Journal of Global Optimization (2020) 77:197–225
https://doi.org/10.1007/s10898-020-00874-3

New bounding schemes and algorithmic options for the Branch-and-Sandwich algorithm

R. Paulavičius[1,2] · **C. S. Adjiman**[1]

Received: 27 February 2019 / Accepted: 4 January 2020 / Published online: 31 January 2020

Abstract
We consider the global solution of bilevel programs involving nonconvex functions. Deterministic global optimization algorithms for the solution of this challenging class of optimization problems have started to emerge over the last few years. We present new schemes to generate valid bounds on the solution of nonconvex inner and outer problems and examine new strategies for branching and node selection. We integrate these within the Branch-and-Sandwich algorithm (Kleniati and Adjiman in J Glob Opt 60:425–458, 2014), which is based on a branch-and-bound framework and enables the solution of a wide range of problems, including those with nonconvex inequalities and equalities in the inner problem. The impact of the proposed modifications is demonstrated on an illustrative example and 10 nonconvex bilevel test problems from the literature. It is found that the performance of the algorithm is improved for all but one problem (where the CPU time is increased by 2%), with an average reduction in CPU time of 39%. For the two most challenging problems, the CPU time required is decreased by factors of over 3 and 10.

Keywords Global optimization · Nonconvex bilevel programming · Branch-and-Sandwich algorithm

1 Introduction

Bilevel programming problems (BPP) have a long history in operations research [6,20] and occur in diverse applications, such as chemical and civil engineering, economics, management, transportation etc.; see e.g. [4,10,14,16] 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:

✉ C. S. Adjiman
c.adjiman@imperial.ac.uk

R. Paulavičius
remigijus.paulavicius@mif.vu.lt

[1] Department of Chemical Engineering, Centre for Process Systems Engineering and Institute for Molecular Science and Engineering, Imperial College London, London SW7 2AZ, UK

[2] Present Address: Vilnius University Institute of Data Science and Digital Technologies, Akademijos 4, 08663 Vilnius, Lithuania

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

where the n-dimensional vector $\mathbf{x} \in X \subsetneq \mathbb{R}^n$ denotes the outer (leader) variables and the m-dimensional vector $\mathbf{y} \in Y \subsetneq \mathbb{R}^m$ 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 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 if for a given $\bar{\mathbf{x}}$ the inner (sub)problem parameterized by upper level variable $\mathbf{x} \in X$:

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

has multiple globally optimal solutions in Y to which the follower is indifferent, one must decide whether to adopt an optimistic or pessimistic formulation [14]. Here the optimistic (co-operative) formulation is assumed, i.e., the leader can choose among globally optimal lower-level solutions to achieve the best outer objective value.

Special cases of bilevel programming have been studied extensively, and many algorithms have been proposed, see e.g., [4,7,8,11,12,14–16,19,28,35,36,43] 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. [27] and the approximation method of Tsoukalas et al. [41] apply to very general nonlinear bilevel problems, restricted solely by the absence of inner equality constraints. Recently, both these approaches were extended as a new discretization-based algorithm for bilevel problems with coupling equality constraints [17].

The Branch-and-Sandwich (B&S) algorithm introduced in [22,23] 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 in [22] were demonstrated practically in [23] by applying the algorithm to a library of test problems [26], using a combination of software and manual application. The impact of some algorithmic options on performance was briefly investigated in [30]. Recent efforts to develop a fully automated implementation of the algorithm have led to further insights and developments that can result in a significant reduction in the computational effort required to solve nonconvex bilevel problems. New theoretical findings, together with a revision of the B&S algorithm, an illustration of the impact of the proposed changes on an illustrative example, and a comparison of the computational performance of the various algorithmic options on 10 problems are presented in this paper. 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. 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 [24], although the specifics of this are beyond the scope of this paper.

The remainder of this paper is organized as follows. Basic concepts of B&S and notations are introduced in Sect. 2. In Sect. 3 new bounding schemes for B&S are introduced based on theoretical analysis. In Sect. 4 the B&S search tree management strategy is extended to include heuristics for branching and node selection. In Sect. 5 the revised and the original B&S approaches are summarized. In Sect. 6 an illustrative example is used to investigate the

impact of the proposed changes. In Sect. 7 a computational comparison of the performance of B&S using the new and original bounding schemes as well is performed, as well as the proposed heuristics. Finally, Sect. 8 gives conclusions.

2 Basic concepts and notation

We start with concepts and definitions required for the Branch-and-Sandwich algorithm. Throughout the text, we use Remarks as a way to emphasize key implications of theorems and corollaries.

Definition 1 (*Node*) A node V^k [equivalently an $(n+m)$-rectangle] in the Branch-and-Sandwich tree is uniquely defined by a set of bounds, representing a subdomain of $X \times Y$:

$$V^k = \left\{ \mathbf{v} = (\mathbf{x}, \mathbf{y})^T \in X^k \times Y^k \subseteq X \times Y \subsetneq \mathbb{R}^{n+m} \right\}. \tag{1}$$

Definition 2 (*Node properties*) Each node V^k in the Branch-and-Sandwich tree has a unique number (index), denoted by superscript k. The root node (the whole $X \times Y$ domain) has $k = 1$. A node V^k has the following attributes (properties):

- $\underline{f}^k$: A valid lower bound on the global solution of the inner problem restricted to node V^k;
- $\bar{f}^k$: A valid upper bound on the global solution of the inner problem restricted to node V^k;
- $f^{\mathrm{UB},k}$: The best inner upper bound over the domain $X^k \times Y$ (note this includes all of Y, rather than the subset of Y in node V^k);
- $\underline{F}^k$: A valid lower bound on the global solution of the bilevel problem restricted to node V^k;
- $\bar{\mathbf{x}}^k$: Outer variable vector corresponding to $\underline{F}^k$;
- s^k: State of node V^k: *active, inner-active* or *inactive*.
- l^k: level (depth) of node V^k in the B&S search tree; for the root node $l^1 = 0$;

Based on the bounding values computed at a node V^k, we may decide to fathom the current node. Two types of fathoming operations take place in the B&S algorithm (cf. Sect. 4.6 in [22]). If a node has been shown not to contain a global solution of the inner problem, i.e., for all $\mathbf{x} \in X^k$, the global solution of the inner problem lies outside of Y^k, it is "*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 for $\mathbf{x} \in X^k$, we need to continue exploring it further via branching. In such a case, it is "*outer-fathomed*" rather than fully fathomed. As a result, the state s^k of a node V^k in the Branch-and-Sandwich tree may have one of three different values:

- *Active* or *open* B&S continues to explore for both the outer and inner problems for such nodes, because they may contain a global solution of the bilevel problem. These nodes are stored in list $\mathcal{L}$, also used in standard branch-and-bound algorithms [21].
- *Inner-active* (*outer-fathomed*) B&S continues to explore such nodes for the inner problem only, because they have been shown not to contain a global solution of the bilevel problem but for some $\mathbf{x} \in X^k$ still may contain a global solution for the inner problem. Such nodes are stored in a list $\mathcal{L}_{\mathrm{In}}$.
- *Inactive* (*fully-fathomed*) all other nodes. No further exploration of these nodes is performed and they are not stored.

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 problem. Thus, the branching scheme for bilevel problems differs from that for single level problems. The algorithm proposed by Mitsos et al. [27] allows branching on the outer variables, but not on the inner variables. The branching scheme used in the B&S algorithm, on the other hand, allows branching on both variable sets $\mathbf{x}$ and $\mathbf{y}$. This is made possible by the existence of list $\mathcal{L}_{\text{In}}$.

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 two further types of lists that link together the nodes in an X-partition (Definition 3). We first define an X-partition as follows.

Definition 3 (*X-partition*) Let $P \subsetneq \mathbb{N}$ be a finite index set and $\mathcal{X}_p$ denote a subdomain of X. Then an X-partition is denoted as $\{\mathcal{X}_p \subseteq X : p \in P\}$ where:

$$X = \bigcup_{p \in P} \mathcal{X}_p \text{ and } \mathcal{X}_p \cap \mathcal{X}_q = \partial\mathcal{X}_p \cap \partial\mathcal{X}_q \text{ for all } p, q \in P,\ p \neq q, \tag{2}$$

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

Remark 1 At any given iteration of the B&S algorithm, a specific X-partition is defined and this in turn defines the index set P. Whenever branching takes place on an $\mathbf{x}$ variable, the partition is updated.

We can then define the concept of a sublist.

Definition 4 (*Sublist*) At a given point in the branch-and-bound process, let us consider the partition of X obtained by using the smallest subdomains generated for element of $\mathbf{x} \in X$. Consider some subdomain in the partition $\mathcal{X}_{p,s}$, $s \in S$ where $S \subsetneq \mathbb{N}$ is a finite index set. The sublist $\mathcal{L}_s^p$ consists of all nodes in $\mathcal{L} \cup \mathcal{L}_{\text{In}}$ for which the X domain contains $\mathcal{X}_{p,s}$.

Then, we define a sublist index set $\mathcal{K}_s^p$ that contains the indices of all nodes within a sublist $\mathcal{L}_s^p$.

Definition 5 (*Sublist index set*) The sublist index set $\mathcal{K}_s^p$ is given by

$$\mathcal{K}_s^p = \left\{k : V^k \in \mathcal{L}_s^p\right\} \tag{3}$$

By construction of the branch-and-bound tree, the Y subdomains of the nodes in $\mathcal{L}_s^p$ are non-overlapping. However, the X subdomains in two different sublists may overlap because not all nodes have undergone the same extent of branching. We thus define an independent list that connects together all sublists that cover overlapping parts of the X domain but different subdomains of Y. Such a list makes it possible to take into account the hierarchical structure of the bilevel problem, in which the outer $\mathbf{x}$ variables drive the decision-making at the inner level, during the exploration of the branch-and-bound tree.

Definition 6 (*Independent list*) An independent list $\mathcal{L}^p$, $p \in P$ is given by:

$$\mathcal{L}^p = \{\mathcal{L}_1^p, \ldots, \mathcal{L}_{s_p}^p\}, \tag{4}$$

where s_p is the number of sublists in $\mathcal{L}^p$ and, when $s_p > 1$, for each $s = 1, \ldots, s_p$, there exists $s' = 1, \ldots, s_p$, $s' \neq s$ and a pair of nodes $(V^k \in \mathcal{L}_s^p, V^{k'} \in \mathcal{L}_{s'}^p)$ such that $\text{int}(X^k) \cup \text{int}(X^{k'}) \neq \emptyset$, where $\text{int}(X)$ denotes the interior of X.

For simplicity, we sometimes use the shorthand $V^k \in \mathcal{L}^p$ to indicate that node V^k is such that there exists at least one $s \in \{1, \ldots, s_p\}$ for which $V^k \in \mathcal{L}_s^p$.

Finally, we define the concept of the independence condition.

