

THE BRANCH-AND-BOUND TREE CLOSURE

MARIUS ROLAND, NAGISA SUGISHITA, ALEXANDRE FOREL,
YOUSSEF EMINE, RICARDO FUKASAWA, THIBAUT VIDAL

ABSTRACT. This paper investigates the a-posteriori analysis of Branch-and-Bound (BB) trees to extract structural information about the feasible region of mixed-binary linear programs. We introduce three novel outer approximations of the feasible region, systematically constructed from a BB tree. These are: a tight formulation based on disjunctive programming, a branching-based formulation derived from the tree’s branching logic, and a mixing-set formulation derived from the on-off properties inside the tree. We establish an inclusion hierarchy, which ranks the approximations by their theoretical strength w.r.t. to the original feasible region. The analysis is extended to the generation of valid inequalities, revealing a separation-time hierarchy that mirrors the inclusion hierarchy in reverse. This highlights a trade-off between the tightness of an approximation and the computational cost of generating cuts from it. Motivated by the computational expense of the stronger approximations, we introduce a new family of valid inequalities called star tree inequalities. Although their closure forms the weakest of the proposed approximations, their practical appeal lies in an efficient, polynomial-time combinatorial separation algorithm. A computational study on multi-dimensional knapsack and set-covering problems empirically validates the theoretical findings. Moreover, these experiments confirm that computationally useful valid inequalities can be generated from BB trees obtained by solving optimization problems considered in practice.

1. INTRODUCTION

The Branch-and-Bound (BB) method proposed by Land and Doig [39], has transformed mixed-integer linear optimization. This central role has motivated extensive research on the design and analysis of branch-and-bound and its many components, notably branching rules, valid inequalities, and primal heuristics. Over time, additional features such as presolve, symmetry handling, parallelization, and restarts have become standard. Each of these elements continues to be the focus of significant attention [19].

Despite their practical effectiveness, BB algorithms are difficult to parameterize. The design space is vast: each component admits many alternatives, and their interactions are subtle. As a result, the parameter choices implemented in state-of-the-art solvers rely heavily on expert judgment and extensive empirical tuning [12]. This reliance on heuristic selection sits uneasily with the prescriptive spirit of optimization and has motivated a recent body of work that seeks a rigorous, mathematical understanding of branch-and-bound. Recent research seeks to address fundamental questions, such as identifying classes of instances that are solvable efficiently using BB [26], and analyzing the circumstances under which specific algorithmic components are most or least effective [27].

The present work contributes to this new perspective but approaches it from an “a-posteriori” angle. We study the following question: given a mixed-binary linear program, what information about its feasible region can be recovered from a branch-and-bound tree (that may or may not certify optimality)? To address this question,

we study how to derive valid inequalities and construct outer approximations of the feasible region using only the information contained in a BB tree.

Beyond its theoretical contributions, this work is motivated by a recurring computational pattern in mathematical optimization: the need to solve sequences of closely related mixed-binary programs. For instance, decomposition methods, such as column generation [33, 46] and Lagrangian decomposition, repeatedly solve subproblems that differ only in their objective coefficients. Similarly, restart strategies within MIP solvers [21] halt and restart the search, creating an opportunity to carry over knowledge from one BB tree to the next. Bilevel optimization algorithms often involve an outer loop that repeatedly queries an optimal solution to a follower’s MIP under varying parameters. The pattern also appears in application-specific contexts, where models for energy systems [45], scheduling [2], or vehicle routing [1] are re-solved with updated data, and in modern machine learning pipelines, where differentiating through an optimization model requires solving many perturbed instances [44, 11, 24, 29]. In all these settings, the ability to extract and reuse structural information from a BB tree, such as the outer approximations we develop, can significantly reduce the cost of subsequent solves.

Because our primary motivations are computational, we emphasize scalable constructions. In particular, we quantify the size of each proposed outer approximation, in terms of variables and constraints, as a function of the number of leaves in the BB tree. For each construction, we also analyze the computational complexity and, where relevant, the formulation size required to separate valid inequalities for the proposed outer approximations.

