Exploiting Overlap Information in Chance-constrained Program with Random Right-hand Side

Wei Lv[a], Wei-Kun Chen[a], Yu-Hong Dai[b,c], and Xiao-Jiao Tong[d]

[a] *School of Mathematics and Statistics, Beijing Institute of Technology, Beijing 100081, China, {lvwei,chenweikun}@bit.edu.cn*

[b] *Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China*

[c] *School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China, dyh@lsec.cc.ac.cn*

[d] *School of Mathematics and Statistics, Hunan First Normal University, Changsha 410205, China, dysftxj@hnfnu.edu.cn*

June 15, 2024

Abstract

We consider the chance-constrained program (CCP) with random right-hand side under a finite discrete distribution. It is known that the standard mixed integer linear programming (MILP) reformulation of the CCP is generally difficult to solve by general-purpose solvers as the branch-and-cut search trees are enormously large, partly due to the weak linear programming relaxation. In this paper, we identify another reason for this phenomenon: the intersection of the feasible regions of the subproblems in the search tree could be nonempty, leading to a wasteful duplication of effort in exploring the uninteresting overlap in the search tree. To address the newly identified challenge and enhance the capability of the MILP-based approach in solving CCPs, we first show that the overlap in the search tree can be completely removed by a family of valid nonlinear if-then constraints, and then propose two practical approaches to tackle the highly nonlinear if-then constraints. In particular, we use the concept of dominance relations between different scenarios of the random variables, and propose a novel branching, called dominance-based branching, which is able to create a valid partition of the problem with a much smaller overlap than the classic variable branching. Moreover, we develop overlap-oriented node pruning and variable fixing techniques, applied at each node of the search tree, to remove more overlaps in the search tree. Computational results demonstrate the effectiveness of the proposed dominance-based branching and overlap-oriented node pruning and variable fixing techniques in reducing the search tree size and improving the overall solution efficiency.

Keywords: Stochastic programming · Integer programming · Chance constraints · Branch-and-cut algorithms · Overlap

1 Introduction

Consider the chance-constrained program (CCP) with random right-hand side:

$$\min\left\{c^\top x\,:\,\mathbb{P}\{Tx \geq \xi\} \geq 1-\epsilon,\ x\in\mathcal{X}\right\}, \tag{CCP}$$

where $\mathcal{X} \subseteq \mathbb{R}^d$ is a polyhedron, T is an $m \times d$ matrix, c is a d-dimensional cost vector, ξ is an m-dimensional *random* vector, and $\epsilon \in (0,1)$ is a confidence parameter chosen by the decision maker, typically near zero. (CCP) is a powerful paradigm to model risk-averse decision-making problems, and has been seen in or served as a building block in many industrial applications in areas such as appointment scheduling [26], energy [21, 50, 57], finance [18], healthcare [10], facility location [11, 16, 52], supply chain logistic [12, 35, 41], and telecommunication [19]. We refer to the surveys of Prékopa [48] and Küçükyavuz and Jiang [29] and the references therein for more applications of (CCP).

(CCP) was introduced by Charnes and Cooper [14], Miller and Wagner [40], Prékopa [45], and Prékopa [46], and has been extensively investigated in the literature [29, 48]. For (CCP) with discrete random variables, Sen [53] derived a relaxation problem using disjunctive programming techniques. Dentcheva et al. [19] used the so-called $(1-\epsilon)$-efficient points [47] to develop various reformulations for (CCP) with discrete random variables, and derived lower and upper bounds for the optimal value of the problem. Using a partial enumeration of the $(1-\epsilon)$-efficient points, Beraldi and Ruszczyński [12] proposed a specialized branch-and-bound algorithm for (CCP) that is based on the relaxation in [19] and guaranteed to find an optimal solution. Lejeune [33] presented a pattern-based solution method for obtaining a global solution of (CCP), which requires the enumeration of the so-called $(1-\epsilon)$-sufficient and -insufficient points [34] and solving an equivalent mixed integer linear programming (MILP) reformulation. Cheon et al. [17] considered the case with a finite discrete distribution of the random vector ξ, that is, ξ takes values $\xi^1, \xi^2, \ldots, \xi^n$ with $\mathbb{P}\{\xi = \xi^i\} = p_i \geq 0$ for $i \in [n] := \{1, 2, \ldots, n\}$ and $\sum_{i=1}^n p_i = 1$. Cheon et al. [17] provided a specialized branch-and-cut (B&C), called branch-reduce-cut, algorithm that finds a global solution of (CCP) by successively partitioning the nonconvex feasible region and using bounding techniques.

The solution algorithms of interest in the paper are the MILP reformulation based approaches in [1, 5, 28, 30, 38, 51, 56, 59], which can be easily adopted by practitioners. These approaches also rely on the assumption that ξ has a finite discrete distribution $\xi^1, \xi^2, \ldots, \xi^n$, and require to solve an MILP reformulation of (CCP), detailed as follows. Without loss of generality, we assume $\xi^i \geq 0$ for all $i \in [n]$ (by applying the transformation in [29, 38] if needed). By introducing for each $i \in [n]$, a binary variable z_i, where $z_i = 0$ guarantees $v = Tx \geq \xi^i$, (CCP) can then be reformulated as the following MILP formulation [51]:

$$\min \left\{ c^\top x \, : \, (1) - (4) \right\}, \tag{MILP}$$

where

$$Tx = v, \; x \in \mathcal{X}, \tag{1}$$

$$v \geq \xi^i (1 - z_i), \; \forall \, i \in [n], \tag{2}$$

$$\sum_{i=1}^n p_i z_i \leq \epsilon, \tag{3}$$

$$v \in \mathbb{R}^m_+, \; z \in \{0,1\}^n. \tag{4}$$

Constraints (2) are referred to as big-M constraints where the big-M coefficients are given by ξ^i_k, $k \in [m]$. It is worthy noting that the assumption on the finite discrete distribution of ξ is not restrictive, as for arbitrary distributions of ξ, we can use sample average approximation or importance sampling technique to obtain an approximation problem that satisfies this assumption; see [6, 8, 37, 42].

It is well-known that due to the presence of the big-M constraints in (2), the linear programming (LP) relaxation of formulation (MILP) is generally very weak, making it difficult to use state-of-the-art MILP solvers to tackle formulation (MILP) directly. To bypass this difficulty, various approaches have been developed in the literature [1, 5, 28, 30, 38, 56, 59]. Specifically, using the lower bounds of variables v (also known as quantile information [20, 32]), Luedtke et al. [38] developed a strengthened version of formulation (MILP) that provides a tighter LP relaxation and has less constraints. By investigating the single mixing set with a knapsack constraint, an important substructure of (MILP) defined by constraints (2) for a fixed $k \in [m]$ and (3), Luedtke et al. [38] employed the mixing inequalities [7, 25] to strengthen the LP relaxation of formulation (MILP). Küçükyavuz [28], Abdi and Fukasawa [1], and Zhao et al. [59] further investigated the polyhedral structure of the single mixing set and proposed new families of inequalities to strengthen the LP relaxation of formulation (MILP). Küçükyavuz [28], Zhao et al. [59], and Kılınç-Karzan et al. [30] investigated the joint mixing set with a knapsack constraint, that considers all constraints in (2) and (3), and developed various aggregated mixing inequalities. Computational evidences in [1, 13, 28, 38, 59] have demonstrated the effectiveness of the mentioned inequalities in enhancing the capability of employing MILP solvers in solving formulation (MILP). Luedtke et al. [38], Vielma et al. [56], Küçükyavuz [28], and Ahmed et al. [5] developed various extended formulations that provide tighter LP relaxations than the direct LP relaxation of formulation (MILP).

In another line of research, Ruszczyński [51] used the concept of dominance relations between scenarios and developed a class of valid inequalities, called dominance inequalities. While it was stated in [17] that the incorporation of dominance inequalities can improve the performance of employing MILP solvers in solving formulation (MILP) (see [9, 27, 54, 55] for similar discussions in other contexts), it was unknown until now how these inequalities improve the performance of MILP solvers. As a byproduct of analysis, this paper closes this research gap by showing that adding the dominance inequalities into formulation (MILP) cannot improve the LP relaxation but can improve the performance of MILP solvers by saving the computational efforts spent in exploring the uninteresting part of the search tree; see Section 3.2.

The goal of this paper is to explore new integer programming techniques to further improve the computational performance of the MILP-based approach to solving (CCP).

1.1 Contributions and outline

Unlike existing approaches [1, 5, 28, 30, 38, 56, 59] that mainly focus on improving the LP relaxation of formulation (MILP), we go for a different direction by identifying another drawback of solving formulation (MILP) using B&C solvers; that is, the intersection of the feasible regions of the subproblems (after removing the common fixed variables) in the B&C search tree could be nonempty, leading to a wasteful duplication of effort in exploring the uninteresting overlap in the search tree. To overcome the drawback and enhance the capability of the MILP-based approach in solving (CCP), we first show that the overlap can be completely removed by adding a family of valid nonlinear if-then constraints into formulation (MILP), and then propose two practically tractable approaches to tackle the highly nonlinear if-then constraints. More specifically,

- By using the concept of dominance relations between different scenarios of the random variables [51], we propose a novel branching, called dominance-based branching, which is able to create a valid partition of the current problem with a much smaller overlap than the classic variable branching. We show that applying the proposed dominance-based branching to formulation (MILP) is equivalent to applying the classic variable branching to formulation

(MILP) with the dominance inequalities in [51] in terms of fixing the same variables and sharing the same LP relaxation bound at two nodes with the same branching variables. We also employ a preprocessing technique to derive more dominance relations between the scenarios of the random variables, thereby leading to a much more effective dominance-based branching.

- By considering the joint mixing set with a knapsack constraint and the newly proposed if-then constraints, we develop node pruning and variable fixing, applied at each node of the search tree, to remove more overlaps in the search tree. Although implementing the proposed overlap-oriented node pruning and variable fixing techniques is proved to be strongly NP-hard, we are able to develop an approximation algorithm that is competitive with the exact algorithm in terms of reducing the tree size while still enjoying an efficient polynomial-time worst-case complexity.

We embed the proposed dominance-based branching and overlap-oriented node pruning and variable fixing techniques into the state-of-the-art open source MILP solver SCIP, and apply the resultant approach to solve the chance-constrained versions of the resource planning (CCRP) problem [26, 36], multiperiod power planning (CCMPP) problem [21], and lot-sizing (CCLS) problem [12]. Extensive computational results show that the two proposed approaches can significantly reduce the B&C search tree size and substantially enhance the capability of SCIP in solving CCPs.

This paper is organized as follows. Section 1.2 presents the notations and assumptions used in this paper. Section 2 shows that applying the classic B&C algorithm to formulation (MILP) leads to overlaps in the search tree, and presents a family of valid nonlinear if-then constraints to remove the overlaps. Section 3 develops the dominance-based branching and analyzes its relation to the direct use of the dominance inequalities in [51]. Section 4 derives overlap-oriented node pruning and variable fixing techniques and provides the complexity and algorithmic design for the implementation. Section 5 reports the computational results. Finally, Section 6 draws the conclusion.

1.2 Notations and assumptions

For a nonnegative integer n, let $[n] = \{1, 2, \ldots, n\}$ where $[n] = \varnothing$ if $n = 0$. Let $\mathbf{0}$ denote the all zeros vector with an appropriate dimension. For two vectors v^1, v^2 of dimension m, $v^1 \geq v^2$ denotes that $v^1_k \geq v^2_k$ holds for all $k \in [m]$; and $v^1 \ngeq v^2$ denotes that $v^1_k < v^2_k$ holds for at least one $k \in [m]$. For vectors $\xi^1, \xi^2, \ldots, \xi^n$ and a subset $\mathcal{S} \subseteq [n]$, we denote $\xi^{\mathcal{S}} = \max_{i \in \mathcal{S}}\{\xi^i\}$, where the max is taken component-wise and $\xi^{\mathcal{S}} = \mathbf{0}$ if $\mathcal{S} = \varnothing$. We follow [39, 43, 44] to characterize the current node of the search tree by $(\mathcal{B}_0, \mathcal{B}_1)$ or $(\mathcal{N}_0, \mathcal{N}_1)$, where $\mathcal{B}_0$ and $\mathcal{B}_1$ are the index sets of variables z that have been branched on 0 and 1, respectively; and $\mathcal{N}_0$ and $\mathcal{N}_1$ are the index sets of variables z that have been fixed to 0 and 1, respectively, by variable branching or other methods like reduced cost fixing or the coming overlap-oriented variable fixing.

For simplicity, we assume that variables x in (CCP) are all continuous variables. However, our proposed approach can be applied to the general case, in which some of the variables x may require to be integers, as long as the branching is allowed to be performed on variables z_i, $i \in [n]$, at a node.

2 Overlaps arising in solving formulation (MILP)

In this section, we first illustrate the weakness of applying the B&C algorithm with the classic variable branching to solve formulation (MILP); that is, the intersection of the feasible regions of the subproblems (after removing the common fixed variables) in the B&C search tree could be nonempty, leading to a wasteful duplication of effort in exploring the uninteresting overlap in the search tree. Then we strengthen formulation (MILP) by presenting a family of valid nonlinear *if-then* constraints to remove the overlaps during the B&C process.

2.1 Overlaps in the search tree

We first apply the classic variable branching to formulation (MILP). Without loss of generality, we suppose that variable z_j, $j \in [n]$, is branched on at the root node, and the two branches are

$$O^{\mathrm{L}} = \min \left\{ c^\top x \,:\, (1) - (4),\ z_j = 0 \right\}, \tag{5}$$
$$O^{\mathrm{R}} = \min \left\{ c^\top x \,:\, (1) - (4),\ z_j = 1 \right\}. \tag{6}$$

By removing the common fixed variable z_j from problems (5) and (6), we obtain

$$O^{\mathrm{L}} = \min \left\{ c^\top x \,:\, (x, v, z) \in \mathcal{C},\ v \geq \xi^j,\ \sum_{i \in [n] \backslash \{j\}} p_i z_i \leq \epsilon \right\}, \tag{7}$$

$$O^{\mathrm{R}} = \min \left\{ c^\top x \,:\, (x, v, z) \in \mathcal{C},\ v \geq \mathbf{0},\ p_j + \sum_{i \in [n] \backslash \{j\}} p_i z_i \leq \epsilon \right\}, \tag{8}$$

where

$$\mathcal{C} = \left\{ (x, v, z) \in \mathcal{X} \times \mathbb{R}_+^m \times \{0, 1\}^{n-1} \,:\, Tx = v,\ v \geq \xi^i (1 - z_i),\ \forall\, i \in [n] \backslash \{j\} \right\}$$

represents the feasible set of formulation (MILP) with constraints $v \geq \xi^j (1 - z_j)$ and $\sum_{i=1}^n p_i z_i \leq \epsilon$ dropped and variable z_j removed. As $\xi^j \geq 0$ and $p_j \geq 0$, the intersection of the feasible sets of the two subproblems (7) and (8), called *overlap*, is

$$\mathcal{O} = \left\{ (x, v, z) \in \mathcal{C} \,:\, v \geq \xi^j,\ p_j + \sum_{i \in [n] \backslash \{j\}} p_i z_i \leq \epsilon \right\}. \tag{9}$$

The overlap $\mathcal{O}$ between the left and right branches is, however, generally nonempty. As the B&C algorithm explores this overlap in both branches, and the restrictions of both branches to this overlap provide the same objective value, there exists some redundancy in the search tree. In particular, considerable efforts are likely to be spent in exploring many subsequent nodes of one branch, whose feasible sets are subsets of the overlap $\mathcal{O}$, but can be avoided as these nodes cannot provide a better solution than that of the other branch. We use the following example to illustrate this weakness.