Definition 7 (*Independence condition* (IC) [22]) If after partitioning of some nodes from $\mathcal{L}^p$, $p \in P$, there exist index sets I and J such that:

$$\begin{aligned} I \cap J = \emptyset, I \cup J = \{1, \ldots, s_p\}, \\ \{\mathcal{L}_i^p : i \in I\} \cap \{\mathcal{L}_j^p : j \in J\} = \emptyset, \end{aligned} \tag{IC}$$

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

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

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

In the revised B&S algorithm, we use the same list management scheme as in original B&S, therefore we refer to Sect. 4.1 in [22] for details. 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 is given in Algorithm 1.

Algorithm 1 Branch-and-Sandwich

```
Initialize
Compute inner and outer bounds at the root node. Apply fathoming-rules.        ▷ Sect. 4.6 in [22]
while list of active nodes is not empty do
    Select node(s) for branching.                                              ▷ Section 4.2
    Branch selected node(s). Update lists.          ▷ Section 4.1 and Sect. 4.1 in [22]
    for each new node do
        Compute inner lower bound. Apply full-fathoming.                       ▷ Sect. 3.1 in [22]
        Compute inner upper bound if needed.                                   ▷ Section 3.1
        Update best inner upper bound. Apply full-fathoming.                   ▷ Section 3.2
        Compute outer lower bound. Apply outer-fathoming.                      ▷ Section 3.3
        Compute inner subproblems.                                             ▷ Section 3.4
        Compute outer upper bound if needed. Apply outer-fathoming.            ▷ Section 3.4
    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
```

3 New bounding schemes

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, i.e., four bounds are obtained for each node. In this section, we present new bounding schemes based on theoretical analysis and practical observations and refer the reader to [22] for the bounding schemes that remain unchanged.

3.1 Inner upper bounding scheme at a node

To generate the inner upper bound at a node $V^k \in (\mathcal{L} \cup \mathcal{L}_{\text{In}})$, the original bounding scheme involves the solution of a challenging optimization problem. Here, we show that this can be avoided at some nodes and that this can result in a guaranteed improvement in the performance of the algorithm.

Two formulations of the inner upper bounding problem have been previously proposed (cf. Sect. 4.4 in [22]): a max–min problem, which yields an exact value of the inner upper bound:

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

or a computationally less expensive relaxed inner upper bound $\bar{f}^k$, which is based on a semi-infinite reformulation of (IUB(V^k)) and a KKT relaxation [37,38]:

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

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.:

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

We consider each formulation in turn.

3.1.1 Bound inheritance for the inner upper bound

We first prove a useful property of the inner upper bound $f^{k,\mathrm{U}}$.

Theorem 1 *The inner upper bound $f^{k,\mathrm{U}}$ is not improving when branching on an inner* ($\mathbf{y}$) *variable takes place, i.e.,*

$$f^{k,\mathrm{U}} = \min\{ f^{k_1,\mathrm{U}},\ f^{k_2,\mathrm{U}} \} \tag{5}$$

where $\{Y^{k_1}, Y^{k_2}\}$ is a partition of Y^k and $X^k = X^{k_1} = X^{k_2}$. Equivalently, replacing each inner upper bound in Eq. (5) *by the corresponding* (*IUB*) *problem*:

$$\begin{aligned} & \max_{\mathbf{x}\in X^k} \min_{\mathbf{y}\in Y^k} \{ f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{0}, \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \} \\ & \quad = \min_{j\in\{k_1,k_2\}} \left\{ \max_{\mathbf{x}\in X^k} \min_{\mathbf{y}\in Y^j} \{ f(\mathbf{x},\mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{0}, \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \} \right\}. \end{aligned} \tag{6}$$

Proof After branching on an inner variable ($\mathbf{y}$) at a node $V^k = X^k \times Y^k$ 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

$$\begin{aligned} &\min_{\mathbf{y}\in Y^k} \{f(\bar{\mathbf{x}}, \mathbf{y}) \text{ s.t. } \mathbf{g}(\bar{\mathbf{x}}, \mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\bar{\mathbf{x}}, \mathbf{y}) = \mathbf{0}\} \\ &\quad \leq \min_{\mathbf{y}\in Y^j} \{f(\bar{\mathbf{x}}, \mathbf{y}) \text{ s.t. } \mathbf{g}(\bar{\mathbf{x}}, \mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\bar{\mathbf{x}}, \mathbf{y}) = \mathbf{0}\}, \quad \forall j \in \{k_1, k_2\}, \end{aligned} \tag{7}$$

where the equality must hold for at least one of k_1 and k_2 since the solution(s) of the left hand side is (are) in $Y^k = Y^{k_1} \cup Y^{k_2}$. Next, from Eq. (7) it follows that over all $\mathbf{x} \in X^k$ we have

$$\begin{aligned} &\max_{\mathbf{x}\in X^k} \min_{\mathbf{y}\in Y^k} \{f(\mathbf{x}, \mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0}\} \\ &\quad \leq \max_{\mathbf{x}\in X^k} \min_{\mathbf{y}\in Y^j} \{f(\mathbf{x}, \mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0}\}, \quad \forall j \in \{k_1, k_2\}, \end{aligned} \tag{8}$$

where the equality must again hold for at least one of k_1 and k_2. Finally, from Eq. (8) it follows that

$$\begin{aligned} &\max_{\mathbf{x}\in X^k} \min_{\mathbf{y}\in Y^k} \{f(\mathbf{x}, \mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0}\} \\ &\quad = \min_{j\in\{k_1,k2\}} \left\{ \max_{\mathbf{x}\in X^k} \min_{\mathbf{y}\in Y^j} \{f(\mathbf{x}, \mathbf{y}) \text{ s.t. } \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0}\} \right\}, \end{aligned} \tag{9}$$

and this concludes the proof. □

A graphical illustration of Theorem 1 can be seen in Fig. 1, 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 of $\bar{x} = 0.5$, we find that $f^{1,\mathrm{U}} = -0.08$, $f^{2,\mathrm{U}} = 0.02$ and $f^{3,\mathrm{U}} = -0.08$ so that $f^{1,\mathrm{U}} = \min\left\{f^{2,\mathrm{U}}, f^{3,\mathrm{U}}\right\}$ in agreement with Theorem 1.

Corollary 1 *After branching at node V^k on a $\mathbf{y}$-variable and creating two child nodes V^{k_1} and V^{k_2}, computation of the inner upper bound $f^{k,\mathrm{U}}$ for the child nodes is unnecessary.*

To describe the proposed inner upper bounding scheme when branching on a $\mathbf{y}$ variable, we first note that all sublists $\mathcal{L}_s^p,\ s \in \{1, 2, \ldots, s_p\}$ containing node V^k are modified in Step 5 of Algorithm 1, such that:

$$\mathcal{L}_s^p \leftarrow (\mathcal{L}_s^p \backslash \{V^k\}) \cup \{V^{k_1}, V^{k_2}\},$$

and that no new sublists are created. Then, we can use Theorem 1 to set

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

Thus computation of the inner upper bound or relaxed inner upper bound only takes place at the root node and following branching on an $\mathbf{x}$ variable. This is a useful property because the computational complexity of solving the inner upper bounding problem is similar to solving the original bilevel problem [18]. Theorem 1 shows that there is no impact on the bounds when avoiding these calculations.

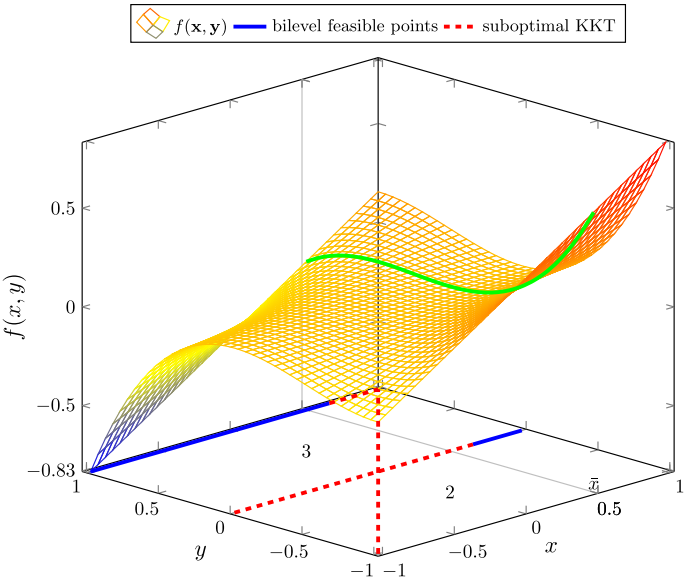


Fig. 1 Graphical illustration of the inner problem from the bilevel instance `mb_1_1_08` [26] together with its minima (thick (blue) lines) and suboptimal KKT points (dashed (red) lines). Numbers within rectangles denote node indices and $V^1 = V^2 \cup V^3$. The graphical illustration shows that the inner upper bound $f^{k,\mathrm{U}}$ is not improving when branching on an inner (y) variable takes place. (Color figure online)

3.1.2 Bound inheritance for the relaxed inner upper bound

Next, we consider the case of the relaxed inner upper bound $\bar{f}^k$. In contrast to (IUB(V^k)), the relaxed inner upper bounding scheme (RIUB(V^k)) does not perform a minimization over Y, but rather a maximization over a subset of Y, therefore an analog to Theorem 1 does not hold. This situation can easily be observed in Fig. 1. The relaxed inner upper bound over the root node V^1 gives $\bar{f}^1 = 0.17$ at the suboptimal KKT point (1, 1); solution over node V^2 yields $\bar{f}^2 = 0.0$ (the former suboptimal KKT point (1, 1) does not belong to this node) and solution over node V^3 gives $\bar{f}^3 = 0.17$; thus, $\bar{f}^1 = 0.17 > \min\{\bar{f}^2, \bar{f}^3\} = 0.0$.

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 could be developed to trade-off the cost of computing $\bar{f}^k$ against the benefit of tighter relaxed inner upper bounds, but this is not explored here. Instead, we update the relaxed inner upper bound for each new node when appropriate (see Remarks 3, 5).

3.2 Inner upper bounding scheme over $X^k \times Y$

Having computed the inner upper bound $f^{k,\mathrm{U}}$ or relaxed inner upper bound $\bar{f}^k$ for a specific node V^k, we need to generate an upper bound that is valid for the inner problem over X^k on the entire Y domain. In this section, we show how the cost of obtaining such a bound can be reduced and how the value of the bound can sometimes be improved (decreased).

3.2.1 Best inner upper bound for $X^k \times Y$ based on 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 8 (*Best inner upper bound for a list* $\mathcal{L}^p$, *cf. Definition* 7 *in* [22]) 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:

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

Remark 2 $f^{\mathrm{UB},p}$ 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 Sec. 4.6 in [22]).

