BASBL: BRANCH-AND-SANDWICH BILEVEL SOLVER. I. THEORETICAL ADVANCES AND ALGORITHMIC IMPROVEMENTS*

REMIGIJUS PAULAVIČIUS† AND CLAIRE S. ADJIMAN‡

Abstract. In this paper, we consider the global solution of bilevel programs involving nonconvex functions. We present algorithmic improvements and extensions to the recently proposed deterministic Branch-and-Sandwich algorithm (Kleniati and Adjiman, J. Glob. Opt. 60, 425–458, 2014), based on the theoretical results and heuristics. Choices in the way each step of the Branch-and-Sandwich algorithm is tackled, revised bounding schemes, as well as different strategies for branching, node selection, and the node lists management, are explored in this work. The potential benefits of the proposed modifications are examined in an illustrative example.

Key words. Nonconvex bilevel programming, Deterministic global optimization, Branch-and-Sandwich algorithm

AMS subject classifications. 65K05, 90C26, 90C30, 90C57

1. Introduction.

Bilevel programming problems (BPP) have a long history in operations research [6, 13] and occur in diverse applications, such as chemical and civil engineering, economics, transportation; see e.g. [3, 8, 10] and references therein. From the mathematical point of view, bilevel problems are hierarchical mathematical programming problems where an outer (upper-level) problem is constrained by an embedded inner (lower-level) problem:

$$\text{(BPP)} \quad \begin{array}{lll} \min\limits_{\mathbf{x},\mathbf{y}} & F(\mathbf{x},\mathbf{y}) & \\ \text{s.t.} & \mathbf{G}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, & \mathbf{H}(\mathbf{x},\mathbf{y}) = \mathbf{0}, \\ & \mathbf{x} \in X, & \mathbf{y} \in \arg\min\limits_{\mathbf{y}\in Y} \left\{ f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \right\}, \end{array}$$

where the n-dimensional vector $\mathbf{x} \in X \subset \mathbb{R}^n$ denotes the outer (leader) variables and the m-dimensional vector $\mathbf{y} \in Y \subset \mathbb{R}^m$ denotes the inner (follower) variables. Functions $F, f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ denote the outer/inner objective functions, $\mathbf{G} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ and $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^r$ are the vector-valued outer/inner inequality constraint functions and $\mathbf{H} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q$ and $\mathbf{h} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^s$ are vector-valued outer/inner equality constraint functions. In this work, all nonconvex bilevel problems that fall within the class (BPP) are considered without any convexity assumptions on the functions in the problem. Notice that whenever strict convexity does not hold for the inner (sub)problem parameterized by upper level variable $\mathbf{x} \in X$, given by:

$$\text{(ISP}(\mathbf{x})\text{)} \qquad w(\mathbf{x}) = \min_{\mathbf{y}\in Y} \left\{ f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \right\},$$

one must decide whether to adopt an optimistic or pessimistic formulation [8]. Here the optimistic (co-operative) formulation is assumed, where if for a given $\bar{\mathbf{x}}$ the inner subproblem has multiple globally optimal solutions in Y to which the follower is

*Centre for Process Systems Engineering, Department of Chemical Engineering, Imperial College London, London SW7 2AZ, UK.

Funding: We gratefully acknowledge funding from the EPSRC through a Leadership Fellowship [EP/J003840/1].

† (remigijus.paulavicius@imperial.ac.uk).

‡ (c.adjiman@imperial.ac.uk).

indifferent, the leader can choose among them, to achieve the best outer objective value. Hence the outer minimization in (BPP) is over the whole set of variables.

Special cases of bilevel programming have been studied extensively, and many algorithms have been proposed, see, e.g., [3, 7, 8, 9, 10, 28, 35] for reviews. However, the general nonconvex form is very challenging and only recently were the first methods to tackle this class of problems proposed. The deterministic approach of Mitsos et al. [23] and the approximation method of Tsoukalas et al. [33] apply to very general nonlinear bilevel problems, restricted solely by the absence of inner equality constraints. The Branch-and-Sandwich (B&S) algorithm introduced in [16, 17] makes it possible to solve general nonconvex bilevel problems that include inner equality constraints provided that a constraint qualification holds for the inner problem.

The theoretical advances and the B&S algorithm presented in [16] were demonstrated practically in [17] by applying the algorithm to a library of test problems [22], using a combination of software and manual application. The impact of some algorithmic options on performance was briefly investigated in [25]. Recent efforts to develop a fully automated implementation of the algorithm have led to further insights and developments that can lead to a significant reduction the computational effort required to solve nonconvex bilevel problems. These findings, together with a revision of the B&S algorithm and an illustration of the impact of the proposed changes on an example problem, are presented in this paper, Part I of a two-part contribution. It is assumed that the reader has some familiarity with bilevel programming and deterministic global optimization. Thus, not all concepts are defined but appropriate references are provided wherever relevant for readers who are new to the field. In the companion paper, Part II [24], an implementation of the algorithm, the `BASBL` solver, is introduced and extensive computational studies are undertaken. Most aspects of the proposed approach are applicable to the case of mixed integer nonlinear bilevel problems and can therefore be incorporated in the MINLP version of B&S [18], although the specifics of this are beyond the scope of this paper.

The remainder of this paper is organized as follows. In section 2 the Branch-and-Sandwich search tree management strategy is extended to include heuristics for branching and node selection. In section 3 the bounding schemes of B&S are revised, and improvements are presented. In section 4 the revised and the original B&S approaches are summarized. In section 5 an illustrative example is used to investigate the impact of the proposed changes. Finally, section 6 gives conclusions.

2. Branch-and-Sandwich search tree management. The B&S algorithm is a deterministic global optimization algorithm based on the idea of spatial branch-and-bound [15]. It offers a guarantee of convergence to an ϵ-optimal solution [17] in finite time [16, 17]. As in all branch-and-bound algorithms, upper and lower bounds on the solution are obtained at each node. Given the bilevel nature of the problem, such bounds are derived for both the outer and inner problems. In addition, a central and unusual concept in the algorithm is that branching takes place on both the outer $\mathbf{x}$ variables and the inner $\mathbf{y}$ variables. While the partitioning of both variable spaces is crucial for convergence and may have computational benefits [17], it poses a challenge in making sure only bilevel feasible solutions [21] are identified. This is because, in order to draw conclusions on the suboptimality of a specific subset of X, the whole of Y must be considered to ensure that the global solution of the inner problem is taken into account. This implies that multiple nodes must be examined when deriving bounds for a specific node. This challenge is addressed by careful management of the branch-and-bound tree. In this section, Branch-and-Sandwich search tree is presented,

and alternative strategies for branching and the management of nodes are introduced.

2.1. Basic definitions and notation. We start with concepts and definitions required for the Branch-and-Sandwich tree description.

DEFINITION 2.1 (Node). *A node k (equivalently an $(n+m)$-rectangle) in the Branch-and-Sandwich tree is uniquely defined by a set of bounds, representing a subdomain of $X \times Y$:*

$$k = \left\{ \mathbf{v} = (\mathbf{x}, \mathbf{y})^T \in X^{(k)} \times Y^{(k)} \subseteq X \times Y \subset \mathbb{R}^{(n+m)} \mid \mathbf{v}^{(k,\mathrm{L})} \leq \mathbf{v} \leq \mathbf{v}^{(k,\mathrm{U})} \right\}, \tag{2.1}$$

where superscripts (k,L) and (k,U) indicate the lower and upper variable bounds for node k, respectively.

DEFINITION 2.2 (Node properties). *Each node k in the Branch-and-Sandwich tree has a unique number, denoted by superscript (k). The root node (the whole $X \times Y$ domain) has $(k) = 1$. A node k has the following attributes (properties):*

- $\underline{f}^{(k)}$: *A valid lower bound on the global solution of the inner problem restricted to node k;*
- $\bar{f}^{(k)}$: *A valid upper bound on the global solution of the inner problem restricted to node k;*
- $\underline{F}^{(k)}$: *A valid lower bound on the global solution of the bilevel problem over node k;*
- $\bar{\mathbf{x}}^{(k)}$: *Outer variable vector corresponding to $\underline{F}^{(k)}$;*
- $s^{(k)}$: *State of node k: active, inner-active or inactive (cf. subsection 2.4).*

To provide some context to the concepts presented in the following sections, a brief statement of the main steps of the Branch-and-Sandwich algorithm [16, 17] is given in Algorithm 2.1.

Algorithm 2.1 Branch-and-Sandwich

```
Initialize
Compute inner and outer bounds at the root node. Apply fathoming-rules.
while list of active nodes is not empty do                          ▷ subsection 2.2.1
    Select node(s) for branching.                                   ▷ subsection 2.3
    Branch selected node(s). Update lists.                          ▷ subsections 2.2.1 and 2.2.2
    for each new node do
        Compute inner lower bound. Apply full-fathoming.            ▷ subsections 2.4 and 3.1
        Compute inner upper bound.                                  ▷ subsection 3.2
        Update best inner upper bound. Apply full-fathoming.        ▷ subsections 2.4 and 3.3
        Compute outer lower bound. Apply outer-fathoming.           ▷ subsections 2.4 and 3.4
        Compute inner subproblems.                                  ▷ subsection 3.5
        Compute outer upper bound. Apply outer-fathoming.           ▷ subsections 2.4 and 3.5
    end for
end while
if problem is feasible then
    return the best found solution and the objective(s) value(s).
else
    return problem is infeasible.
end if
```

2.2. Branching scheme. In this section, it is assumed that bounds are available. Strategies to derive bounds are discussed in section 3. As mentioned, the branching scheme for bilevel problems differs from that for single level problems. The sequential decision making inherent in bilevel problems implies that the entire host set Y must be considered to bound the solution of the inner level problem. To ensure this, the algorithm proposed by Mitsos et al. [23] allows branching on the outer variables, but not on the inner variables. The branching scheme introduced in the original B&S algorithm [16], on the other hand, allows branching on both variable sets without any distinction between $\mathbf{x}$ and $\mathbf{y}$ variables.

2.2.1. List management. The replacement of the parent node k by new child nodes in the Branch-and-Sandwich search tree is managed by using appropriate lists of nodes with special properties, in addition to the classical list of *open* (unfathomed/active) nodes $\mathcal{L}$ used in standard branch-and-bound algorithms [15]. In particular, the list $\mathcal{L}_{\text{In}}$ is the list of *outer-fathomed* (or *inner-active*) nodes, i.e., nodes that are known not to contain the bilevel solution but that may contain bilevel feasible points; these nodes should be explored further with respect to the inner problem only. Lists $\mathcal{L}$ and $\mathcal{L}_{\text{In}}$ are disjoint, i.e.,: $\mathcal{L} \cap \mathcal{L}_{\text{In}} = \emptyset$ and their union $\mathcal{L} \cup \mathcal{L}_{\text{In}}$ contains all the active nodes. In addition to lists $\mathcal{L}$ and $\mathcal{L}_{\text{In}}$, we use auxiliary (independent) lists $\mathcal{L}^p, p \in P$ corresponding to each X-partition (Definition 2.3).

DEFINITION 2.3 (X-partition). *Let $P \subset \mathbb{N}$ be a finite index set and $\mathcal{X}_p$ denotes a subdomain of X. Then an X-partition is denoted as $\{\mathcal{X}_p \subseteq X : p \in P\}$ where:*

$$X = \bigcup_{p \in P} \mathcal{X}_p \text{ and } \mathcal{X}_p \cap \mathcal{X}_q = \partial\mathcal{X}_p \cap \partial\mathcal{X}_q \text{ for all } p, q \in P, p \neq q, \tag{2.2}$$

and $\partial\mathcal{X}_p$ denotes the relative boundary [5] of $\mathcal{X}_p$.

The independent lists link active nodes appropriately such that the hierarchical structure of the bilevel problem is maintained, i.e., a list $\mathcal{L}^p$, corresponding to subdomain $\mathcal{X}_p$, consists of a collection of *sublists* containing nodes with X subdomains in $\mathcal{X}_p$, but non-overlapping Y subdomains:

$$\mathcal{L}^p = \{\mathcal{L}^p_1, \ldots, \mathcal{L}^p_{s_p}\}. \tag{2.3}$$

A list $\mathcal{L}^p, p \in P$, is generated as follows: given the set of nodes in $\mathcal{X}_p$ we create the minimum number s_p of sublists $\mathcal{L}^p_s, s \in \{1, \ldots, s_p\}$, such that every node in $\mathcal{X}_p$ is in at least one sublist $\mathcal{L}^p_s \in \mathcal{L}^p$. This is formally stated in Definition 2.4.

DEFINITION 2.4 (List ($\mathcal{L}^p$) generation process). *In order to generate $\mathcal{L}^p, p \in P$ corresponding to the set $\mathcal{X}_p$, we generate sublist(s) $\mathcal{L}^p_s$, such that*

$$\mathcal{L}^p_s = \{k \in \mathcal{L} \cup \mathcal{L}_{\text{In}} : \mathit{relint}(X^{(k)}) \cap \mathit{relint}(\mathcal{X}_p) \neq \emptyset\}, \quad s \in \{1, \ldots s_p\}, \tag{2.4}$$

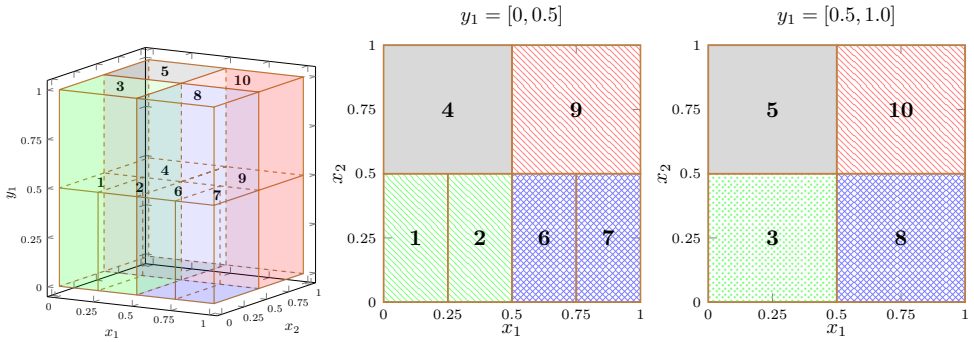
where $\mathit{relint}(\cdot)$ denotes the relative interior of a set [5] and

$$X^{(k)} \in \mathcal{X}_p \Leftrightarrow \exists s \in \{1, \ldots, s_p\} : X^{(k)} \in \mathcal{L}^p_s, \tag{2.5}$$

$$\mathit{relint}(X^{(i)}) \cap \mathit{relint}(X^{(j)}) \neq \emptyset \quad \textit{for all } i, j \in \mathcal{L}^p_s, i \neq j, s \in \{1, \ldots, s_p\}, \tag{2.6}$$

$$\mathit{relint}(Y^{(i)}) \cap \mathit{relint}(Y^{(j)}) = \emptyset \quad \textit{for all } i, j \in \mathcal{L}^p_s, i \neq j, s \in \{1, \ldots, s_p\}. \tag{2.7}$$

In particular, every sublist $\mathcal{L}^p_s, s \in \{1, \ldots, s_p\}$ must represent a Y partition. Namely, the collection of subdomains $\{Y^{(k)}\}, k \in \mathcal{L}^p_s$, is a partition of Y for each



(a) 3-d representation (b) 2-d representation (c) 2-d representation

FIG. 2.1. (a) *Illustration of the list generation process in* $X \times Y = [0,1]^2 \times [0,1]$. *All lists and corresponding sublists are shown in Table 2.1. The use of the same color for the cuboids (Figure 2.1a) denotes nodes that belong to the same independent list corresponding to different* X*-partitions:* $\mathcal{X}_1 = [0,0.5] \times [0,0.5]$, $\mathcal{X}_2 = [0,0.5] \times [0.5,1]$, $\mathcal{X}_3 = [0.5,1] \times [0,0.5]$, $\mathcal{X}_4 = [0.5,1] \times [0.5,1]$. (b) 2-$d$ *projection of the* $y_1 \in [0,0.5]$. *The use of the same pattern for several rectangles in Figures 2.1b and 2.1c denotes nodes that belong to the same independent list.* (c) 2-d *projection of the* $y_1 \in [0.5,1.0]$.