Example 2.1. *Consider*

$$\min \left\{ 6x_1 + x_2 + 3x_3 \,:\, \mathbb{P}\{x \geq \xi\} \geq 1 - \epsilon,\ x \in \mathbb{R}_+^3 \right\}, \tag{10}$$

where $\epsilon = \frac{4}{7}$ and ξ is a random vector with 7 equi-probable scenarios:

$$\xi^1 = \begin{pmatrix} 2 \\ 1 \\ 12 \end{pmatrix}, \xi^2 = \begin{pmatrix} 3 \\ 1 \\ 10 \end{pmatrix}, \xi^3 = \begin{pmatrix} 4 \\ 2 \\ 7 \end{pmatrix},$$
$$\xi^4 = \begin{pmatrix} 5 \\ 2 \\ 6 \end{pmatrix}, \xi^5 = \begin{pmatrix} 6 \\ 2 \\ 6 \end{pmatrix}, \xi^6 = \begin{pmatrix} 7 \\ 1 \\ 4 \end{pmatrix}, \xi^7 = \begin{pmatrix} 12 \\ 1 \\ 2 \end{pmatrix}.$$

In formulation (MILP) of this example, we have $x_1 = v_1$, $x_2 = v_2$, and $x_3 = v_3$. As a result, formulation (MILP) reduces to

$$\min \left\{ 6v_1 + v_2 + 3v_3 \, : \, v \geq \xi^i(1 - z_i), \ \forall \, i \in [7], \ \frac{1}{7}\sum_{i=1}^{7} z_i \leq \frac{4}{7}, \ v \in \mathbb{R}^3_+, \ z \in \{0,1\}^7 \right\}. \tag{11}$$

The optimal value of problem (11) is 59. *We apply the* B&C *algorithm to solve problem (11). We assume that a feasible solution of objective value* 59 *is found at the root node (e.g., by some heuristic algorithm). For simplicity of illustration, we use the most infeasible branching rule [3] to choose the variable to branch on. The* B&C *search tree is drawn in Figure 1. At each node, we report the optimal value z_{LP} of its LP relaxation.*

The feasible regions of nodes 2 *and* 3 *(after removing the common fixed variable z_4) are*

$$\mathcal{F}^2 = \Big\{ (v, z) \in \mathbb{R}^3_+ \times \{0,1\}^6 \, : \ \frac{1}{7}(z_1 + z_2 + z_3 + z_5 + z_6 + z_7) \leq \frac{4}{7},$$
$$v \geq \xi^4 = (5,\, 2,\, 6)^\top, \ v \geq \xi^i(1 - z_i), \ \forall \, i \in \{1,2,3,5,6,7\} \Big\},$$
$$\mathcal{F}^3 = \Big\{ (v, z) \in \mathbb{R}^3_+ \times \{0,1\}^6 \, : \ \frac{1}{7}(z_1 + z_2 + z_3 + z_5 + z_6 + z_7) \leq \frac{3}{7},$$
$$v \geq \mathbf{0}, \ v \geq \xi^i(1 - z_i), \ \forall \, i \in \{1,2,3,5,6,7\} \Big\},$$

and the overlap is

$$\mathcal{O} = \mathcal{F}^2 \cap \mathcal{F}^3 = \Big\{ (v, z) \in \mathbb{R}^3_+ \times \{0,1\}^6 \, : \, \frac{1}{7}(z_1 + z_2 + z_3 + z_5 + z_6 + z_7) \leq \frac{3}{7},$$
$$v \geq \xi^4 = (5,\, 2,\, 6)^\top, v \geq \xi^i(1 - z_i), \ \forall \, i \in \{1,2,3,5,6,7\} \Big\}.$$

Consider node 6*, a subsequent node of node* 3*, whose feasible region (after removing the fixed variable z_4) is*

$$\mathcal{F}^6 = \Big\{ (v, z) \in \mathbb{R}^3_+ \times \{0,1\}^6 \, : \, z_5 = 0, \ \frac{1}{7}(z_1 + z_2 + z_3 + z_5 + z_6 + z_7) \leq \frac{3}{7},$$
$$v \geq \xi^5 = (6,\, 2,\, 6)^\top, v \geq \xi^i(1 - z_i), \ \forall \, i \in \{1,2,3,5,6,7\} \Big\}.$$

Clearly, $\mathcal{F}^6 \subseteq \mathcal{O} \subseteq \mathcal{F}^2$, implying that the optimal value of node 6 *cannot be better than that of node* 2*. Thus, node* 6 *(and its descendant nodes* 12, 13, 20, 21, 30, *and* 31*) can be pruned.*

This example demonstrates the weakness of applying the B&C *algorithm with the classic variable branching to solve formulation (MILP): a wasteful duplication of efforts is spent in exploring the uninteresting overlap in the search tree.*

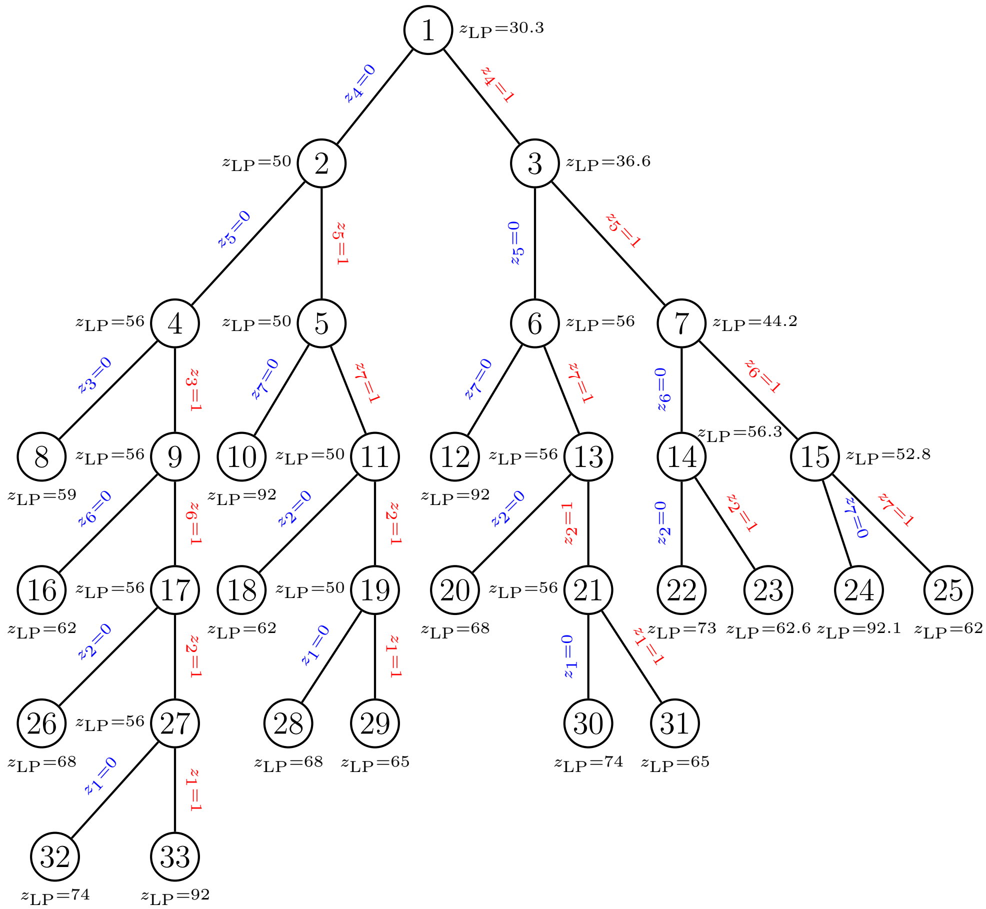

Figure 1: The B&C search tree of the problem in Example 2.1 with the classic variable branching applied.

2.2 Removing the overlaps

To remove the overlap $\mathcal{O}$ in the search tree, let us first divide the right branch (8) into the following two subproblems:

$$O^{\mathrm{R}_1} = \min\left\{ c^\top x \,:\, (x,v,z)\in\mathcal{C},\ v\geq\xi^j,\ p_j+\sum_{i\in[n]\setminus\{j\}} p_i z_i \leq \epsilon \right\}, \tag{12}$$

and

$$O^{\mathrm{R}_2} = \min\left\{ c^\top x \,:\, (x,v,z)\in\mathcal{C},\ v\ngeq\xi^j,\ p_j+\sum_{i\in[n]\setminus\{j\}} p_i z_i \leq \epsilon \right\}. \tag{13}$$

Here $v\ngeq\xi^j$ denotes that $v<\xi^j_k$ holds for at least one $k\in[m]$. Observe that the feasible region of subproblem (12) is identical to the overlap $\mathcal{O}$ in (9) and thus a subset of the feasible region of problem (7). This, together with the fact that the objective functions of problems (7) and (12) are identical, implies $O^{\mathrm{R}_1}\geq O^{\mathrm{L}}$. Consequently, instead of exploring problem (8) with a potentially large feasible region, we can explore its restriction (13) in the search tree. Notice that since constraints $v=Tx\geq\xi^j$ and $v=Tx\ngeq\xi^j$ appear in the left and (new) right branches (7) and (13), respectively, no overlap exists between the feasible regions (and their projections onto the x space) of the two branches.

Remark 2.2. *By dividing the left branch* (7) *into*

$$O^{\mathrm{L}_1} = \min\left\{ c^\top x \,:\, (x,v,z) \in \mathcal{C},\ v \geq \xi^j,\ \sum_{i\in[n]\setminus\{j\}} p_i z_i \leq \epsilon - p_j \right\}, \tag{14}$$

$$O^{\mathrm{L}_2} = \min\left\{ c^\top x \,:\, (x,v,z) \in \mathcal{C},\ v \geq \xi^j,\ \epsilon - p_j < \sum_{i\in[n]\setminus\{j\}} p_i z_i \leq \epsilon \right\}, \tag{15}$$

and noting that $O^{\mathrm{L}_1} \geq O^{\mathrm{R}}$, we can also keep the right branch (8) *but add the new left branch* (15) *with a smaller feasible region than that of* (7) *into the search tree. Clearly, the feasible regions of problems* (15) *and* (8) *also do not contain an overlap. However, different from those of branches* (7) *and* (13)*, the projections of the feasible regions of branches* (15) *and* (8) *onto the x space may still contain overlaps. An illustrative example for this is provided in Appendix A. Thus, the redundancy still exists in the search tree as an optimal solution x^* of* (CCP) *may simultaneously define feasible solutions of the two branches* (15) *and* (8)*.*

Remark 2.3. *The overlap in the* B&C *search tree has also been identified by* Qiu et al. [49] *and* Chen et al. [15] *in the context of solving the k-violation linear programming and covering location problems. However, unlike those in* [49] *and* [15] *where the overlap can be removed by linear constraints, we need to use the highly nonlinear constraints $v \ngeq \xi^j$ to remove the overlap arising in solving the CCPs.*

The overlap $\mathcal{O}$ in (9) can also be removed by refining formulation (MILP). To do this, let us first present problem (CCP) as

$$\min\left\{ c^\top x \,:\, Tx = v,\ x \in \mathcal{X},\ \sum_{i=1}^{n} p_i \chi(v \ngeq \xi^i) \leq \epsilon \right\}, \tag{CCP'}$$

where χ is an indicator function:

$$\chi(v \ngeq \xi^i) = \begin{cases} 1, & \text{if } v \ngeq \xi^i; \\ 0, & \text{otherwise,} \end{cases} \quad \forall\, i \in [n]. \tag{16}$$

Let $\mathcal{F}^{\mathrm{CCP}}$ and $\mathcal{F}^{\mathrm{CCP'}}$ denote the feasible regions of problems (CCP) and (CCP'), respectively. Then we must have $\mathcal{F}^{\mathrm{CCP}} = \mathrm{Proj}_x(\mathcal{F}^{\mathrm{CCP'}})$, and hence problems (CCP) and (CCP') are equivalent. Unlike formulation (CCP') where (16) simultaneously ensures $\chi(v \ngeq \xi^i) = 0 \Rightarrow v \geq \xi^i$ and $\chi(v \ngeq \xi^i) = 1 \Rightarrow v \ngeq \xi^i$, formulation (MILP) with (2) and $z_i \in \{0,1\}$ only guarantees $z_i = 0 \Rightarrow v \geq \xi^i$ but cannot guarantee $z_i = 1 \Rightarrow v \ngeq \xi^i$. In other words, it is possible that formulation (MILP) has a feasible solution (x^*, v^*, z^*) such that $z_i^* = 1$ and $v \geq \xi^i$ (though setting $z_i^* = 0$ yields another feasible solution of formulation (MILP) with the same objective value and hence formulation (MILP) is a valid formulation for (CCP)).

However, only enforcing $z_i = 0 \Rightarrow v \geq \xi^i$ in formulation (MILP) makes the overlaps in the search tree. To see this, let us add the valid nonlinear *if-then* constraints

$$z_i = 1 \Rightarrow v \ngeq \xi^i,\ \forall\, i \in [n], \tag{17}$$

into formulation (MILP) and obtain a mixed integer nonlinear programming (MINLP) formulation for (CCP):

$$\min\left\{ c^\top x \,:\, (1) - (4), (17) \right\}. \tag{MINLP}$$

Let $\mathcal{F}^{\text{MILP}}$ and $\mathcal{F}^{\text{MINLP}}$ be the feasible sets of formulations (MILP) and (MINLP), respectively. As $\text{Proj}_x(\mathcal{F}^{\text{MILP}}) = \text{Proj}_x(\mathcal{F}^{\text{MINLP}}) = \mathcal{F}^{\text{CCP}}$, formulations (MINLP) and (MILP) are equivalent. Now, for formulation (MINLP), branching on variable z_j will not lead to an overlap in the x space as $v = Tx \geq \xi^j$ and $v = Tx \ngeq \xi^j$ are enforced in the left and right branches, respectively. As constraints (2) and (17) are all enforced in problem (MINLP), a node $a = (\mathcal{N}_0^a, \mathcal{N}_1^a)$ in the search tree includes the linear constraints $v \geq \xi^i$, $i \in \mathcal{N}_0^a$, and the *disjunctive constraints* $v \ngeq \xi^i$, $i \in \mathcal{N}_1^a$. Therefore, the result can be generalized to any two descendant nodes of the left and right branches of a node, as detailed in the following theorem.

Theorem 2.4. *Let $a = (\mathcal{N}_0^a, \mathcal{N}_1^a)$ and $b = (\mathcal{N}_0^b, \mathcal{N}_1^b)$ be any two nodes in the search tree of formulation (MINLP) and c be their first common ancestor. If c differs from a and b, then*

$$\text{Proj}_x(\mathcal{F}^{\text{MINLP}}(\mathcal{N}_0^a, \mathcal{N}_1^a)) \cap \text{Proj}_x(\mathcal{F}^{\text{MINLP}}(\mathcal{N}_0^b, \mathcal{N}_1^b)) = \varnothing, \tag{18}$$

where $\mathcal{F}^{\text{MINLP}}(\mathcal{N}_0^a, \mathcal{N}_1^a)$ and $\mathcal{F}^{\text{MINLP}}(\mathcal{N}_0^b, \mathcal{N}_1^b)$ are the feasible regions of nodes a and b, respectively.

Proof. Without loss of generality, suppose that z_j is branched on at node c, and nodes a and b are in the left and right branches of node c, respectively. As $v = Tx \geq \xi^j$ and $v = Tx \ngeq \xi^j$ are enforced in the left and right branches, respectively, (18) holds. □

By Theorem 2.4, in the search tree of formulation (MINLP), no $x \in \mathbb{R}^d$ can simultaneously define feasible solutions for any two descendant nodes of the left and right branches of a node. This is intrinsically different from the search tree of formulation (MILP) in which a solution x may define feasible solutions for two descendant nodes of the left and right branches of a node. Thus, it can be expected that the search tree of applying the B&C algorithm to formulation (MINLP) is smaller than that of applying the B&C algorithm to formulation (MILP).