Theorem 2 *If after branching at a node* V^k *on an* $\mathbf{x}$*-variable, the independence condition* (*see Definition* 7) *is not satisfied, then the best inner upper bound* $f^{\mathrm{UB},p}$ *cannot improve.*

Proof For clarity, all lists, sublists, index sets and the best inner upper bound before branching are indicated by the superscript 0 and those after branching are indicated by the superscript n for "new". Before branching, the best inner upper bound ($f^{\mathrm{UB},p,0}$) for a list $\mathcal{L}^{p,0}$ is

$$f^{\mathrm{UB},p,0} = \max\left\{ \min_{j\in\mathcal{K}_1^{p,0}}\left\{\bar{f}^j\right\}, \ldots, \min_{j\in\mathcal{K}_{s_p}^{p,0}}\left\{\bar{f}^j\right\}\right\}. \tag{10}$$

After bisecting on an $\mathbf{x}$-variable, two child nodes V^{k_1} and V^{k_2} are created such that $X^k = X^{k_1} \cup X^{k_2}$, $Y^k = Y^{k_1} = Y^{k_2}$. We then replace all sublists of $\mathcal{L}^{p,0}$ containing node V^k (i.e., sublists $\mathcal{L}_s^{p,0}$, $s \in \{1, \ldots, s_p^0 : V^k \in \mathcal{L}_s^{p,0}\}$) by one or two new sublists that contain V^{k_1} or V^{k_2} instead of V^k (cf Definition 6 in [22]). If the independence condition (IC) is not satisfied, no new independent list is created. Moreover, from the properties of $\bar{f}^k$, it follows in such a case that

$$\max\{\bar{f}^{k_1}, \bar{f}^{k_2}\} = \bar{f}^k. \tag{11}$$

One can thus see that a value $\bar{f}^k$ will only be removed from the maximization in (10) when (IC) is satisfied. More formally, three cases are possible:

$$\text{Case 1:} \quad \begin{cases} \bar{f}^{k_1} = \bar{f}^k \\ \bar{f}^{k_2} < \bar{f}^k \end{cases} \tag{12}$$

$$\text{Case 2:} \quad \begin{cases} \bar{f}^{k_1} < \bar{f}^k \\ \bar{f}^{k_2} = \bar{f}^k \end{cases} \tag{13}$$

$$\text{Case 3:} \quad \begin{cases} \bar{f}^{k_1} = \bar{f}^k \\ \bar{f}^{k_2} = \bar{f}^k \end{cases} \tag{14}$$

There is no possible improvement to $f^{\mathrm{UB},p,0}$ when (14) holds; without loss of generality, consider the case when (12) holds, noting that equivalent arguments can be made when (13) is satisfied. From (11) and (12) it follows that

$$\min_{j\in\mathcal{K}_{s_i}^{p,n}}\left\{\bar{f}^j\right\} = \min_{j\in\mathcal{K}_s^{p,0}}\left\{\bar{f}^j\right\}, \forall s_i \in \left\{s' = 1,\ldots,s_p^n\colon V^{k_1} \in \mathcal{L}_{s'}^{p,n}\right\}, \forall s \in \left\{s' = 1,\ldots,s_p^0\colon V^k \in \mathcal{L}_{s'}^{p,0}\right\}, \tag{15}$$

$$\min_{j\in\mathcal{K}_{s_i}^{p,n}}\left\{\bar{f}^j\right\} \leq \min_{j\in\mathcal{K}_s^{p,0}}\left\{\bar{f}^j\right\}, \forall s_i \in \{s' = 1,\ldots,s_p^n\colon V^{k_2} \in \mathcal{L}_{s'}^{p,n}\}, \forall s \in \left\{s' = 1,\ldots,s_p^0\colon V^k \in \mathcal{L}_{s'}^{p,0}\right\}, \tag{16}$$

and

$$f^{\mathrm{UB},p,n} = \max\left\{\min_{j\in\mathcal{K}_1^{p,n}}\left\{\bar{f}^j\right\},\ldots,\min_{j\in\mathcal{K}_{s_p^n}^{p,n}}\left\{\bar{f}^j\right\}\right\}. \tag{17}$$

Therefore, from (15) to (17) it follows that $f^{\mathrm{UB},p,n} = f^{\mathrm{UB},p,0}$. □

This situation is illustrated in Example 1.

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

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

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

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

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

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

Remark 3 From Theorem 2 and Remark 2, it follows that for the newly created nodes after branching on an **x**-variable, the computation of the relaxed inner upper bound $\bar{f}^k$ (as well as the best inner upper bound $f^{\mathrm{UB},p}$) can be postponed until the list satisfies the independence condition (IC).

Remark 4 It further follows that in cases where all nodes within an independent list are outer-fathomed before the independence condition is met, the computation of the relaxed inner upper bound $\bar{f}^k$ can be avoided altogether.

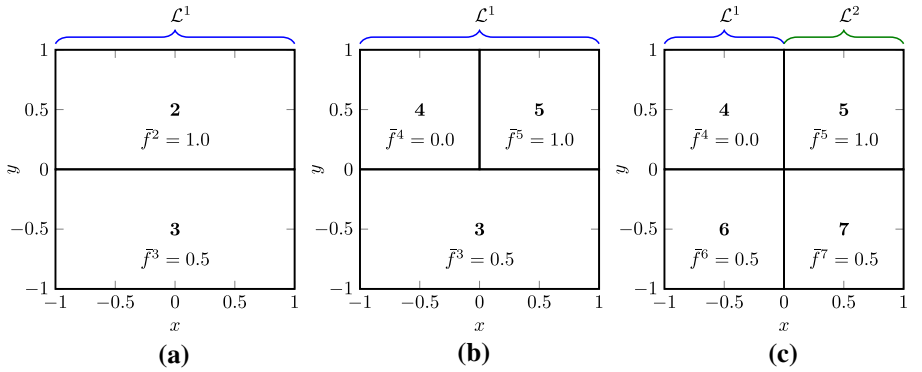


Fig. 2 Illustration of a case where branching on a variable x takes place twice, starting from the partitioning in (**a**). Numbers within rectangles denote node indices. In (**b**), while a better inner upper bound ($\bar{f}^4 = 0.0$) is obtained, the best inner upper bound (BIUB($\mathcal{L}^p$)) does not improve. In (**c**), it improves as it is possible to the independence condition [see Eq. (IC) in [22]] is satisfied (**c**)

3.2.2 Best inner upper bound for $X^k \times Y$ based on a subset of list $\mathcal{L}^p$

Although Theorem 2 indicates that the best inner upper bound cannot improve with the original inner upper bound scheme when the independence condition does not hold, this can be circumvented by modifying the bounding scheme. We take advantage of the fact that a given node V^k which belongs to at least one sublist in $\mathcal{L}^p$ may not belong to all sublists $\mathcal{L}^p_s \in \mathcal{L}^p : s \in \{1, \ldots, s_p\}$. Then, a best inner upper bound that is valid for node V^k can be obtained by taking into account only those sublists that contain V^k, i.e., by considering smaller subsets of subdomain $\mathcal{X}_p$ where possible.

Definition 9 (*Best inner upper bound over the set of sublists of $\mathcal{L}^p$ containing node V^k*)

$$f^{\mathrm{UB},k} = \max \left\{ \min_{j \in \mathcal{K}^p_s} \left\{ \bar{f}^j \right\}, s \in \{s' = 1, \ldots, s_p : V^k \in \mathcal{L}^p_{s'}\} \right\}. \qquad (\mathrm{BIUB}(\mathcal{L}^p_s))$$

Note that this best inner upper bound is differentiated from the best inner upper bound defined over all of $\mathcal{L}^p$ by the use of the superscript k rather than p.

Next, this new bound is shown to have the useful property that it is at least as tight as the original best inner upper bound (BIUB($\mathcal{L}^p$)).

Theorem 3 *The best inner upper bound ($f^{\mathrm{UB},k}$) for the set of sublists in $\mathcal{L}^p$ that contain node V^k is at least as tight as the best inner upper bound ($f^{\mathrm{UB},p}$) for the list $\mathcal{L}^p$ containing node V^k, i.e.,*

$$f^{\mathrm{UB},k} \leq f^{\mathrm{UB},p}. \qquad (18)$$

Proof The set $\mathcal{L}^{p,k}$ of sublists to which node V^k belongs is a subset of all sublists in independent list $\mathcal{L}^p = \{\mathcal{L}^p_1, \ldots, \mathcal{L}^p_{s_p}\}$ containing node V^k:

$$\mathcal{L}^{p,k} = \left\{ \mathcal{L}^p_s : s = 1, \ldots, s_p : V^k \in \mathcal{L}^p_s \right\} \subseteq \mathcal{L}^p. \qquad (19)$$

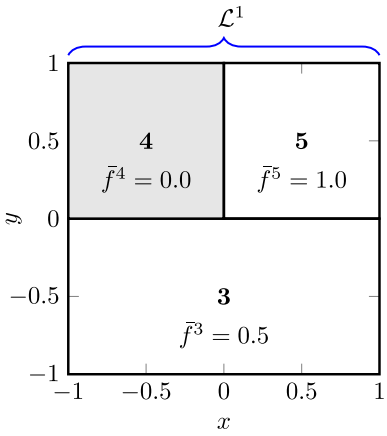


Fig. 3 Illustration of a case in which the best inner upper bound for node V^k based on a set of sublists (BIUB($\mathcal{L}_s^p$)) yields a tighter bound compared to the original approach in which the best inner upper bound (BIUB($\mathcal{L}^p$)) is derived based on $\mathcal{L}^p$. As before, the numbers within rectangles denote node indices

From (19) it follows that the set of minimal inner upper bound values for all the sublists containing node V^k is also a subset of the set of minimal values for all sublists in $\mathcal{L}^p$:

$$\left\{ \min_{j \in \mathcal{K}_s^p} \left\{ \bar{f}^j \right\}, s \in \{s' = 1, \ldots, s_p : V^k \in \mathcal{L}_{s'}^p\} \right\} \subseteq \left\{ \min_{j \in \mathcal{K}_s^p} \left\{ \bar{f}^j \right\}, s = 1, \ldots, s_p \right\}. \quad (20)$$

Thus, from (19) and (20) it follows that

$$\max \left\{ \min_{j \in \mathcal{K}_s^p} \left\{ \bar{f}^j \right\}, s \in \{s' = 1, \ldots, s_p : V^k \in \mathcal{L}_{s'}^p\} \right\} \leq \max \left\{ \min_{j \in \mathcal{K}_s^p} \left\{ \bar{f}^j \right\}, s = 1, \ldots, s_p \right\},$$

and this concludes the proof. □

Example 2 This approach is illustrated in Fig. 3 to show the advantage of deriving a best inner upper bound based on only those sublists in $\mathcal{L}^p$ that contain node V^k. In the case shown in Fig. 3, there is one independent list containing two sublists: $\mathcal{L}^1 = \{\mathcal{L}_1^1, \mathcal{L}_2^1\}$ with $\mathcal{L}_1^1 = \{V^3, V^4\}$ and $\mathcal{L}_2^1 = \{V^3, V^5\}$. Using the original (BIUB($\mathcal{L}^p$)) 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($\mathcal{L}_s^p$)), the best inner upper bound for nodes V^3 and V^5 is the same as $f^{\mathrm{UB},p=1}$: $f^{\mathrm{UB},k=3} = f^{\mathrm{UB},k=5} = \min\{0.5, 1.0\} = 0.5$, but for node V^4, a lower (tighter) value is obtained: $f^{\mathrm{UB},k=4} = \min\{0.5, 0.0\} = 0.0$.