TABLE 2.1
Independent lists and corresponding sublists appearing in partitioning example shown in Figure 2.1

$\mathcal{X}_1 = [0,0.5]^2$	$\mathcal{X}_2 = [0,0.5] \times [0.5,1]$	$\mathcal{X}_3 = [0.5,1] \times [0,0.5]$	$\mathcal{X}_4 = [0.5,1]^2$
$\mathcal{L}^1 = \{\mathcal{L}^1_1, \mathcal{L}^1_2\}$	$\mathcal{L}^2 = \{\mathcal{L}^2_1\}$	$\mathcal{L}^3 = \{\mathcal{L}^3_1, \mathcal{L}^3_2\}$	$\mathcal{L}^4 = \{\mathcal{L}^4_1\}$
$\mathcal{L}^1_1 = \{1,3\}$ $\mathcal{L}^1_2 = \{2,3\}$	$\mathcal{L}^2_1 = \{4,5\}$	$\mathcal{L}^3_1 = \{6,8\}$ $\mathcal{L}^3_2 = \{7,8\}$	$\mathcal{L}^4_1 = \{9,10\}$

$s \in \{1, \ldots, s_p\}$. In other words, for all $\mathbf{x} \in \mathcal{X}_p$ the "whole" Y is maintained, that is $\forall \mathbf{x} \in \mathcal{X}_p, \forall \mathbf{y} \in Y, (\mathbf{x}, \mathbf{y}) \in X^{(k)} \times Y^{(k)} \Rightarrow \exists \mathcal{L}^p_s \in \mathcal{L}^p : X^{(k)} \times Y^{(k)} \in \mathcal{L}^p_s$. This highlights the significance of independent lists: each such list $\mathcal{L}^p$ contains all information on subset $\mathcal{X}_p$ of X and in particular covers all of Y. Thus an independent list $\mathcal{L}^p$ contains all the information necessary to examine the suboptimality of a node whose domain $\mathcal{X}^{(k)}$ is in $\mathcal{X}_p$. In some cases, the partition of Y may be incomplete where it has been shown that a subset of $\mathcal{X}_p \times Y$ cannot contain the global solution of the inner problem.

EXAMPLE 2.1. *The implications of Definition 2.4 are illustrated in Figure 2.1 and Table 2.1, where for each* $\mathcal{X}_p, p \in \{1, \ldots, 4\}$, *more than one sublist may satisfy (2.4)–(2.7) and a given node can appear in more than one sublist. In this situation, independent lists* $\mathcal{L}^1$ *and* $\mathcal{L}^3$ *(corresponding to partitions* $\mathcal{X}_1$ *and* $\mathcal{X}_3$*) have two sublists each:* $\mathcal{L}^1 = \{\mathcal{L}^1_1, \mathcal{L}^1_2\}$ *and* $\mathcal{L}^3 = \{\mathcal{L}^3_1, \mathcal{L}^3_2\}$ *and node* $k = 3$ *belongs to both sublists in* $\mathcal{L}^1$ *while node* $k = 8$ *belongs to both sublists in* $\mathcal{L}^3$.

REMARK 2.1. *Definition 2.4 is valid for any partition. As we will see in subsection 2.2.2, the branching scheme adopted here results in regular partitions [14] such as the one presented in Example 2.1.*

2.2.2. Branching variable and branching point selection. The branching scheme of B&S is specified by two rules: a branching variable selection rule and a node

selection rule. The choice of branching variable (coordinate axis) influences the structure of the tree generated and can significantly affect the computational performance of the algorithm. Let the variable vector $\mathbf{v} = (\mathbf{x}, \mathbf{y})^{\mathrm{T}} = (v_1, \ldots, v_n, v_{n+1}, \ldots, v_{n+m})^{\mathrm{T}}$ be formed first from the n outer variables and then from the m inner variables. In the original B&S approach [16], the branching variable is selected by giving the highest priority to the variable with the largest range (longest edge). If several edges satisfy the longest edge requirement, then the one with the smallest index is selected to be subdivided.

It is easy to notice that, if some variable bounds differ significantly in magnitude, branching on the variables with the shortest edge is not performed until the ranges for all variables are sufficiently reduced. In the revised version we select the branching variable with the largest normalized range as has long been practised in branch-and-bound algorithms (e.g., see [1, 27]). Also, when an ambiguous situation arises, i.e., when several variables satisfy the longest (normalized) edge requirement we employ two different strategies, where the priority is given either to the variable with the lowest variable index (strategy (XY), as this gives priority to outer variables $\mathbf{x}$) or the variable with the highest index (strategy (YX)).

DEFINITION 2.5 (Branching variable selection rules). *Consider a node $k \in (\mathcal{L} \cup \mathcal{L}_{\mathrm{In}}) \cap \mathcal{L}^p$. Find the lowest-index (XY)/highest-index (YX) branching variable v_{br} with the largest normalized range (edge):*

$$\text{(XY)} \qquad v_{\mathrm{br}} = \min \left\{ \underset{i=1,\ldots,n+m}{\arg\max} \left\{ \left(v_i^{(k,\mathrm{U})} - v_i^{(k,\mathrm{L})} \right) \Big/ \left(v_i^{(1,\mathrm{U})} - v_i^{(1,\mathrm{L})} \right) \right\} \right\};$$

$$\text{(YX)} \qquad v_{\mathrm{br}} = \max \left\{ \underset{i=1,\ldots,n+m}{\arg\max} \left\{ \left(v_i^{(k,\mathrm{U})} - v_i^{(k,\mathrm{L})} \right) \Big/ \left(v_i^{(1,\mathrm{U})} - v_i^{(1,\mathrm{L})} \right) \right\} \right\};$$

Having identified a branching variable, a standard bisection strategy is adopted to select a branching point, although other approaches could be adopted [2].

DEFINITION 2.6 (Branching point selection rule). *For the selected branching variable v_{br}, the branching point is found using exact bisection [34].*

A description of the proposed subdivision process, i.e., the selection of the branching variable and branching point as well as the list management, is formally stated in Definition 2.7.

DEFINITION 2.7 (Subdivision process). *Consider a node $k \in (\mathcal{L} \cup \mathcal{L}_{\mathrm{In}}) \cap \mathcal{L}^p$. Then*

Step 1 *Find the branching variable v_{br} using Definition 2.5.*

Step 2 *Create two new child nodes $k_i, i = 1, 2$ by branching on the variable v_{br} (Definition 2.6).*

Step 3 *Update nodes in the sublists based on the type of branching variable:*

Step 3.1 *If v_{br} is an inner $(\mathbf{y})$ variable, modify all sublists $\mathcal{L}_s^p$, $s \in \{1, 2, \ldots, s_p\}$ containing k:*

$$\text{If } X^{(k)} \subseteq \mathcal{L}_s^p, \quad \mathcal{L}_s^p = (\mathcal{L}_s^p \setminus \{k\}) \cup \{k_1, k_2\}, \quad s \in \{1, 2, \ldots, s_p\}.$$

Step 3.2 *If v_{br} is an outer $(\mathbf{x})$ variable, replace all sublists $\mathcal{L}_s^p$ containing k by one or two new sublists:*

Step 3.2.1 *For $s \in \{1, 2, \ldots, s_p\}$ and for $i = 1, 2$, if $\exists j \in \mathcal{L}_s^p \setminus \{k\}$: $\mathit{relint}(X^{(k_i)}) \cap \mathit{relint}(X^{(j)}) \neq \emptyset$, create:*

$$\mathcal{L}_{s_i}^p = (\mathcal{L}_s^p \setminus \{k\}) \cup \{k_i\}.$$

Step 3.2.2 *Check an independence condition* (IC) *[16], i.e., if there exist index sets I and J such that:*

$$\text{(IC)} \qquad \begin{gathered} I \cap J = \emptyset, I \cup J = \{1, \dots, s_p\}, \\ \{k_1 \in \mathcal{L}_i^p : i \in I\} \cap \{k_2 \in \mathcal{L}_j^p : j \in J\} = \emptyset, \end{gathered}$$

replace $\mathcal{L}^p$ with two new independent lists $\mathcal{L}^{p_1}$ and $\mathcal{L}^{p_2}$:

$$\begin{aligned} \mathcal{L}^{p_1} &= \{\mathcal{L}_i^p\}, i \in I \subset \{1, \dots, s_p\}, \\ \mathcal{L}^{p_2} &= \{\mathcal{L}_j^p\}, j \in J \subset \{1, \dots, s_p\}, \end{aligned}$$

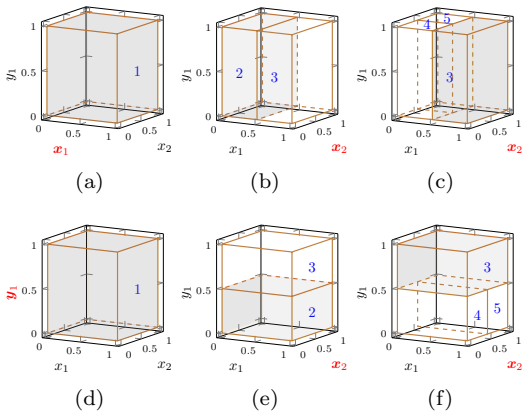
where $p_1 = p$ and $p_2 = |P| + 1$.

An illustration of the proposed subdivision process is presented in Example 2.2.

EXAMPLE 2.2. Consider the six partitioning examples of the three-dimensional space $X \times Y = [0,1]^2 \times [0,1]$ presented in Figure 2.2. In these pictures, the grey color highlights a node that has been selected for branching. Numbers inside the cuboids are the unique node numbers (k). At the first iteration (Figures 2.2a and 2.2d), the selected branching variable v_{br} depends on the branching variable selection rule, as all three variables satisfy the normalized longest range requirement (see Definition 2.7, Step 1). Therefore, when the (XY) strategy is used, i.e., when priority is given to the variable with the lowest index, the selected branching variable is $v_{\text{br}} = x_1$ (as noted Figure 2.2a); however when the (YX) strategy is used, the selected branching variable is $v_{\text{br}} = y_1$ (see Figure 2.2d). After branching (bisection) on the selected variable (see Definition 2.7, Step 2) in each case, we obtain two completely different partitions illustrated in Figures 2.2b and 2.2e, respectively. After bisection on variable x_1, there are two independent lists with one sublist each: $\mathcal{L}^1 = \mathcal{L}_1^1 = \{2\}$ and $\mathcal{L}^2 = \mathcal{L}_1^2 = \{3\}$, but there is only one independent list after bisection on variable y_1: $\mathcal{L}^1 = \mathcal{L}_1^1 = \{2, 3\}$.

If we assume that node 2 is selected at the next iteration, the same variable, x_2, is chosen using both branching variable selection rules. However, after branching we obtain two completely different Branch-and-Sandwich trees. In the (XY) case, which is shown in Figure 2.2c, we have three independent lists: $\mathcal{L}^1 = \mathcal{L}_1^1 = \{4\}$, $\mathcal{L}^2 = \mathcal{L}_1^2 = \{3\}$ and $\mathcal{L}^3 = \mathcal{L}_1^3 = \{5\}$, while in the (YX) case, we continue to have one independent list but now with two sublists: $\mathcal{L}^1 = \{\mathcal{L}_1^1, \mathcal{L}_2^1\}$ where $\mathcal{L}_1^1 = \{3, 4\}, \mathcal{L}_2^1 = \{3, 5\}$. Thus, the (XY) branching variable strategy speeds up the refinement of the X partition and produces a higher number of independent lists, which can be examined completely independently, while the (YX) strategy results in a smaller number of independent lists and faster refinement of the Y space. While performance of the two approaches may be problem-dependent in a serial implementation, strategy (XY) thus appears to be better suited to a parallel implementation.

2.3. Node selection rules. The selection of nodes for branching influences the structure of the branch-and-bound tree and can have a significant impact on the number of iterations needed for the convergence [19, 26]. However, well-known heuristics for node selection from single-level global optimization cannot be applied directly to the bilevel case, as they consider only the "outer" level information. To select the "most promising" nodes in $\mathcal{L} \cup \mathcal{L}_{\text{In}}$ based on the information from both levels, B&S first identifies the most promising independent list (X-partition) taking into account outer-level information. When a list is found, B&S continues to look for the best candidate only among nodes belonging to this list and selects an appropriate node by taking into account inner-level information. In the revised version presented



	$\mathcal{L}^1$	$\mathcal{L}^2$	$\mathcal{L}^3$
a	$\mathcal{L}_1^1 = \{1\}$	–	–
b	$\mathcal{L}_1^1 = \{2\}$	$\mathcal{L}_1^2 = \{3\}$	–
c	$\mathcal{L}_1^1 = \{4\}$	$\mathcal{L}_1^2 = \{3\}$	$\mathcal{L}_1^3 = \{5\}$

	$\mathcal{L}^1$	$\mathcal{L}^2$	$\mathcal{L}^3$
d	$\mathcal{L}_1^1 = \{1\}$	–	–
e	$\mathcal{L}_1^1 = \{2, 3\}$	–	–
f	$\mathcal{L}_1^1 = \{3, 4\}, \mathcal{L}_2^1 = \{3, 5\}$	–	–

FIG. 2.2. *An illustration of the partitions in Example 2.2 is shown on the left-hand side. Cuboids in Figures 2.2a to 2.2c depict the (XY) strategy for three consecutive nodes. Cuboids in Figures 2.2d to 2.2f depict the (YX) strategy for three consecutive nodes. The selected branching variable is shown in bold (red). The independent lists and the sublists appearing in each partitioning are shown in the tables on the right-hand side.*

here, we employ four variants of the node selection operation. In the same vein as in the original selection procedure, B&S selects one node from the list of active nodes $\mathcal{L}$ at each iteration and one from the list of inner-active nodes $\mathcal{L}_{\text{In}}$, if non-empty. The node selection procedure is described in Definition 2.8. For each of the two steps in the node selection procedure, two options are given and the four variants are obtained by taking all combinations of these options.

DEFINITION 2.8 (Extended node selection rules).