In contrast to formulation (MILP) which is an MILP problem, formulation (MINLP) is, however, a relatively hard MINLP problem due to the nonlinear if-then constraints (17). Moreover, to the best of our knowledge, the highly nonlinear if-then constraints (17) cannot be directly tackled by state-of-the-art MINLP solvers. In the next two sections, we will propose a novel branching and overlap-oriented node pruning and variable fixing techniques based on formulation (MILP) that explicitly employ the if-then constraints (17) in removing the overlaps in the search tree.

3 Dominance-based branching

As shown in Section 2.2, while using (17) guarantees that no overlap exists between the left branch (7) and right branch (13), it also leads to a hard MINLP subproblem (13). To resolve the difficulty, in this section, we first leverage the concept of *dominance* between scenarios [51] and propose a novel branching, called dominance-based branching, which achieves a better tradeoff between the size of the overlap and the solution efficiency of the two branches. Then we discuss its relation to the classic variable branching that is applied to problem (MILP) with the dominance inequalities in [51]. Subsequently, we enhance the proposed branching by presenting a preprocessing technique to derive more dominance pairs between different scenarios. Finally, we use an example to illustrate the effectiveness of the proposed dominance-based branching for solving problem (MILP).

3.1 Description of the dominance-based branching

A scenario i is dominated by a scenario j if $\xi^i \leq \xi^j$, denoted as $i \preceq j$. Let

$$\mathcal{N}_j^- = \left\{ i \in [n] \,:\, \xi^i \leq \xi^j \right\} \tag{19}$$

be the index set of scenarios that are dominated by scenario ξ^j (including scenario ξ^j). For a feasible solution (x, v, z) of problem (MINLP) with $z_j = 0$, we have $v \geq \xi^j$. This, together with $\xi^i \leq \xi^j$, $z_i \in \{0, 1\}$, and the if-then constraints $z_i = 1 \Rightarrow v \not\geq \xi^i$ in (17) for $i \in \mathcal{N}_j^-$, implies $z_i = 0$ for all $i \in \mathcal{N}_j^-$ and

$$
\begin{aligned}
&\{(x, v, z) \,:\, (1) - (4),\ (17),\ z_j = 0\} && (20)\\
&\quad\subseteq \left\{(x, v, z) \,:\, (1) - (4),\ z_i = 0,\ \forall\, i \in \mathcal{N}_j^-\right\} && (21)\\
&\quad\subseteq \{(x, v, z) \,:\, (1) - (4),\ z_j = 0\}\,. && (22)
\end{aligned}
$$

Observe that (20) and (22) are the feasible sets of problems (MINLP) and (MILP) with $z_j = 0$, respectively. By (20)–(22) and the fact that exploring a left branch with its feasible set being (20) or (22) in the search tree can all return a correct optimal solution for problem (CCP), we can develop a new branching that explores a new *linear* left branch with its feasible set being (21):

$$
O^{\mathrm{L_D}} = \min\left\{c^\top x \,:\, (1) - (4),\ z_i = 0,\ \forall\, i \in \mathcal{N}_j^-\right\}. \tag{23}
$$

Similarly, we can also derive a *linear* right branch with a potentially more compact feasible set. In particular, let

$$
\mathcal{N}_j^+ = \left\{i \in [n] \,:\, \xi^j \leq \xi^i\right\} \tag{24}
$$

be the index set of scenarios that dominate scenario ξ^j (including scenario ξ^j), and (x, v, z) be a feasible solution of problem (MINLP) with $z_j = 1$. From (17), $v \not\geq \xi^j$ holds, which, together with $\xi^j \leq \xi^i$ for $i \in \mathcal{N}_j^+$, implies that $v \not\geq \xi^i$ must hold. From $v \geq \xi^i(1 - z_i)$ and $z_i \in \{0, 1\}$, $z_i = 1$ must be satisfied for $i \in \mathcal{N}_j^+$. Using a similar argument as that of deriving the left branch (23), we can derive a new equivalent right branch:

$$
O^{\mathrm{R_D}} = \min\left\{c^\top x \,:\, (1) - (4),\ z_i = 1,\ \forall\, i \in \mathcal{N}_j^+\right\}. \tag{25}
$$

Note that branches (23) and (25) are of the form of (MILP), and so the above dominance-based branching can be applied to any node of the search tree. In particular, for a node $(\mathcal{B}_0, \mathcal{B}_1)$ in the search tree, the index sets of variables, that can be fixed at zero and one, are

$$
\mathcal{N}_0 = \bigcup_{j \in \mathcal{B}_0} \mathcal{N}_j^- \text{ and } \mathcal{N}_1 = \bigcup_{j \in \mathcal{B}_1} \mathcal{N}_j^+, \tag{26}
$$

respectively.

Compared with the two branches (7) and (8) with the classic variable branching applied, the new branches (23) and (25) with the proposed dominance-based branching applied are still MILP problems but with more compact feasible regions. Thus, compared with the overlap $\mathcal{O}$ in (9), the overlap between the feasible regions of branches (23) and (25) (after removing the common fixed variable z_j), i.e.,

$$
\mathcal{O}' = \Bigg\{(x, v, z) \in \mathcal{C} \,:\, v \geq \xi^j,\ p_j + \sum_{i \in [n] \setminus \{j\}} p_i z_i \leq \epsilon,
\\
z_i = 0,\ \forall\, i \in \mathcal{N}_j^- \setminus \{j\},\ z_i = 1,\ \forall\, i \in \mathcal{N}_j^+ \setminus \{j\}\Bigg\},
$$

is much smaller, especially when $\mathcal{N}_j^-$ and $\mathcal{N}_j^+$ are large. Consequently, certain nodes that are explored with the classic variable branching applied may not be explored with the proposed dominance-based branching applied (i.e., the feasible sets of these nodes are subsets of $\mathcal{O}\backslash\mathcal{O}'$). Due to this advantage, it can be expected that the B&C search tree with the proposed dominance-based branching applied will be potentially much smaller than that with the classic variable branching applied.

The more compact feasible region of the new right branch (25) may lead to infeasibility or enable to provide a potentially much stronger LP relaxation bound than that provided by the old one (8). An example for illustrating this will be provided in Section 3.4. This advantage opens up more possibilities to apply the infeasibility or bound exceeding based node pruning and guides the selection of branching variables (if strong branching or its variants [4] are applied), thereby enabling to further reduce the tree size. Notice that for an optimal solution (x, v, z) of the LP relaxation of branch (7), setting $z_i = 0$ for all $i \in \mathcal{N}_j^-$ yields a feasible solution of branch (23) with the same objective value, and thus the new left branch (23) provides the same LP relaxation bound as that provided by the old one (7).

It is worthwhile remarking that the proposed dominance-based branching can be implemented along with various *branching strategies* such as strong branching and the most infeasible branching [3] that choose a variable to branch on for the current LP relaxation. In Section 5, we will describe how to implement the proposed dominance-based branching into a state-of-the-art open source MILP solver while using its default fine-tuned branching strategy.

3.2 Relation to the result in [51]

As shown in the previous subsection, for a feasible solution (x, v, z) of problem (MINLP), if $z_j = 0$, then $z_i = 0$ holds for $i \in [n]\backslash\{j\}$ with $\xi^i \leq \xi^j$ (or equivalently, if $z_i = 1$, then $z_j = 1$ holds for $j \in [n]\backslash\{i\}$ with $\xi^i \leq \xi^j$). Thus, the following *dominance inequalities*

$$z_i \leq z_j, \ (i, j) \in \mathcal{A} := \{(i, j) \in [n] \times [n] \, : \, \xi^i \leq \xi^j, \ i \neq j\} \tag{27}$$

are valid for problem (MINLP). This, together with the equivalence of problems (MINLP) and (MILP), shows that inequalities (27) are also valid for problem (MILP). This result, first established by Ruszczyński [51], is formally stated as follows.

Theorem 3.1 ([51])**.** *The MILP problem*

$$\min \left\{ c^\top x \, : \, (1) - (4), (27) \right\} \tag{28}$$

is equivalent to problem (MILP) *in terms of sharing at least one identical optimal solution.*

Remark 3.2. *If $\xi^i \leq \xi^s \leq \xi^j$ for some distinct $i, j, s \in [n]$, the dominance inequality $z_i \leq z_j$ is implied by inequalities $z_i \leq z_s$ and $z_s \leq z_j$. Thus, constraints* (27) *in problem* (28) *can be simplified as $z_i \leq z_j$ for $(i, j) \in \mathcal{A}'$ where*

$$\mathcal{A}' := \left\{(i, j) \in \mathcal{A} \, : \, \textit{no } s \in [n]\backslash\{i, j\} \textit{ with } \xi^i \leq \xi^s \leq \xi^j \textit{ exists}\right\}.$$

We can further strengthen the result in Theorem 3.1 by showing that the LP relaxations of problems (MILP) and (28) are equivalent. This indicates that the dominance inequalities cannot improve the LP relaxation of problem (MILP).

Proposition 3.3. *The LP relaxations of problems* (MILP) *and* (28) *are equivalent.*

Proof. It suffices to show there exists an optimal solution (x^*, v^*, z^*) of the LP relaxation of problem (MILP) such that $z_i^* \leq z_j^*$ holds for all $(i,j) \in \mathcal{A}$. Let $(\bar{x}, \bar{v}, \bar{z})$ be an optimal solution of the LP relaxation of problem (MILP). If $\bar{z}_i \leq \bar{z}_j$ for all $(i,j) \in \mathcal{A}$, the statement follows. Otherwise, there exists $(i_0, j_0) \in \mathcal{A}$ such that $\bar{z}_{i_0} > \bar{z}_{j_0}$. In this case, we construct a new point $(\hat{x}, \hat{v}, \hat{z})$ by setting

$$\hat{x} = \bar{x},\ \hat{v} = \bar{v},\ \hat{z}_{i_0} = \bar{z}_{j_0},\ \text{and } \hat{z}_i = \bar{z}_i,\ \text{for all } i \in [n] \backslash \{i_0\}. \tag{29}$$

Clearly, the objective values of the LP relaxation of problem (MILP) at points $(\bar{x}, \bar{v}, \bar{z})$ and $(\hat{x}, \hat{v}, \hat{z})$, respectively, are the same. In addition, $(\hat{x}, \hat{v}, \hat{z})$ is also a feasible solution of the LP relaxation of problem (MILP) as

$$\hat{v} = \bar{v} \geq \xi^{j_0}(1 - \bar{z}_{j_0}) \geq \xi^{i_0}(1 - \bar{z}_{j_0}) = \xi^{i_0}(1 - \hat{z}_{i_0}),$$

where the second inequality follows from $\xi^{i_0} \leq \xi^{j_0}$ and $\bar{z}_{j_0} \leq 1$. Recursively applying the above operation and using the fact that the value of each z_i decreases at most n times, we will obtain an optimal solution (x^*, v^*, z^*) of the LP relaxation of problem (MILP) such that $z_i^* \leq z_j^*$ holds for all $(i,j) \in \mathcal{A}$. □

Next, we show that (i) applying the proposed dominance-based branching in Section 3.1 to problem (MILP) is theoretically equivalent to (ii) applying the classic variable branching to problem (28) in the following two aspects. First, for two nodes in the two search trees constructed by (i) and (ii) characterizing by the same branching variables $\mathcal{B}_0$ and $\mathcal{B}_1$, the index sets of variables fixed at zero and one are all identical. Indeed, for case (i), as shown in Section 3.1, the index sets of variables fixed at zero and one are the $\mathcal{N}_0$ and $\mathcal{N}_1$, respectively, defined in (26); and for case (ii), the index sets of variables fixed at zero and one at node $(\mathcal{B}_0, \mathcal{B}_1)$ (due to branching or fixing by the dominance inequalities in (27)) are also $\mathcal{N}_0$ and $\mathcal{N}_1$, respectively. Second, the LP relaxations of the two nodes are also equivalent, as detailed in the following corollary.

Corollary 3.4. *Consider a node $(\mathcal{B}_0, \mathcal{B}_1)$ in the search tree constructed by* (i) *or* (ii) *and let $\mathcal{N}_0$ and $\mathcal{N}_1$ be defined as in* (26)*. The LP relaxation of problem* (MILP) *with $z_i = 0$ for $i \in \mathcal{N}_0$ and $z_i = 1$ for $i \in \mathcal{N}_1$ is equivalent to the LP relaxation of problem* (28) *with $z_i = 0$ for $i \in \mathcal{N}_0$ and $z_i = 1$ for $i \in \mathcal{N}_1$.*

Proof. The result follows from Proposition 3.3 and the fact that a subproblem of (MILP) or (28) with $z_i = 0$ for $i \in \mathcal{N}_0$ and $z_i = 1$ for $i \in \mathcal{N}_1$ still takes the form of (MILP) or (28), respectively. □

Corollary 3.4 implies that the LP relaxations of the two nodes $(\mathcal{B}_0, \mathcal{B}_1)$ in the two search trees share at least one identical optimal solution (if they are feasible). Thus, if we choose the same variable to branch on for the identical (fractional) LP relaxation solution using, e.g., strong branching strategy [3], the search trees constructed by (i) and (ii) will also be identical.

The theoretical equivalence of (i) and (ii) shows that although the dominance inequalities in (27) cannot strengthen the LP relaxation of problem (MILP), they can, as the proposed dominance branching, enhance the B&C algorithm of MILP solvers by removing the uninteresting overlap in the search tree. This theoretical equivalence also sheds useful insights on the proposed dominance-based branching. More specifically, the dominance-based branching can be treated as an enhanced version of classic variable branching to formulation (MILP) that additionally uses the dominance relations in (27) for fixing variables. As such, the dominance-based branching relies only on the dominance relations in (27) but does not rely on the underlying formulation, which allows for the use of sophisticated MILP methodologies such as cutting planes and preprocessing techniques. In particular, effective techniques for CCPs like mixing cuts [7, 25] and their variants [1, 28, 30, 38, 59] and the preprocessing technique in [38] can all be applied along with the proposed dominance-based branching.

Although applying the dominance-based branching to problem (MILP) is theoretically equivalent to applying the classic variable branching to problem (28), the former, however, can avoid solving a possibly large LP relaxation of problem (28) (due to the addition of the dominance inequalities in (27)). Therefore, it can be expected that applying the dominance-based branching to problem (MILP) is more computationally efficient. In Section 5, we will further present computational results to illustrate this.

3.3 Preprocessing

The dominance-based branching depends critically on the existence of the dominance pairs $i \preceq j$. In general, the more the dominance pairs, the smaller the overlap $\mathcal{O}'$, and thus the more effective the dominance-based branching. However, the condition $\xi^i \leq \xi^j$ is quite restrictive, and in some applications, the number of dominance pairs is extremely small, leading to the ineffectiveness of the dominance-based branching. To overcome this weakness, below we apply the preprocessing technique in [31, 38] to problem (MILP) so that we can derive more dominance pairs $i \preceq j$ from the equivalent transformed problem.

To proceed, for $k \in [m]$, let $\{\pi_k(1), \pi_k(2), \ldots, \pi_k(n)\}$ be a permutation of $[n]$ such that $\xi_k^{\pi_k(1)} \geq \xi_k^{\pi_k(2)} \geq \cdots \geq \xi_k^{\pi_k(n)}$. Define $\tau_k := \min\{s \,:\, \sum_{i=1}^{s} p_{\pi_k(i)} > \epsilon\}$. From the knapsack constraint (3), $z_{\pi_k(t)} = 1$, $t = 1, 2, \ldots, \tau_k$, cannot simultaneously hold for a feasible solution (x, v, z) of formulation (MILP). Using this observation and the fact that $v_k \geq \xi_k^{\pi_k(t)}(1 - z_{\pi_k(t)})$ and $z_{\pi_k(t)} \in \{0, 1\}$ for $t = 1, 2, \ldots, \tau_k$, a lower bound $\xi_k^0 := \xi_k^{\pi_k(\tau_k)}$ for variable v_k can be derived.

Lemma 3.5 ([19, 31, 32, 38]). *$v_k \geq \xi_k^0$, $k \in [m]$, are valid for formulation* (MILP).