The impact of generating a potentially tighter best inner upper bound $f^{\mathrm{UB},k}$ for the domain $X^k \times Y$ is investigated in Sect. 7. We note that the computational cost of computing $f^{\mathrm{UB},k}$ is similar to that of computing $f^{\mathrm{UB},p}$.

Remark 5 From Theorem 3, it follows that to take full advantage of the fact $f^{\mathrm{UB},k}$ is a tighter bound than $f^{\mathrm{UB},p}$, the computation of $f^{\mathrm{UB},k}$ should take place whenever branching on an $\mathbf{x}$-variable, because $f^{\mathrm{UB},k}$ can improve after branching on an $\mathbf{x}$-variable even when the independence condition is not satisfied, in contrast to $f^{\mathrm{UB},p}$ (cf., Theorem 2 and Remark 3).

Remark 6 The use of the potentially tighter $f^{\mathrm{UB},k}$ instead of $f^{\mathrm{UB},p}$ to check for full-fathoming (Sec. 4.6 in [22]) can lead to the faster removal of nodes from all lists.

Remark 7 Finally, recall that it was proven in Lemma 2 in [22] that the original fathoming scheme guarantees that a subregion where a global optimal solutions lies remains inner-active (i.e., in at least one list $\mathcal{L}^p$, $p \in P$) until that global solution is identified. It can be shown trivially that this property is preserved by construction with the revised scheme.

3.3 Outer lower bounding scheme

The formulation of the outer lower bounding problem at a node $V^k \in \mathcal{L} \cap \mathcal{L}^p$ is based on the single-level reformulation of the bilevel problem [14] and makes use of the best inner upper bound for the corresponding $X^k \times Y$ domain (cf. Sect. 3.2 of [22]) as a constraint on the value of the inner objective function:

$$f(\mathbf{x}, \mathbf{y}) \leq f^{\mathrm{UB},p}. \tag{21}$$

We therefore update the formulation to use $f^{\mathrm{UB},k}$ rather than $f^{\mathrm{UB},p}$:

$$f(\mathbf{x}, \mathbf{y}) \leq f^{\mathrm{UB},k}, \tag{22}$$

thereby obtaining a lower bound $\underline{F}^k$ on the bilevel objective function that is as least as tight as that proposed in [22].

3.4 Outer upper bounding scheme

In the revised B&S, we propose using an outer upper bounding scheme based on [27]. To motivate this choice, we undertake an analysis of the behaviour of the outer upper bounding scheme in [22], discussing observed drawbacks and suggesting possible ways to overcome them.

3.4.1 Analysis of existing outer upper bounding scheme

We begin by summarizing the outer bounding scheme in [22], given a node $V^k \in \mathcal{L} \cap \mathcal{L}^p$ and $(\bar{\mathbf{x}}^k, \bar{\mathbf{y}}^k)$, the solution of the corresponding outer lower bounding problem. We set $\bar{\mathbf{x}}$ equal to $\bar{\mathbf{x}}^k$, and seek a node $V^{k'} \in (\mathcal{L} \cup \mathcal{L}_{\mathrm{In}}) \cap \mathcal{L}^p$ based on (MinRISP($\bar{\mathbf{x}}$, $V^{k'}$)) to formulate the outer upper bounding problem. We first introduce relevant definitions.

Definition 10 (*Node index set based on* $\bar{\mathbf{x}}$) Given a domain X^k, the corresponding list $\mathcal{L}^p$ in which all nodes that cover X^k are stored, and a point $\bar{\mathbf{x}} \in X^k$, the index set $\mathcal{K}^{p,\bar{\mathbf{x}}}$ is defined as the set of the indices of those nodes V^j that belong to a sublist $\mathcal{L}^p_s \in \mathcal{L}^p$ and for which $\bar{\mathbf{x}} \in X^j$:

$$\mathcal{K}^{p,\bar{\mathbf{x}}} = \{j : \exists s \in \{1, \ldots, s_p\} : V^j \in \mathcal{L}^p_s \text{ and } \bar{\mathbf{x}} \in X^j\}.$$

Next, recall the definition of $\underline{w}^j(\bar{\mathbf{x}})$ at some node V^j for which $\bar{\mathbf{x}} \in X^j$, given by [22]:

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

where $\breve{f}$, $\breve{g}$, $\breve{h}$ and $\breve{h}^-$ represent convex underestimators (e.g., [2,39]) of functions f, g, h and $-h$. We are interested in the node(s) that yield the smallest value of $\underline{w}^j(\bar{\mathbf{x}})$ over all sublists $\mathcal{L}^p_s \in \mathcal{L}^p$, $s = 1, \ldots, s_p$ and among those, we choose the node with the lowest value of the outer lower bound. Then, the index k' of node $V^{k'}$ is given by:

$$k' = \min\left\{\arg\min_{\bar{k}\in\mathcal{K}^{p,\bar{\mathbf{x}}}_{\underline{w}}}\left\{\underline{F}^{\bar{k}}\right\}\right\}, \qquad \text{(MinRISP}(\bar{\mathbf{x}}, V^{k'}))$$

where

$$\mathcal{K}^{p,\bar{\mathbf{x}}}_{\underline{w}} = \left\{\bar{k}: \bar{k} = \arg\min_{j\in\mathcal{K}^{p,\bar{\mathbf{x}}}}\left\{\underline{w}^{j}(\bar{\mathbf{x}})\right\}\right\}, \tag{23}$$

i.e., $\mathcal{K}^{p,\bar{\mathbf{x}}}_{\underline{w}}$ denotes the set of the indices of all nodes $\bar{k}$ in the sublists of $\mathcal{L}^p$ for which $\underline{w}^{\bar{k}}(\bar{\mathbf{x}})$ is at the minimum value.

Remark 8 The use of convex underestimators means that problem (RISP($\bar{\mathbf{x}}$, V^j)) is convex and hence relatively inexpensive to solve, while maintaining the convergent property of the bounding scheme (cf. [22]). This may however come at a cost in terms of the quality of the upper bounds generated and this issue is discussed further in this section.

Remark 9 The choice of the node with the lowest value of the outer lower bound is consistent with the requirement in optimistic bilevel programming that the leader consider the best candidate among indifferent solutions for the follower. 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($\bar{\mathbf{x}}$, V^j)) may be infeasible over some or even over all nodes whose indices belong to $\mathcal{K}^{p,\bar{\mathbf{x}}}$. 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, however, a node $V^k \in \mathcal{L}\cup\mathcal{L}_{\text{In}}$ such that the corresponding (RISP($\bar{\mathbf{x}}$, V^k)) problem is feasible, the following outer upper bounding problem is solved for $\mathbf{x} = \bar{\mathbf{x}}$:

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

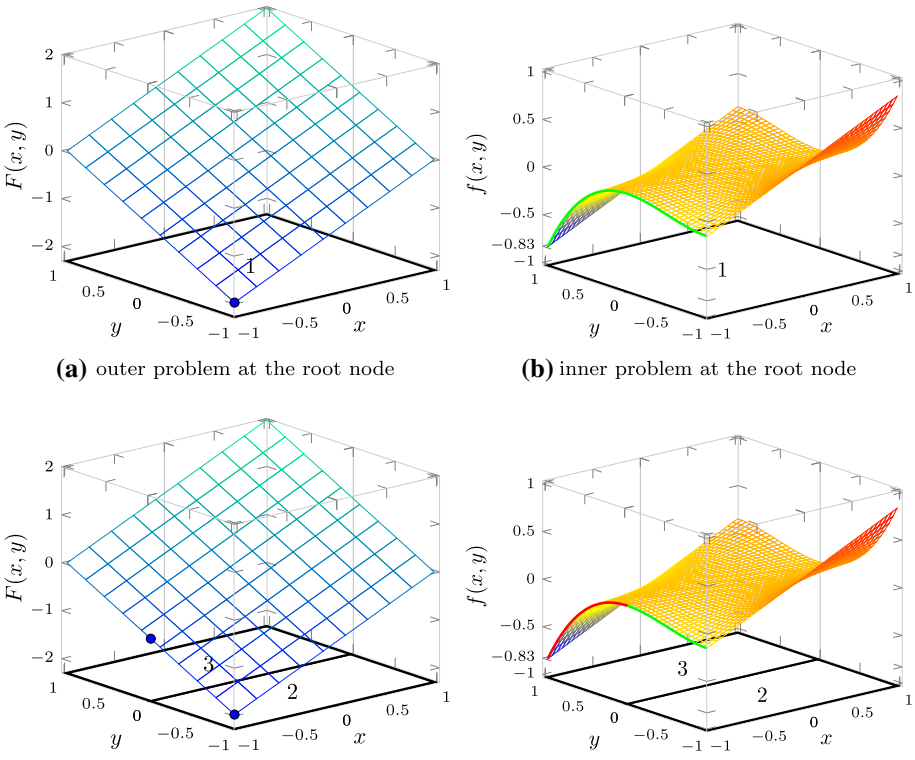
where ε_f is the optimality tolerance for the inner problem. If problem (UB($\bar{\mathbf{x}}$, $V^{k'}$)) is infeasible, then no solution exists for $\bar{\mathbf{x}}$; otherwise, an upper bound is obtained in the sense of an ε_f-feasible point [27]. The implications of seeking an ε-optimal bilevel solution have been discussed in the literature and been shown in [3,9] that there is no clear pattern in the way suboptimality in the inner problem affects the outer objective function, i.e, two close inner solutions may cause very different outer objective function values.

Alternatively, a valid upper bound can be found through any feasible point of problem (UB($\bar{\mathbf{x}}$, $V^{k'}$)), e.g., a point obtained from a local solution.

In the following set of remarks, we explore the behavior of this scheme for generating outer upper bounds.

Remark 10 When node $V^{k'}$ 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($\bar{\mathbf{x}}$, $V^{k'}$)).

Remark 11 While we would like to avoid repeating unnecessary calculations, the solution of (RISP($\bar{\mathbf{x}}$, V^k)) can only be avoided if another (RISP($\bar{\mathbf{x}}'$, V^j)) has already been solved with $\bar{\mathbf{x}}' = \bar{\mathbf{x}}$ and $Y^j = Y^k$. These conditions are restrictive as illustrated in the following example.



(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. 4 Graphical illustration of the outer and inner problems together with found solution points from the lower bounding problems (blue dots) and Y subdomains over which (RISP($\bar{\mathbf{x}}$, V^j)) is solved (green and red lines). As before, numbers within rectangles indicate node indices. (Color figure online)

Example 3 For the case shown in Fig. 4a, b, we have one list containing one sublist at the root node: $\mathcal{L}^1 = \{\mathcal{L}_1^1\} = \{V^1\}$. Solving the outer lower bounding problem 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 Fig. 4b). The value found is used in (UB($\bar{\mathbf{x}}$, $V^{k'}$)).