1.1. Contributions.

- (1) We introduce three novel outer approximations of the feasible region of mixed-binary programs that can be systematically constructed from a BB tree. The first approximation is derived from disjunctive programming [4], the second is related to binary polynomial optimization [28], and the third is based on the mixing-set [35, 3]. For each approximation, we propose a linear extended formulation.
- (2) We establish a strict inclusion hierarchy among them, demonstrating that they form a sequence of increasingly weaker approximations.
- (3) We develop separation procedures for each outer approximation by projecting its extended formulation onto the original space of variables. In particular, we adapt the cut-generating linear program framework [7, 6] to the extended formulations. We analyze the size of each extended formulation in terms of the number of leaf nodes in the BB tree and the number of decision variables. This analysis reveals a trade-off: the computational effort required for separation is inversely related to the tightness of the approximation. This trade-off justifies the study of each formulation, including those that are comparatively weaker.
- (4) To provide a computationally cheaper alternative to solving a linear separation program, we introduce a new class of valid inequalities, which we call *star tree inequalities*. We prove the validity of these inequalities and present a combinatorial, polynomial-time algorithm for the corresponding separation problem. We show that the closure of the star tree inequalities contains the disjunctive- and branching-based outer approximations.
- (5) We conduct a computational study on multi-dimensional knapsack and set-covering problems instances to evaluate the practical effectiveness of the proposed outer approximations. The experiments illustrate numerically the inclusion and separation hierarchies emphasized throughout the manuscript. Further, they demonstrate that useful structural information about the

feasible region can be extracted a posteriori from BB trees. In fact, by varying the structure of the BB tree, we empirically show that the choice of tree significantly influences separation.

1.2. Relevant literature. Our work takes place within a recent and growing body of research that seeks to understand the mathematical and computational properties of BB. This research often investigates the theoretical limitations and performance of the BB algorithm. Several foundational papers have established that even seemingly simple integer programs can be difficult for BB. Jeroslow [36] constructs a class of simple zero-one integer programs that require an exponential number of nodes to be solved by any BB algorithm. Similarly, Chvátal [18] identifies classes of knapsack problems that are computationally hard to solve. More recently, Dey et al. [26] extends these complexity results by constructing packing, set covering, and traveling salesman problem instances for which any general BB tree must be of exponential size. Mahajan and Ralphs [41] show that selecting an optimal disjunction for branching is an NP-hard problem. Gläser and Pfetsch [34] further reinforce this by proving that approximating the size of the smallest possible BB tree is computationally hard, unless the strong exponential time hypothesis fails.

Despite these worst-case complexity results, the practical success of BB has motivated research into its performance under specific conditions. Dey et al. [25] provide a theoretical justification for this success by showing that, for random binary integer programs with a fixed number of constraints, the BB algorithm is expected to terminate in polynomial time. The choice of branching rule is also critical to performance. Dey et al. [27] conduct a detailed theoretical and computational analysis of full strong-branching, identifying classes of problems where it performs provably well. Complementing this, Owen and Mehrotra [43] demonstrate experimentally that using general disjunctions, instead of simple variable disjunctions, can significantly reduce the size of the BB tree for general mixed-integer linear programs.

Another stream of research compares the strength of BB with other well-known techniques, particularly cutting planes. Basu et al. [9] investigate the theoretical complexity of BB and cutting plane (CP) algorithms, showing that for convex 0/1 problems, CPs are at least as powerful as BB based on variable disjunctions. Cornuéjols and Dubey [22] introduced “skewed k-trees” to show that the hierarchy of relaxations from BB is incomparable to classical lift-and-project hierarchies. Fleming et al. [32] analyzed the Stabbing Planes proof system, which models the reasoning in modern solvers, and related its power back to the Cutting Planes system.