Let

$$\bar{\xi}_k^i := \max\left\{\xi_k^i,\, \xi_k^0\right\},\ \forall\, i \in [n],\ k \in [m], \tag{30}$$

and

$$v \geq \bar{\xi}^i(1 - z_i),\ \forall\, i \in [n]. \tag{2'}$$

From Lemma 3.5 and $z \in \{0, 1\}^n$, (2) and (2') are equivalent. Thus, applying this preprocessing technique, we can obtain the following new equivalent MILP formulation for problem (CCP):

$$\min\left\{c^\top x \,:\, (1),\ (2'),\ (3),\ (4)\right\}. \tag{31}$$

In the new MILP formulation (31), if $\bar{\xi}^i \leq \bar{\xi}^j$, scenario i is dominated by scenario j. Observe that from (30), $\bar{\xi}^i \leq \bar{\xi}^j$ is more likely to be appeared than $\xi^i \leq \xi^j$, and thus more dominance relations

$$z_i \leq z_j,\ \forall\, (i, j) \in \bar{\mathcal{A}} := \left\{(i, j) \in [n] \times [n] \,:\, \bar{\xi}^i \leq \bar{\xi}^j,\ i \neq j\right\} \tag{32}$$

can be derived. Now, applying the dominance-based branching with the dominance relations in $\bar{\mathcal{A}}$, we will obtain two more compact branches than (23) and (25):

$$\min\left\{c^\top x \,:\, (1) - (4),\ z_i = 0,\ \forall\, i \in \bar{\mathcal{N}}_j^-\right\}, \tag{33}$$

$$\min\left\{c^\top x \,:\, (1) - (4),\ z_i = 1,\ \forall\, i \in \bar{\mathcal{N}}_j^+\right\}, \tag{34}$$

where $\bar{\mathcal{N}}_j^- := \left\{i \in [n] : \bar{\xi}^i \leq \bar{\xi}^j\right\}$ and $\bar{\mathcal{N}}_j^+ := \left\{i \in [n] : \bar{\xi}^j \leq \bar{\xi}^i\right\}$. Moreover, with the decreasing of ϵ, ξ_k^0 will become larger and more the dominance pairs $i \preceq j$ are likely to appear, implying that branches (33) and (34) will also become much more compact.

Remark 3.6. *Adding*

$$z_i \leq z_j, \ \forall \ (i,j) \in \bar{\mathcal{A}}' := \left\{(i,j) \in \bar{\mathcal{A}} \,:\, \textit{no } s \in [n]\backslash\{i,j\} \textit{ with } \bar{\xi}^i \leq \bar{\xi}^s \leq \bar{\xi}^j \textit{ exists}\right\}, \tag{35}$$

the simplified version of (32) (*see Remark 3.2*), *into formulation* (31) *yields another equivalent MILP formulation. This formulation is stronger than* (28) *in terms of providing a more compact feasible region.*

3.4 An illustrative example

We now apply the dominance-based branching to problem (11) in Example 2.1 to demonstrate its effectiveness over the classic variable branching.

We first note that for problem (11), the lower bounds ξ^0 for variables v stated in Lemma 3.5 reads $\begin{pmatrix} 4 \\ 1 \\ 6 \end{pmatrix}$. Applying the preprocessing technique in Section 3.3, we will obtain an equivalent problem of (11) where ξ^i is replaced by $\bar{\xi}^i$:

$$\bar{\xi}^1 = \begin{pmatrix} 4 \\ 1 \\ 12 \end{pmatrix}, \bar{\xi}^2 = \begin{pmatrix} 4 \\ 1 \\ 10 \end{pmatrix}, \bar{\xi}^3 = \begin{pmatrix} 4 \\ 2 \\ 7 \end{pmatrix},$$

$$\bar{\xi}^4 = \begin{pmatrix} 5 \\ 2 \\ 6 \end{pmatrix}, \bar{\xi}^5 = \begin{pmatrix} 6 \\ 2 \\ 6 \end{pmatrix}, \bar{\xi}^6 = \begin{pmatrix} 7 \\ 1 \\ 6 \end{pmatrix}, \bar{\xi}^7 = \begin{pmatrix} 12 \\ 1 \\ 6 \end{pmatrix}.$$

With the preprocessing technique, we can detect 3 dominance pairs: $2 \preceq 1$, $6 \preceq 7$, and $4 \preceq 5$. In contrast, only a single dominance pair $4 \preceq 5$ can be detected without the preprocessing technique.

Figure 2 displays the search tree constructed by using the dominance-based branching to solve problem (11) with the most infeasible branching rule applied. Let us consider the branching of variable z_4 at the root node. Since $4 \preceq 5$, we obtain $\mathcal{N}_4^+ = \{4, 5\}$. The right branch 3 of node 1 is associated with $\mathcal{N}_0 = \varnothing$ and $\mathcal{N}_1 = \{4, 5\}$. Thus, node 6 and its descendant nodes 12, 13, 20, 21, 30, and 31 in the previous search tree in Figure 1 do not need to be explored with the proposed dominance-based branching applied.

Using the classic variable branching, the LP relaxation of the right branch 27 of node 17 in Figure 1 is still feasible with an optimal value of 56, while using the proposed dominance-based branching, the LP relaxation of the right branch 27 of node 17 in Figure 2 is infeasible, thereby avoiding further branching at node 27. Similarly, using the proposed dominance-based branching, the LP relaxation of the right branch 19 of node 11 in Figure 2 has an optimal value of 65, which is larger than the optimal value of problem (11) (i.e., 59), thereby also avoiding further branching at node 19.

This example shows the effectiveness of the proposed dominance-based branching over the classic variable branching in reducing the search tree size. Overall, using the proposed dominance-based branching with the preprocessing technique, only 19 nodes need to be explored, which is 14 less than that of the search tree in Figure 1 where the classic variable branching is applied.

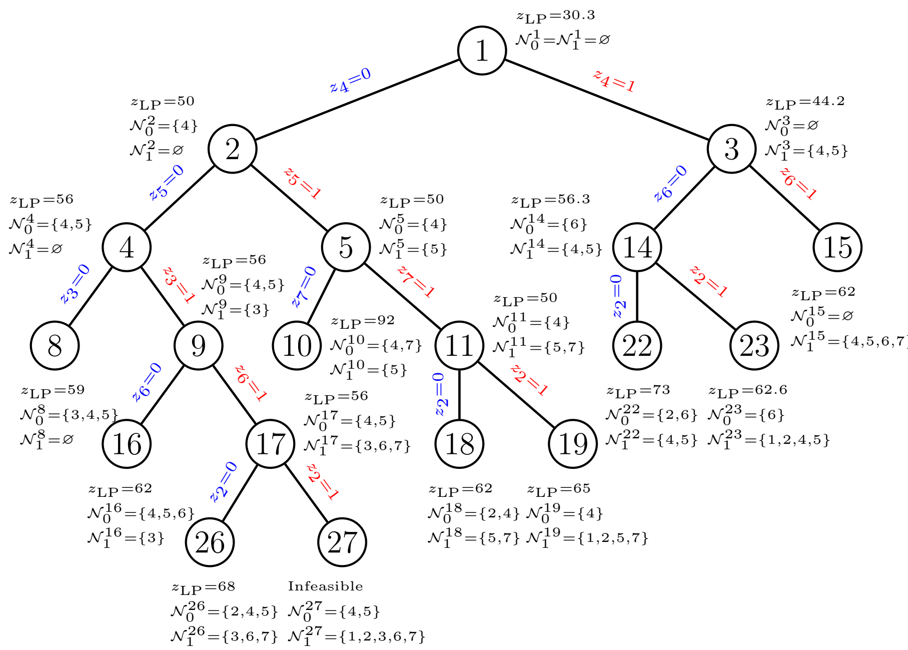

Figure 2: The B&C search tree of the problem in Example 2.1 with the proposed dominance-based branching applied.

4 Overlap-oriented node pruning and variable fixing

The dominance-based branching creates two subproblems (33) and (34) (with more fixed variables) using the dominance relations in (32). Such relations are derived from constraints (2)–(4) and the if-then constraints (17) at the root node of the search tree (that removes the overlaps). It is possible, however, that at other nodes in the search tree, with the additional fixings of variables to 0 and 1, more reductions, i.e., node pruning and variable fixing, can be derived by exploiting the overlap information. To further enhance the dominance-based branching, in this section, we shall perform overlap-oriented node pruning and variable fixing at each node $(\mathcal{N}_0, \mathcal{N}_1)$ by considering the set

$$\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) := \{(v, z) \,:\, (2) - (4),\ (17),\ z_i = 0,\ \forall\, i \in \mathcal{N}_0,\ z_i = 1,\ \forall\, i \in \mathcal{N}_1\}\,. \tag{36}$$

Here $\mathcal{N}_0$ and $\mathcal{N}_1$ are the index sets of variables z fixed to 0 and 1, respectively, at the current node. This set is a variant of the joint mixing set with a knapsack constraint [28, 59] that additionally includes the if-then constraints (17) and the variable fixings at the current node. Specifically, we can

(R1) prune node $(\mathcal{N}_0, \mathcal{N}_1)$ when $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) = \varnothing$ is detected; or

(R2) fix variables in $\mathcal{R}_0$ and $\mathcal{R}_1$ to 0 and 1 at node $(\mathcal{N}_0, \mathcal{N}_1)$ when $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) \neq \varnothing$, where $\mathcal{R}_0$ and $\mathcal{R}_1$ are disjoint subsets of $\mathcal{N}_f := [n] \backslash (\mathcal{N}_0 \cup \mathcal{N}_1)$ satisfying

$$\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) = \mathcal{C}(\mathcal{N}_0 \cup \mathcal{R}_0, \mathcal{N}_1 \cup \mathcal{R}_1). \tag{37}$$

Note that the dominance relations in (32) can be derived from $\mathcal{C}(\varnothing, \varnothing)$, and thus the reductions by (R1) and (R2) include the reductions by the dominance relations in (32) (in the dominance-based branching). In the following, we shall present exact and approximation approaches to detect when $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) = \varnothing$ holds, or find the largest subsets $\mathcal{R}_0$ and $\mathcal{R}_1$ of $\mathcal{N}_f$ satisfying (37).

4.1 The exact approach

We first consider the overlap-oriented node pruning in (R1), i.e., identify the condition under which $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) = \varnothing$ holds. Let $(v, z) \in \mathcal{C}(\mathcal{N}_0, \mathcal{N}_1)$. For $i \in \mathcal{N}_0$, from (2) and $z_i = 0$, we obtain $v \geq \xi^i$, and thus $v \geq \xi^{\mathcal{N}_0} := \max_{i\in\mathcal{N}_0}\{\xi^i\}$ (where the max is taken component-wise and $\xi^{\mathcal{N}_0} = \mathbf{0}$ if $\mathcal{N}_0 = \varnothing$); and for $j \in \mathcal{N}_1$, from the if-then constraint $z_j = 1 \Rightarrow v \not\geq \xi^j$ in (17) and $z_j = 1$, we obtain $v \not\geq \xi^j$, or equivalently, $\bigvee_{k\in[m]} \left(v_k < \xi^j_k\right) = 1$. Therefore,

$$1 = \bigwedge_{j\in\mathcal{N}_1} \bigvee_{k\in[m]} \left(v_k < \xi^j_k\right) = \bigwedge_{j\in\mathcal{N}_1} \bigvee_{k\in[m],\, \xi^j_k > \xi^{\mathcal{N}_0}_k} \left(v_k < \xi^j_k\right), \tag{38}$$

where the second equality follows from $v \geq \xi^{\mathcal{N}_0}$. Letting $\mathcal{M}_j = \left\{k \in [m] \,:\, \xi^j_k > \xi^{\mathcal{N}_0}_k\right\}$ and $\mathcal{L} = \prod_{j\in\mathcal{N}_1} \mathcal{M}_j$, then (38) can be rewritten as

$$\bigvee_{\ell\in\mathcal{L}} d_\ell(v) = 1, \tag{39}$$

where

$$d_\ell(v) = \bigwedge_{j\in\mathcal{N}_1} \left(v_{\ell_j} < \xi^j_{\ell_j}\right). \tag{40}$$

Notice that $\mathcal{M}_j$, $\mathcal{L}$, and $d_\ell(v)$ indeed depend on $\mathcal{N}_0$ or $\mathcal{N}_1$ (or both of them) but we omit this dependence for notation convenience. The following theorem provides a necessary and sufficient condition for $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) \neq \varnothing$.

Theorem 4.1. *$\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) \neq \varnothing$ holds if and only if there exists some $\ell \in \mathcal{L}$ such that*

$$\sum_{i\in\mathcal{N}_\ell\cup\mathcal{N}_1} p_i \leq \epsilon, \ \textit{where } \mathcal{N}_\ell = \left\{i \in \mathcal{N}_f \,:\, d_\ell(\xi^i) = 0\right\}. \tag{41}$$

Proof. Necessity. Suppose that $(v, z) \in \mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) \neq \varnothing$. By (39), there must exist some $\ell \in \mathcal{L}$ such that $d_\ell(v) = 1$. For $i \in \mathcal{N}_\ell$, we have $d_\ell(\xi^i) = 0$, which together with $d_\ell(v) = 1$ and the fact that $d_\ell(v^1) \leq d_\ell(v^2)$ holds for any $v^1, v^2 \in \mathbb{R}^m_+$ with $v^1 \geq v^2$, implies $v \not\geq \xi^i$, and thus $z_i = 1$. Combining with $z_j = 1$ for $j \in \mathcal{N}_1$ and (3), this indicates

$$\sum_{i\in\mathcal{N}_\ell\cup\mathcal{N}_1} p_i = \sum_{i\in\mathcal{N}_\ell\cup\mathcal{N}_1} p_i z_i \leq \sum_{i=1}^{n} p_i z_i \leq \epsilon.$$

Sufficiency. Suppose that (41) holds for some $\ell \in \mathcal{L}$. We define a point $(\hat{v}, \hat{z}) \in \mathbb{R}^m_+ \times \{0, 1\}^n$ as follows:

$$\hat{v}_k = \begin{cases} \displaystyle\min_{\ell_j = k,\ j\in\mathcal{N}_1} \{\xi^j_{\ell_j}\} - \delta, & \text{if } k \in \mathcal{M}'; \\ \displaystyle\max_{i\in[n]} \{\xi^i_k\}, & \text{otherwise,} \end{cases} \quad \forall\ k \in [m], \tag{42}$$

$$\hat{z}_i = \begin{cases} 1, & \text{if } i \in \mathcal{N}_1 \cup \mathcal{N}_\ell; \\ 0, & \text{otherwise,} \end{cases} \quad \forall\ i \in [n], \tag{43}$$

where $\delta > 0$ is a sufficiently small value and $\mathcal{M}' = \{\ell_j : j \in \mathcal{N}_1\}$. From $\ell \in \mathcal{L}$, we have $\ell_j \in \mathcal{M}_j$ and $\xi^j_{\ell_j} > \xi^{\mathcal{N}_0}_{\ell_j} \geq 0$ for all $j \in \mathcal{N}_1$. By (42) and the fact that $\delta > 0$ is sufficiently small, we can derive $\hat{v} \geq \xi^{\mathcal{N}_0} \geq 0$. Together with (41) and (43), this implies that (3) and (4) hold at point $(\hat{v}, \hat{z})$. In the following, we shall prove $(\hat{v}, \hat{z}) \in \mathcal{C}(\mathcal{N}_0, \mathcal{N}_1)$ by showing that constraints (2) and (17) hold at $(\hat{v}, \hat{z})$.