Next, after branching on variable y, the case shown in Figs. 4c, d is obtained, i.e., there is one independent list containing one sublist: $\mathcal{L}^1 = \{\mathcal{L}_1^1\} = \{\{V^2, V^3\}\}$. After solving both outer lower bounding problems 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($\bar{\mathbf{x}}$, V^j)) (RISP($\bar{\mathbf{x}}$, V^2)) 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($\bar{\mathbf{x}}$, $V^{k'}$)) 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.

Finally use of convex relaxations in problem (RISP($\bar{\mathbf{x}}$, V^j)) can cause slow convergence, described in Remark 12.

Remark 12 The value obtained from the solution of problem (RISP($\bar{\mathbf{x}}$, V^j)) plays a significant role in the (UB($\bar{\mathbf{x}}$, $V^{k'}$)) 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($\bar{\mathbf{x}}$, V^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($\bar{\mathbf{x}}$, V^j)) can be very loose due to the size of the domain. A natural way to overcome this is to replace subproblem (RISP($\bar{\mathbf{x}}$, V^j)) with the computationally more expensive but tighter approach:

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

and to select the node $V^{k'}$ for problem (UB($\bar{\mathbf{x}}$, $V^{k'}$)) using

$$k' = \min \left\{ \arg\min_{\bar{k} \in \mathcal{K}_w^{p,\bar{\mathbf{x}}}} \left\{ \underline{F}^{\bar{k}} \right\} \right\}, \tag{MinISP($\bar{\mathbf{x}}$, $V^{k'}$))}$$

where

$$\mathcal{K}_w^{p,\bar{\mathbf{x}}} = \left\{ \bar{k} : \bar{k} = \arg\min_{j \in \mathcal{K}^{p,\bar{\mathbf{x}}}} \left\{ w^j(\bar{\mathbf{x}}) \right\} \right\}. \tag{24}$$

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.

3.4.2 Upper bounding problem based on [27]

Let us observe that the original upper bounding procedure can be computationally demanding, especially during the formulation of (MinRISP($\bar{\mathbf{x}}$, $V^{k'}$)) which requires the solution of (RISP($\bar{\mathbf{x}}$, V^j)) over several nodes $V^j \in \mathcal{L}^p$. Therefore the original upper bounding procedure becomes more and more expensive as the size of the independent lists increases. Thus motivated by the performance of the upper bounding scheme in [27], we adopt this alternative approach to obtain a valid upper bound that avoids this overhead. Using the notation in our current work, this requires solving (ISP($\mathbf{x}$)) defined over the whole of Y:

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

This is followed by the solution of the upper bounding problem using $w(\bar{\mathbf{x}})$ and considering the whole of Y:

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

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, highlighted in the following remarks.

Remark 13 This 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 14 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 ISP($\bar{\mathbf{x}}$, $V^{k'}$), i.e.,

$$w(\bar{\mathbf{x}}) = w^{k'}(\bar{\mathbf{x}}), \tag{25}$$

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

$$\bar{F}(\bar{\mathbf{x}}) \leq \bar{F}^{k'}(\bar{\mathbf{x}}). \tag{26}$$

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

Remark 15 Following the identification of a value $\bar{\mathbf{x}}$ as the solution of the lower bounding problem, 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

$$\mathbb{X} = \{\bar{\mathbf{x}}_i \in X\colon\ \forall i \neq j \in \{1, \ldots, |\mathbb{X}|\},\ \bar{\mathbf{x}}_i \neq \bar{\mathbf{x}}_j\}, \tag{27}$$

corresponding to unique solution points of outer-variables ($\bar{\mathbf{x}}$). This procedure is much simpler than identifying duplicate problems in the original bounding scheme (see Remark 11).

4 New branching and node selection rules

In this section, new strategies for branching and the management of nodes are introduced.

4.1 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 [22], 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 [1,33]). 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)]. The rules for branching variable selection are stated in Algorithm 2.

Algorithm 2 Branching variable selection

Consider a node $V^k \in (\mathcal{L} \cup \mathcal{L}_{\text{In}})$.

1: Select the branching variable v_{br} with the largest normalized range (edge).

2: If several variables satisfy the largest (normalized) range (edge) requirement, there are two options:

Option (XY): Find the branching variable v_{br} with the lowest-index:

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

Option (YX): Find the branching variable v_{br} with the highest-index:

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

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

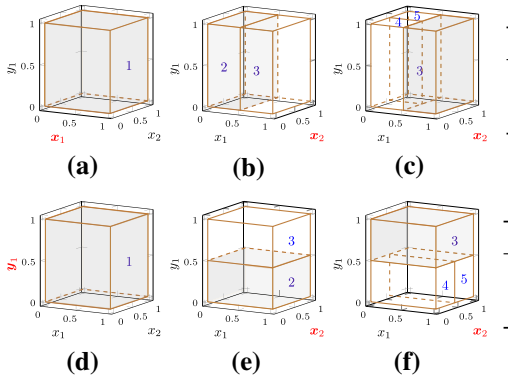
Having identified a branching variable, a standard bisection strategy is adopted to select a branching point, although other approaches could be adopted [2].

Property 1 (Branching point selection rule) *For the selected branching variable v_{br}, the branching point is found using exact bisection* [42].

An illustration of the subdivision process, i.e., the selection of the branching variable and branching point as well as the list management, is presented in Example 4.

Example 4 Consider the six partitioning examples of the three-dimensional space $X \times Y = [0, 1]^2 \times [0, 1]$ presented in Fig. 5. In these pictures, the grey color highlights a node that has been selected for branching. Numbers inside the cuboids are the unique node indices k. At the first iteration (Fig. 5a, d), the selected branching variable v_{br} depends on the branching variable selection rule, as all three variables satisfy the normalized longest range requirement (see Algorithm 2, 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 $v_{br} = x_1$ (as noted in Fig. 5a); however when the (YX) strategy is used, the selected branching variable is $v_{br} = y_1$ (see Fig. 5d). After branching (bisection) on the selected variable in each case, we obtain two completely different partitions illustrated in Fig. 5b, e, respectively. After bisection on variable x_1, there are two independent lists with one sublist each: $\mathcal{L}^1 = \{\mathcal{L}_1^1\} = \{\{V^2\}\}$ and $\mathcal{L}^2 = \{\mathcal{L}_1^2\} = \{\{V^3\}\}$, but there is only one independent list after bisection on variable y_1: $\mathcal{L}^1 = \{\mathcal{L}_1^1\} = \{\{V^2, V^3\}\}$.

If we assume that node V^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 Fig. 5c, we have three independent lists: $\mathcal{L}^1 = \{\mathcal{L}_1^1\} = \{\{V^4\}\}$, $\mathcal{L}^2 = \{\mathcal{L}_1^2\} = \{\{V^3\}\}$ and $\mathcal{L}^3 = \{\mathcal{L}_1^3\} = \{\{V^5\}\}$, while in the (YX) case, we continue to have one independent list but now with two sublists: $\mathcal{L}^1 = \{\mathcal{L}_1^1, \mathcal{L}_2^1\} = \{\{V^3, V^4\}, \{V^3, V^5\}\}$. 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.



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

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

Fig. 5 An illustration of the partitions in Example 4 is shown on the left-hand side. Numbers within cuboids refer to node indices. Cuboids in **a**–**c** depict the (XY) strategy for three consecutive nodes. Cuboids in **d**–**f** 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. (Color figure online)

4.2 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 convergence [25,31,32]. 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 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 Algorithm 3. 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.

The original B&S algorithm node selection procedure [22] 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). An ambiguity can arise in Step 1 when there are nodes with the same smallest outer lower bound value in different independent lists. We address this in ($\underline{F}l$)–($l\,\underline{f}$) so that B&S selects the list containing the comparatively less reduced 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.

Algorithm 3 Extended node selection

1: Select an independent list. There are two options for this step:
Option ($\underline{Fl}$): Find the independent list $\mathcal{L}^p$, $p \in P$, containing a node $V^k \in \mathcal{L}^p \cap \mathcal{L}$ with the lowest lower bound ($\underline{F}^k$). If several such nodes exist, select a node with the smallest level (l^k), and if several such nodes exist, take the node with the smallest index:

$$\mathcal{L}^p : V^k \in \mathcal{L}^p, \text{where } k = \min_{k' \in \mathcal{I}^l} \{k'\} \text{ with } \mathcal{I}^l = \left\{k' : k' = \arg\min_{k'' \in \mathcal{I}^F} \left\{l^{k''}\right\}\right\} \text{ and } \mathcal{I}^F = \left\{k' : k' = \arg\min_{k'' \in \mathcal{L}} \left\{\underline{F}^{k''}\right\}\right\}; \tag{$\underline{F}l$}$$

Option ($l\underline{F}$): Find $\mathcal{L}^p$ containing a node V^k with the smallest level (l^k). If several nodes with the smallest level exist, select a node with the lowest lower bound ($\underline{F}^k$), and if several such nodes exist, choose the node with the smallest index:

$$\mathcal{L}^p : V^k \in \mathcal{L}^p, \text{where } k = \min_{k' \in \mathcal{I}^F} \{k'\} \text{ with } \mathcal{I}^F = \left\{k' : k' = \arg\min_{k'' \in \mathcal{I}^l} \left\{\underline{F}^{k''}\right\}\right\} \text{ and } \mathcal{I}^l = \left\{k' : k' = \arg\min_{k'' \in \mathcal{L}} \left\{l^{k''}\right\}\right\}; \tag{$l\underline{F}$}$$

2: Select nodes for branching. Having found $\mathcal{L}^p$, there are two options here:
Option ($l\underline{f}$): Select a node $V^{k^*} \in \mathcal{L} \cap \mathcal{L}^p$ and a node $V^{k^*_{\text{In}}} \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p$, if $\mathcal{L}_{\text{In}} \neq \emptyset$, with the smallest levels l^{k^*} and $l^{k^*_{\text{In}}}$, respectively. If several nodes with the smallest level exist, select nodes with the lowest inner lower bounds, $\underline{f}^{k^*}$ and $\underline{f}^{k^*_{\text{In}}}$, respectively, and if several such nodes exist, choose the node with the smallest index:

$$k^* = \min_{k' \in \mathcal{I}^{\underline{f}}} \{k'\} \text{ with } \mathcal{I}^{\underline{f}} = \left\{k' : k' = \arg\min_{k'' \in \mathcal{I}^l} \left\{\underline{f}^i\right\}\right\} \text{ and } \mathcal{I}^l = \left\{k' : k' = \arg\min_{k'' \in \mathcal{L} \cap \mathcal{L}^p} \left\{l^{k''}\right\}\right\}$$
$$k^*_{\text{In}} = \min_{k' \in \mathcal{I}^{\underline{f}}} \{k'\} \text{ with } \mathcal{I}^{\underline{f}} = \left\{k' : k' = \arg\min_{k'' \in \mathcal{I}^l} \left\{\underline{f}^i\right\}\right\} \text{ and } \mathcal{I}^l = \left\{k' : k' = \arg\min_{k'' \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p} \left\{l^{k''}\right\}\right\}; \tag{$l\underline{f}$}$$

Option ($l\bar{f}$): Select nodes $V^{k^*} \in \mathcal{L} \cap \mathcal{L}^p$ and $V^{k^*_{\text{In}}} \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p$ with the smallest levels l^{k^*} and $l^{k^*_{\text{In}}}$, respectively. If several nodes with the lowest level exist, select the nodes with the smallest relaxed inner upper bounds, $\bar{f}^{k^*}$ and $\bar{f}^{k^*_{\text{In}}}$, respectively:

$$k^* = \min_{k' \in \mathcal{I}^{\bar{f}}} \{k'\} \text{ with } \mathcal{I}^{\bar{f}} = \left\{k' : k' = \arg\min_{k'' \in \mathcal{I}^l} \left\{\bar{f}^i\right\}\right\} \text{ and } \mathcal{I}^l = \left\{k' : k' = \arg\min_{k'' \in \mathcal{L} \cap \mathcal{L}^p} \left\{l^{k''}\right\}\right\}$$
$$k^*_{\text{In}} = \min_{k' \in \mathcal{I}^{\bar{f}}} \{k'\} \text{ with } \mathcal{I}^{\bar{f}} = \left\{k' : k' = \arg\min_{k'' \in \mathcal{I}^l} \left\{\bar{f}^i\right\}\right\} \text{ and } \mathcal{I}^l = \left\{k' : k' = \arg\min_{k'' \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p} \left\{l^{k''}\right\}\right\}. \tag{$l\bar{f}$}$$

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 5.