Step 1 *Selecting an independent list. There are two options for this step:*

Option $(\underline{F}l)$*: Find the independent list* $\mathcal{L}^p, p \in P$*, containing the node* $k \in \mathcal{L}^p \cap \mathcal{L}$ *with the lowest lower bound* $(\underline{F}^{(k)})$ *and, if several nodes with the smallest lower bound exist, the smallest level* $(l^{(k)})$*:*

$$k = \arg\min_i \left\{ l^{(i)} : i = \arg\min_{j \in \mathcal{L}} \left\{ \underline{F}^{(j)} \right\} \right\}; \tag{$\underline{F}l$}$$

Option $(l\underline{F})$*: Find* $\mathcal{L}^p$ *with a node* k *with the smallest level* $(l^{(k)})$ *and, if several nodes with the smallest level exists, with the lowest lower bound* $(\underline{F}^{(k)})$*:*

$$k = \arg\min_i \left\{ \underline{F}^{(i)} : i = \arg\min_{j \in \mathcal{L}} \left\{ l^{(j)} \right\} \right\}; \tag{$l\underline{F}$}$$

Step 2 *Selecting nodes for branching. Again, there are two options here:*

Option $(l\underline{f})$*: 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, with the smallest levels* ($l^{(k)}$ *and* $l^{(k_{\text{In}})}$ *respectively) and, if several nodes with the smallest level exist, the lowest inner lower bounds* ($\underline{f}^{(k)}$ *and* $\underline{f}^{(k_{\text{In}})}$ *respectively):*

$$\begin{aligned} k &= \arg\min_i \{ \underline{f}^{(i)} : i = \arg\min_{j \in \mathcal{L} \cap \mathcal{L}^p} \{ l^{(j)} \} \} \\ k_{\text{In}} &= \arg\min_i \{ \underline{f}^{(i)} : i = \arg\min_{j \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p} \{ l^{(j)} \} \}; \end{aligned} \tag{$l\underline{f}$}$$

Option $(l\bar{f})$*: Select nodes* k *and* k_{In} *with the smallest levels* $(l^{(k)})$ *and, if several nodes with the lowest level exist, the smallest relaxed inner upper bounds* $(\bar{f}^{(k)})$*:*

$$(l\bar{f}) \qquad \begin{aligned} k &= \arg\min_i \{\bar{f}^{(i)} : i = \arg\min_{j \in \mathcal{L} \cap \mathcal{L}^p} \{l^{(j)}\}\} \\ k_{\text{In}} &= \arg\min_i \{\bar{f}^{(i)} : i = \arg\min_{j \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p} \{l^{(j)}\}\}. \end{aligned}$$

The original B&S algorithm node selection procedure described in [16] is very similar to the $(\underline{F}l)$-$(l\underline{f})$ heuristic described here. It implies that preference is given in Step 1 to the independent list containing the node(s) with the best (lowest) outer lower bounding value and in Step 2, to the nodes with the smallest level, i.e, covering the largest subdomain of $X \times Y$. When several nodes in the selected list are at the same smallest level in the Branch-and-Sandwich tree, the node corresponding to the lowest inner lower bound $\underline{f}$ is chosen (the best candidate from the perspective of the inner problem). In our proposed approach, we resolve the ambiguity that arises in Step 1, when there are nodes with the same smallest outer lower bound value in different independent lists. In this case, B&S selects the list containing the larger node, i.e., the one with the smallest l value.

The $(l\bar{f})$ heuristic is an alternative strategy to $(l\underline{f})$ where in Step 2, if there are several nodes at the same level of the tree in the selected independent list, the node with the lowest inner upper bound value is chosen. In this way, we choose the best candidate in the selected list from the perspective of the outer problem, as tighter inner upper bound values lead to larger values of the outer lower bound and are thus likely to result in faster outer-fathoming (see subsection 2.4).

In the remaining two node selection variants, the independent list corresponding to $(l\underline{F})$ is selected, i.e., we focus on the list containing the largest node (that is, the smallest level l).

The four variants are illustrated in Example 2.3.

EXAMPLE 2.3. *(Illustration of Definition 2.8) Consider the partitioning of a two-dimensional space* $X \times Y = [-1, 1] \times [-1, 1]$*, shown in Figure 2.3a with the corresponding tree shown in Figure 2.3b. All leaf nodes belong to the active list, i.e.,* $\mathcal{L} = \{3, 4, 5\}$ *and* $\mathcal{L}_{\text{In}} = \emptyset$*. There are two independent lists corresponding to each member set* $\mathcal{X}_p$ *of the* X *partition with one sublist each: for* $\mathcal{X}_1$*,* $\mathcal{L}^1 = \mathcal{L}^1_1 = \{4, 5\}$ *and for* $\mathcal{X}_2$*,* $\mathcal{L}^2 = \mathcal{L}^2_1 = \{3\}$*. In Step 1, using the* $(l\underline{F})$ *heuristic, B&S selects independent list* $\mathcal{L}^2$ *as it contains node 3, the only active node at level 1. As node 3 is the sole candidate in* $\mathcal{L}^2$*, B&S selects this node in Step 2. However, when the* $(\underline{F}l)$ *selection heuristic is used in Step 1, B&S selects list* $\mathcal{L}^1$ *as it contains node 4 which has the smallest lower bound value* $\underline{F}^{(4)} = 0.00$*. But in Step 2, the two heuristics based on inner-level information return different candidates, as* $l^{(4)} = l^{(5)} = 2$ *but* $\underline{f}^{(4)} < \underline{f}^{(5)}$ *and* $\bar{f}^{(4)} > \bar{f}^{(5)}$*. As a result, variant* $(\underline{F}l)$-$(l\underline{f})$ *leads to the selection of node 4 and* $(\underline{F}l)$-$(l\bar{f})$ *to that of node 5.*

2.4. Node fathoming rules. Based on the bounding values computed at a node k, we may decide to fathom the current node. Two types of fathoming operations take place in the B&S algorithm. If a node has been shown not to contain a global solution of the inner problem, it is removed from all lists ("*fully-fathomed*"). When a node has been shown not to contain a global solution of the bilevel problem, but it may nevertheless contain global solutions of the inner problem over the whole Y, we need to continue exploring it further via branching. In such a case, it is "*outer-fathomed*"

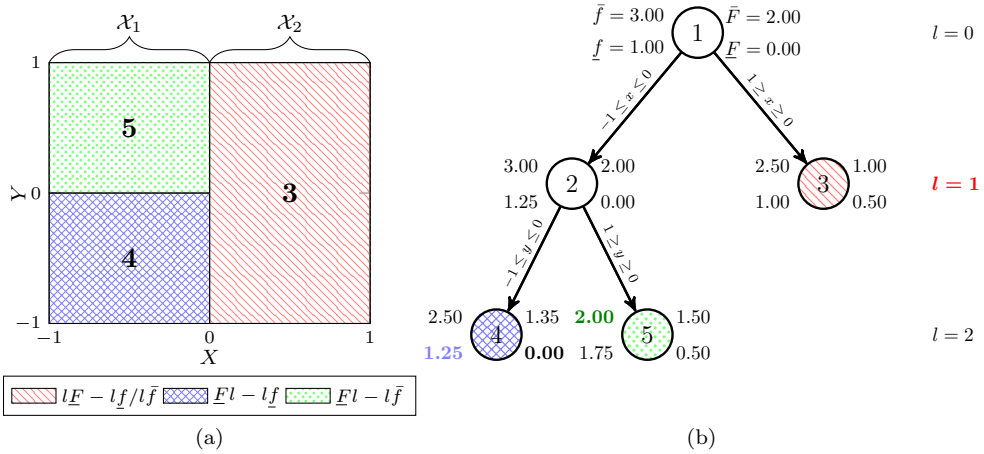


FIG. 2.3. *Illustration of the nodes selected for branching when using different node selection heuristics. See text for further description.*

rather than fully fathomed. As a result, a node in the Branch-and-Sandwich tree may have three different states (s):

- *active* or *open* leaf nodes which B&S continues to explore for both the outer and inner problems, because they may contain a global solution of the bilevel problem;
- *inner-active (outer-fathomed)*: those leaf nodes which B&S continues to explore on the inner problem only, because they have been shown not to contain a global solution of the bilevel problem but may contain a global solution for the inner problem.
- *inactive (fully-fathomed)* all other nodes. These nodes are deleted from all lists and no further exploration of these nodes is performed.

DEFINITION 2.9 (Fathoming rules). *Given node $k \in (\mathcal{L} \cup \mathcal{L}_{\text{In}}) \cap \mathcal{L}^p$, if one of the following hold:*

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

then delete k from $\mathcal{L}$ (or $\mathcal{L}_{In}$) and $\mathcal{L}^p$.

REMARK 2.2. *Note that we fully fathom an inner-active node if the inner upper bounding problem is infeasible, i.e., $\bar{f}^{(k)} = \infty$. Infeasibility of this problem occurs when the domain of interest (node k) does not contain feasible bilevel points; thus it is safe to remove this node from consideration.*

DEFINITION 2.10 (Outer-fathoming rules). *Given outer objective tolerance ε_F, outer-fathom node k if:*

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

then move k from $\mathcal{L}$ to $\mathcal{L}_{In}$.

2.5. List deletion rules. A node k that has not yet been fully fathomed belongs either to $\mathcal{L}$ or to $\mathcal{L}_{\text{In}}$. It must also belong to an independent list $\mathcal{L}^p$ for some $p \in P$,

i.e., $k \in (\mathcal{L} \cup \mathcal{L}_{\text{In}}) \cap \mathcal{L}^p$. However, if a certain sublist contains only outer-fathomed nodes, i.e., it no longer contains nodes in $\mathcal{L}$ which are active from the perspective of the overall problem, then it can be deleted. All the nodes that belong only to the deleted sublist and to no other sublist can be fully-fathomed too.

DEFINITION 2.11 (List deletion rules). *Consider a sublist $\mathcal{L}_i^p \in \mathcal{L}^p, i \in \{1, \dots, s_p\}$.*

1. *If $\mathcal{L}_i^p \cap \mathcal{L} = \emptyset$ and $\mathcal{L}_i^p \cap \mathcal{L}_j^p = \emptyset$ for all $i \neq j \in \{1, \dots, s_p\}$, then fully-fathom all nodes $k \in \mathcal{L}_i^p$, i.e., delete them from $\mathcal{L}_{\text{In}}$ and $\mathcal{L}^p$. Also delete sublist $\mathcal{L}_i^p$, i.e., set $\mathcal{L}^p = \mathcal{L}^p \setminus \{\mathcal{L}_i^p\}$ and decrease s_p by 1.*
2. *If $\mathcal{L}_i^p \cap \mathcal{L} = \emptyset$ and $\mathcal{L}_i^p \cap \mathcal{L}_j^p \neq \emptyset$ for some $i \neq j \in \{1, \dots, s_p\}$ then delete (fully-fathom) all the nodes from $\mathcal{L}_{\text{In}}$ and $\mathcal{L}^p$ belonging only to sublist $\mathcal{L}_i^p$. Then delete sublist $\mathcal{L}^p = \mathcal{L}^p \setminus \{\mathcal{L}_i^p\}$ and decrease s_p by 1.*
3. *If $s_p = 0$, delete "independent" list $\mathcal{L}^p$ and decrease $|P|$ by 1.*

REMARK 2.3. *Note that in Step 2 of the list deletion rules (Definition 2.11), when sublist $\mathcal{L}_i^p$ contains only outer-fathomed nodes, we delete all the nodes belonging to this sublist only from $\mathcal{L}_{\text{In}}$ and from $\mathcal{L}^p$ before deletion of $\mathcal{L}_i^p$ takes place.*

REMARK 2.4. *If the node $k \in \mathcal{L}^p$ is the sole candidate and was outer-fathomed based on Definition 2.10, it can also be full-fathomed.*

Remark 2.4 suggests that it could be beneficial to investigate branching heuristics in which branching first takes place on the outer variables ($\mathbf{x}$), with the aim to locate the most promising regions from the perspective of the outer problem as fast as possible. If during this exploration *outer-fathomed* regions are detected, they can also be fully-fathomed. Once the most promising subsets of X have been identified, one could then start branching on the inner variables ($\mathbf{y}$), keeping the number of sublists small and improving the best inner upper bound, which helps to *fully-fathom* regions that are bilevel-infeasible, i.e., that do not contain a global solution of the inner problem.

3. Revised bounding schemes. In this section, we present extensions and improvements to the original B&S bounding scheme. The B&S algorithm makes use of an equivalent single-level (BPP) formulation [11]:

$$\text{(slBPP)} \qquad \begin{aligned} \min_{\mathbf{x},\mathbf{y}} \ & F(\mathbf{x},\mathbf{y}) \\ \text{s.t. } & \mathbf{G}(\mathbf{x},\mathbf{y}) \leq \mathbf{0},\ \mathbf{H}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & f(\mathbf{x},\mathbf{y}) \leq w(\mathbf{x}) \\ & \mathbf{x} \in X, \mathbf{y} \in Y, \end{aligned}$$

where for a fixed $\bar{\mathbf{x}} \in X$ the *optimal value function* $w(\bar{\mathbf{x}})$ of the inner problem is defined in (ISP($\mathbf{x}$)). The bounding scheme of the B&S algorithm is partly based on the observation [23] that a modification of (slBPP) and particularly of the constraint $f(\mathbf{x},\mathbf{y}) \leq w(\mathbf{x})$ to include an upper bound on the optimal value function $w(\mathbf{x})$ yields a relaxation of (a lower bound on) the overall problem (slBPP) and consequently, a relaxation of the original (BPP). Similarly, a lower bound on $w(\mathbf{x})$ gives a restriction of (an upper bound on) the overall problem (slBPP). Given this, the outer bounding scheme is based on finding valid lower and upper bounds on the inner optimal value $w(\mathbf{x})$ for the domain of interest. In this way, we can derive the smallest and the largest possible inner objective values corresponding to the feasible bilevel points for the domain of interest.

3.1. Inner lower bounding scheme. For an *active* or *inner-active* node $k \in (\mathcal{L} \cup \mathcal{L}_{\text{In}})$, the inner lower bounding problem (ILB(k)) over the current subdomain $X^{(k)} \times Y^{(k)}$ is:

$$\text{(ILB}(k)\text{)} \qquad f^{(k),\text{L}} = \min_{\mathbf{x}\in X^{(k)},\mathbf{y}\in Y^{(k)}} \{f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}\},$$

where the minimum over all values of the outer ($\mathbf{x}$) and the inner ($\mathbf{y}$) variables vectors is taken. Note that fixing the outer variables and minimizing with respect to the inner variables only would yield an invalid inner lower bound [16]. This situation is illustrated in Example 3.1.

EXAMPLE 3.1 (`mb_1_1_08` instance from [22]).

$$\begin{aligned} &\min_{x,y} x + y \\ &s.t.\ y \in \arg\min_{y} \frac{xy^2}{2} - \frac{y^3}{3} \\ &\quad x \in [-1,1], y \in [-1,1] \end{aligned}$$

Fixing $\bar{x} = 0.5$ and solving (ILB(k)) over the whole of Y, i.e., $y \in [-1,1]$ (over the green line in Figure 3.4) yields -0.08. However, the exact inner lower bound is $f^{(1),\text{L}} = -0.83$, proving that -0.08 is invalid.