To prove that (2) holds at $(\hat{v}, \hat{z})$, it suffices to show $\hat{v} \geq \xi^i$ for every $i \in \mathcal{N}_0 \cup (\mathcal{N}_f \backslash \mathcal{N}_\ell)$. For $i \in \mathcal{N}_0$, $\hat{v} \geq \xi^i$ follows from $\hat{v} \geq \xi^{\mathcal{N}_0}$. Now consider the case $i \in \mathcal{N}_f \backslash \mathcal{N}_\ell$. First, from the definition of $\hat{v}$ in (42), $\hat{v}_k \geq \xi^i_k$ holds for all $k \in [m] \backslash \mathcal{M}'$. Second, by the definition of $\mathcal{N}_\ell$ in (41) and $i \in \mathcal{N}_f \backslash \mathcal{N}_\ell$, we obtain $d_\ell(\xi^i) = \bigwedge_{j \in \mathcal{N}_1} \left(\xi^i_{\ell_j} < \xi^j_{\ell_j} \right) = 1$, which, together with the fact that $\delta > 0$ is a sufficiently small value, implies $\xi^i_{\ell_j} \leq \xi^j_{\ell_j} - \delta$ for all $j \in \mathcal{N}_1$. Therefore, for $k \in \mathcal{M}'$, it also follows $\xi^i_k \leq \min_{\ell_j = k,\ j \in \mathcal{N}_1} \{\xi^j_{\ell_j}\} - \delta = \hat{v}_k$.

Finally, we show that (17) holds at point $(\hat{v}, \hat{z})$, which can be done by proving $\hat{v} \ngeq \xi^i$ for all $i \in \mathcal{N}_1 \cup \mathcal{N}_\ell$. From the definition of $\hat{v}$ in (42) and $\delta > 0$, it is simple to see $d_\ell(\hat{v}) = 1$ and $\hat{v} \ngeq \xi^i$ for all $i \in \mathcal{N}_1$. For $i \in \mathcal{N}_\ell$, we have $d_\ell(\xi^i) = 0$, which together with $d_\ell(\hat{v}) = 1$ and the fact that $d_\ell(v^1) \leq d_\ell(v^2)$ holds for any $v^1, v^2 \in \mathbb{R}^m_+$ with $v^1 \geq v^2$, implies $\hat{v} \ngeq \xi^i$. $\square$

Theorem 4.1 enables to determine whether $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) \neq \varnothing$ by solving an MILP problem. Specifically, for $j \in \mathcal{N}_1$ and $i \in \mathcal{N}_f$, let $\mathcal{M}_{ji} = \{k \in \mathcal{M}_j : \xi^j_k \leq \xi^i_k\}$; for $j \in \mathcal{N}_1$ and $k \in \mathcal{M}_j$, let $w_{jk} \in \{0,1\}$ denote whether $v_k < \xi^j_k$ is included in (40); and for $i \in \mathcal{N}_f$, let $z_i \in \{0,1\}$ denote whether $i \in \mathcal{N}_\ell$ holds. Then it follows from Theorem 4.1 that

Corollary 4.2. *$(v,z) \in \mathcal{C}(\mathcal{N}_0, \mathcal{N}_1)$ holds if and only if there exists a vector w for which (w,z) satisfies*

$$\sum_{k \in \mathcal{M}_j} w_{jk} = 1,\ \forall\ j \in \mathcal{N}_1, \tag{44}$$

$$\sum_{k \in \mathcal{M}_{ji}} w_{jk} \leq z_i,\ \forall\ j \in \mathcal{N}_1,\ i \in \mathcal{N}_f, \tag{45}$$

$$\sum_{i \in \mathcal{N}_f} p_i z_i \leq \epsilon - \sum_{j \in \mathcal{N}_1} p_j, \tag{46}$$

$$w_{jk} \in \{0,1\},\ \forall\ j \in \mathcal{N}_1,\ k \in \mathcal{M}_j,\ z_i \in \{0,1\},\ \forall\ i \in \mathcal{N}_f. \tag{47}$$

Therefore, to determine whether $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) \neq \varnothing$, we can solve the following MILP problem

$$o = \min_{w,\, z} \left\{ \sum_{i \in \mathcal{N}_f} p_i z_i \ :\ (44),\ (45),\ (47) \right\}. \tag{48}$$

If $o \leq \epsilon - \sum_{j \in \mathcal{N}_1} p_j$, then $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) \neq \varnothing$ and the optimal solution z can define a feasible solution (v,z) of $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1)$; otherwise, $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) = \varnothing$.

Next, we attempt to derive variable fixings for the case $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) \neq \varnothing$. Let $(v,z) \in \mathcal{C}(\mathcal{N}_0, \mathcal{N}_1)$. Observe that $z_i = 0$ (respectively, $z_i = 1$) holds for all $(v,z) \in \mathcal{C}(\mathcal{N}_0, \mathcal{N}_1)$ if and only if $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1 \cup \{i\}) = \varnothing$ (respectively, $\mathcal{C}(\mathcal{N}_0 \cup \{i\}, \mathcal{N}_1) = \varnothing$) holds for $i \in \mathcal{N}_f$. Hence, the largest subsets $\mathcal{R}_0$ and $\mathcal{R}_1$ satisfying (37) can be written as

$$\mathcal{R}_0 = \{i \in \mathcal{N}_f : \mathcal{C}(\mathcal{N}_0, \mathcal{N}_1 \cup \{i\}) = \varnothing\} \text{ and } \mathcal{R}_1 = \{i \in \mathcal{N}_f : \mathcal{C}(\mathcal{N}_0 \cup \{i\}, \mathcal{N}_1) = \varnothing\}.$$

As a result, determining the largest subsets $\mathcal{R}_0$ and $\mathcal{R}_1$ satisfying (37) can be done by solving $2|\mathcal{N}_f|$ MILPs of the form (48). Notice that for $i \in \mathcal{N}_f \backslash (\mathcal{R}_0 \cup \mathcal{R}_1)$, both $\mathcal{C}(\mathcal{N}_0 \cup \mathcal{R}_0 \cup \{i\}, \mathcal{N}_1 \cup \mathcal{R}_1) \neq \varnothing$ and $\mathcal{C}(\mathcal{N}_0 \cup \mathcal{R}_0, \mathcal{N}_1 \cup \mathcal{R}_1 \cup \{i\}) \neq \varnothing$ must hold. Therefore,

Remark 4.3. *If the overlap-oriented variable fixing is performed to find the largest subsets $\mathcal{R}_0$ and $\mathcal{R}_1$ satisfying (37) at all nodes of the search tree, then no overlap-oriented node pruning can be performed. That is, for any node $(\mathcal{N}_0, \mathcal{N}_1)$ in the search tree, it must follow $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) \neq \varnothing$.*

Example 4.4 (continued)**.** *Applying the dominance-based branching and overlap-oriented variable fixing to the problem in Example 2.1, we obtain the search tree in Figure 3. All variables fixed to 0 or 1 by overlap-oriented variable fixing are underlined in Figure 3.*

Let us perform variable fixing at node 5 where, at the beginning, $\mathcal{N}_0 := \mathcal{B}_0 = \{4\}$ and $\mathcal{N}_1 := \mathcal{B}_1 = \{5\}$. By simple computations, we have $\mathcal{M}_5 = \{1\}$, $\mathcal{M}_{51} = \mathcal{M}_{52} = \mathcal{M}_{53} = \varnothing$, and $\mathcal{M}_{56} = \mathcal{M}_{57} = \{1\}$. Hence, problem (48) reduces to

$$\min \left\{ \frac{1}{7} \sum_{i \in [7] \backslash \{4,5\}} z_i \,:\, w_{51} = 1, w_{51} \leq z_6, w_{51} \leq z_7, w_{51} \in \{0,1\}, z_i \in \{0,1\}, \forall\, i \in [7] \backslash \{4,5\} \right\}.$$

As the optimal value of the above problem is $\frac{2}{7} \leq \epsilon - \sum_{j \in \mathcal{N}_1} p_j = \frac{3}{7}$, from Corollary 4.2 $\mathcal{C}(\{4\},\{5\}) \neq \varnothing$ holds. Similarly, we can show that $\mathcal{C}(\{4,6\},\{5\}) = \mathcal{C}(\{4,7\},\{5\}) = \mathcal{C}(\{4\},\{2,5\}) = \mathcal{C}(\{4\},\{3,5\}) = \varnothing$, $\mathcal{C}(\{1,4\},\{5\}) \neq \varnothing$, and $\mathcal{C}(\{4\},\{1,5\}) \neq \varnothing$. Thus, we can fix $z_2 = z_3 = 0$ and $z_6 = z_7 = 1$ at node 5.

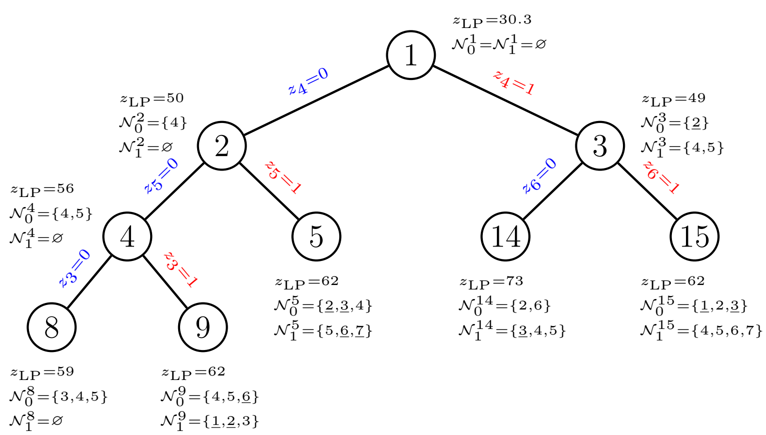

Figure 3: The B&C search tree of the problem in Example 2.1 with the proposed dominance-based branching and overlap-oriented variable fixing applied.

The above example helps to demonstrate the advantage of the overlap-oriented variable fixing. For instance, for node 5 in the search tree in Figure 3, the overlap-oriented variable fixing improves the LP relaxation bound from 50 to 62 and avoids further branching. Overall, applying the dominance-based branching and overlap-oriented variable fixing, only 9 nodes are explored while applying the vanilla dominance-based branching, 19 nodes are explored; see Figures 2 and 3.

To apply the overlap-oriented node pruning and variable fixing, we need to determine whether the system of the form (44)–(47) has a feasible solution (which can be done by solving MILP problems of the form (48)). Unfortunately, the following theorem shows that there does not exist a polynomial-time exact algorithm for solving the above problem.

Theorem 4.5. *Given $\mathcal{N}_0, \mathcal{N}_1 \subseteq [n]$, determining whether (44)–(47) has a feasible solution is strongly NP-complete.*

Proof. We shall prove the strong NP-completeness of the problem of deciding whether (44)–(47) has a feasible solution by establishing a polynomial-time reduction from the strongly NP-hard problem: set covering (SC) problem [24]. We first introduce the SC problem: given t subsets $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_t$ of $\mathcal{J}$ where $\mathcal{J}$ is a finite set of r elements, does there exist $\mathcal{S} \subseteq [t]$ such that $|\mathcal{S}| \leq B$ and $\cup_{i\in\mathcal{S}}\mathcal{J}_i = \mathcal{J}$? The SC problem is equivalent to deciding whether the following system

$$\sum_{i=1}^{t} z_i \leq B,\ \sum_{i\in\mathcal{I}_j} z_i \geq 1,\ \forall\, j \in \mathcal{J},\ z_i \in \{0,1\},\ \forall\, i \in [t], \tag{49}$$

has a feasible solution z, where $\mathcal{I}_j = \{i \in [t] : j \in \mathcal{J}_i\}$. For notations purpose, we denote $\mathcal{J} = \{t+1, t+2, \ldots, t+r\}$.

Given any instance of the SC problem, we construct an instance of deciding whether (44)–(47) has a feasible solution as follows:

(i) $n := t+r$, $m := t$, $p_i := \frac{1}{t+r}$, $i \in [t+r]$, $\epsilon := \frac{B+r}{t+r}$, $\mathcal{N}_0 := \varnothing$, and $\mathcal{N}_1 := \mathcal{J} = \{t+1, t+2, \ldots, t+r\}$;

(ii) For each $i \in [t]$, we define $\xi^i := \boldsymbol{e}_i$ where $\boldsymbol{e}_i$ is the i-th unit vector of dimension t, and for each $j \in \{t+1, t+2, \ldots, t+r\}$, we define ξ^j as follows:

$$\xi^j_k := \begin{cases} 1, & \text{if } k \in \mathcal{I}_j; \\ 0, & \text{otherwise,} \end{cases} \quad \forall\, k \in [t]. \tag{50}$$

By the definitions of p_i, ϵ, $\mathcal{N}_0$, and $\mathcal{N}_1$, (46) reduces to

$$\sum_{i=1}^{t} z_i \leq B. \tag{51}$$

As $\mathcal{N}_0 = \varnothing$, it follows $\xi^{\mathcal{N}_0} = \mathbf{0}$. For $j \in \mathcal{N}_1$, it follows from (50) that $\mathcal{M}_j = \{k \in [t] : \xi^j_k > 0\} = \{k \in [t] : \xi^j_k = 1\} = \mathcal{I}_j$. Thus (44) and (47) reduce to

$$\sum_{i\in\mathcal{I}_j} w_{ji} = 1,\ \forall\, j \in \mathcal{N}_1, \tag{52}$$

$$w_{ji} \in \{0,1\},\ \forall\, j \in \mathcal{N}_1,\ i \in \mathcal{I}_j,\ z_i \in \{0,1\},\ \forall\, i \in [t]. \tag{53}$$

For $j \in \mathcal{N}_1$ and $i \in [t]$, we have $\mathcal{M}_{ji} = \{k \in \mathcal{M}_j : \xi^j_k \leq \xi^i_k\} = \{k \in \mathcal{I}_j : \xi^j_k \leq \xi^i_k\} = \{k \in \mathcal{I}_j : 1 \leq \xi^i_k\}$, which, together with $\xi^i = \boldsymbol{e}_i$, implies

$$\mathcal{M}_{ji} = \begin{cases} \{i\}, & \text{if } i \in \mathcal{I}_j; \\ \varnothing, & \text{otherwise.} \end{cases}$$

Therefore, (45) reduces to

$$w_{ji} \leq z_i,\ \forall\, j \in \mathcal{N}_1,\ i \in \mathcal{I}_j. \tag{54}$$

Observe that (x, z) satisfies (52)–(54) if and only if $\sum_{i\in\mathcal{I}_j} z_i \geq 1$ and $z_i \in \{0,1\}$ hold for all $j \in \mathcal{N}_1$ and $i \in [t]$, respectively. Therefore, (51)–(54) has a feasible solution if and only if (49) has a feasible solution, which completes the proof. □

Due to the negative result in Theorem 4.5, we shall develop an efficient heuristic algorithm to apply the overlap-oriented node pruning and variable fixing in the following.

4.2 The polynomial-time approximation approach

In this subsection, we present a polynomial-time approximation approach to apply overlap-oriented node pruning and variable fixing. The proposed approach iteratively performs the following two steps until no more reduction is found: (i) deriving lower bounds for variables v from the constraints in $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1)$, and (ii) using the derived lower bounds to identify when $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) = \varnothing$ or determine $\mathcal{R}_0$ and $\mathcal{R}_1$ satisfying (37). Before going into the details, we note that if $\epsilon - \sum_{i\in\mathcal{N}_1} p_i < 0$, it must follow $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) = \varnothing$, and therefore, to perform the reductions, it suffices to consider the case

$$\epsilon - \sum_{i\in\mathcal{N}_1} p_i \geq 0. \tag{55}$$

4.2.1 Deriving lower bounds for variables v

As noted in Section 4.1, using constraints (2) and $z_i = 0$ for $i \in \mathcal{N}_0$ in $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1)$, we obtain the following lower bounds for variables v:

$$v_k \geq \xi_k^{\mathcal{N}_0}, \ \forall\, k \in [m], \tag{56}$$

where we recall that $\xi_k^{\mathcal{N}_0} = \max_{i\in\mathcal{N}_0}\{\xi_k^i\}$. Another way to derive lower bounds for variables v is to use the technique as in Section 3.3. Specifically, for each $k \in [m]$, let $\pi_k(1), \pi_k(2), \ldots, \pi_k(|\mathcal{N}_f|)$ be a permutation of $\mathcal{N}_f$ satisfying

$$\xi_k^{\pi_k(1)} \geq \xi_k^{\pi_k(2)} \geq \cdots \geq \xi_k^{\pi_k(|\mathcal{N}_f|)}, \tag{57}$$

and let

$$\tau_k := \min\left\{ s \in [|\mathcal{N}_f|] \ : \ \sum_{i=1}^{s} p_{\pi_k(i)} > \epsilon - \sum_{i\in\mathcal{N}_1} p_i \right\}. \tag{58}$$

By equation (55), we must have $\tau_k \geq 1$. Using the same technique in Section 3.3, the following must hold:

$$v_k \geq \xi_k^{\pi_k(\tau_k)}, \ \forall\, k \in [m]. \tag{59}$$

Combining (56) and (59), we can obtain tighter lower bounds for variables v:

$$v_k \geq \xi_k^{\mathcal{N}_0,\mathcal{N}_1} := \max\left\{\xi_k^{\mathcal{N}_0}, \xi_k^{\pi_k(\tau_k)}\right\}, \ \forall\, k \in [m]. \tag{60}$$

It is worthy noting that the lower bounds $\xi_k^{\mathcal{N}_0,\mathcal{N}_1}$ depend on both $\mathcal{N}_0$ and $\mathcal{N}_1$: the larger the $\mathcal{N}_0$ and $\mathcal{N}_1$, the tighter the lower bounds $\xi_k^{\mathcal{N}_0,\mathcal{N}_1}$.