Example 5 (Illustration of Algorithm 3) Consider the partitioning of a two-dimensional space $X \times Y = [-1, 1] \times [-1, 1]$, shown in Fig. 6a with the corresponding tree shown in Fig. 6b. All leaf nodes belong to the active list, i.e., $\mathcal{L} = \{V^3, V^4, V^5\}$ 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 $\mathcal{X}_1 = [-1, 0]$, $\mathcal{L}^1 = \{\mathcal{L}^1_1\} = \{\{V^4, V^5\}\}$ and for $\mathcal{X}_2 = [0, -1]$, $\mathcal{L}^2 = \{\mathcal{L}^2_1\} = \{\{V^3\}\}$. In Step 1, using the ($l\underline{F}$) heuristic, B&S selects independent list $\mathcal{L}^2$ as it contains node V^3, the only active node at level 1. As node V^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

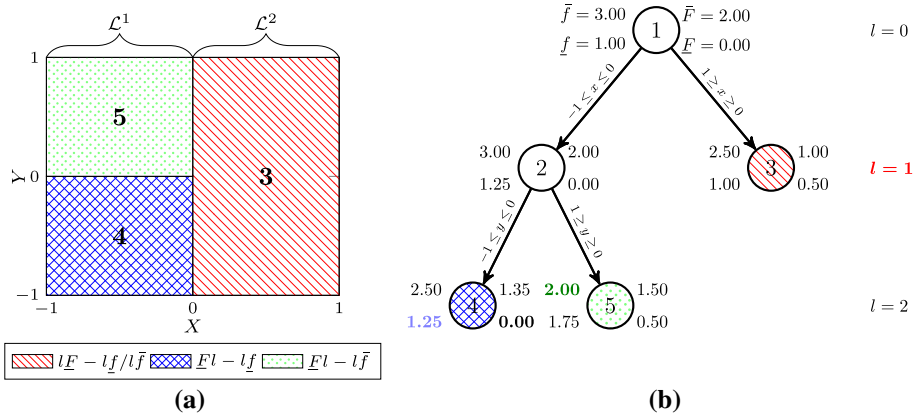


Fig. 6 Illustration of the nodes selected for branching in Example 5 when using different node selection heuristics from Algorithm 3. As before, numbers within rectangles and circles indicate node indices. Refer to the text for an interpretation of the figure

list $\mathcal{L}^1$ as it contains node V^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 V^4 and $(\underline{F}l)$–$(l\,\bar{f})$ to that of node V^5.

5 Summary of the revised and the original B&S

In this section, in Table 1, we provide a comparative summary of the options available in the two versions of the B&S algorithm (1): the revised B&S version presented in this paper, and the original B&S [22]. We note that it is not possible to reproduce exactly the configuration in [22] as some small ambiguities were present (as was explained in Sects. 4.1 and 4.2) in the algorithm described. We thus 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 [22]. In the revised algorithm, changes have been made to the node selection and branching rules used in Steps 4 and 5 of the algorithm, while the bounding schemes in Steps 9, 11, 12 have been revised, either by making changes to the bounding problems or by making changes to the timing of the solution of the bounding problems.

6 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 by using the default options shown in Table 1. We concentrate on the bilevel instance used in Example 1 in [23], where the behavior of the B&S algorithm at each step was described and explained in detail:

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

Table 1 Comparison of the available options for the mains steps in the revised and original B&S

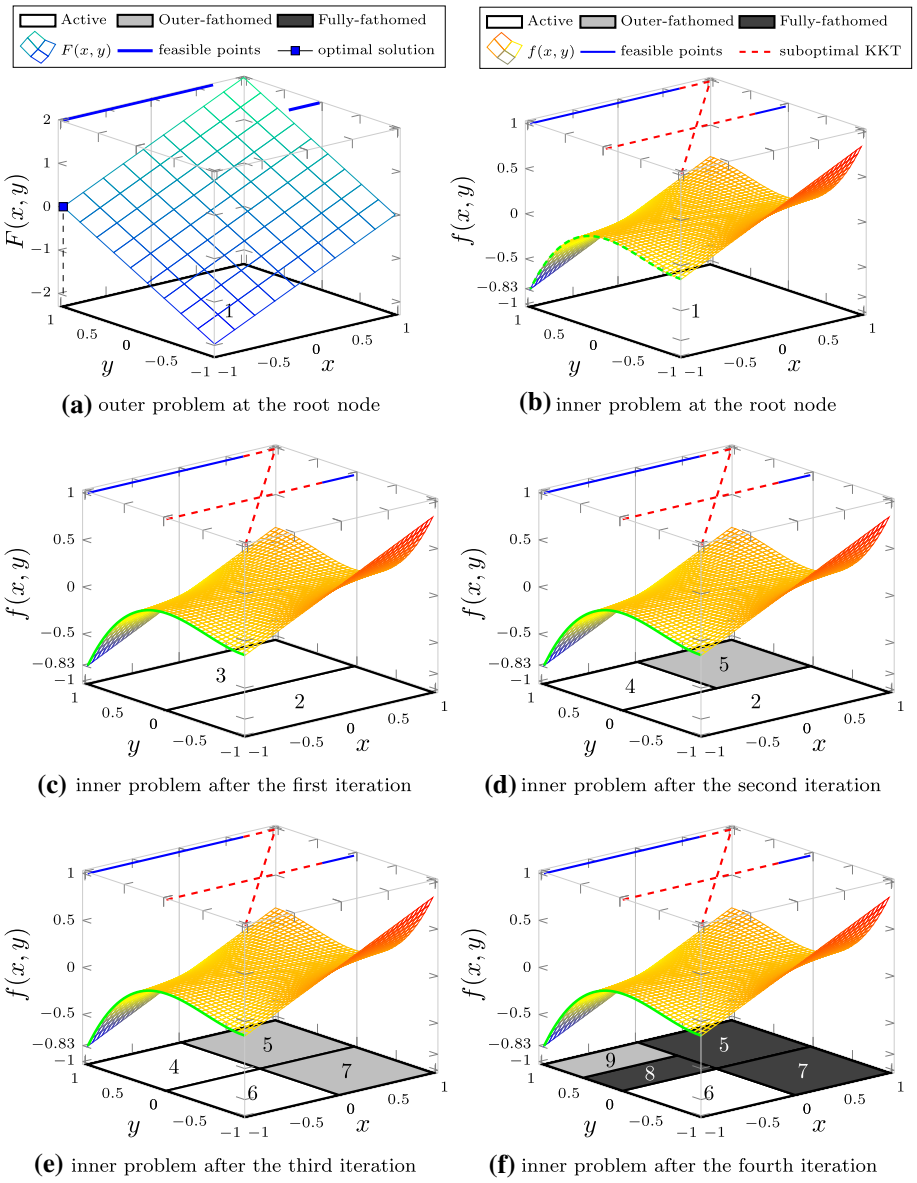
Algorithmic step	References	Revised B&S		Original B&S	
		All options	Default	All options	Default
Step 4: Node Selection	Sect. 4.2	($\underline{F}l$)–($l\underline{f}$), ($\underline{F}l$)–($l\bar{f}$), ($l\underline{F}$)–($l\underline{f}$), ($l\underline{F}$)–($l\bar{f}$)	($\underline{F}l$)–($l\underline{f}$)	($\underline{F}l$)–($l\underline{f}$)	($\underline{F}l$)–($l\underline{f}$)
Step 5: Branching rule	Sect. 4.1	(XY), (YX)	(YX)	(YX)	(YX)
Step 7: Inner lower bound	[22]	$f^{k,\mathrm{L}}$, $\check{f}^{k,\mathrm{L}}$	$f^{k,\mathrm{L}}$	$f^{k,\mathrm{L}}$, $\check{f}^{k,\mathrm{L}}$	$\check{f}^{k,\mathrm{L}}$
Step 8: Inner upper bound	Sect. 3.1	$f^{k,\mathrm{U}}$, $\bar{f}^{k}$	$\bar{f}^{k}$	$f^{k,\mathrm{U}}$, $\bar{f}^{k}$	$\bar{f}^{k}$
Step 9: Best inner upper bound	Sect. 3.2	$f^{\mathrm{UB},p}$, $f^{\mathrm{UB},k}$	$f^{\mathrm{UB},k}$	$f^{\mathrm{UB},p}$	$f^{\mathrm{UB},p}$
Step 10: Outer lower bound[a]	Sect. 3.3	$\underline{F}^{k}$	$\underline{F}^{k}$	$\underline{F}^{k}$	$\underline{F}^{k}$
Step 11: ISP scheme	Sects. 3.4.1 and 3.4.2	$\underline{w}^{j}(\bar{\mathbf{x}})$, $w^{j}(\bar{\mathbf{x}})$, $w(\bar{\mathbf{x}})$	$w(\bar{\mathbf{x}})$	$\underline{w}^{j}(\bar{\mathbf{x}})$	$\underline{w}^{j}(\bar{\mathbf{x}})$
Step 12: Outer upper bound	[27] & Sect. 3.4.2	$\bar{F}^{k'}(\bar{\mathbf{x}})$, $\bar{F}(\bar{\mathbf{x}})$	$\bar{F}(\bar{\mathbf{x}})$	$\bar{F}^{k'}(\bar{\mathbf{x}})$	$\bar{F}^{k'}(\bar{\mathbf{x}})$

All steps are with reference to Algorithm 1
[a] Uses best inner upper bound from Step 9

The inner problem satisfies the linear independence constraint qualification LICQ and the problem has a unique global 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 Fig. 7a, b). All bound values obtained using the B&S algorithm with the revised and original bounding schemes are summarized in Table 2a, b, respectively.