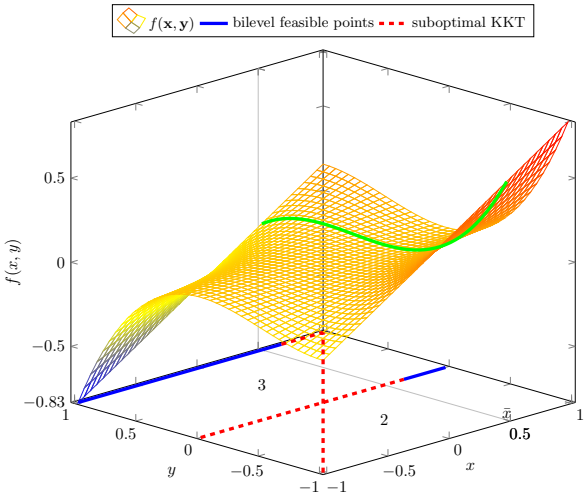


FIG. 3.4. *Graphical illustration of the inner problem from the bilevel `mb_1_1_08` instance [22] together with its minima (thick (blue) lines) and suboptimal KKT points (dashed (red) lines). The graphical illustration shows that a valid inner lower bound (ILB(k)) cannot be found by simply fixing $\bar{x}$, e.g., $\bar{x} = 0.5$ (thick (green) curve on the surface), and then solving (ILB(k)) over the whole of Y.*

A computationally less expensive relaxed inner lower bound can be obtained by generating a valid lower bound on the solution of (ILB(k)), either by calling a global optimization NLP solver for a limited number of iterations (or with a loose optimality tolerance) or by solving the following convexification of (ILB(k)) [16]:

$$\text{(RILB}(k)\text{)} \qquad \breve{f}^{(k),\text{L}} = \min_{\mathbf{x}\in X^{(k)},\mathbf{y}\in Y^{(k)}} \{\breve{f}(\mathbf{x},\mathbf{y}) \text{ s.t. } \breve{\mathbf{g}}(\mathbf{x},\mathbf{y}) \leq \mathbf{0},\ \breve{\mathbf{h}}(\mathbf{x},\mathbf{y}) = \mathbf{0}\},$$

where $\breve{f}, \breve{g}$ and $\breve{h}$ represent convex underestimators (e.g.,[2, 31]) of functions f, g and h on $X^{(k)} \times Y^{(k)}$.

REMARK 3.1. *The lower bound* $\underline{f}$ *introduced in Definition 2.2 and used in the node selection strategies and fathoming rules can be obtained by solving* (ILB(k)) ($\underline{f} = f^{(k),\mathrm{L}}$) *or* (RILB($k$)) ($\underline{f} = \breve{f}^{(k),\mathrm{L}}$).

3.2. Inner upper bounding scheme. For a node $k \in (\mathcal{L} \cup \mathcal{L}_{\mathrm{In}})$ a valid inner upper bound (IUB(k)) is obtained by accounting for all values of the outer variables $\mathbf{x}$ and taking the worst-case scenario with respect to those:

$$\text{(IUB}(k)\text{)} \qquad f^{(k),\mathrm{U}} = \max_{\mathbf{x} \in X^{(k)}} \min_{\mathbf{y} \in Y^{(k)}} \{f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}\}.$$

Before providing a practical way of obtaining an upper bound on this challenging problem, we formulate and prove a useful property of the inner upper bounding scheme.

THEOREM 3.12. *The inner upper bounding scheme* (IUB(k)) *is not improving when branching on an inner* ($\mathbf{y}$) *variable takes place, i.e.,*

$$f^{(k),\mathrm{U}} \leq \min\{f^{(k_1),\mathrm{U}}, f^{(k_2),\mathrm{U}}\} \tag{3.8}$$

where $Y^{(k)} = Y^{(k_1)} \cup Y^{(k_2)}$ *and* $X^{(k)} = X^{(k_1)} = X^{(k_2)}$. *Equivalently in a more comprehensive form:*

$$\begin{aligned} &\max_{\mathbf{x} \in X^{(k)}} \min_{\mathbf{y} \in Y^{(k)}} \{f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}\} \\ &\qquad \leq \min_{j \in \{k_1,k_2\}} \left\{ \max_{\mathbf{x} \in X^{(k)}} \min_{\mathbf{y} \in Y^{(j)}} \{f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}\} \right\}. \end{aligned} \tag{3.9}$$

Proof. After branching at a node k on an inner variable ($\mathbf{y}$) we obtain two new child nodes $X^{(k)} \times Y^{(k_1)}$ and $X^{(k)} \times Y^{(k_2)}$ such that $Y^{(k)} = Y^{(k_1)} \cup Y^{(k_2)}$. Then, for each fixed $\bar{\mathbf{x}} \in X^{(k)}$ we have

$$\begin{aligned} &\min_{\mathbf{y} \in Y^{(k)}} \{f(\bar{\mathbf{x}},\mathbf{y}) \text{ s.t. } \mathbf{g}(\bar{\mathbf{x}},\mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\bar{\mathbf{x}},\mathbf{y}) = \mathbf{0}\} \\ &\qquad \leq \min_{\mathbf{y} \in Y^{(j)}} \{f(\bar{\mathbf{x}},\mathbf{y}) \text{ s.t. } \mathbf{g}(\bar{\mathbf{x}},\mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\bar{\mathbf{x}},\mathbf{y}) = \mathbf{0}\}, \quad \forall j \in \{k_1,k_2\}. \end{aligned} \tag{3.10}$$

Next, from Eq. (3.10) it follows that over all $\mathbf{x} \in X^{(k)}$ we have

$$\begin{aligned} &\max_{\mathbf{x} \in X^{(k)}} \min_{\mathbf{y} \in Y^{(k)}} \{f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}\} \\ &\qquad \leq \max_{\mathbf{x} \in X^{(k)}} \min_{\mathbf{y} \in Y^{(j)}} \{f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}\}, \quad \forall j \in \{k_1,k_2\}. \end{aligned} \tag{3.11}$$

Finally, from Eq. (3.11) it follows that

$$\begin{aligned} &\max_{\mathbf{x} \in X^{(k)}} \min_{\mathbf{y} \in Y^{(k)}} \{f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}\} \\ &\qquad \leq \min_{j \in \{k_1,k2\}} \left\{ \max_{\mathbf{x} \in X^{(k)}} \min_{\mathbf{y} \in Y^{(j)}} \{f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0}\} \right\}, \end{aligned} \tag{3.12}$$

and this concludes the proof. ∎

REMARK 3.2. *A graphical illustration of Equation (3.10) can be seen in Figure 3.4, where the bisection takes place at point $y = 0$. After branching, the newly obtained child nodes are: $X^{(2)} \times Y^{(2)} = [-1, 1] \times [-1, 0]$ and $X^{(3)} \times Y^{(3)} = [-1, 1] \times [0, 1]$. Selecting a fixed value here of $\bar{x} = 0.5$ we find that over node 3, Eq. (3.10) holds as an equality (both sides are equal to -0.08), whereas it is a strict inequality in node 2 ($-0.08 < 0.02$).*

COROLLARY 3.13. *After branching at node k on a $\mathbf{y}$-variable and creating two child nodes k_1 and k_2, solving (IUB(k)) for the child nodes is unnecessary.*

From Definition 2.7, Step 2.1 we know, that branching on a $\mathbf{y}$ variable modifies all sublists $\mathcal{L}_s^p$, $s \in \{1, 2, \ldots, s_p\}$, containing node k such that:

$$\mathcal{L}_s^p = (\mathcal{L}_s^p \setminus \{k\}) \cup \{k_1, k_2\},$$

i.e., we do not create new sublists. Then, we can use Theorem 3.12 to set

$$f^{(k_1),\mathrm{U}} = f^{(k_2),\mathrm{U}} = f^{(k),\mathrm{U}}.$$

Thus problem (IUB(k)) only needs to be solved at the root node and following branching on an $\mathbf{x}$ variable. This is a useful property because solving problem (IUB(k)) is as hard as solving the original bilevel problem [12]. When it becomes necessary to solve (IUB(k)), in order to overcome the difficulty of such a nonconvex max-min problem, we employ a cheaper bounding scheme proposed in the original approach [16, 17], based on a semi-infinite reformulation and KKT relaxation [29, 30]:

$$\begin{aligned} \bar{f}^{(k)} = \max_{\mathbf{x}_0,\mathbf{x},\mathbf{y},\boldsymbol{\mu}} \quad & \mathbf{x}_0 \\ \text{s.t.} \quad & \mathbf{x}_0 - f(\mathbf{x},\mathbf{y}) \le 0 \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{0},\ \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \nabla_{\mathbf{y}} f(\mathbf{x},\mathbf{y}) + \boldsymbol{\mu}^{\mathrm{T}} \nabla_{\mathbf{y}} \tilde{\mathbf{g}}(\mathbf{x},\mathbf{y}) + \boldsymbol{\lambda}^{\mathrm{T}} \nabla_{\mathbf{y}} \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \boldsymbol{\mu}^T \tilde{\mathbf{g}}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \mathbf{0} \le \boldsymbol{\mu} \le \boldsymbol{\mu}^{\mathrm{U}} \\ & (\mathbf{x},\mathbf{y}) \in X^{(k)} \times Y^{(k)}, \end{aligned} \tag{RIUB(k))}$$

where $\tilde{\mathbf{g}}$ is the concatenation of the inner inequalities $\mathbf{g}$ and the bound constraints $\mathbf{y}^{\mathrm{L}} \le \mathbf{y} \le \mathbf{y}^{\mathrm{U}}$, i.e.:

$$\tilde{g}_j(\mathbf{x},\mathbf{y}) = \begin{cases} \tilde{g}_j(\mathbf{x},\mathbf{y}) & j = 1, \ldots, r \\ y_{j-r}^{\mathrm{L}} - y_{j-r} & j = r+1, \ldots, r+m \\ y_{j-r-m} - y_{j-r-m}^{\mathrm{U}} & j = r+m+1, \ldots, r+2m. \end{cases}$$

REMARK 3.3. *Note that in this formulation, the host set $\mathbf{y} \in Y$ for the KKT conditions should not be restricted to $Y^{(k)}$ because this may generate spurious KKT points at the boundary of $Y^{(k)}$ [23].*

The use of the KKT necessary conditions requires the satisfaction of a constraint qualification for the inner problem for all values of $\mathbf{x}$ and the specification of upper bounds $\boldsymbol{\mu}^{\mathrm{U}}$ for the Lagrange multipliers. The importance of choosing bounds for the multipliers was discussed in [23].

It is easy to notice that the useful property of the (IUB(k)) bounding scheme does not hold for the (RIUB(k)) case because after branching on a $\mathbf{y}$ variable, we exclude part of suboptimal KKT points. This situation can easily be observed in Figure 3.4.

The relaxed inner upper bound over the root node ($k = 1$) gives $\bar{f}^{(1)} = 0.17$ at the suboptimal KKT point $(1, 1)$; solution over node 2 yields $\bar{f}^{(2)} = 0.0$ (the former suboptimal KKT point $(1, 1)$ does not belong to this node) and solution over node 3 gives $\bar{f}^{(3)} = 0.17$; thus, $\bar{f}^{(1)} > \min\{\bar{f}^{(2)}, \bar{f}^{(3)}\}$.

We note, however, that $\bar{f}^{(1)}$ remains a valid upper bound on the solution of the inner problem at both child nodes and can therefore be inherited by the child nodes. Heuristics can be used to trade-off the cost of solving (RIUB(k)) against the benefit of tighter inner upper bounds.

Finally, both inner bounding subproblems (ILB(k)) and (RIUB(k)) are nonconvex and must be solved globally, e.g., with [1, 20, 32].

3.3. Valid inner upper bound over Y. Having computed the relaxed inner upper bound (RIUB(k)) for a specific node k, we need to ensure we have an upper bound that is valid for the inner problem on the entire Y domain. When $Y^{(k)} \subset Y$, this entails considering other nodes in addition to node (k). We first discuss the generation of a valid bound based on the subdomain $\mathcal{X}_p$ corresponding to node k, and then present a tighter bounding strategy.

3.3.1. Best inner upper bound for list $\mathcal{L}^p$. We define the best inner upper bound $f^{\mathrm{UB},p}$ over each independent list $\mathcal{L}^p$. Within each sublist of $\mathcal{L}^p$ an entire Y partition is present for a given subdomain of $\mathcal{X}_p$. Therefore the minimum value of the inner upper bound within a sublist expresses the lowest inner upper bound with respect to Y. Furthermore, different sublists contain overlapping X. Taking this into consideration, we obtain the following definition of the best inner upper bound over $\mathcal{L}^p$.

DEFINITION 3.14 (Best inner upper bound for a list $\mathcal{L}^p$). *The best inner upper bound ($f^{\mathrm{UB},p}$) for the list $\mathcal{L}^p$ is the lowest value of the inner upper bound over the* $\mathbf{y}$ *variables but the largest over the* $\mathbf{x}$ *variables:*

$$\text{(BIUB}(p)) \qquad f^{\mathrm{UB},p} = \max\left\{ \min_{j \in \mathcal{L}_1^p} \left\{ \bar{f}^{(j)} \right\}, \ldots, \min_{j \in \mathcal{L}_{s_p}^p} \left\{ \bar{f}^{(j)} \right\} \right\}.$$

REMARK 3.4. *$f^{\mathrm{UB},p}$ is a valid inner upper bound for all nodes belonging to list $\mathcal{L}^p$ (corresponding to the X-partition, $\mathcal{X}_p$). It is used to check whether a full-fathoming of some nodes is possible (see subsection 2.4).*

COROLLARY 3.15. *If after branching at a node k on an* $\mathbf{x}$*-variable the independence condition (IC) is not satisfied, then $f^{\mathrm{UB},p}$ cannot improve.*

Proof. Before branching the best inner upper bound ($f_0^{\mathrm{UB},p}$) for a list $\mathcal{L}^p$ is

$$f_0^{\mathrm{UB},p} = \max\left\{ \min_{j \in \mathcal{L}_1^p} \left\{ \bar{f}^{(j)} \right\}, \ldots, \min_{j \in \mathcal{L}_{s_p}^p} \left\{ \bar{f}^{(j)} \right\} \right\}. \tag{3.13}$$

After bisecting on an $\mathbf{x}$-variable, two child nodes k_1 and k_2 are created such that $X^{(k)} = X^{(k_1)} \cup X^{(k_2)}, Y^{(k)} = Y^{(k_1)} = Y^{(k_2)}$. From Definition 2.7, Step 3.2, we know that after branching at a node k on an outer variable, we replace all sublists $\mathcal{L}_s^p : s = 1, \ldots, s_p : k \in \mathcal{L}_s^p$ containing node k by one or two new sublists $\mathcal{L}_{s_i}^p = (\mathcal{L}_s^p \setminus \{k\}) \cup \{k_i\}, i \in \{1, 2\}$. If the independence condition (IC) is not satisfied, no new independent list $\mathcal{L}^p$ is created. Moreover, from (RIUB(k)) it follows, that

$$\max\{\bar{f}^{(k_1)}, \bar{f}^{(k_2)}\} = \bar{f}^{(k)}. \tag{3.14}$$

Thus, three cases are possible:

$$\text{(3.15)}\quad \begin{cases} \bar{f}^{(k_1)} = \bar{f}^{(k)} \\ \bar{f}^{(k_2)} < \bar{f}^{(k)} \end{cases} \qquad \text{(3.16)}\quad \begin{cases} \bar{f}^{(k_1)} < \bar{f}^{(k)} \\ \bar{f}^{(k_2)} = \bar{f}^{(k)} \end{cases} \qquad \text{(3.17)}\quad \begin{cases} \bar{f}^{(k_1)} = \bar{f}^{(k)} \\ \bar{f}^{(k_2)} = \bar{f}^{(k)} \end{cases}$$