While the aforementioned literature provides a deep understanding of BB trees, our work differs by focusing on the a-posteriori extraction of structural information from a single, already computed tree. To the best of our knowledge, we are the first to reuse the information of a BB tree to generate outer approximations of the feasible region and valid inequalities. The most closely related works are discussed next.

The work of Muñoz et al. [42] on tree compression also performs an a-posteriori analysis of a BB tree. Their goal is to compress the tree into a smaller one with an equivalent or stronger dual bound, which can serve as a more compact certificate of optimality or help identify strong disjunctions. Fischetti and Monaci [31] propose “backdoor branching”, a method that identifies a small set of critical branching variables by sampling fractional solutions during a preliminary phase. This “backdoor” set is then used to guide the branching process in a subsequent, full solve. Their approach shares the idea of using information gathered from one process to improve another, but it does not analyze the structure of a complete BB tree. The recent work by Becu et al. [10] proves that for a family of instances

with right-hand-sides belonging to a lattice, the Gomory Mixed-Integer Cut (GMIC) closure can be obtained using the same finite list of aggregation weights. This result motivates a simple heuristic to efficiently select aggregations for generating GMICs from historical data of similar instances. Finally, Kılınç et al. [37] is perhaps the most related work in spirit. The authors propose using the “discarded” information from strong-branching to generate valid inequalities. This is similar to our goal of using dual bounds to create valid inequalities. However, their method extracts information during the node selection process of an active BB search, whereas our approach performs an a-posteriori analysis on an entire, static BB tree that is already generated.

1.3. Outline. Section 2 introduces the first and tightest outer approximation, which is based on disjunctive programming. It also discusses its extended formulation and analyzes the computational cost of generating valid inequalities from it. Section 3 presents a second, novel outer approximation derived from the branching logic of the tree. This section shows that while this approximation is looser than the first, its corresponding separation problem is computationally more manageable. Section 4 introduces a third outer approximation based on a mixing-set formulation and establishes its relationship to the second approximation, showing that its continuous relaxation is weaker and offers no computational benefits for cut generation. Motivated by the computational challenges of the previous methods, Section 5 introduces a new family of valid inequalities called Star Tree Inequalities, proves their validity, and demonstrates that they can be separated efficiently using a polynomial-time combinatorial algorithm. Section 6 provides a computational analysis to empirically evaluate the trade-offs between the theoretical strength of the proposed approximations and the practical cost of generating cuts from them. Finally, Section 7 concludes with a summary of the main findings and outlines potential avenues for future research.

2. A DISJUNCTIVE PROGRAMMING BASED OUTER APPROXIMATION

In this section, we introduce an outer approximation based on disjunctive programming. We begin by defining the approximation and formally proving its validity. Next, we present its linear reformulation in an extended space and analyze its size. We then describe the use of a Cut-Generating Linear Program (CGLP) to derive valid inequalities in the original variable space. Finally, we examine the dimensions of this CGLP to motivate the development of more computationally tractable approximations in subsequent sections.

2.1. Notation and Assumptions. We first introduce the notation for the class of optimization problems under study. We consider Mixed-Binary linear Problems (MBPs) in the following standard form:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax \geq b, \\ & x_i \in \{0, 1\}, \quad i \in I_B, \\ & \underline{b}_i \leq x_i \leq \bar{b}_i, \quad i \in I_C. \end{aligned} \tag{1}$$

Here, $c \in \mathbb{R}^n$ is the objective vector, $A \in \mathbb{R}^{m \times n}$ is the constraint matrix, and $b \in \mathbb{R}^m$ is the right-hand side vector. The vector $x \in \mathbb{R}^n$ represents the decision variables. The index set of these variables $I := \{1, \dots, n\}$ is partitioned into a set for binary variables, $I_B = \{1, \dots, n_b\}$, and a set for continuous variables, $I_C = \{n_b + 1, \dots, n\}$. The continuous variables are bounded by \underline{b}_j and \bar{b}_j , which may be infinite. The

feasible region of Problem (1) is given by $F := \{x \in \mathcal{X} : Ax \geq b\}$, where \mathcal{X} is the set constructed by considering only the variable bounds, i.e.,

$$\mathcal{X} := \left\{ x \in \mathbb{R}^n : \begin{array}{ll} x_i \in \{0, 1\}, & i \in I_B, \\ \underline{b}_i \leq x_i \leq \bar{b}_i, & i \in I_C. \end{array} \right\}.$$