From the values obtained at the *root node* ($k = 1$), we observe that the inner lower bound $f^{k,\mathrm{L}}$ gives a tighter value than the relaxed inner lower bound $\check{f}^{k,\mathrm{L}}$, i.e., $f^{1,\mathrm{L}} = -0.83 > \check{f}^{1,\mathrm{L}} = -2.50$. The same applies for values obtained using the (ISP($\bar{\mathbf{x}}$, Y)) bounding scheme (introduced in Sect. 3.4.2), and the (RISP($\bar{\mathbf{x}}$, V^j)) scheme used in the original approach, 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 solution ($\bar{x}^k$, $\bar{y}^k$) of the outer lower bounding problem is identified. Using the revised approach, we maintain a set $\mathbb{X}$ [defined in (27)] 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 upper bounding problem from Sect. 3.4.2, 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 $\underline{w}(-1.0) = -1.50$, no feasible solution is found at this step.

Next, at the *first iteration*, both B&S versions select node V^1 (based on Algorithm 3) and bisect on the selected branching variable y (based on Algorithm 2), generating two new nodes (V^2 and V^3), as shown in Fig. 7c. After branching, both Branch-and-Sandwich trees consist of one list with one sublist: $\mathcal{L}^1 = \{\mathcal{L}_1^1\} = \{\{V^2, V^3\}\}$. 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 lower bounding problems, $(\bar{x}^2, \bar{y}^2) = (-1.0, -1.0)$ and $(\bar{x}^3, \bar{y}^3) = (-1.0, 0.0)$.



(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. 7 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). Numbers within rectangles indicate node indices. 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. (Color figure online)

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($\bar{\mathbf{x}}$, V^j)) problems and then one (UB($\bar{\mathbf{x}}$, $V^{k'}$)) 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.

Table 2 Detailed comparison of the B&S algorithm on `mb_1_1_08` bilevel problem

k	$f^{k,\mathrm{L}}$	$\bar{f}^k$	$f^{\mathrm{UB},k}$	$\underline{F}^k$	$(\bar{x}, \bar{y})$	$w(\bar{x})$	$\bar{F}(\bar{x})$
(a) Using revised bounding schemes							
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
k	$\check{f}^{k,\mathrm{L}}$	$\bar{f}^k$	$f^{\mathrm{UB},p}$	$\underline{F}^k$	$(\bar{x}, \bar{y})$	$\underline{w}^j(\bar{x})$	$\bar{F}^{k'}(\bar{x})$
(b) Using original bounding schemes							
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

At the *second iteration*, in both versions of B&S node V^3 is selected and bisected on variable x, resulting in two new nodes (V^4 and V^5), as shown in Fig. 7d. The two Branch-and-Sandwich trees then consist of one list with two sublists: $\mathcal{L}^1 = \{\mathcal{L}_1^1, \mathcal{L}_2^1\} = \{\{V^2, V^4\}, \{V^2, V^5\}\}$. Note that using results from Theorem 2 and Remark 3, the original B&S approach can be improved by postponing the redundant computation of the relaxed inner upper bound $\bar{f}^k$ and of the best inner upper bound $f^{\mathrm{UB},p}$ until the independence condition is satisfied. However this strategy is not followed for the revised B&S scheme, as stated in Remark 5. The biggest difference in algorithmic behavior 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 $\underline{w}^4(\bar{x}^4)$ is the same as $\underline{w}^3(\bar{x}^3)$ as both these RISP problems are solved over the same $Y = [0, 1]$ and with the same $\bar{x} = \bar{x}^3 = \bar{x}^4$.

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, the original B&S checks the outer-fathoming rule (Sect. 4.6 in [22]) for node V^5, leading to it being moved from the list of open nodes to $\mathcal{L}_{\text{In}}$. 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. Using the 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 solutions of the lower bounding problem: $\mathbb{X} = \mathbb{X} \cup \{0.0\} = \{0.0, 1.0\}$ and perform the revised outer upper bounding procedure for this solution point.

At the *third iteration*, both versions of B&S select node V^2 and bisect on variable x, generating two new nodes (V^6 and V^7), as shown in Fig. 7e. After branching, the independence condition is satisfied so that the inner upper and the best inner upper bound need to be evaluated (Remark 3) and the two Branch-and-Sandwich trees therefore consist of two lists with one sublist each: $\mathcal{L}^1 = \{\mathcal{L}^1_1\} = \{\{V^4, V^6\}\}$, $\mathcal{L}^2 = \{\mathcal{L}^2_1\} = \{\{V^5, V^7\}\}$.

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 V^2 is no longer in the Branch-and-Sandwich tree; therefore the original approach fails to recognize this unnecessary evaluation. In both approaches, we first perform outer-fathoming of node V^7; this is followed by the deletion of list $\mathcal{L}^2$ and the full-fathoming of nodes V^5 and V^7.

At the *fourth iteration*, both versions of B&S select node V^4 and bisect on $y = 0.5$, generating two new nodes (V^8 and V^9) shown in Fig. 7f. After this branching step, the two Branch-and-Sandwich trees consist of $\mathcal{L}^1 = \{\mathcal{L}^1_1\} = \{\{V^6, V^8, V^9\}\}$. At this iteration, the best inner upper bound using both schemes, i.e., $f^{\text{UB},p=1} = f^{\text{UB},k} = -0.33$ for each $k \in \{6, 8, 9\}$. This leads to the full-fathoming of node V^8, followed by outer-fathoming of node V^9. Finally, using the revised scheme we can fully-fathom node V^6 as $f^{6,\text{L}} = -0.17$, delete list $\mathcal{L}^1$, and terminate the B&S algorithm successfully. However, using the original approach we cannot fully-fathom node V^6 as $\breve{f}^{6,\text{L}} = -0.41$ is lower than the current best inner upper bound. This leads to an additional B&S iteration and the exploration of new nodes, V^{10}, V^{11}, V^{12}, V^{13}. A further aspect worthy of note is that the timing of fathoming and list deletion tests can help to reduce the number of relaxed inner lower bounding problems that need to be solved 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 lower bounding problem.

7 Computational comparison of the revised B&S versus the original B&S

In this section, we perform an analysis of the computational impact of the proposed extensions, modifications and improvements. We have extended the `BASBL` solver [30], an initial implementation of the original B&S approach. The revised bounding schemes and the proposed heuristics for the node selection and branching have been implemented within `BASBL` and compared against the initial version, which is based on the original algorithm. To test

Table 3 The number of iterations ($Iter$), total number of solved subproblems (#P) and total CPU time ($t(s)$) using revised and original B&S bounding schemes

Label	Revised B&S			Original B&S		
	$Iter$	#P	$t(s)$	$Iter$	#P	$t(s)$
mb_1_1_05	5	41	2.41	5	47	2.35
mb_1_1_06	60	577	60.43	130	3627	655.05
mb_1_1_06v	5	47	2.98	7	99	5.20
mb_1_1_07	6	45	2.28	7	69	3.27
mb_1_1_08	4	30	1.48	4	37	1.79
mb_1_1_09	6	51	2.68	7	82	3.84
mb_1_1_10	5	41	2.32	5	48	2.46
mb_1_1_11	28	231	13.19	36	567	43.70
mb_1_1_13	7	53	2.74	7	65	3.33
mb_1_1_14	3	27	1.55	3	31	1.88
Average	12.90	114.30	9.21	21.10	467.20	72.32

the revised implementation, we use the same ten challenging bilevel instances from [30], originally introduced in [26], with some modifications introduced in [23]. All test instances are solved using the BASBL solver with the revised and original bounding schemes. BASBL subproblems are solved globally with BARON version 17.4.1 [34,40], and accessed using GAMS version 24.8.4 [13] through system calls. Absolute and relative termination tolerances for BARON are set to 10^{-5} for all problems. The same outer and inner objective tolerances as in [30] were used, i.e., $\varepsilon_F = 10^{-3}$ and $\varepsilon_f = 10^{-5}$ are set for all problem instances except for mb_1_1_06 and mb_1_1_11 where $\varepsilon_F = 0.1$ is used. All experiments are performed on the same computer as in [30], i.e., a 64-bit Intel(R) Xeon(R) CPU E5-2667 v2 @ 3.30 GHz processor running Linux OS. The closest combination of branching and node selection rules to the original B&S [23] was used, i.e., (XY) for branching and ($\underline{F}l$)–($l\underline{f}$) for node selection.

We begin by investigating the impact of the new bounding schemes on the performance of the revised B&S algorithm. The number of iterations ($Iter$), total number of inner and outer (sub)problems (#P) and wall-clock CPU time ($t(s)$) obtained are reported in Table 3. On average, using the new bounding schemes, the total number of iterations is 39% smaller, the number of solved subproblems decreases by 75% and the execution time is reduced by 87%. This shows the significant benefits that can be achieved through better bounding.

Next, we test the impact of different branching and node selection rules on the performance of the revised B&S algorithm. We report results in Table 4 using three different combinations of branching and node selection rules. Note that we do not include node selection strategies with the ($l\underline{F}$) rule, as we find that the choice of ($\underline{F}l$) or ($l\underline{F}$) does not have a significant effect on the performance of revised B&S for the test problems used. The combination (YX)-($\underline{F}l$)–($l\underline{f}$) is closest to the original B&S [23]. The other combinations are based on changing one of the options at a time.

Comparing the different branching variable selection rules ((XY) vs. (YX)), we can observe that branching preferentially on **x** variables ((XY) strategy) reduces the total number of solved (sub)problems (#P) by around 11% on average. This is especially evident on the two most challenging problems, mb_1_1_06 and mb_1_1_11, where the (XY) strategy reduces

Table 4 Revised B&S performance based on different variants of the branching and node selection rules

Label	(YX)–($\underline{Fl}$)–($l\underline{f}$)			(YX)–($\underline{Fl}$)–($l\bar{f}$)			(XY)–($\underline{Fl}$)–($l\underline{f}$)		
	Iter	#P	$t(s)$	*Iter*	#P	$t(s)$	*Iter*	#P	$t(s)$
mb_1_1_05	5	41	2.41	5	41	2.52	8	61	3.41
mb_1_1_06	60	577	60.43	62	597	62.10	49	452	54.15
mb_1_1_06v	5	47	2.98	5	47	3.22	4	37	2.11
mb_1_1_07	6	45	2.28	7	49	2.57	10	66	3.54
mb_1_1_08	4	30	1.48	4	30	1.62	6	41	2.02
mb_1_1_09	6	51	2.68	5	45	2.31	4	31	1.48
mb_1_1_10	5	41	2.32	5	41	2.50	8	57	3.36
mb_1_1_11	28	231	13.19	27	217	11.74	22	177	9.42
mb_1_1_13	7	53	2.74	7	53	3.04	11	83	4.48
mb_1_1_14	3	27	1.55	3	27	1.59	2	17	1.19
Average	12.90	114.30	9.21	13.00	114.70	9.32	12.40	102.20	8.52

the total number of solved (sub)problems by 22% on average. While the (XY) strategy gives the best average performance, it is not optimal for all tested problems.