There is no possible improvement to $f^{\mathrm{UB},p}$ when Eq. (3.17) holds; without loss of generality, consider the case when Eq. (3.15) holds, noting that equivalent arguments can be made when Eq. (3.16) is satisfied. From Eqs. (3.14) and (3.15) it follows that

$$\min_{j\in\mathcal{L}^p_{s_i}:k_1\in\mathcal{L}^p_{s_i}} \left\{\bar{f}^{(j)}\right\} = \min_{j\in\mathcal{L}^p_s:k\in\mathcal{L}^p_s} \left\{\bar{f}^{(j)}\right\}, \tag{3.18}$$

$$\min_{j\in\mathcal{L}^p_{s_i}:k_2\in\mathcal{L}^p_{s_i}} \left\{\bar{f}^{(j)}\right\} \leq \min_{j\in\mathcal{L}^p_s:k\in\mathcal{L}^p_s} \left\{\bar{f}^{(j)}\right\}, \tag{3.19}$$

and

$$f^{\mathrm{UB},p}_{\mathrm{new}} = \max\left\{\min_{j\in\mathcal{L}^p_1}\left\{\bar{f}^{(j)}\right\}, \ldots, \min_{j\in\mathcal{L}^p_{s_p}}\left\{\bar{f}^{(j)}\right\}\right\}. \tag{3.20}$$

Note, that in Eq. (3.20) all sublists $\min_{j\in\mathcal{L}^p_s:k\in\mathcal{L}^p_{s_i}}\left\{\bar{f}^{(j)}\right\}$ containing node k were replaced by new sublists $\min_{j\in\mathcal{L}^p_{s_i}:k_1\in\mathcal{L}^p_{s_i}}\left\{\bar{f}^{(j)}\right\}$ and $\min_{j\in\mathcal{L}^p_{s_i}:k_2\in\mathcal{L}^p_{s_i}}\left\{\bar{f}^{(j)}\right\}$ containing child nodes k_1 and k_2. Therefore, from Eqs. (3.18)–(3.20) it follows that $f^{\mathrm{UB},p}_{\mathrm{new}} = f^{\mathrm{UB},p}_0$. ∎

This situation is illustrated in Example 3.2.

EXAMPLE 3.2. *Assume, that the current Branch-and-Sandwich tree consists of one independent list containing one sublist:* $\mathcal{L}^1 = \{\mathcal{L}^1_1\} = \{2, 3\}$ *with the inner upper bound values shown in Figure 3.5a. Thus, from* (BIUB(p))

$$f^{\mathrm{UB},p=1} = \max\{\min\{0.5, 1.0\}\} = \max\{0.5\} = 0.5.$$

Next, assume that node $k = 2$ *is selected and bisection on variable* x *takes place. We have one independent list containing two sublists:* $\mathcal{L}^1 = \{\mathcal{L}^1_1, \mathcal{L}^1_2\}$ *with the nodes* $\mathcal{L}^1_1 = \{3, 4\}, \mathcal{L}^1_2 = \{3, 5\}$ *(see Figure 3.5b). While a better (lower) inner upper bound is achieved for the node 4* ($\underline{f}^{(4)} = 0.0$)*, this does not improve the best inner upper bound using* (BIUB(p))*:*

$$f^{\mathrm{UB},p=1} = \max\{\min\{0.5, 0.0\}, \min\{0.5, 1.0\}\} = \max\{0.0, 0.5\} = 0.5.$$

Finally, assume that node $k = 3$ *is selected and bisected again on variable* x*. We obtain two new child nodes,* $k_1 = 6$ *and* $k_2 = 7$*. This time, the independence condition* (IC) *is satisfied, i.e., two independent lists containing one sublist each are created:* $\mathcal{L}^1 = \mathcal{L}^1_1 = \{4, 6\}$ *and* $\mathcal{L}^2 = \mathcal{L}^2_1 = \{5, 7\}$ *(see Figure 3.5c). As a result a tighter best inner upper bound over list* $\mathcal{L}^1$ *is obtained:*

$$f^{\mathrm{UB},p=1} = \max\{\min\{0.0, 0.5\}\} = 0.0,$$
$$f^{\mathrm{UB},p=2} = \max\{\min\{1.0, 0.5\}\} = 0.5.$$

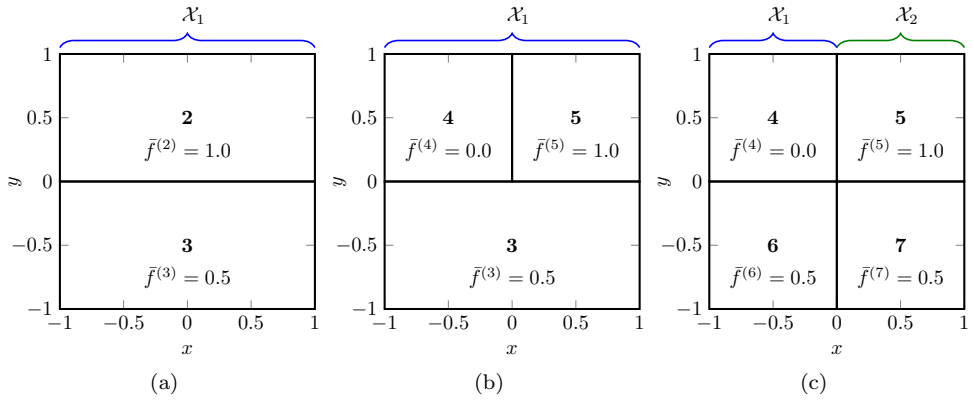


FIG. 3.5. *Illustration of a case where branching on a variable x takes place twice. While a better inner upper bound ($\underline{f}^{(4)} = 0.0$) is obtained in Figure 3.5b, the best inner upper bound (BIUB(p)) improves only when the independence condition (IC) is satisfied (Figure 3.5c).*

3.3.2. Best inner upper bound for a set of sublists. Note that, if a certain node $k \in \mathcal{L}^p$ does not belong to all sublists $\mathcal{L}^p_s \in \mathcal{L}^p : s \in \{1, \dots, s_p\}$ then a best inner upper bound that is valid for node k can be obtained by taking into account only those sublists that contain k:

DEFINITION 3.16 (Best inner upper bound for a node k).

$$\text{(BIUB}(k)) \qquad f^{\text{UB},k} = \max\left\{\min_{j \in \mathcal{L}^p_s}\left\{\bar{f}^{(j)}\right\}, s = 1, \dots, s_p : k \in \mathcal{L}^p_s\right\}.$$

This revised bound can be shown to be at least as tight as the best inner upper bound (BIUB(p)).

THEOREM 3.17. *The best inner upper bound ($f^{\text{UB},k}$) for the set of sublists in $\mathcal{L}^p$ that contain node k is at least as tight as the best inner upper bound ($f^{\text{UB},p}$) for the list $\mathcal{L}^p$ containing node k, i.e.,*

$$f^{\text{UB},k} \leq f^{\text{UB},p}. \tag{3.21}$$

Proof. The set of sublists to which a certain node k belongs is a subset of all sublist corresponding to the independent list $\mathcal{L}^p = \{\mathcal{L}^p_1, \dots, \mathcal{L}^p_{s_p}\}$ containing node k:

$$\{\mathcal{L}^p_s : s = 1, \dots, s_p : k \in \mathcal{L}^p_s\} \subseteq \mathcal{L}^p. \tag{3.22}$$

From Eq. (3.22) it follows that the set of minimal inner upper bound values for each of sublists containing node k is also a subset of the set containing minimal values for all sublists corresponding to the current independent list:

$$\left\{\min_{j \in \mathcal{L}^p_s}\left\{\bar{f}^{(j)}\right\}, s = 1, \dots, s_p : k \in \mathcal{L}^p_s\right\} \subseteq \left\{\min_{j \in \mathcal{L}^p_s}\left\{\bar{f}^{(j)}\right\}, s = 1, \dots, s_p\right\}. \tag{3.23}$$

Thus, from Equations (3.22) and (3.23) it follows that

$$\max\left\{\min_{j \in \mathcal{L}^p_s}\left\{\bar{f}^{(j)}\right\}, s = 1, \dots, s_p : k \in \mathcal{L}^p_s\right\} \leq \max\left\{\min_{j \in \mathcal{L}^p_s}\left\{\bar{f}^{(j)}\right\}, s = 1, \dots, s_p\right\},$$

and this concludes the proof. ∎

EXAMPLE 3.3. *This approach is illustrated in Figure 3.6 to show the advantage of deriving a best inner upper bound based on only those sublists that contain node k. In the case shown in Figure 3.6, there is one independent list containing two sublists: $\mathcal{L}^1 = \{\mathcal{L}_1^1, \mathcal{L}_2^1\}$ with $\mathcal{L}_1^1 = \{3, 4\}$ and $\mathcal{L}_2^1 = \{3, 5\}$. Using the original (BIUB(p)) approach, the best inner upper bound, for list $\mathcal{L}^p$ is*

$$f^{\mathrm{UB},p=1} = \max\{\min\{0.5, 0.0\}, \min\{0.5, 1.0\}\} = \max\{0.0, 0.5\} = 0.5,$$

and this is valid for all nodes. Using the alternative approach (BIUB(k)), the best inner upper bound for nodes 3 and 5 is the same as $f^{\mathrm{UB},p=1}$: $f^{\mathrm{UB},3} = f^{\mathrm{UB},5} = \min\{0.5, 1.0\} = 0.5$, but for node $k = 4$, a lower (tighter) value is obtained: $f^{\mathrm{UB},4} = \min\{0.5, 0.0\} = 0.0$.

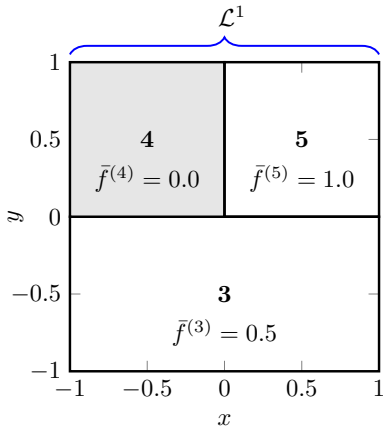


FIG. 3.6. *Illustration of a case in which the best inner upper bound for node k based on a set of sublists (BIUB(k)) yields a tighter bound compared to the original approach in which the best inner upper bound (BIUB(p)) is derived based on $\mathcal{L}^p$.*

3.4. Lower bounding scheme. The best inner upper bound serves as a constant bound cut in the outer lower bounding problem (LB(k)). For $k \in \mathcal{L} \cap \mathcal{L}^p$ the lower bounding problem is:

$$\begin{aligned}
\underline{F}^{(k)} = \min_{\mathbf{x},\mathbf{y},\boldsymbol{\mu}} \quad & F(\mathbf{x},\mathbf{y}) \\
\text{s.t.} \quad & \mathbf{G}(\mathbf{x},\mathbf{y}) \leq \mathbf{0},\ \mathbf{H}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\
& \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0},\ \ \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\
& f(\mathbf{x},\mathbf{y}) \leq f^{\mathrm{UB},k} \\
& \nabla_{\mathbf{y}} f(\mathbf{x},\mathbf{y}) + \boldsymbol{\mu}^{\mathrm{T}} \nabla_{\mathbf{y}} \tilde{\mathbf{g}}(\mathbf{x},\mathbf{y}) + \boldsymbol{\lambda}^{\mathrm{T}} \nabla_{\mathbf{y}} \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\
& \boldsymbol{\mu}^{T} \tilde{\mathbf{g}}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\
& \mathbf{0} \leq \boldsymbol{\mu} \leq \boldsymbol{\mu}^{\mathrm{U}} \\
& (\mathbf{x},\mathbf{y}) \in X^{(k)} \times Y^{(k)},
\end{aligned} \tag{LB(k)}$$

where $\tilde{\mathbf{g}}$ is defined as in (RIUB(k)) and the newly introduced $f^{\mathrm{UB},k}$, derived from the solution of (BIUB(k)), is used to bound the inner objective function for node k.

REMARK 3.5. *Note that in the third constraint of (LB(k)), i.e., in the inequality $f(\mathbf{x},\mathbf{y}) \leq f^{\mathrm{UB},k}$ the best inner upper bound based on a set of sublists $f^{\mathrm{UB},k}$ can be replaced with the best inner upper bound for $\mathcal{L}^p$, $f^{\mathrm{UB},p}$.*

3.5. Upper bounding scheme. To define the outer upper bounding scheme, we start from the description of the original upper bounding scheme, discussing observed drawbacks and suggesting possible ways to overcome them.

3.5.1. Upper bounding scheme based on the original approach. Given $k \in \mathcal{L} \cap \mathcal{L}^p$ and $(\bar{\mathbf{x}}^{(k)}, \bar{\mathbf{y}}^{(k)})$, the solution of the lower bounding problem (LB(k)), $\bar{\mathbf{x}}$ is set equal to $\bar{\mathbf{x}}^{(k)}$, and a node $k' \in (\mathcal{L} \cup \mathcal{L}_{\text{In}}) \cap \mathcal{L}^p$ is sought such that:

$$\text{(MinRISP}(\bar{\mathbf{x}}, k')) \qquad k' = \arg\min_i \left\{ \underline{F}^{(i)} : i = \arg\min_{j \in \mathcal{L}_s^p, s=1,\ldots,s_p : \bar{\mathbf{x}} \in \mathcal{L}_s^p} \left\{ \underline{w}^{(j)}(\bar{\mathbf{x}}) \right\} \right\}.$$

with $\underline{w}^{(j)}(\bar{\mathbf{x}})$ given by:

$$\text{(RISP}(\bar{\mathbf{x}}, j)) \qquad \underline{w}^{(j)}(\bar{\mathbf{x}}) = \min_{\mathbf{y} \in Y^{(j)}} \{ \breve{f}(\bar{\mathbf{x}}, \mathbf{y}) \ \text{ s.t. } \ \breve{\mathbf{g}}(\bar{\mathbf{x}}, \mathbf{y}) \leq \mathbf{0}, \ \breve{\mathbf{h}}(\bar{\mathbf{x}}, \mathbf{y}) = \mathbf{0} \}.$$

REMARK 3.6. *To find the solution of* (MinRISP($\bar{\mathbf{x}}, k'$)) *it suffices to solve subproblem* (RISP($\bar{\mathbf{x}}, j$)) *for each node in each sublist containing node k, i.e., over all $j \in \mathcal{L}_s^p, s \in \{1, \ldots, s_p\}$ such that $\bar{\mathbf{x}} \in X^{(j)}$.*

REMARK 3.7. *When the solution of* (RISP($\bar{\mathbf{x}}, j$)) *over all j yields several nodes with the same lowest value we give a preference to the node with the lowest outer lower bound ($\underline{F}^{(k)}$): This approach is consistent with the requirement in optimistic bilevel programming that the leader consider the best candidate among indifferent solutions for the follower. Also, under this assumption, the algorithmic behavior is not affected by which global minimum is obtained during the solution of the inner problem.*