Throughout this document, we use the “hat” notation, as in \hat{F} , to denote the continuous relaxation of a set. For instance, \hat{F} is the feasible region of the continuous relaxation of Problem (1), where the binary constraints $x_i \in \{0, 1\}$ are relaxed to $x_i \in [0, 1]$ for all $i \in I_B$.

Next, we introduce the notation and assumptions regarding the BB tree considered. The BB method solves an MBP by constructing a search tree, $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each node $v \in V$ corresponds to a subproblem of the original MBP, while each edge represents a branching decision that partitions the feasible set of the parent node’s subproblem.

We make two common assumptions regarding the BB tree G that we consider:

- (1) Branching is restricted to elementary binary branching, where any child node is created by fixing a binary variable to either zero or one.
- (2) Every node has either zero or two children. Nodes with zero children are the leaf nodes, which we denote by the set $L \subseteq V$.

These assumptions imply that the leaf nodes L induce a partition of the binary variable space and, by extension, of the feasible region F . Any feasible point $x \in F$ satisfies the branching decisions corresponding to exactly one leaf node. Following standard literature terminology [13], we refer to a tree with this structure as a “full binary tree”.

We remark that considering full binary trees is not restrictive in practice. Any binary tree that does not directly imply a partition of the feasible region, for instance, due to pruning of some nodes in the BB process, can be completed by reintroducing the missing nodes. Since the methods discussed in this paper only require valid dual bounds for any node of the tree, we can assign them the dual bound of their parent, which is necessarily valid.

We also define notations capturing the relationships between nodes in the trees. For any non-root node $v \in V \setminus \{r\}$, we denote its unique parent and sibling as $p(v)$ and $s(v)$, respectively. Similarly, for any non-leaf node $v \in V \setminus L$, its children created by branching on variable x_j are denoted by $c_0(v)$ (such that $x_j = 0$) and $c_1(v)$ (where $x_j = 1$). The path from the root r to any node v is defined by a series of variable fixings. We define the sets $\mathbb{0}_v \subseteq I_B$ and $\mathbb{1}_v \subseteq I_B$ as the indices of the binary variables fixed to 0 and 1, respectively, along this unique path. We let $i(v) \in I_B$ be the index of the variable on which branching occurred to create node v .

As mentioned earlier, we assume that a valid dual bound l_v is available at each node $v \in V$. This value is a lower bound on the optimal objective of the subproblem at node v :

$$l_v \leq \min \left\{ c^\top x : \begin{array}{ll} x \in F & \\ x_i = 0, & i \in \mathbb{0}_v, \\ x_i = 1, & i \in \mathbb{1}_v, \end{array} \right\}.$$

2.2. Definition and Validity of the Outer Approximation. We construct the outer approximation using the disjunctive theory established by Balas [4]. Central to this construction are the *atoms* of the branch-and-bound tree, a concept recently employed in [27, 26]. For each leaf node $v \in L$, we define the corresponding atom \mathcal{A}_v as

$$\mathcal{A}_v := \left\{ x \in \hat{\mathcal{X}} : \begin{array}{ll} c^\top x \geq l_v, & \\ x_i = 0, & i \in \mathbb{0}_v, \\ x_i = 1, & i \in \mathbb{1}_v, \end{array} \right\}. \quad (2)$$

Each atom \mathcal{A}_v is a polyhedron that incorporates three sets of constraints: the branching decisions that define the path to node v , the local dual bound $c^\top x \geq l_v$ at that node, and the original variable bounds contained within the definition of $\hat{\mathcal{X}}$. We recall that $\hat{\mathcal{X}}$ is the continuous relaxation of the set \mathcal{X} .