Obviously, the computation of $\xi_k^{\mathcal{N}_0}$, $k \in [m]$, in (56) can be done in the complexity of $\mathcal{O}(m|\mathcal{N}_0|)$. As for the computation of $\xi_k^{\pi_k(\tau_k)}$, $k \in [m]$, we need to, for each $k \in [m]$, sort ξ_k^i, $i \in \mathcal{N}_f$, satisfying (57) and then determine τ_k satisfying (58), which can be implemented with the complexity of $\mathcal{O}(m|\mathcal{N}_f|\log(|\mathcal{N}_f|))$. Therefore, the complexity for the computation of lower bounds for variables v is $\mathcal{O}(m|\mathcal{N}_0| + m|\mathcal{N}_f|\log(|\mathcal{N}_f|))$.

4.2.2 Node pruning and variable fixing

Using the lower bounds of variables v in (60), we are able to give a sufficient condition under which node $(\mathcal{N}_0, \mathcal{N}_1)$ can be pruned, as detailed in the following proposition.

Proposition 4.6. *Let $\mathcal{N}_0$, $\mathcal{N}_1 \subseteq [n]$ be such that $\mathcal{N}_0 \cap \mathcal{N}_1 = \varnothing$. If $\xi^j \leq \xi^{\mathcal{N}_0,\mathcal{N}_1}$ for some $j \in \mathcal{N}_1$, then $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) = \varnothing$.*

Proof. Let $j \in \mathcal{N}_1$ such that $\xi^j \leq \xi^{\mathcal{N}_0,\mathcal{N}_1}$. Suppose, otherwise, that $(v, z) \in \mathcal{C}(\mathcal{N}_0, \mathcal{N}_1)$. By (60), we have $v \geq \xi^{\mathcal{N}_0,\mathcal{N}_1} \geq \xi^j$, which, together with $z_j \in \{0, 1\}$ and (17), implies that $z_j = 0$. However, this contradicts with $z_j = 1$ (as $j \in \mathcal{N}_1$) and thus $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) = \varnothing$. □

Next, we attempt to perform variable fixing for the case that the condition in Proposition 4.6 does not hold, i.e.,

$$\xi^j \nleq \xi^{\mathcal{N}_0,\mathcal{N}_1}, \ \forall \ j \in \mathcal{N}_1. \tag{61}$$

To begin with, we note that using the lower bounds of variables v in (60), constraints $v \ngeq \xi^j$, or equivalently, $\bigvee_{k\in[m]} \left(v_k < \xi^j_k\right) = 1$ in $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1)$ can be simplified as

$$c_j(v) := \bigvee_{k\in\bar{\mathcal{M}}_j} \left(v_k < \xi^j_k\right) = 1, \ \forall \ j \in \mathcal{N}_1, \tag{62}$$

where

$$\bar{\mathcal{M}}_j := \left\{k \in [m] \ : \ \xi^j_k > \xi^{\mathcal{N}_0,\mathcal{N}_1}_k\right\}. \tag{63}$$

Let

$$\begin{aligned} \mathcal{R}_1 &:= \left\{i \in \mathcal{N}_f \ : \ c_j(\xi^i) = 0 \text{ holds for some } j \in \mathcal{N}_1\right\}, \\ \mathcal{R}_0 &:= \left\{i \in \mathcal{N}_f \ : \ p_i > \epsilon - \sum_{j\in\mathcal{N}_1\cup\mathcal{R}_1} p_j \text{ or } \xi^i \leq \xi^{\mathcal{N}_0,\mathcal{N}_1}\right\}. \end{aligned} \tag{64}$$

Note that compared with the definition $\mathcal{M}_j$ in Section 4.1, we use the tighter lower bounds $\xi^{\mathcal{N}_0,\mathcal{N}_1}_k$ for the definition of $\bar{\mathcal{M}}_j$, and thus $\bar{\mathcal{M}}_j \subseteq \mathcal{M}_j$. Also note that the tighter the lower bounds $\xi^{\mathcal{N}_0,\mathcal{N}_1}_k$, the smaller the $\bar{\mathcal{M}}_j$, and the larger the $\mathcal{R}_0$ and $\mathcal{R}_1$.

The following proposition shows that for node $(\mathcal{N}_0, \mathcal{N}_1)$, variables z_i for $i \in \mathcal{R}_0$ can be fixed to zero; and variables z_i for $i \in \mathcal{R}_1$ can be fixed to one.

Proposition 4.7. *Let $\mathcal{N}_0$, $\mathcal{N}_1 \subseteq [n]$ be such that $\mathcal{N}_0 \cap \mathcal{N}_1 = \varnothing$ and (61) hold, and $\mathcal{R}_0$ and $\mathcal{R}_1$ be defined in (64). Then $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) = \mathcal{C}(\mathcal{N}_0 \cup \mathcal{R}_0, \mathcal{N}_1 \cup \mathcal{R}_1)$.*

Proof. We shall show that for any $(v, z) \in \mathcal{C}(\mathcal{N}_0, \mathcal{N}_1)$, $z_i = 1$ and $z_i = 0$ hold for $i \in \mathcal{R}_1$ and $i \in \mathcal{R}_0$, respectively.

(i) For $i \in \mathcal{R}_1$, $c_j(\xi^i) = 0$ holds for some $j \in \mathcal{N}_1$. By (62), we have $c_j(v) = 1$. Then $v \ngeq \xi^i$ follows from the definition of $c_j(\cdot)$ in (62), that is, for any $v^1, v^2 \in \mathbb{R}^m_+$ with $v^1 \leq v^2$, it follows $c_j(v^2) \leq c_j(v^1)$. Together with $z_i \in \{0, 1\}$ and (2), this implies $z_i = 1$.

(ii) For $i \in \mathcal{R}_0$, if $p_i > \epsilon - \sum_{j\in\mathcal{N}_1\cup\mathcal{R}_1} p_j$, then it follows from (3) and $z_j = 1$ for all $j \in \mathcal{N}_1 \cup \mathcal{R}_1$ that $z_i = 0$. Otherwise, $\xi^i \leq \xi^{\mathcal{N}_0,\mathcal{N}_1}$, and by (60), $v \geq \xi^{\mathcal{N}_0,\mathcal{N}_1} \geq \xi^i$ holds. Together with $z_i \in \{0, 1\}$ and (17), this implies $z_i = 0$. □

Notice that using Proposition 4.6 to determine whether $\mathcal{C}(\mathcal{N}_0, \mathcal{N}_1) = \varnothing$ and computing $c_j(v)$, $j \in \mathcal{N}_1$, can all be done in the complexity of $\mathcal{O}(m|\mathcal{N}_1|)$. The computations of $\mathcal{R}_0$ and $\mathcal{R}_1$ in (64) can be implemented with the complexity of $\mathcal{O}(m|\mathcal{N}_1||\mathcal{N}_f|)$. Therefore, the overall complexity of performing node pruning and variable fixing is $\mathcal{O}(m|\mathcal{N}_1||\mathcal{N}_f|)$.

4.2.3 The overall algorithmic framework

After performing the variable fixing in Section 4.2.2, we may compute tighter lower bounds for variables v using the procedure Section 4.2.1 again, which, in turn, opens up new possibilities to detect more reductions using the procedure in Section 4.2.2. We illustrate this using the following example.

Example 4.8. *Let us consider node* 5 *in Figure 2 where the dominance-based branching is applied to solve the problem in Example 2.1. At the beginning,* $\mathcal{N}_0 := \mathcal{B}_0 = \{4\}$ *and* $\mathcal{N}_1 := \mathcal{B}_1 = \{5\}$*. Applying the procedure in Section 4.2.1, we obtain the lower bounds* $\xi^{\mathcal{N}_0,\mathcal{N}_1} = (5,\, 2,\, 6)^\top$ *for variables* v*. Applying the procedure in Section 4.2.2, we obtain* $\mathcal{R}_0 = \varnothing$ *and* $\mathcal{R}_1 = \{6, 7\}$*, and thus* $\mathcal{C}(\{4\}, \{5\}) = \mathcal{C}(\{4\}, \{5, 6, 7\})$*.*

Now letting $\mathcal{N}_0 := \{4\}$ *and* $\mathcal{N}_1 := \{5, 6, 7\}$*, applying the procedure in Section 4.2.1 again, we obtain the tighter lower bounds* $\xi^{\mathcal{N}_0,\mathcal{N}_1} = (5,\, 2,\, 10)^\top$ *for variables* v*. Similarly, applying the procedure in Section 4.2.2 again, we can obtain* $\mathcal{R}_0 = \{2, 3\}$ *and* $\mathcal{R}_1 = \varnothing$*, and thus* $\mathcal{C}(\{4\}, \{5, 6, 7\}) = \mathcal{C}(\{2, 3, 4\}, \{5, 6, 7\})$*.*

Example 4.8 offers a hint to the algorithmic design for the overlap-oriented node pruning and variable fixing. Specifically, we can iteratively apply the procedures in Sections 4.2.1 and 4.2.2 to detect reductions for node $(\mathcal{N}_0, \mathcal{N}_1)$ until no more reduction is found (i.e., either the node is pruned or $\mathcal{R}_0 = \mathcal{R}_1 = \varnothing$). The details are summarized in Algorithm 1. Note that in one iteration of Algorithm 1, either the procedure is terminated or at least one more variable is fixed. Therefore, the number of iterations in Algorithm 1 is at most $|\mathcal{N}_f|$, and the worst-case complexity of Algorithm 1 is polynomial. Moreover, as will be demonstrated in Section 5, Algorithm 1 is indeed competitive with the exact approach in Section 4.1 in terms of reducing the tree size while enjoying a high computational efficiency.

Algorithm 1: An iterative procedure for overlap-oriented node pruning and variable fixing

Input: Node $(\mathcal{N}_0, \mathcal{N}_1)$.
repeat
    If $\epsilon - \sum_{i \in \mathcal{N}_1} p_i < 0$, **stop** and claim that the node is infeasible;
    Compute the lower bounds $\xi^{\mathcal{N}_0,\mathcal{N}_1}$ as in (60);
    If $\xi^j \leq \xi^{\mathcal{N}_0,\mathcal{N}_1}$ for some $j \in \mathcal{N}_1$, **stop** and claim that the node is infeasible;
    Compute $\mathcal{R}_0$ and $\mathcal{R}_1$ as in (64);
    Update $\mathcal{N}_0 \leftarrow \mathcal{N}_0 \cup \mathcal{R}_0$ and $\mathcal{N}_1 \leftarrow \mathcal{N}_1 \cup \mathcal{R}_1$;
until *no more reduction is found*;

5 Computational results

In this section, we present computational results to illustrate the effectiveness of the proposed dominance-based branching and overlap-oriented node pruning and variable fixing techniques. The proposed methods were implemented in C language and linked to the state-of-the-art open source MILP solver SCIP 8.0.0 [13], using SoPlex 6.0.0 as the LP solver. SCIP includes a routine of domain propagation methods [3] that is applied at each node of the search tree to tighten the variable bounds (including variable fixings) or detect infeasible subproblems. Therefore, we implemented the proposed dominance-based branching and overlap-oriented node pruning and variable fixing techniques as domain propagation methods. With this implementation, other sophisticated components of SCIP including the default fine-tuned branching strategy (called *hybrid branching*) can still be used. Moreover, as demonstrated in [22, 23], when selecting the branching variables, the default branching strategy of SCIP will invoke the domain propagation methods; and thus, this implementation also enables the proposed dominance-based branching and overlap-oriented node pruning and variable fixing techniques to guide the selection of branching variables.

In order to improve the overall solution efficiency, we follow [38] to apply the preprocessing technique to strengthen formulation (MILP) of problem (CCP). Specifically, since ξ^0 are the lower bounds for variables v (see Lemma 3.5), we can remove constraint $v_k \geq \xi_k^i(1 - z_i)$ from the problem formulation if $\xi_k^i \leq \xi_k^0$ and strengthen it as $v_k \geq \xi_k^i - (\xi_k^i - \xi_k^0) z_i$ otherwise. In addition, we also implemented the (strengthened) mixing cuts [7, 25], which is recognized as one of the most effective cutting planes for solving formulation (MILP) of problem (CCP) [38], to speed up the solution process.

In our computational study, we consider three CCPs studied in the literature, which are the CCRP problem [26, 36], CCMPP problem [21], and CCLS problem [12]. Our testset consists of 135 CCRP instances, 135 CCMPP instances, and 180 CCLS instances. The descriptions of the problems and the instance construction procedures are provided in Appendix B. Except where explicitly stated, all computations were performed on a cluster of Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz computers running Linux, with a time limit of 4 hours and a relative gap of 0%. Throughout this section, all averages are reported as the shifted geometric mean with shifts of 1 second and 100 nodes for the CPU time and the number of explored nodes, respectively. The shifted geometric mean of values $x_1, x_2, \ldots, x_n$ with shift s is defined as $\prod_{k=1}^n (x_k + s)^{1/n} - s$; see [3].

5.1 Comparison of the dominance-based branching with the direct use of dominance inequalities

In this subsection, we compare the performance of the dominance-based branching with the direct use of dominance inequalities. Specifically, we compare the performance of the following two settings.

- `DB`: solving formulation (MILP) using the B&C algorithm with the proposed dominance-based branching (but without the overlap-oriented node pruning and variable fixing in Section 4);
- `DI`: solving formulation (MILP) with the dominance inequalities in (32) using the B&C algorithm with the classic variable branching;

For benchmarking purposes, we also report the results of solving formulation (MILP) with the weaker version of dominance inequalities in (27) (in the sense that inequalities in (32) imply all inequalities in (27)), denoted as `w-DI`. When adding the dominance inequalities into formulation (MILP), we only add the nonredundant ones; see Remarks 3.2 and 3.6.

Table 1 summarizes the computational results of the three problems [1]. In column `#`, we report

[1]Throughout, we report the aggregated results (for each of the three problems). Detailed statistics of instance-wise computational results can be found in the online supplement available at https://drive.google.com/file/d/1hZnv0jgoFUjyIS7Fwyo6bA_6p1tu9yil/view?usp=share_link.

the number of instances that can be solved to optimality within the time limit by at least one setting. For each setting, we report the number of solved instances (`S`), the average CPU time in seconds (`T`) (which includes the CPU time spent in the implementation of dominance-based branching), and the average number of explored nodes (`N`). For settings `w-DI` and `DI`, we additionally report the percentage of dominance pairs between scenarios and the percentage of nonredundant dominance inequalities, defined by $\Delta\mathtt{DP} := \mathtt{\#DP}/\mathtt{\#MDP} \times 100$ and $\Delta\mathtt{NDI} := \mathtt{\#NDI}/\mathtt{\#MDP} \times 100$, respectively. Here, `#DP` is the number of dominance pairs, $\mathtt{\#MDP} = n(n-1)/2$ is the maximum number of possible dominance pairs, and `#NDI` is the number of nonredundant dominance inequalities.