For a given $\bar{\mathbf{x}}$, problem (RISP($\bar{\mathbf{x}}, j$)) may be infeasible over some or even over all nodes from a list $\mathcal{L}^p$. In the latter case, no solution to the bilevel problem exists and no upper bound can be obtained for the point $\bar{\mathbf{x}}$. If a feasible and active node k' exists and ε_f is the inner objective tolerance, the following outer upper bounding problem is solved for $\mathbf{x} = \bar{\mathbf{x}}$:

$$\text{(UB}(\bar{\mathbf{x}}, k')) \qquad \begin{aligned} \bar{F}^{(k')}(\bar{\mathbf{x}}) = \min_{\mathbf{y} \in Y^{(k')}} \ & F(\bar{\mathbf{x}}, \mathbf{y}), \\ \text{s.t.} \ & \mathbf{G}(\bar{\mathbf{x}}, \mathbf{y}) \leq \mathbf{0}, \ \mathbf{H}(\bar{\mathbf{x}}, \mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\bar{\mathbf{x}}, \mathbf{y}) \leq \mathbf{0}, \ \ \mathbf{h}(\bar{\mathbf{x}}, \mathbf{y}) = \mathbf{0} \\ & f(\bar{\mathbf{x}}, \mathbf{y}) \leq \underline{w}^{(k')}(\bar{\mathbf{x}}) + \varepsilon_f. \end{aligned}$$

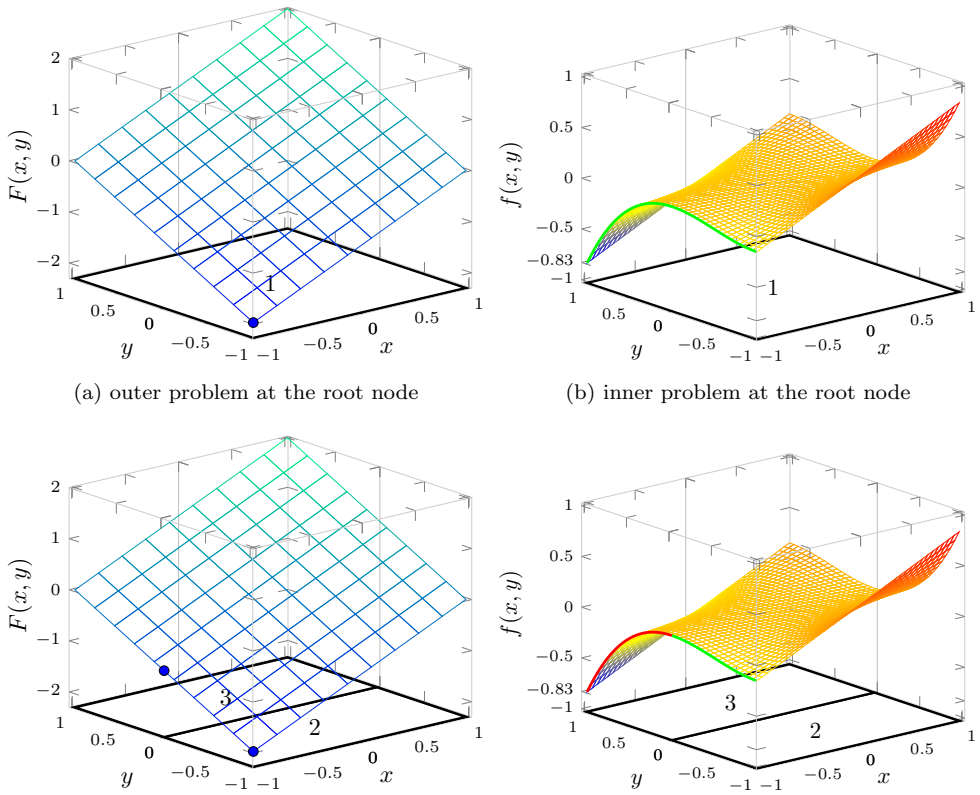
If problem (UB($\bar{\mathbf{x}}, k'$)) is infeasible, then no solution exists for $\bar{\mathbf{x}}$; otherwise, an upper bound is obtained in the sense of an ε_f-feasible point [23]. Alternatively, the point $(\bar{\mathbf{x}}^{(k')}, \bar{\mathbf{y}}^{(k')})$ corresponding to the solution k' of (MinRISP($\bar{\mathbf{x}}, k'$)) can be used to obtain a valid upper bound without solving (UB($\bar{\mathbf{x}}, k'$)) at all. Similarly, a valid upper bound can be found through any feasible point of problem (UB($\bar{\mathbf{x}}, k'$)), e.g., a point obtained from a local solution.

REMARK 3.8. *When node k' is inner-active, i.e., it has been outer-fathomed, we already know that the global solution of the bilevel problem cannot lie in this subdomain $X^{(k')} \times Y^{(k')}$ and we can skip the solution of* (UB($\bar{\mathbf{x}}, k'$)).

REMARK 3.9. *While we would like to avoid repeating unnecessary calculations, the solution of* (RISP($\bar{\mathbf{x}}, j$)) *can only be avoided if another* (RISP($i, \bar{\mathbf{x}}'$)) *has already been solved with $\bar{\mathbf{x}}' = \bar{\mathbf{x}}$ and $Y^{(i)} = Y^{(j)}$. These conditions are restrictive.*

EXAMPLE 3.4. *For the case shown in Figures 3.7a and 3.7b, we have one list containing one sublist at the root node:* $\mathcal{L}^1 = \{\mathcal{L}_1^1\} = \{1\}$. *Solving lower bounding problem* (LB(k)) *at the root node gives solution point* $(\bar{x}^{(1)}, \bar{y}^{(1)}) = (-1.0, -1.0)$. *Next, we set* $\bar{x} = \bar{x}^{(1)} = -1.0$ *and obtain* $\underline{w}^{(1)}(-1.0) = -1.50$, *taking into consideration the whole of* Y *(over the green line in Figure 3.7b). The value found is used in* (UB($\bar{\mathbf{x}}, k'$)).

Next, after branching on variable y, *the case shown in Figures 3.7c and 3.7d is obtained, i.e., there is one list containing one sublist:* $\mathcal{L}^1 = \{\mathcal{L}_1^1\} = \{2, 3\}$. *After solving both outer lower bounding problems* (LB(k)) *the resulting solution points are:* $(\bar{x}^{(2)}, \bar{y}^{(2)}) = (-1.0, -1.0)$ *and* $(\bar{x}^{(3)}, \bar{y}^{(3)}) = (-1.0, 0.0)$. *Next, we set* $\bar{x} = \bar{x}^{(2)} = -1.0$. *Despite the fact that* $\bar{x} = \bar{x}^{(2)} = -1.0$ *is the same point as in the previous iteration it must explored again in the new* (RISP($\bar{\mathbf{x}}, j$)) *subproblem as different domains of* Y *are considered. Different values are obtained:* $\underline{w}^{(2)}(-1.0) = -0.23$ *and* $\underline{w}^{(3)}(-1.0) = -0.83$ *and two* (UB($\bar{\mathbf{x}}, k'$)) *subproblems must then be solved. Finally, since* $\bar{x} = \bar{x}^{(3)} = -1.0$, *note that we have already computed* $\underline{w}^{(2)}(-1.0)$ *and* $\underline{w}^{(3)}(-1.0)$; *hence, no further computation is necessary.*



(a) outer problem at the root node

(b) inner problem at the root node

(c) outer problem(s) after the first iteration

(d) inner problem(s) after the first iteration

FIG. 3.7. *Graphical illustration of the outer and inner problems together with found solution points from* (LB(k)) *problems (blue dots) and* Y *subdomains over which* (RISP($\bar{\mathbf{x}}, j$)) *is solved (green and red lines).*

Finally use of convex relaxations in problem (RISP($\bar{\mathbf{x}}, j$)) can cause slow convergence, described in Remark 3.10.

REMARK 3.10. *The value obtained from the solution of problem* (RISP($\bar{\mathbf{x}}, j$)) *plays a significant role in the* (UB($\bar{\mathbf{x}}, k'$)) *procedure, specifically due to the presence of the inner objective constraint* $f(\bar{\mathbf{x}}, \mathbf{y}) \leq \underline{w}^{(k')}(\bar{\mathbf{x}}) + \varepsilon_f$. *The lower the value of* $\underline{w}^{(k')}(\bar{\mathbf{x}})$ *obtained from* (RISP($\bar{\mathbf{x}}, j$)), *the more likely it is that the identification of a bilevel feasible point is delayed. This situation can cause slow outer-fathoming, and thus cause the Branch-and-Sandwich tree to expand. This is especially a concern in the early stages of the algorithm, when the convex relaxations in* (RISP($\bar{\mathbf{x}}, j$)) *can be very loose due to the size of the domain. A natural way to overcome this is to replace subproblem* (RISP($\bar{\mathbf{x}}, j$)) *with the computationally more expensive but tighter approach:*

$$\text{(ISP}(\bar{\mathbf{x}}, j)) \qquad w^{(j)}(\bar{\mathbf{x}}) = \min_{\mathbf{y} \in Y^{(j)}} \{f(\bar{\mathbf{x}}, \mathbf{y}) \ \text{ s.t. } \ \mathbf{g}(\bar{\mathbf{x}}, \mathbf{y}) \leq \mathbf{0}, \ \mathbf{h}(\bar{\mathbf{x}}, \mathbf{y}) = \mathbf{0}\},$$

and

$$\text{(MinISP}(\bar{\mathbf{x}}, k')) \qquad k' = \arg\min_i \left\{ \underline{F}^{(i)} : i = \underset{j \in \mathcal{L}_s^p, s=1,\ldots,s_p : \bar{\mathbf{x}} \in \mathcal{L}_s^p}{\arg\min} \left\{ w^{(j)}(\bar{\mathbf{x}}) \right\} \right\}.$$

Using tighter $w^{(j)}(\bar{\mathbf{x}})$ *values, the B&S algorithm is likely to locate bilevel feasible points faster. This can help to improve the best known upper bound value* (F^{UB}). *This can, in turn, lead to earlier fathoming of nodes and thus help to keep the Branch-and-Sandwich tree smaller and to reduce the total number of subproblems solved.*

3.5.2. An alternative upper bounding scheme. Let us observe that the original upper bounding procedure can be computationally demanding, especially during the formulation of (MinRISP($\bar{\mathbf{x}}, k'$)) which requires the solution of (RISP($\bar{\mathbf{x}}, j$)) over all nodes $j \in \mathcal{L}^p$. Therefore the original upper bounding procedure becomes more and more expensive as the size of independent lists increases. Thus motivated by [23], we propose an alternative approach to obtain a valid upper bound that avoids this overhead. First, we replace (RISP($\bar{\mathbf{x}}, j$)) with (ISP($\bar{\mathbf{x}}, j$)), taking into account Remark 3.10. Next, we fix $\bar{\mathbf{x}}$ and solve one (ISP($\mathbf{x}$)) subproblem over the whole of Y, i.e.,

$$\text{(ISP}(\bar{\mathbf{x}}, Y)) \qquad w(\bar{\mathbf{x}}) = \min_{\mathbf{y} \in Y} \{f(\bar{\mathbf{x}}, \mathbf{y}) \ \text{ s.t. } \ \mathbf{g}(\bar{\mathbf{x}}, \mathbf{y}) \leq \mathbf{0}, \ \mathbf{h}(\bar{\mathbf{x}}, \mathbf{y}) = \mathbf{0}\}.$$

For comparison, in the bounding scheme based on the original approach, we have to solve (ISP($\bar{\mathbf{x}}, j$)) over all nodes in a sublist $j \in \mathcal{L}_s^p, s = 1, \ldots, s_p : \bar{\mathbf{x}} \in \mathcal{L}_s^p$ and select the node k' with the smallest $w^{(j)}(\bar{\mathbf{x}})$ value.

Next we modify the upper bounding problem by using $w(\bar{\mathbf{x}})$ and solving the problem over the whole of Y:

$$\text{(UB}(\bar{\mathbf{x}}, Y)) \qquad \begin{aligned} \bar{F}(\bar{\mathbf{x}}) = \ & \min_{\mathbf{y} \in Y} && F(\bar{\mathbf{x}}, \mathbf{y}), \\ & \text{s.t.} && \mathbf{G}(\bar{\mathbf{x}}, \mathbf{y}) \leq \mathbf{0}, \ \mathbf{H}(\bar{\mathbf{x}}, \mathbf{y}) = \mathbf{0} \\ & && \mathbf{g}(\bar{\mathbf{x}}, \mathbf{y}) \leq \mathbf{0}, \ \ \mathbf{h}(\bar{\mathbf{x}}, \mathbf{y}) = \mathbf{0} \\ & && f(\bar{\mathbf{x}}, \mathbf{y}) \leq w(\bar{\mathbf{x}}) + \varepsilon_f. \end{aligned}$$

This alternative approach of solving only one (ISP($\bar{\mathbf{x}}, Y$)) and one (UB($\bar{\mathbf{x}}, Y$)) for each unique $\bar{\mathbf{x}}$, can have at least three significant benefits.

REMARK 3.11. *The alternative bounding scheme significantly reduces the total number of optimal value functions* ($w(\bar{\mathbf{x}})$) *that must be calculated during the course of the algorithm.*

REMARK 3.12. *When solving* (UB($\bar{\mathbf{x}}, Y$)) *globally over the whole of* Y*, the behavior of the B&S algorithm is not affected by which global minimum is obtained in* (ISP($\bar{\mathbf{x}}, Y$))*. While equality holds for the bounds obtained using* (ISP($\bar{\mathbf{x}}, Y$)) *and* (MinISP($\bar{\mathbf{x}}, k'$))*, i.e.,*

$$w(\bar{\mathbf{x}}) = w^{(k')}(\bar{\mathbf{x}}), \tag{3.24}$$

the upper bound obtained with (UB($\bar{\mathbf{x}}, Y$)) *is at least as tight as the one obtained with* (UB($\bar{\mathbf{x}}, k'$))*, i.e.,*

$$\bar{F}(\bar{\mathbf{x}}) \leq \bar{F}^{(k')}(\bar{\mathbf{x}}). \tag{3.25}$$

This is due to the fact that for (UB($\bar{\mathbf{x}}, Y$)) *is solved over the whole of* Y *while* (UB($\bar{\mathbf{x}}, k'$)) *is solved over a subset of* Y*.*

REMARK 3.13. *Following the identification of a value* $\bar{\mathbf{x}}$ *as the solution of problem* (LB(k))*, it is trivial to determine whether* (ISP($\bar{\mathbf{x}}, Y$)) *has been solved at a previous iteration. Since every* (ISP($\bar{\mathbf{x}}, Y$)) *problem is solved over the whole of* Y*, it suffices to check whether the same value* $\bar{\mathbf{x}}$ *has been generated previously. For this, we maintain the set*

$$\mathbb{X} = \{\bar{\mathbf{x}}_i \in X : \forall i \neq j \in \{1, \ldots, |\mathbb{X}|\},\ \bar{\mathbf{x}}_i \neq \bar{\mathbf{x}}_j\}, \tag{3.26}$$

corresponding to unique solution points of outer-variables ($\bar{\mathbf{x}}$)*. This procedure is much simpler than identifying duplicate problems in the bounding scheme based on the original approach (see Remark 3.9).*

4. Summary of the revised and the original B&S. In this section, in Table 4.2, we provide a comparative summary of the options available in the two versions of the B&S algorithm (Algorithm 2.1): the revised B&S version, presented in this paper and the original B&S [16]. The algorithmic options used in section 5 are highlighted in bold font. Changes have been made on the basis of theoretical arguments to Steps 9, 11, 12 and to the node fathoming rules used in Steps 2, 7 and 9. Furthermore, additional heuristics have been introduced in Steps 4 and 5.