Using this notation, the first outer approximation O_1 is defined as the convex hull of the union of all atoms:

$$O_1 := \text{conv} \left(\bigcup_{v \in L} \mathcal{A}_v \right). \quad (3)$$

This formulation differs from the classical approach in disjunctive programming [4], where the sets in the union are typically defined by the original problem constraints $Ax \geq b$ rather than by the dual bound inequality $c^\top x \geq l_v$. This choice is motivated by our objective to construct approximations using only the information contained within the BB tree. The dual bound l_v is a fundamental piece of data generated during the BB process for each node. This decision also results in a more compact formulation, introducing the central trade-off between approximation tightness and computational cost that we explore throughout this work.

We now formally establish that O_1 is a valid outer approximation of the feasible set F .

Proposition 1. *The set O_1 is an outer approximation of the set F ; that is,*

$$F \subseteq O_1.$$

Proof. Consider an arbitrary point $\bar{x} \in F$. We show that $\bar{x} \in O_1$. Since by assumption (see Section 2.1), we assume that the BB tree partitions the feasible region, \bar{x} must satisfy the branching conditions corresponding to a unique leaf node $u \in L$. Therefore, we have

$$\bar{x} \in \left\{ x \in \hat{\mathcal{X}} : \begin{array}{ll} x_i = 0, & i \in \mathbb{0}_u, \\ x_i = 1, & i \in \mathbb{1}_u \end{array} \right\}.$$

Furthermore, because $\bar{x} \in F$, it must also satisfy the local dual bound at node u , so $c^\top \bar{x} \geq l_u$. It follows that $\bar{x} \in \mathcal{A}_u$. Consequently,

$$\bar{x} \in \mathcal{A}_u \subseteq \bigcup_{v \in L} \mathcal{A}_v \subseteq \text{conv} \left(\bigcup_{v \in L} \mathcal{A}_v \right) = O_1.$$

□

Since each set \mathcal{A}_v in the union shares the same recession cone, their convex hull O_1 is a polyhedron. This set admits a linear reformulation in an extended variable space [20]. To express this, we first write each atom as $\mathcal{A}_v = \{x \in \mathbb{R}^n : D_v x \geq f_v\}$, where the matrix $D_v \in \mathbb{R}^{k \times n}$ and vector $f_v \in \mathbb{R}^k$ are formed by combining the dual bound constraint, the branching constraints (written as inequalities), and the variable bounds from $\hat{\mathcal{X}}$. Note that the number of rows, k , is the same for all atoms and is given by $k = 1 + 2|I_B| + o$, where o is the number of bound constraints on the continuous variables.

The extended formulation of O_1 is constructed by introducing auxiliary variables $z^v \in \mathbb{R}^n$ and $z_0^v \in \mathbb{R}$ for each leaf node $v \in L$. We collect these auxiliary variables into a single vector $z \in \mathbb{R}^q$, where $q = |L|(n+1)$. The extended formulation,

which we denote by $O_{1,\text{lin}}$, is then given by

$$O_{1,\text{lin}} := \left\{ (x, z) \in \mathbb{R}^n \times \mathbb{R}^q : \begin{array}{l} D_v z^v \geq f_v z_0^v, \quad v \in L, \\ \sum_{v \in L} z^v = x, \\ \sum_{v \in L} z_0^v = 1, \\ z_0^v \geq 0, \quad v \in L \end{array} \right\}. \quad (4)$$

The following theorem establishes the equivalence between O_1 and the projection of $O_{1,\text{lin}}$ in the x -space.