Table 1: Performance comparison of settings `DB`, `DI`, and `w-DI`.

Probs	#	DB			DI					w-DI				
		S	T	N	S	T	N	ΔDP	ΔNDI	S	T	N	ΔDP	ΔNDI
CCRP	110	**109**	**156.6**	120	108	210.9	**82**	51.7	1.2	100	284.2	202	5.0	4.3
CCMPP	110	**110**	**35.9**	330	108	44.7	**273**	40.3	0.2	83	294.4	3843	0.1	0.1
CCLS	174	**174**	**134.9**	631	170	182.0	**493**	90.0	0.4	160	234.3	745	34.8	7.3
All	394	**393**	**97.4**	351	386	128.4	**275**	63.9	0.6	343	263.6	875	15.7	4.4

We first compare the performance of settings `DB` and `DI`. As shown in Section 3.2, the two settings are theoretically equivalent under certain conditions in terms of exploring the same search trees. In Table 1, we observe that the numbers of explored nodes returned by the two settings are similar; overall, `DI` returns a slightly smaller number of explored nodes than `DB`. Notice that the latter is reasonable as directly adding the dominance inequalities into the formulation can enhance other components of MILP solvers, such as triggering more internal cuts generation and more conflict analysis [2, 58]. However, the smaller number of explored nodes returned `DI` cannot compensate for the overhead of a large problem size, as the dominance inequalities in (32) need to be added as constraints into the formulation. In contrast, the proposed `DB` avoids solving a problem with a large problem size and thus achieves an overall better performance. In particular, the proposed `DB` enables to solve 7 more instances to optimality and to reduce the CPU time by a factor of 1.3.

Next, we can observe from Table 1 that the number of dominance pairs in (32) is much larger than that in (27), which shows the effectiveness of the preprocessing technique in Section 3.3 in detecting more dominance pairs. Although the number of dominance pairs in (32) is very large, only a small proportion of the inequalities (i.e., the nonredundant ones) needs to be added into the formulation; in some cases, the number of nonredundant inequalities in (32) is even smaller than that of the nonredundant ones in (27). Due to these two advantages, `DI` performs much better than `w-DI`. In particular, using `DI`, 43 more instances can be solved to optimality, and the CPU time and the number of explored nodes are reduced by factors of 2.1 and 3.2, respectively.

5.2 Performance effect of the overlap-oriented node pruning and variable fixing

We now evaluate the performance effect of the overlap-oriented node pruning and variable fixing in Section 4. Table 2 summarizes the computational results of settings `DB` and `DB+AOPF`, where `DB+AOPF` is denoted by

- `DB+AOPF`: solving formulation (MILP) using the B&C algorithm with the proposed dominance-based branching and the approximation approach of the overlap-oriented node pruning and variable fixing in Section 4.2.

In Table 2, column `PT` reports the CPU time spent in the implementation of the proposed dominance-based branching (with the approximation approach of the overlap-oriented node pruning and variable fixing). The rows "$\geq s$" collect the subsets of instances that can be solved by at least one setting within the time limit and for which the number of explored nodes returned by at least one of the two settings is at least s. With the increasing s, this provides a hierarchy of subsets of increasing difficulty (as the tree size is large).

As shown in Table 2, the overlap-oriented node pruning and variable fixing can effectively improve the performance of solving all three problems while the computational overhead of the approximation approach for implementing them is fairly small. In total, equipped with the overlap-oriented node pruning and variable fixing, `DB+AOPF` can solve 27 more instances to optimality. Additionally, the average CPU time and number of explored nodes are reduced by factors of 1.3 and 1.9, respectively. For hard instances that require to explore at least 1000 nodes by at least one of the two settings, we can even observe a factor of 1.7 runtime speed-up and a factor of 2.7 tree size reduction. This shows the effectiveness of the overlap-oriented node pruning and variable fixing in further improving the performance of the dominance-based branching for solving CCPs.

Table 2: Performance comparison of settings `DB` and `DB+AOPF`.

Probs	#	DB				DB+AOPF			
		S	T	N	PT	S	T	N	PT
CCRP	119	109	220.7	146	**1.5**	**119**	**156.3**	**66**	9.6
CCMPP	123	110	68.3	547	**1.6**	**123**	**48.0**	**182**	3.0
CCLS	178	174	149.9	685	**1.5**	**178**	**141.9**	**477**	18.0
All	420	393	133.0	434	**1.5**	**420**	**106.3**	**229**	9.2
≥ 10	312	285	287.9	842	**2.1**	**312**	**202.7**	**391**	13.2
≥ 100	245	218	542.2	1519	**2.7**	**245**	**343.0**	**629**	18.4
≥ 1000	151	124	1200.1	3764	**4.2**	**151**	**706.3**	**1381**	30.2

In order to illustrate the effectiveness of the proposed approximation approach in Section 4.2 for implementing the overlap-oriented node pruning and variable fixing, we compare it with the exact approach in Section 4.1 (denoted by `DB+EOPF`). In Table 3, we report the computational results of 287 instances that can be solved to optimally in both `DB+AOPF` and `DB+EOPF` within 24 hours. We observe that `DB+AOPF` is much more efficient than `DB+EOPF`. This is quite expected because `DB+EOPF` needs to solve several MILP problems of the form (48) to perform the reductions, which is very time-consuming, as shown in column `PT` under setting `DB+EOPF`. On the other hand, the number of explored nodes returned by `DB+EOPF` is only slightly smaller than that of `DB+AOPF`. These results demonstrate that the proposed approximation algorithm for implementing the overlap-oriented node pruning and variable fixing is competitive with the exact algorithm in terms of reducing the tree size while enjoying a high computational efficiency.

Table 3: Performance comparison of settings DB+AOPF and DB+EOPF.

Probs	#	DB+AOPF				DB+EOPF			
		S	T	N	PT	S	T	N	PT
CCRP	81	**81**	**48.6**	13	**6.7**	**81**	667.2	**12**	520.4
CCMPP	90	**90**	**16.0**	**44**	**1.4**	**90**	279.4	**44**	233.9
CCLS	116	**116**	**48.5**	147	**7.9**	**116**	1420.1	**138**	1262.9
All	287	**287**	**34.5**	67	**4.7**	**287**	689.4	**64**	579.8
≥ 10	133	**133**	**55.1**	192	**6.6**	**133**	3610.9	**180**	3440.6
≥ 100	82	**82**	**84.4**	394	**9.7**	**82**	12873.3	**367**	12763.8
≥ 1000	14	**14**	**182.5**	1391	**28.5**	**14**	34748.9	**1232**	34605.7

5.3 Comparison with the classic branch-and-cut algorithm

We now compare the performance of applying the proposed dominance-based branching and (the approximation approach of) overlap-oriented node pruning and variable fixing techniques to solve the CCP based on formulation (MILP) against the default B&C algorithm of SCIP with the classic variable branching (denoted by cB&C). The results are summarized in Table 4. We can observe from Table 4 that DB+AOPF significantly outperforms cB&C. In particular, using the proposed DB+AOPF, 73 more instances can be solved to optimality within the given 4 hours limits; the average number of exploded nodes is reduced by a factor of 5.5; and the average CPU time is reduced by a factor of 2.8. Moreover, for hard instances that require to explore at least 1000 nodes by at least one of the two settings, we can observe a drastic node reduction factor of 10.4 and a drastic runtime speed-up factor of 5.8. In addition, among the three problems, we can observe a tremendous improvement on CCMPP instances where the tree size constructed by cB&C is relatively large. Indeed, using the proposed DB+AOPF, the average number of explored nodes and CPU time decrease from 4614 and 436.5 seconds to 182 and 48.0 seconds, respectively, with 39 more solved instances. These results show the efficiency of the proposed dominance-based branching and overlap-oriented node pruning and variable fixing techniques in reducing the search tree size and improving the efficiency of solving CCPs, especially for hard instances.

Table 4: Performance comparison of settings cB&C and DB+AOPF.

Probs	#	cB&C			DB+AOPF			
		S	T	N	S	T	N	PT
CCRP	119	102	323.7	232	**119**	**156.3**	**66**	9.6
CCMPP	123	84	436.5	4614	**123**	**48.0**	**182**	3.0
CCLS	178	161	213.4	1401	**178**	**141.9**	**477**	18.0
All	420	347	296.2	1269	**420**	**106.3**	**229**	9.2
≥ 10	364	291	502.1	1938	**364**	**147.5**	**293**	10.9
≥ 100	320	247	731.2	2833	**320**	**187.1**	**370**	12.4
≥ 1000	221	148	1707.2	6842	**221**	**295.6**	**658**	16.4

6 Conclusion

In this paper, it has been shown that the presence of the overlap in the search tree makes the standard MILP formulation of the CCP difficult to be solved by state-of-the-art MILP solvers. In an effort to remedy this, we have developed several approaches to remove the overlap during the B&C process. In particular, we have showed that a family of valid nonlinear if-then constraints is able to remove all overlaps in the search tree. To tackle the highly nonlinear if-then constraints, we have proposed the dominance-based branching, which, compared with the classic variable branching, is able to partition the current problem into two MILP subproblems with much smaller feasible regions, especially when the number of dominance pairs between scenarios is large. Moreover, by considering the joint mixing set with a knapsack constraint and the if-then constraints, we have developed overlap-oriented node pruning and variable fixing, applied at each node of the search tree, to remove more overlaps in the tree. Both of the proposed dominance-based branching and overlap-oriented node pruning and variable fixing are easily to be embedded into the B&C framework along with other sophisticated components of MILP solvers. By extensive computational experiments, we have demonstrated that the proposed dominance-based branching and overlap-orient node pruning and variable fixing techniques can significantly reduce the B&C search tree size and substantially improve the computational performance of the MILP-based approach to solving CCPs.

Appendix A An example to illustrate Remark 2.2

In this section, we present an example to illustrate Remark 2.2. The example is based on the problem obtained by modifying scenario ξ^6 to $(6,\,2,\,5)^\top$ in problem (10). As such, formulation (MILP) for the modified problem reduces to

$$\min\left\{6v_1+v_2+3v_3 : v \geq \xi^i(1-z_i),\ \forall\, i\in[7],\ \frac{1}{7}\sum_{i=1}^{7} z_i \leq \frac{4}{7},\ v\in\mathbb{R}^3_+,\ z\in\{0,1\}^7\right\}, \tag{65}$$

where

$$\xi^1=\begin{pmatrix}2\\1\\12\end{pmatrix},\ \xi^2=\begin{pmatrix}3\\1\\10\end{pmatrix},\ \xi^3=\begin{pmatrix}4\\2\\7\end{pmatrix},$$
$$\xi^4=\begin{pmatrix}5\\2\\6\end{pmatrix},\ \xi^5=\begin{pmatrix}6\\2\\6\end{pmatrix},\ \xi^6=\begin{pmatrix}6\\2\\5\end{pmatrix},\ \xi^7=\begin{pmatrix}12\\1\\2\end{pmatrix}.$$

We choose variable z_4 to branch on. The left and right branches (after removing the fixed variable z_4) reduces to

$$\begin{aligned}O^{\mathrm{L}}=\min\Bigg\{&6v_1+v_2+3v_3 : \frac{1}{7}(z_1+z_2+z_3+z_5+z_6+z_7)\leq\frac{4}{7},\ v\geq\xi^4=(5,2,6)^\top,\\ &v\geq\xi^i(1-z_i),\ \forall\, i\in\{1,2,3,5,6,7\},\ v\in\mathbb{R}^3_+,\ z\in\{0,1\}^6\Bigg\},\end{aligned} \tag{66}$$

$$O^{\mathrm{R}} = \min\Bigg\{6v_1 + v_2 + 3v_3 \,:\, \frac{1}{7}(z_1 + z_2 + z_3 + z_5 + z_6 + z_7) \le \frac{3}{7},\ v \ge \mathbf{0},$$
$$v \ge \xi^i(1 - z_i),\ \forall\, i \in \{1, 2, 3, 5, 6, 7\},\ v \in \mathbb{R}^3_+,\ z \in \{0, 1\}^6\Bigg\}. \tag{67}$$

By Remark 2.2, the new left branch O^{L_2} reduces to

$$\min\Bigg\{6v_1 + v_2 + 3v_3 \,:\, \frac{3}{7} < \frac{1}{7}(z_1 + z_2 + z_3 + z_5 + z_6 + z_7) \le \frac{4}{7},\, v \ge \xi^4 = (5, 2, 6)^\top,$$
$$v \ge \xi^i(1 - z_i),\ \forall\, i \in \{1, 2, 3, 5, 6, 7\},\ v \in \mathbb{R}^3_+,\ z \in \{0, 1\}^6\Bigg\}. \tag{68}$$

Although the feasible regions of branches (67) and (68) do not contain an overlap, their projection on the v space, however, do contain overlaps. Indeed, point $v^* = (6,\, 2,\, 7)^\top$ can define feasible solutions

$$(v^*, \hat{z}) = (6,\, 2,\, 7,\, 1,\, 1,\, 0,\, 0,\, 0,\, 1)^\top \text{and } (v^*, \bar{z}) = (6,\, 2,\, 7,\, 1,\, 1,\, 0,\, 0,\, 1,\, 1)^\top$$

of the two branches (67) and (68), respectively.

Appendix B Test problems

Here we introduce the CCRP, CCMPP, and CCLS problems considered in the computational experiments and the details of the procedures to generate the instances.

B.1 Chance-constrained resource planning problem [26, 36]

Consider a set of resources $\mathcal{I}$ with unit cost c_i for each $i \in \mathcal{I}$ and a set of customers $\mathcal{J}$ with random demand $\tilde{\lambda}_j$ for each $j \in \mathcal{J}$. The CCRP problem attempts to choose the quantities of the resources and allocate these resources to customers such that the total cost of resources is minimized, while requiring that the allocation does not exceed the available resources and meets the customer demand with a probability at least $1 - \epsilon$.

Let x_i denote the quantity of resource $i \in \mathcal{I}$ and y_{ij} denote the amount of resource i allocated to customer $j \in \mathcal{J}$. The mathematical formulation of this problem can be stated as:

$$\min_{(x,y)\in\mathbb{R}_+^{|\mathcal{I}|+|\mathcal{I}||\mathcal{J}|}} \left\{\sum_{i\in\mathcal{I}} c_i x_i : \sum_{j\in\mathcal{J}} y_{ij} \le \rho_i x_i, \forall\, i \in \mathcal{I},\, \mathbb{P}\left\{\sum_{i\in\mathcal{I}} \mu_{ij} y_{ij} \ge \tilde{\lambda}_j, \forall\, j \in \mathcal{J}\right\} \ge 1 - \epsilon\right\},$$

where $\rho_i \in (0, 1]$ is the yield of resource i and $\mu_{ij} \ge 0$ is the service rate of resource i to customer j. We use the data available at `https://jrluedtke.github.io`; see [36] for the detailed generation procedure. The numbers of resources and customers, represented as $(|\mathcal{I}|, |\mathcal{J}|)$, are chosen from $\{(20, 30), (40, 50), (50, 100)\}$. The scenario size n for the random variables $\{\tilde{\lambda}_j\}$ and the confidence parameter ϵ are taken from $\{1000, 2000, 3000\}$ and $\{0.10, 0.15, 0.20\}$, respectively. The probabilities p_i, $i \in [n]$, are all set to $1/n$. For each $(|\mathcal{I}|, |\mathcal{J}|)$, n, and ϵ, there are 5 instances, leading to a testbed of 135 CCRP instances.