TABLE 4.2
Comparison of the available options for the mains steps in the revised and original B&S

Algorithmic step (option)	**Revised B&S**	**Original B&S**
Algorithm 2.1, Step 4: Node Selection	**($\underline{F}l$)-($l\underline{f}$)**, ($\underline{F}l$)-($l\bar{f}$), ($l\underline{F}$)-($l\underline{f}$), ($l\underline{F}$)-($l\bar{f}$)	**($\underline{F}l$)-($l\underline{f}$)**
Algorithm 2.1, Step 5: Branching rule	(XY), **(YX)**	**(YX)**†
Algorithm 2.1, Step 7: ILB scheme	$\boldsymbol{f^{(k),\mathrm{L}}}$ (ILB(k)), $\check{f}^{(k),\mathrm{L}}$ (RILB(k))	$f^{(k),\mathrm{L}}$ (ILB(k)), $\boldsymbol{\check{f}^{(k),\mathrm{L}}}$ (RILB(k))
Algorithm 2.1, Step 8: IUB scheme	$f^{(k),\mathrm{U}}$ (IUB(k)) $\boldsymbol{\bar{f}^{(k)}}$ (RIUB(k))	$f^{(k),\mathrm{U}}$ (IUB(k)) $\boldsymbol{\bar{f}^{(k)}}$ (RIUB(k))
Algorithm 2.1, Step 9: BIUB scheme	$f^{\mathrm{UB},p}$ (BIUB(p)), $\boldsymbol{f^{\mathrm{UB},k}}$ (BIUB(k))	$\boldsymbol{f^{\mathrm{UB},p}}$ (BIUB(p))
Algorithm 2.1, Step 10: LB scheme	$\boldsymbol{\underline{F}^{(k)}}$ (LB(k))	$\boldsymbol{\underline{F}^{(k)}}$ (LB(k))
Algorithm 2.1, Step 11: ISP scheme	$\underline{w}^{(j)}(\bar{\mathbf{x}})$ (RISP($\bar{\mathbf{x}}, j$)) $w^{(j)}(\bar{\mathbf{x}})$ (ISP($\bar{\mathbf{x}}, j$)) $\boldsymbol{w}(\bar{\mathbf{x}})$ (ISP($\bar{\mathbf{x}}, Y$))	$\boldsymbol{\underline{w}^{(j)}}(\bar{\mathbf{x}})$ (RISP($\bar{\mathbf{x}}, j$))
Algorithm 2.1, Step 12: UB scheme	$\bar{F}^{(k')}(\bar{\mathbf{x}})$ (UB($\bar{\mathbf{x}}, k'$)), $\boldsymbol{\bar{F}}(\bar{\mathbf{x}})$ (UB($\bar{\mathbf{x}}, Y$)),	$\boldsymbol{\bar{F}^{(k')}}(\bar{\mathbf{x}})$ (UB($\bar{\mathbf{x}}, k'$))
Node fathoming rule	Extended full-fathoming, Outer-fathoming	Full-fathoming, Outer-fathoming

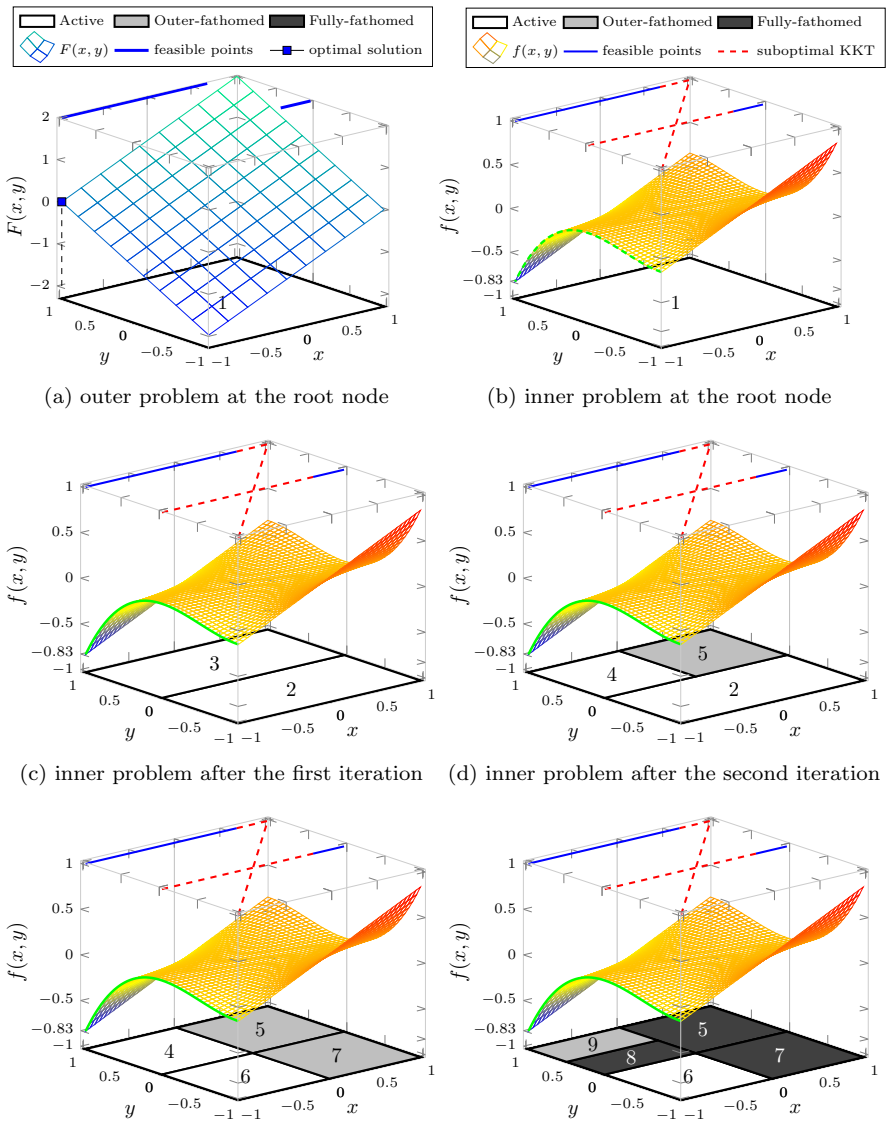
† The branching rule described in [16] does not correspond directly to (XY) or (YX) but the examples in that publication are based on a rule similar to (YX). So this is used here.

5. Algorithmic comparison of the revised B&S versus the original B&S. In this section the main emphasis is on illustrating the influence of the different bounding procedures on the B&S algorithm. The impact of using different branching and node selection strategies is investigated in the second part of this manuscript. To provide as fair a comparison as possible we use the (YX) branching and ($\underline{F}l$)-($l\underline{f}$) node selection strategies, as they are closest to those used in the original paper [16]. Moreover, we concentrate on the bilevel instance used in Example 3.1 of our current work and Example 1 in [17], where the behavior of the B&S algorithm at each step was described and explained in detail:

$$\begin{aligned} &\min_{x,y} x + y \\ &\text{s.t. } y \in \arg\min_{y} \frac{xy^2}{2} - \frac{y^3}{3} \\ &\qquad x \in [-1, 1], y \in [-1, 1]. \end{aligned}$$

The inner problem satisfies the Adabie constraint qualification [4, 22]. Stationarity of the inner objective gives $xy - y^2 = 0$ and therefore $y = 1, y = 0$ and $y = x$ are KKT points of the inner problem. Further, $y = 1$ is optimal for $-1 \leq x \leq \frac{2}{3}$ and $y = 0$ is optimal for $\frac{2}{3} \leq x \leq 1$. Thus, the problem has a unique optimal solution at $(x^*, y^*) = (-1, 1)$ yielding $F^* = 0$ and $f^* = -0.83$ (see the visualization of the outer and the inner problems in Figures 5.8a and 5.8b). All bound values obtained using

the B&S algorithm with the revised and original bounding schemes are summarized in Tables 5.4a and 5.4b, respectively.



(a) outer problem at the root node

(b) inner problem at the root node

(c) inner problem after the first iteration

(d) inner problem after the second iteration

(e) inner problem after the third iteration

(f) inner problem after the fourth iteration

FIG. 5.8. *Graphical illustration of the outer (over the root node ($k = 1$)) an inner problems (over all nodes) together with bilevel feasible points (thick (blue) lines) and suboptimal KKT points (dashed (red) lines). The light-gray nodes denote nodes that are first outer-fathomed, i.e., moved to the list $\mathcal{L}_{\text{In}}$, while dark-gray nodes denote those that are fully-fathomed due to full-fathoming rules.*

From the values obtained at the *root node* ($k = 1$), we observe that the (ILB(k)) bounding scheme gives a tighter inner lower bound value than (RILB(k)), i.e., $f^{(1),\text{L}} = -0.83 > \underline{f}^{(1)} = -2.50$. The same applies for values obtained using the (ISP($\bar{\mathbf{x}}, Y$)) and (RISP($\bar{\mathbf{x}}, j$)) schemes, i.e., $w(-1.0) = -0.83 > \underline{w}(-1.0) = -1.50$. A further significant difference is how the uniqueness of the solution points ($\bar{x}$) obtained from the

TABLE 5.3
Detailed comparison of the B&S algorithm on `mb_1_1_08` *bilevel problem*

k	$f^{(k),\mathrm{L}}$	$\bar{f}^{(k)}$	$f^{\mathrm{UB},k}$	$\underline{F}^{(k)}$	$(\bar{x},\bar{y})$	$w(\bar{x})$	$\bar{F}(\bar{x})$
1	-0.83	0.17	0.17	-2.0	$(-1.0,-1.0)$	-0.83	0.0
2	-0.17	0.0	0.0	-2.0	$(-1.0,-1.0)$	–	–
3	-0.83	0.17	0.0	-1.0	$(-1.0,0.0)$	–	–
4	-0.83	0.0	0.0	-1.0	$(-1.0,0.0)$	–	–
5	-0.33	0.17	0.0	0.0	$(0.0,0.0)$	-0.33	1.0
6	-0.17	0.0	0.0	-2.0	$(-1.0,-1.0)$	–	–
7	0.0	0.0	0.0	0.0	$(0.0,0.0)$	–	–
8	-0.17	0.0	–	–	–	–	–
9	-0.83	-0.33	-0.33	0.0	$(-1.0,0.5)$	–	–
#Total	9	9	–	8	–	2	2

(a) Using revised bounding schemes

k	$\breve{f}^{(k),\mathrm{L}}$	$\bar{f}^{(k)}$	$f^{\mathrm{UB},p}$	$\underline{F}^{(k)}$	$(\bar{x},\bar{y})$	$\underline{w}^{(j)}(\bar{x})$	$\bar{F}^{(k')}(\bar{x})$
1	-2.50	0.17	0.17	-2.0	$(-1.0,-1.0)$	-1.50	∞
2	-0.6	0.0	0.0	-2.0	$(-1.0,-1.0)$	$-0.24;-0.83$	0.0
3	-0.96	0.17	0.0	-1.0	$(-1.0,0.0)$	–	–
4	-0.83	0.0	0.0	-1.0	$(-1.0,0.0)$	-0.83	0.0
5	-0.49	0.17	0.0	0.0	$(0.0,0.0)$	–	–
6	-0.41	0.0	0.0	-2.0	$(-1.0,-1.0)$	-0.24	0.0
7	-0.20	0.0	0.0	0.0	$(0.0,0.0)$	–	–
8	-0.17	0.0	–	–	–	–	–
9	-0.83	-0.33	-0.33	0.0	$(-1.0,0.5)$	–	–
10	-0.20	–	–	–	–	–	–
11	-0.11	–	–	–	–	–	–
12	-0.83	–	–	–	–	–	–
13	-0.58	–	–	–	–	–	–
#Total	13	9	–	8	–	5	4

(b) Using original bounding schemes

solution $(\bar{x}^{(k)},\bar{y}^{(k)})$ of (LB(k)) is identified. Using the revised approach, we maintain a set $\mathbb{X}$ (defined in Eq. (3.26)) consisting of unique points $\bar{x}$. In the revised B&S we thus add $\bar{x}^{(1)}=-1.0$ to the set $\mathbb{X}=\{-1.0\}$. Finally, observe that using the revised approach with the exact $w(-1.0)=-0.83$ value makes it possible to find the first

bilevel-feasible solution $(\bar{x}^{(1)}, \bar{y}^{(1)}) = (-1.0, -1.0)$ with $F^{\mathrm{UB}} = 0.0$. When using the original B&S and solving the outer upper bounding problem with $\underline{w}(-1.0) = -0.83$ no feasible solution is found at this step.

Next, at the *first iteration*, both B&S versions select node $k = 1$ (based on Definition 2.8) and bisect on the selected branching variable y (based on Definition 2.5), generating two new nodes ($k = 2$ and $k = 3$), as shown in Figure 5.8c. After branching, both Branch-and-Sandwich trees consist of one list with one sublist: $\mathcal{L}^1 = \{\mathcal{L}^1_1\} = \{2, 3\}$. At this iteration, the two inner lower bounding approaches generate different lower bounds once more. None of the bounds derived allow the fathoming of a subregion. A significant difference in the algorithmic behavior occurs in setting up the outer upper bounding problems with the solutions of the two corresponding outer outer lower bounding problems, $(\bar{x}^{(2)}, \bar{y}^{(2)}) = (-1.0, -1.0)$ and $(\bar{x}^{(3)}, \bar{y}^{(3)}) = (-1.0, 0.0)$. Using the revised upper bounding scheme, we can safely skip the upper bounding procedure for both nodes, as $\bar{x} = \bar{x}^{(2)} = \bar{x}^{(3)} = -1.0 \in \mathbb{X}$. Using the original upper bounding scheme, B&S first solves two (RISP($\bar{\mathbf{x}}, j$)) problems and then one (UB($\bar{\mathbf{x}}, k'$)) problem, which yields the first bilevel-feasible solution $\bar{F}^3(-1.0) = 0.0$; this solution was located at the root node using the revised approach.