Finally, only a small difference (of up to around 5%) is observed when choosing one node selection strategy over the other (($l\underline{f}$) vs. ($l\bar{f}$)). We only show the results obtained with (XY)–($\underline{Fl}$)–($l\underline{f}$), as similar results are found with the (XY) branching strategy.

8 Conclusions

We have presented algorithmic improvements and extensions to the recently proposed bilevel deterministic global optimization algorithm, Branch-and-Sandwich based on a combination of new theoretical results and heuristic rules. Our algorithmic extensions include changes to the bounding schemes, including an alternative way to obtain tighter the best inner upper bounds, a simpler and significantly faster outer upper bounding scheme as well as alternative choices for branching and node selection. Detailed algorithmic and computational comparisons have been used to demonstrate the beneficial impact of the proposed extensions and modifications on the number of iterations and solution performance for a set of challenging problems. In particular, the number of nonconvex subproblems to be solved and the total execution time were found to be greatly reduced. Numerical comparisons revealed that a notable performance improvement may be achieved by selecting specific node selection and branching rules.

Finally, the detailed description of the fully-fledged `BASBL` implementation and the application to the larger problems is considered in a recent submission [29].

Acknowledgements We gratefully acknowledge funding from the Leverhulme Trust through the Philip Leverhulme Prize and from the EPSRC through a Leadership Fellowship (`EP/J003840/1`).

Data access statement Data underlying this article can be accessed on Zenodo at https://doi.org/10.5281/zenodo.3266835, and used under the Creative Commons Attribution license.

References

1. Adjiman, C.S., Androulakis, I.P., Floudas, C.A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs—II. Implementation and computational results. Comput. Chem. Eng. **22**(9), 1159–1179 (1998). https://doi.org/10.1016/S0098-1354(98)00218-X
2. Adjiman, C.S., Dallwig, S., Floudas, C.A., Neumaier, A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs—I. Theoretical advances. Comput. Chem. Eng. **22**(9), 1137–1158 (1998). https://doi.org/10.1016/S0098-1354(98)00027-1
3. Angelo, J.S., Barbosa, H.J.: A study on the use of heuristics to solve a bilevel programming problem. Int. Trans. Oper. Res. **22**(5), 861–882 (2015). https://doi.org/10.1111/itor.12153
4. Bard, J.F.: Practical bilevel optimization. In: Nonconvex Optimization and Its Applications, vol. 30. Springer, New York (1998). https://doi.org/10.1007/978-1-4757-2836-1
5. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
6. Bracken, J., McGill, J.T.: Mathematical programs with optimization problems in the constraints. Oper. Res. **21**(1), 37–44 (1973). https://doi.org/10.1287/opre.21.1.37
7. Calvete, H.I., Galé, C.: The bilevel linear/linear fractional programming problem. Eur. J. Oper. Res. **114**(1), 188–197 (1999). https://doi.org/10.1016/S0377-2217(98)00078-2
8. Calvete, H.I., Galé, C., Mateo, P.M.: A new approach for solving linear bilevel problems using genetic algorithms. Eur. J. Oper. Res. **188**(1), 14–28 (2008). https://doi.org/10.1016/j.ejor.2007.03.034
9. Casas-Ramírez, M.S., Camacho-Vallejo, J.F., Martínez-Salazar, I.A.: Approximating solutions to a bilevel capacitated facility location problem with customer's patronization toward a list of preferences. Appl. Math. Comput. **319**, 369–386 (2018). https://doi.org/10.1016/j.amc.2017.03.051
10. Cecchini, M., Ecker, J., Kupferschmid, M., Leitch, R.: Solving nonlinear principal-agent problems using bilevel programming. Eur. J. Oper. Res. **230**(2), 364–373 (2013). https://doi.org/10.1016/j.ejor.2013.04.014
11. Chen, Q., Paulavičius, R., Adjiman, C.S., García-Muñoz, S.: An optimization framework to combine operable space maximization with design of experiments. AIChE J. **64**(11), 3944–3957 (2018). https://doi.org/10.1002/aic.16214
12. Colson, B., Marcotte, P., Savard, G.: An overview of bilevel optimization. Ann. Oper. Res. **153**(1), 235–256 (2007). https://doi.org/10.1007/s10479-007-0176-2
13. Corporation, G.D.: General Algebraic Modeling System (GAMS) Release 24.8.4. http://www.gams.com/ (2017). Accessed 20 Sept 2017
14. Dempe, S.: Foundations of bilevel programming. In: Nonconvex Optimization and Its Applications, vol. 61. Kluwer, Boston (2002). https://doi.org/10.1007/b101970
15. Dempe, S.: Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. Optimization **52**(3), 333–359 (2003). https://doi.org/10.1080/0233193031000149894
16. Dempe, S., Kalashnikov, V., Pérez-Valdés, G.A., Kalashnykova, N.: Bilevel programming problems. In: Energy Systems. Springer, Berlin (2015). https://doi.org/10.1007/978-3-662-45827-3
17. Djelassi, H., Glass, M., Mitsos, A.: Discretization-based algorithms for generalized semi-infinite and bilevel programs with coupling equality constraints. J. Glob. Optim. (2019). https://doi.org/10.1007/s10898-019-00764-3
18. Falk, J.E., Hoffman, K.: A nonconvex max–min problem. Nav. Res. Logist. Q. **24**(3), 441–450 (1977). https://doi.org/10.1002/nav.3800240307
19. Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: A new general-purpose algorithm for mixed-integer bilevel linear programs. Oper. Res. **65**(6), 1615–1637 (2017). https://doi.org/10.1287/opre.2017.1650
20. Fortuny-Amat, J., McCarl, B.: A representation and economic interpretation of a two-level programming problem. J. Oper. Res. Soc. (1981). https://doi.org/10.1057/jors.1981.156
21. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches, 3rd edn. Springer, Berlin (1996). https://doi.org/10.1007/978-3-662-03199-5

22. Kleniati, P.M., Adjiman, C.S.: Branch-and-Sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part I: theoretical development. J. Glob. Optim. **60**(3), 425–458 (2014). https://doi.org/10.1007/s10898-013-0121-7
23. Kleniati, P.M., Adjiman, C.S.: Branch-and-Sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part II: convergence analysis and numerical results. J. Glob. Optim. **60**(3), 459–481 (2014). https://doi.org/10.1007/s10898-013-0120-8
24. Kleniati, P.M., Adjiman, C.S.: A generalization of the Branch-and-Sandwich algorithm: from continuous to mixed-integer nonlinear bilevel problems. Comput. Chem. Eng. **72**, 373–386 (2014). https://doi.org/10.1016/j.compchemeng.2014.06.004
25. Linderoth, J.T., Savelsbergh, M.W.: A computational study of search strategies for mixed integer programming. INFORMS J. Comput. **11**(2), 173–187 (1999). https://doi.org/10.1287/ijoc.11.2.173
26. Mitsos, A., Barton, P.I.: A Test Set for Bilevel Programs. http://www.researchgate.net/publication/228455291 (2007). (Last updated 19 Sept. 2007)
27. Mitsos, A., Lemonidis, P., Barton, P.I.: Global solution of bilevel programs with a nonconvex inner program. J. Glob. Optim. **42**(4), 475–513 (2008). https://doi.org/10.1007/s10898-007-9260-z
28. Moore, J.T., Bard, J.F.: The mixed integer linear bilevel programming problem. Oper. Res. **38**(5), 911–921 (1990). https://doi.org/10.1287/opre.38.5.911
29. Paulavičius, R., Gao, J., Kleniati, P.M., Adjiman, C.S.: BASBL: Branch-And-Sandwich BiLevel solver: implementation and computational study with the BASBLib test set. Comput. Chem. Eng. (2020). https://doi.org/10.1016/j.compchemeng.2019.106609
30. Paulavičius, R., Kleniati, P.M., Adjiman, C.S.: Global optimization of nonconvex bilevel problems: implementation and computational study of the Branch-and-Sandwich algorithm. In: Kravanja, Z., Bogataj, M. (eds.) 26th European Symposium on Computer Aided Process Engineering, Computer Aided Chemical Engineering, vol. 38, pp. 1977–1982. Elsevier, Amsterdam (2016). https://doi.org/10.1016/B978-0-444-63428-3.50334-9
31. Paulavičius, R., Žilinskas, J.: Simplicial Global Optimization. Springer Briefs in Optimization. Springer, New York (2014). https://doi.org/10.1007/978-1-4614-9093-7
32. Paulavičius, R., Žilinskas, J., Grothey, A.: Investigation of selection strategies in branch and bound algorithm with simplicial partitions and combination of Lipschitz bounds. Optim. Lett. **4**(2), 173–183 (2010). https://doi.org/10.1007/s11590-009-0156-3
33. Sahinidis, N.V.: BARON 14.4.0: Global Optimization of Mixed-Integer Nonlinear Programs, User's Manual (2014)
34. Sahinidis, N.V.: BARON 17.4.1: global optimization of mixed-integer nonlinear programs, User's Manual (2017)
35. Shimizu, K., Ishizuka, Y., Bard, J.F.: Nondifferentiable and Two-Level Mathematical Programming, vol. 102. Kluwer, Boston (1997). https://doi.org/10.1016/S0377-2217(97)00228-2
36. Sinha, A., Malo, P., Deb, K.: Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. Eur. J. Oper. Res. **257**(2), 395–411 (2017). https://doi.org/10.1016/j.ejor.2016.08.027
37. Stein, O., Still, G.: On generalized semi-infinite optimization and bilevel optimization. Eur. J. Oper. Res. **142**(3), 444–462 (2002). https://doi.org/10.1016/S0377-2217(01)00307-1
38. Still, G.: Solving generalized semi-infinite programs by reduction to simpler problems. Optimization **53**(1), 19–38 (2004). https://doi.org/10.1080/02331930410001661190
39. Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: theory, algorithms, software, and applications. In: Nonconvex Optimization and Its Applications. Kluwer, Boston (2002). https://doi.org/10.1007/978-1-4757-3532-1
40. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**(2), 225–249 (2005). https://doi.org/10.1007/s10107-005-0581-8
41. Tsoukalas, A., Rustem, B., Pistikopoulos, E.N.: A global optimization algorithm for generalized semi-infinite, continuous minimax with coupled constraints and bi-level problems. J. Glob. Optim. **44**(2), 235–250 (2008). https://doi.org/10.1007/s10898-008-9321-y
42. Tuy, H.: Convex Analysis and Global Optimization, vol. 22. Springer, Berlin (2013). https://doi.org/10.1007/978-1-4757-2809-5
43. Vicente, L.N., Calamai, P.H.: Bilevel and multilevel programming: a bibliography review. J. Glob. Optim. **5**(3), 291–306 (1994). https://doi.org/10.1007/BF01096458