B.2 Chance-constrained multiperiod power planning problem [21]

The CCMPP problem attempts to expand the electric power capacity of a state by building new coal and nuclear power plants to meet the demand of the state over a planning horizon of T periods. Coal and nuclear plants are operational for T_c and T_n time periods after their construction, respectively. The objective of the problem is to minimize the total capital cost associated with the construction of the power plants while requiring that the fraction of nuclear capacity to the total capacity does not exceed a predetermined threshold f (as required by legal restrictions mandate), and the demands are met with a probability at least $1 - \epsilon$.

Let x_t and y_t be the amount of coal and nuclear capacity (in megawatt) brought on line at the beginning of period t, respectively. Then, the mathematical formulation of this problem is given by

$$\min_{(x,y)\in\mathbb{R}^{2T}_{+}} \left\{ \sum_{t\in[T]} (c_t x_t + n_t y_t) \, : \, \sum_{i=\tau_n(t)}^{t} y_i \leq f \cdot \left(e_t + \sum_{i=\tau_c(t)}^{t} x_i + \sum_{i=\tau_n(t)}^{t} y_i \right), \, \forall \, t \in [T], \right.$$
$$\left. \mathbb{P}\left\{ e_t + \sum_{i=\tau_c(t)}^{t} x_i + \sum_{i=\tau_n(t)}^{t} y_i \geq \tilde{d}_t, \; \forall \, t \in [T] \right\} \geq 1 - \epsilon \right\},$$

where c_t and n_t are the capital costs per megawatt for coal and nuclear power plants, respectively, e_t is the electric capacity from existing resources in period t, $\tau_c(t) = \max\{1, \, t - T_c + 1\}$, and $\tau_n(t) = \max\{1, \, t - T_n + 1\}$, $t \in [T]$. We use a similar procedure of [21] to construct CCMPP instances. Specifically,

- the electricity demands $\{\tilde{d}_t\}$ are independent random integers, and their scenarios are uniformly chosen from $\{300, 301, \ldots, 700\}$;
- the costs c_t and n_t are uniformly chosen from $\{100, 101, \ldots, 300\}$ and $\{100, 101, \ldots, 200\}$, respectively;
- the initial capacity resource e_1 is an integer uniformly chosen $\{100, 101, \ldots, 500\}$ and the capacity resources in the subsequent periods are calculated as $e_t = e_1 \cdot r^{t-1}$, where $t = 2, 3, \ldots, T$, and r is uniformly chosen from $[0.7, 1)$;
- the lifespans of coal and nuclear power plants, T_c and T_n, are set to 15 and 10, respectively;
- the number of periods T is taken from $\{10, 20, 30\}$;
- the scenario size n for the random variables $\{\tilde{d}_t\}$ and the confidence parameter ϵ are taken from $\{1000, 2000, 3000\}$ and $\{0.05, 0.10, 0.20\}$, respectively;
- the probabilities p_i, $i \in [n]$, are all set to $1/n$.

For each T, n, and ϵ, we randomly generate 5 instances, and thus in total, there are 135 CCMPP instances.

B.3 Chance-constrained lot-sizing problem [12]

The CCLS problem attempts to determine a production schedule for T periods that minimizes a summation of the fixed setup cost, the production cost, and the expected inventory cost while satisfying the capacity constraint in each period and meeting the customer demand with a probability at least $1-\epsilon$.

Let x_t be the binary variable denoting whether a setup of production is performed in period t, and y_t be the continuous variable characterizing the corresponding quantity to be produced. Mathematically, the CCLS problem can be written by

$$\min_{(x,y)\in\{0,1\}^T\times\mathbb{R}^T_+} \left\{ \sum_{t\in[T]} \left(f_t x_t + c_t y_t + h_t \mathbb{E}\left(\left(\sum_{j\in[t]} y_j - \sum_{j\in[t]} \tilde{d}_j \right)^+ \right) \right) : \right.$$
$$\left. y_t \le C_t x_t,\ \forall\, t\in[T],\ \mathbb{P}\left\{ \sum_{j\in[t]} y_j \ge \sum_{j\in[t]} \tilde{d}_j,\ \forall\, t\in[T] \right\} \ge 1-\epsilon \right\},$$

where for each $t\in[T]$, c_t is the fixed setup cost per production run, f_t is the unit production cost, h_t is the unit holding cost, C_t is the production capacity, and $\tilde{d}_t$ is the random demand. Notice that the CCLS problem can also be transformed into an MILP problem of the form (MILP); see [59]. We use a similar procedure as in [1] to construct the CCLS instances. Specifically,

- the demands $\{\tilde{d}_j\}$ are independent variables, and their scenarios are uniformly chosen from $(1,100)$;
- the setup cost f_t, the unit production cost c_t, and the unit holding cost h_t are uniformly chosen from $(1,1000)$, $(1,10)$, and $(1,5)$, respectively;
- the number of periods T is taken from $\{5,10,15,20\}$;
- the scenario size n for the random variables $\{\tilde{d}_j\}$ and the confidence parameter ϵ are taken from $\{1000,2000,3000\}$ and $\{0.05,0.10,0.20\}$, respectively;
- the probabilities p_i, $i\in[n]$, are all set to $1/n$.

For each T, n, and ϵ, we randomly generate 5 instances, and thus in total, there are 180 CCLS instances.

References

[1] Abdi, A. and Fukasawa, R. (2016). On the mixing set with a knapsack constraint. *Math. Program.*, 157:191–217.

[2] Achterberg, T. (2007a). Conflict analysis in mixed integer programming. *Discrete Optim.*, 4(1):4–20.

[3] Achterberg, T. (2007b). *Constraint Integer Programming.* Ph.D. thesis, Technical University of Berlin.

[4] Achterberg, T., Koch, T., and Martin, A. (2005). Branching rules revisited. *Oper. Res. Lett.*, 33(1):42–54.

[5] Ahmed, S., Luedtke, J., Song, Y., and Xie, W. (2017). Nonanticipative duality, relaxations, and formulations for chance-constrained stochastic programs. *Math. Program.*, 162(1):51–81.

[6] Ahmed, S. and Shapiro, A. (2008). Solving chance-constrained stochastic programs via sampling and integer programming. *Tutor. Oper. Res. (INFORMS)*, 10:261–269.

[7] Atamtürk, A., Nemhauser, G. L., and Savelsbergh, M. W. (2000). The mixed vertex packing problem. *Math. Program.*, 89:35–53.

[8] Barrera, J., Homem-de Mello, T., Moreno, E., Pagnoncelli, B. K., and Canessa, G. (2016). Chance-constrained problems and rare events: an importance sampling approach. *Math. Program.*, 157(1):153–189.

[9] Beraldi, P. and Bruni, M. E. (2010). An exact approach for solving integer problems under probabilistic constraints with random technology matrix. *Ann. Oper. Res.*, 177(1):127–137.

[10] Beraldi, P., Bruni, M. E., and Conforti, D. (2004). Designing robust emergency medical service via stochastic programming. *Eur. J. Oper. Res.*, 158(1):183–193.

[11] Beraldi, P. and Ruszczyński, A. (2002). The probabilistic set-covering problem. *Oper. Res.*, 50(6):956–967.

[12] Beraldi, P. and Ruszczyński, A. (2002). A branch and bound method for stochastic integer problems under probabilistic constraints. *Optim. Methods Softw.*, 17(3):359–382.

[13] Bestuzheva, K., Besançon, M., Chen, W.-K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., Gottwald, L., Graczyk, C., Halbig, K., Hoen, A., Hojny, C., van der Hulst, R., Koch, T., Lübbecke, M., Maher, S. J., Matter, F., Mühmer, E., Müller, B., Pfetsch, M. E., Rehfeldt, D., Schlein, S., Schlösser, F., Serrano, F., Shinano, Y., Sofranac, B., Turner, M., Vigerske, S., Wegscheider, F., Wellner, P., Weninger, D., and Witzig, J. (2021). The SCIP Optimization Suite 8.0. Technical report, Optimization Online.

[14] Charnes, A. and Cooper, W. W. (1963). Deterministic equivalents for optimizing and satisficing under chance constraints. *Oper. Res.*, 11(1):18–39.

[15] Chen, L., Chen, S.-J., Chen, W.-K., Dai, Y.-H., Quan, T., and Chen, J. (2023). Efficient presolving methods for solving maximal covering and partial set covering location problems. *Eur. J. Oper. Res.*, 311(1):73–87.

[16] Chen, W.-K., Chen, Y.-L., Dai, Y.-H., and Lv, W. (2024). Towards large-scale probabilistic set covering problem: an efficient Benders decomposition approach. In *2024 Informs Optimization Society conference*.

[17] Cheon, M.-S., Ahmed, S., and Al-Khayyal, F. (2006). A branch-reduce-cut algorithm for the global optimization of probabilistically constrained linear programs. *Math. Program.*, 108(2):617–634.

[18] Dentcheva, D., Lai, B., and Ruszczyński, A. (2004). Dual methods for probabilistic optimization problems. *Math. Methods Oper. Res.*, 60(2):331–346.

[19] Dentcheva, D., Prékopa, A., and Ruszczyński, A. (2000). Concavity and efficient points of discrete distributions in probabilistic programming. *Math. Program.*, 89(1):55–77.

[20] Dentcheva, D., Prékopa, A., and Ruszczyński, A. (2001). On convex probabilistic programming with discrete distributions. *Nonlinear Anal.*, 47(3):1997–2009.

[21] Dey, S. S., Dubey, Y., Molinaro, M., and Shah, P. (2024). A theoretical and computational analysis of full strong-branching. *Math. Program.*, 205(1):303–336.

[22] Gamrath, G. (2014). Improving strong branching by domain propagation. *EURO J. Comput. Optim.*, 2(3):99–122.

[23] Gamrath, G. (2020). *Enhanced Predictions and Structure Exploitation in Branch-and-Bound.* Ph.D. thesis, Technical University of Berlin.

[24] Garey, M. R. and Johnson, D. S. (1978). "Strong" NP-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508.

[25] Günlük, O. and Pochet, Y. (2001). Mixing mixed-integer inequalities. *Math. Program.*, 90(3):429–457.

[26] Gurvich, I., Luedtke, J., and Tezcan, T. (2010). Staffing call centers with uncertain demand forecasts: A chance-constrained optimization approach. *Manag. Sci.*, 56(7):1093–1115.

[27] Henrion, R. and Römisch, W. (2022). Problem-based optimal scenario generation and reduction in stochastic programming. *Math. Program.*, 191(1):183–205.

[28] Küçükyavuz, S. (2012). On mixing sets arising in chance-constrained programming. *Math. Program.*, 132:31–56.

[29] Küçükyavuz, S. and Jiang, R. (2022). Chance-constrained optimization under limited distributional information: A review of reformulations based on sampling and distributional robustness. *EURO J. Comput. Optim.*, 10:100030.

[30] Kılınç-Karzan, F., Küçükyavuz, S., and Lee, D. (2022). Joint chance-constrained programs and the intersection of mixing sets through a submodularity lens. *Math. Program.*, 195(1):283–326.

[31] Lejeune, M. and Noyan, N. (2010). Mathematical programming approaches for generating p-efficient points. *Eur. J. Oper. Res.*, 207(2):590–600.

[32] Lejeune, M. A. (2008). Preprocessing techniques and column generation algorithms for stochastically efficient demand. *J. Oper. Res. Soc.*, 59(9):1239–1252.

[33] Lejeune, M. A. (2012a). Pattern-based modeling and solution of probabilistically constrained optimization problems. *Oper. Res.*, 60(6):1356–1372.

[34] Lejeune, M. A. (2012b). Pattern definition of the p-efficiency concept. *Ann. Oper. Res.*, 200(1):23–36.

[35] Lejeune, M. A. and Ruszczyński, A. (2007). An efficient trajectory method for probabilistic production-inventory-distribution problems. *Oper. Res.*, 55(2):378–394.

[36] Luedtke, J. (2014). A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. *Math. Program.*, 146:219–244.

[37] Luedtke, J. and Ahmed, S. (2008). A sample approximation approach for optimization with probabilistic constraints. *SIAM J. Optim.*, 19(2):674–699.

[38] Luedtke, J., Ahmed, S., and Nemhauser, G. L. (2010). An integer programming approach for linear programs with probabilistic constraints. *Math. Program.*, 122:247–272.

[39] Margot, F. (2002). Pruning by isomorphism in branch-and-cut. *Math. Program.*, 94(1):71–90.

[40] Miller, B. L. and Wagner, H. M. (1965). Chance constrained programming with joint constraints. *Oper. Res.*, 13(6):930–945.

[41] Murr, M. R. and Prékopa, A. (2000). Solution of a product substitution problem using stochastic programming. In Uryasev, S. P., editor, *Probabilistic Constrained Optimization: Methodology and Applications*, pages 252–271. Springer US, Boston, MA.

[42] Nemirovski, A. and Shapiro, A. (2006). Scenario approximations of chance constraints. In Calafiore, G. and Dabbene, F., editors, *Probabilistic and Randomized Methods for Design under Uncertainty*, pages 3–47. Springer, London.

[43] Ostrowski, J., Linderoth, J., Rossi, F., and Smriglio, S. (2011). Orbital branching. *Math. Program.*, 126(1):147–178.

[44] Pfetsch, M. E. and Rehn, T. (2019). A computational comparison of symmetry handling methods for mixed integer programs. *Math. Program. Comput.*, 11(1):37–93.

[45] Prékopa, A. (1971). On probabilistic constrained programming. In *Proceedings of the Princeton Symposium on Mathematical Programming*, pages 113–138. Princeton University Press, Princeton.

[46] Prékopa, A. (1973). Contributions to the theory of stochastic programming. *Math. Program.*, 4(1):202–221.

[47] Prékopa, A. (1990). Dual method for the solution of a one-stage stochastic programming problem with random rhs obeying a discrete probability distribution. *Z. Oper. Res.*, 34(6):441–461.

[48] Prékopa, A. (2003). Probabilistic programming. In Ruszczyński, A. and Shapiro, A., editors, *Stochastic Programming*, volume 10 of *Handbooks Oper. Res. Management Sci.*, pages 267–351. Elsevier, Amsterdam.

[49] Qiu, F., Ahmed, S., Dey, S. S., and Wolsey, L. A. (2014). Covering linear programming with violations. *INFORMS J. Comput.*, 26(3):531–546.

[50] Qiu, F. and Wang, J. (2015). Chance-constrained transmission switching with guaranteed wind power utilization. *IEEE Trans. Power Syst.*, 30(3):1270–1278.

[51] Ruszczyński, A. (2002). Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. *Math. Program.*, 93(2):195–215.

[52] Saxena, A., Goyal, V., and Lejeune, M. A. (2010). MIP reformulations of the probabilistic set covering problem. *Math. Program.*, 121(1):1–31.

[53] Sen, S. (1992). Relaxations for probabilistically constrained programs with discrete random variables. *Oper. Res. Lett.*, 11(2):81–86.

[54] Song, Y. and Luedtke, J. R. (2013). Branch-and-cut approaches for chance-constrained formulations of reliable network design problems. *Math. Program. Comput.*, 5(4):397–432.

[55] Song, Y., Luedtke, J. R., and Küçükyavuz, S. (2014). Chance-constrained binary packing problems. *INFORMS J. Comput.*, 26(4):735–747.

[56] Vielma, J. P., Ahmed, S., and Nemhauser, G. L. (2012). Mixed integer linear programming formulations for probabilistic constraints. *Oper. Res. Lett.*, 40(3):153–158.

[57] Wang, Q., Guan, Y., and Wang, J. (2012). A chance-constrained two-stage stochastic program for unit commitment with uncertain wind power output. *IEEE Trans. Power Syst.*, 27(1):206–215.

[58] Witzig, J., Berthold, T., and Heinz, S. (2021). Computational aspects of infeasibility analysis in mixed integer programming. *Math. Program. Comput.*, 13(4):753–785.

[59] Zhao, M., Huang, K., and Zeng, B. (2017). A polyhedral study on chance constrained program with random right-hand side. *Math. Program.*, 166(1-2):19–64.