Theorem 1 (Balas [5]). *The set $O_{1,\text{lin}}$ defined in (4) provides an extended formulation of O_1 . Its projection onto the space of the original x -variables is precisely O_1 ; that is,*

$$\text{proj}_x(O_{1,\text{lin}}) = O_1.$$

This formulation introduces $|L|(n+1)$ auxiliary variables and $|L|(k+1)+4$ inequality constraints.

2.3. Cut Separation. Outer approximations of a set in an extended variable space are computationally impractical. Therefore, we focus on generating valid inequalities in the original space of x -variables. We consider the Cut-Generating Linear Program (CGLP) to perform this projection, following the approach of Balas et al. [7, 6].

To formalize this process and make it applicable to the outer approximations presented in further sections, we adopt generic notations. Let the feasible region in the extended space be represented by a generic polyhedron

$$P = \{(x, z) \in \mathbb{R}^n \times \mathbb{R}^q \mid W_x x + W_z z \geq h\},$$

where $W_x \in \mathbb{R}^{r \times n}$ and $W_z \in \mathbb{R}^{r \times q}$. The number of constraints, $r \in \mathbb{Z}_+$, depends on the specific extended formulation used.

The goal of cut separation is to find a valid inequality $\pi^\top x \geq \pi_0$ that is the most violated by a given point $\bar{x} \in \mathbb{R}^n$. The CGLP is designed to find the components of such a cut. It employs dual multipliers $u \in \mathbb{R}_+^r$ associated with the constraints of P . The CGLP reads

$$\begin{aligned} \max_{\pi, u} \quad & \pi_0 - \pi^\top \bar{x} \\ \text{s.t.} \quad & h^\top u = \pi_0, \\ & W_x^\top u = \pi, \\ & W_z^\top u = 0, \\ & f(u) = 1, \\ & u \geq 0, \end{aligned}$$

where constraint $f(u) = 1$ is added to normalize the dual multipliers u . We recall that r and q represent the number of constraints of the generic formulation and the number of auxiliary variables used to construct the extended formulation, respectively. By eliminating the variables π and π_0 , the CGLP can be simplified to a more compact form with r variables and $q+1$ constraints [20].

In this paper, we emphasize the construction of computationally tractable outer approximations. Therefore, we analyze the size of the CGLP associated with the extended formulation $O_{1,\text{lin}}$. From Theorem 1, we know that for $O_{1,\text{lin}}$, the number of auxiliary variables is $q = |L|(n+1)$ and the number of constraints is $r = |L|(k+1)+4$, where $k = 1 + 2|I_B| + o$ and o represents the number of variable bound constraints. This leads to the following observation on the size of the resulting CGLP.

Observation 1. *The CGLP associated with the extended formulation $O_{1,\text{lin}}$ requires the construction of $\mathcal{O}(|L|(|I_B| + o))$ variables and $\mathcal{O}(|L|n)$ constraints.*

Observation 1 highlights the computational challenge associated to the CGLP of $O_{1,\text{lin}}$. While O_1 provides a tight outer approximation of F , the size of its corresponding CGLP grows with the number of leaf nodes in the BB tree and the number of variables. Solving a linear problem that is $|L|$ times larger than the original problem to generate each valid inequality not only increases the number of iterations needed for the resolution step to finish, but also requires constructing and storing a model that is $|L|$ times larger in memory. This is observed and discussed in, e.g., [15, 16]. In the remainder of this paper, we explore alternative approximations that, while less tight than O_1 , allow for the generation of valid inequalities with computational requirements that scale better with the size of the BB tree.

3. A BRANCHING-BASED OUTER APPROXIMATION

This section presents the second outer approximation. Unlike the disjunctive approximation in Section 2, this new approximation is extracted from an interpretable structure, namely, a binary polynomial set. We first define this outer approximation and prove its validity. We then focus on a subset of its defining constraints, for which we derive a linear reformulation. This reformulation can subsequently be tightened and simplified by leveraging the disjunctive structure of the BB tree. We show that the resulting outer approximation is weaker than the one presented in Section 2. Finally, we analyze the size of the corresponding CGLP. This analysis reveals why the approximation, despite its relative weakness, remains computationally attractive.