At the *second iteration*, in both versions of B&S node $k = 3$ is selected and bisected on variable x, resulting in two new nodes ($k = 4$ and $k = 5$), as shown in Figure 5.8d. The two Branch-and-Sandwich trees then consist of one list with two sublists: $\mathcal{L}^1 = \{\mathcal{L}^1_1, \mathcal{L}^1_2\} = \{\{2, 4\}, \{2, 5\}\}$. The biggest difference in the algorithmic behaviors is observed when the solution points from the lower bounding procedure are obtained: $(\bar{x}^{(4)}, \bar{y}^{(4)}) = (-1.0, 0.0)$ and $(\bar{x}^{(5)}, \bar{y}^{(5)}) = (0.0, 0.0)$. First, the original B&S approach fails to recognize that $\bar{x}^{(4)} = -1.0$ is the same as $\bar{x}^{(3)} = -1.0$ and has thus already been explored at the previous iteration. This leads to the redundant evaluation of $\underline{w}^{(4)}(\bar{x}) = -0.83$. A further issue arises with the second solution point $\bar{x}^{(5)} = 0.0$. First, we observe that this is a new solution point from the perspective of the outer problem. However, before starting the upper bounding procedure, B&S checks the outer-fathoming rule (see Definition 2.10) for node $k = 5$, leading to it being moved from the list of open nodes to $\mathcal{L}_{\mathrm{In}}$. As a result, the upper bounding procedure is bypassed completely and this can delay the identification of an improved upper bound, therefore slowing down convergence. At this step we already know that the bilevel solution does not lie at node $k = 5$, as it was outer-fathomed, but (MinRISP($\bar{\mathbf{x}}, k'$)) requires the selection of node k' to be based on *all* the nodes in the list containing $\bar{x}^{(5)}$, i.e., $\mathcal{L}^1_2 = \{1, 2\}$. Therefore, using the original approach one should solve $\underline{w}^{(2)}(0.0)$ and $\underline{w}^{(5)}(0.0)$. Then, if $k' = 2$, evaluate $\bar{F}^{(2)}(0.0)$; otherwise, i.e., if $k' = 5$ it is safe to skip $\bar{F}^{(5)}(0.0)$ as we already know that this node is outer-fathomed and cannot contain a better solution.

Using revised upper bounding scheme, such a situation cannot arise, as the whole of Y is considered for each unique solution of the outer upper bounding problem. Therefore, we add the new solution point $\bar{x} = 0.0$ to the set of unique (LB(k)) solutions: $\mathbb{X} = \mathbb{X} \cup \{0.0\} = \{0.0, 1.0\}$ and perform the revised outer upper bounding procedure for this solution point.

At the *third iteration*, both versions of B&S select node $k = 2$ and bisect on variable x generating two new nodes ($k = 4$ and $k = 5$), as shown in Figure 5.8e. After this branching, the independence condition (IC) is satisfied and the two Branch-and-Sandwich trees therefore consist of two lists with one sublist each: $\mathcal{L}^1 = \{\mathcal{L}^1_1\} = \{4, 6\}, \mathcal{L}^2 = \{\mathcal{L}^2_1\} = \{5, 7\}$.

The most obvious difference between the two approaches at this iteration is that the original approach performs an unnecessary evaluation of $\underline{w}^{(6)}(-1.0) = -0.24$.

Observe, that $\underline{w}^{(6)}(-1.0) = \underline{w}^{(2)}(-1.0)$, but node $k = 2$ is no longer in the Branch-and-Sandwich tree; therefore the original approach fails to recognize this unnecessary evaluation. Using both approaches, we first perform outer-fathoming of node $k = 7$; this is followed by the deletion of list $\mathcal{L}^2$ (see Definition 2.11) and the full-fathoming of nodes $k = 5$ and $k = 7$.

At the *fourth iteration*, both versions of B&S select node $k = 4$ and bisect on $y = 0.5$, generating two new nodes ($k = 8$ and $k = 9$) shown in Figure 5.8f. After this branching step, the two Branch-and-Sandwich trees consist: $\mathcal{L}^1 = \{\mathcal{L}^1_1\} = \{6, 8, 9\}$. At this iteration, the best inner upper bound using both schemes, i.e., $f^{\mathrm{UB},p=1} = f^{\mathrm{UB},k} = -0.33$ for each $k \in \{6, 8, 9\}$. This leads to the full-fathoming of node $k = 8$ (see Definition 2.9), followed by outer-fathoming of node $k = 9$. Finally, using the revised scheme we can fully-fathom node $k = 6$ as $f^{(6),\mathrm{L}} = -0.17$, delete list $\mathcal{L}^1$, and terminate the B&S algorithm successfully. However, using the original approach we cannot fully-fathom node 6 as $\underline{f}^{(6)} = -0.41$ is lower than the current best inner upper bound. This leads to additional B&S iteration and the exploration of new nodes, $k \in \{10, 11, 12, 13\}$. A further aspect worthy of note is that the timing of fathoming and list deletion tests can help to reduce the number of solved inner lower bounding problems (RILB(k)) from 13 to 11. Specifically inner open nodes should only be explored after it has been ascertained that the corresponding list $\mathcal{L}^p$ contains at least one active node. To conclude, the revised approach reduces the total number of solved subproblems and provides a clearer framework to avoid unnecessary calculations or issues with the management of the solution points obtained from (LB(k)).

Finally, Figure 5.9 shows a graphical comparison based on the CPU time using both bounding schemes. As can be seen, in the original scheme, a larger number of inner lower bounding problems is solved, at a smaller unit cost than in the revised scheme. The overall time spent of solving inner lower bounding problems is reduced in the revised scheme. A similar trend is seen for the solution of the outer upper bounding problems which involves the optimal value function ($w(\bar{x})$ or $\underline{w}^{(j)}(\bar{x})$) and the remaining bounding problem ($\bar{F}(\bar{x})$ or $\bar{F}^{(k')}(\bar{x})$). The inner upper bound ($\bar{f}^{(k)}$) and outer lower bound ($\underline{F}^{(k)}$) are unaffected by the choice of the scheme.

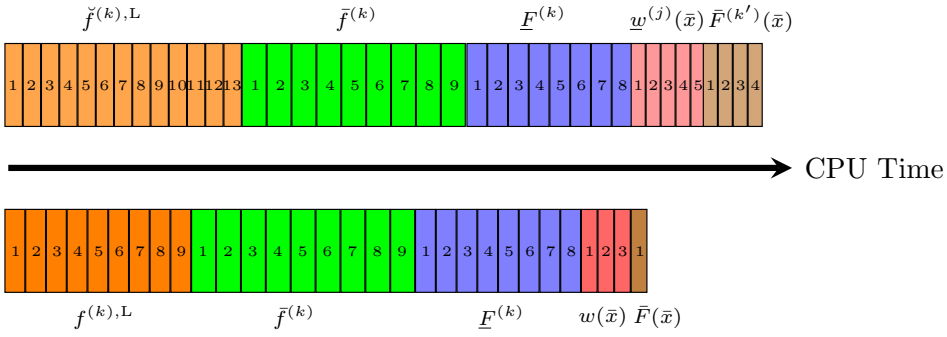


FIG. 5.9. *A graphical comparison of solved subproblems and execution time for each class of subproblems and the total cumulative time using the original bounding schemes (top graph) and the revised bounding schemes (bottom graph). The width of each box represents the CPU time needed.*

818 **6. Conclusions.** We have presented algorithmic improvements and extensions
819 to the recently proposed bilevel deterministic global optimization algorithm, Branch-
820 and-Sandwich based on a combination of theoretical results and heuristic rules. Our
821 algorithmic extensions included revised bounding scheme, including an alternative
822 way to obtain tighter best inner upper bounds, a simpler and significantly faster outer
823 upper bounding scheme as well as alternative choices for branching and node selection.
824 A detailed algorithmic comparison has been used to demonstrate the beneficial impact
825 of the proposed extensions and modifications on the number of iterations and solution
826 performance. In particular, the number of nonconvex subproblems to be solved is
827 greatly reduced. The implementation of the revised Branch-and-Sandwich algorithm
828 as well as a detailed comparison of the performance of different algorithmic options
829 is considered in the second part of this paper [24].

830 **Acknowledgements.** We gratefully acknowledge funding from the Leverhulme
831 Trust through the Philip Leverhulme Prize and from the EPSRC through a Leadership
832 Fellowship [EP/J003840/1].

833 **Data access statement.** Data underlying this article can be accessed on Zenodo
834 at http://dx.doi.org/10.5281/zenodo.44997, and used under the Creative Commons
835 Attribution licence.

836 REFERENCES

837 [1] C. S. Adjiman, I. P. Androulakis, and C. A. Floudas, *A global optimization method, αBB,*
838 *for general twice-differentiable constrained NLPs–II. Implementation and computational*
839 *results*, Computers & Chemical Engineering, 22 (1998), pp. 1159–1179, doi:10.1016/S0098-
840 1354(98)00218-X.
841 [2] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier, *A global optimization method,*
842 *αBB, for general twice-differentiable constrained NLPsI. Theoretical advances*, Computers
843 & Chemical Engineering, 22 (1998), pp. 1137–1158, doi:10.1016/S0098-1354(98)00027-1.
844 [3] J. F. Bard, *Practical Bilevel Optimization*, vol. 30 of Nonconvex Optimization and Its Appli-
845 cations, Springer US, 1998, doi:10.1007/978-1-4757-2836-1.
846 [4] M. S. Bazaraa, C. M. Shetty, and H. D. Sherali, *Nonlinear programming: theory and*
847 *algorithms*, John Wiley & Sons, 1993, doi:10.1002/0471787779.
848 [5] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.
849 [6] J. Bracken and J. T. McGill, *Mathematical programs with optimization problems in the*
850 *constraints*, Operations Research, 21 (1973), pp. 37–44, doi:10.1287/opre.21.1.37.
851 [7] B. Colson, P. Marcotte, and G. Savard, *An overview of bilevel optimization*, Annals of
852 operations research, 153 (2007), pp. 235–256, doi:10.1007/s10479-007-0176-2.
853 [8] S. Dempe, *Foundations of Bilevel Programming*, vol. 61 of Nonconvex Optimization and Its
854 Applications, Kluwer Academic Publishers, Boston, 2002, doi:10.1007/b101970.
855 [9] S. Dempe, *Annotated Bibliography on Bilevel Programming and Mathematical Pro-*
856 *grams with Equilibrium Constraints*, Optimization, 52 (2003), pp. 333–359,
857 doi:10.1080/0233193031000149894.
858 [10] S. Dempe, V. Kalashnikov, G. A. Pérez-Valdés, and N. Kalashnykova, *Bilevel Program-*
859 *ming Problems*, Energy Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015,
860 doi:10.1007/978-3-662-45827-3.
861 [11] S. Dempe and A. B. Zemkoho, *The generalized Mangasarian-Fromowitz constraint qualifica-*
862 *tion and optimality conditions for bilevel programs*, Journal of Optimization Theory and
863 Applications, 148 (2011), pp. 46–68, doi:10.1007/s10957-010-9744-8.
864 [12] J. E. Falk and K. Hoffman, *A nonconvex max-min problem*, Naval Research Logistics Quar-
865 terly, 24 (1977), pp. 441–450, doi:10.1002/nav.3800240307.
866 [13] J. Fortuny-Amat and B. McCarl, *A representation and economic interpretation of a two-*
867 *level programming problem*, Journal of the operational Research Society, (1981), pp. 783–
868 792, doi:10.1057/jors.1981.156.
869 [14] R. L. Graham, B. L. Rothschild, and J. H. Spencer, *Ramsey theory*, vol. 20, John Wiley
870 & Sons, second ed., 1990.

[15] R. Horst and H. Tuy, *Global optimization: Deterministic approaches*, Springer, Berlin, Heidelberg, third ed., 1996, doi:10.1007/978-3-662-03199-5.
[16] P.-M. Kleniati and C. S. Adjiman, *Branch-and-Sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part I: Theoretical development*, Journal of Global Optimization, 60 (2014), pp. 425–458, doi:10.1007/s10898-013-0121-7.
[17] 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*, Journal of Global Optimization, 60 (2014), pp. 459–481, doi:10.1007/s10898-013-0120-8.
[18] P.-M. Kleniati and C. S. Adjiman, *A generalization of the Branch-and-Sandwich algorithm: from continuous to mixed-integer nonlinear bilevel problems*, Computers & Chemical Engineering, 72 (2014), pp. 373–386, doi:10.1016/j.compchemeng.2014.06.004.
[19] J. T. Linderoth and M. W. Savelsbergh, *A computational study of search strategies for mixed integer programming*, INFORMS Journal on Computing, 11 (1999), pp. 173–187, doi:10.1287/ijoc.11.2.173.
[20] R. Misener and C. A. Floudas, *ANTIGONE: Algorithms for coNTinuous / Integer Global Optimization of Nonlinear Equations*, Journal of Global Optimization, 59 (2014), pp. 503–526, doi:10.1007/s10898-014-0166-2.
[21] A. Mitsos and P. I. Barton, *Issues in the development of global optimization algorithms for bilevel programs with a nonconvex inner program*, technical report, Massachusetts Institute of Technology, 2006.
[22] A. Mitsos and P. I. Barton, *A Test Set for Bilevel Programs*. http://www.researchgate.net/publication/228455291, 2007. [Last updated September 19, 2007].
[23] A. Mitsos, P. Lemonidis, and P. I. Barton, *Global solution of bilevel programs with a nonconvex inner program*, Journal of Global Optimization, 42 (2008), pp. 475–513, doi:10.1007/s10898-007-9260-z.
[24] R. Paulavičius, P.-M. Kleniati, and C. S. Adjiman, *BASBL: Branch-And-Sandwich BiLevel solver. II. Implementation and Computational study*, SIAM Journal on Optimization, (2016). Submitted.
[25] R. Paulavičius, P.-M. Kleniati, and C. S. Adjiman, *Global optimization of nonconvex bilevel problems: implementation and computational study of the Branch-and-Sandwich algorithm*, in 26th European Symposium on Computer Aided Process Engineering, Z. Kravanja and M. Bogataj, eds., vol. 38 of Computer Aided Chemical Engineering, Elsevier, 2016, pp. 1977–1982, doi:10.1016/B978-0-444-63428-3.50334-9.
[26] R. Paulavičius, J. Žilinskas, and A. Grothey, *Investigation of selection strategies in branch and bound algorithm with simplicial partitions and combination of Lipschitz bounds*, Optimization Letters, 4 (2010), pp. 173–183, doi:10.1007/s11590-009-0156-3.
[27] N. V. Sahinidis, *BARON 14.4.0: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2014.
[28] K. Shimizu, Y. Ishizuka, and J. F. Bard, *Nondifferentiable and Two-Level Mathematical Programming*, vol. 102, Kluwer Academic Publishers, Boston, 1997, doi:10.1016/S0377-2217(97)00228-2.
[29] O. Stein and G. Still, *On generalized semi-infinite optimization and bilevel optimization*, European Journal of Operational Research, 142 (2002), pp. 444–462, doi:10.1016/S0377-2217(01)00307-1.
[30] G. Still, *Solving generalized semi-infinite programs by reduction to simpler problems*, Optimization, 53 (2004), pp. 19–38, doi:10.1080/02331930410001661190.
[31] M. Tawarmalani and N. V. Sahinidis, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: theory, algorithms, software, and applications*, Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, Boston, 2002, doi:10.1007/978-1-4757-3532-1.
[32] M. Tawarmalani and N. V. Sahinidis, *A polyhedral branch-and-cut approach to global optimization*, Mathematical Programming, 103 (2005), pp. 225–249, doi:10.1007/s10107-005-0581-8.
[33] A. Tsoukalas, B. Rustem, and E. N. Pistikopoulos, *A global optimization algorithm for generalized semi-infinite, continuous minimax with coupled constraints and bi-level problems*, Journal of Global Optimization, 44 (2008), pp. 235–250, doi:10.1007/s10898-008-9321-y.
[34] H. Tuy, *Convex analysis and global optimization*, vol. 22, Springer Science & Business Media, 2013, doi:10.1007/978-1-4757-2809-5.
[35] L. N. Vicente and P. H. Calamai, *Bilevel and multilevel programming: A bibliography review*,

933 Journal of Global Optimization, 5 (1994), pp. 291–306, doi:10.1007/BF01096458.