3.1. Definition and Validity of the Outer Approximation. We introduce the second outer approximation of the set F , which we denote by O_2 . This approximation is the intersection of two sets, B and H . The set B models the activation of leaf nodes, while the set H enforces the local validity of the dual bound. Before proceeding to the formal definition, we offer a remark to avoid ambiguity.

Remark 1. *Contrary to O_1 , the second outer approximation O_2 is directly defined using auxiliary variables z . In addition, these variables satisfy $(x, z) \in \mathcal{X} \times \{0, 1\}^{|L|}$. This is done on purpose and is relaxed later when the continuous relaxation is considered.*

The first component of O_2 is the set B , which we call the binary polynomial set. It is defined as

$$B := \left\{ (x, z) \in \mathcal{X} \times \{0, 1\}^{|L|} : z_v = \prod_{i \in 0_v} (1 - x_i) \cdot \prod_{i \in 1_v} x_i, v \in L \right\}. \quad (6)$$

This set links the branching decisions in the tree to the auxiliary variables z . For any point $(x, z) \in B$, the variable z_v is equal to one if and only if the decision vector x satisfies all branching decisions along the path from the root to the leaf node v .

The second component is the set H , defined as

$$H := \left\{ (x, z) \in \mathcal{X} \times \{0, 1\}^{|L|} : c^\top x \geq \sum_{v \in L} l_v z_v \right\}. \quad (7)$$

The set H relates the objective function value to the dual bounds of the leaf nodes, activated by the z variables.

The outer approximation O_2 is then defined as the intersection of these two sets:

$$O_2 := B \cap H. \quad (8)$$

We now show that O_2 defines a valid outer approximation of F .

Proposition 2. *The set $\text{proj}_x(O_2)$ is an outer-approximation of the set F ; that is,*

$$F \subseteq \text{proj}_x(O_2).$$

Proof. Let $x \in F$ be a feasible point. We show that there exists a $z \in \{0, 1\}^{|L|}$ such that $(x, z) \in O_2$. By assumption, see Section 2.1, there exists exactly one leaf node $u \in L$ such that x satisfies all branching decisions to reach that node. We set $z_u = 1$, and $z_v = 0$ for all $v \in L \setminus \{u\}$. The pair (x, z) satisfies the defining equations of B . For (x, z) , the defining inequality of H reduces to $c^\top x \geq \sum_{v \in L} l_v z_v = l_u$, which is necessarily satisfied for x since l_u is a locally valid dual bound. \square

3.2. Reformulation and Tightening of the Binary Polynomial Set. We now discuss the reformulation and tightening of the binary polynomial set B . We first present the linear reformulation of B , denoted as B_{lin} . It requires the introduction of auxiliary binary variables for the non-leaf nodes. We then show how to strengthen B_{lin} by taking advantage of the disjunctive structure of the BB tree. This leads to a tightened set, B_{tight} . Finally, we define the linear reformulation $O_{2,\text{lin}}$ of O_2 , using this tightened set.

To define the linear reformulation B_{lin} , we introduce an auxiliary binary variable z_v for each non-leaf node $v \in V \setminus L$. We also define the following subsets of nodes. The set V_0 contains nodes reached by branching on a variable and fixing it to zero, while V_1 contains nodes reached by fixing a variable to one. Formally,

$$V_0 := \{v \in V \setminus \{r\} : x_{i(v)} = 0\}, \quad V_1 := \{v \in V \setminus \{r\} : x_{i(v)} = 1\}. \quad (9)$$

The set B_{lin} is then defined as

$$B_{\text{lin}} := \left\{ (x, z) \in \mathcal{X} \times \{0, 1\}^{|V|} : \begin{array}{l} z_r = 1, \\ z_v \leq z_{p(v)}, \quad v \in V \setminus \{r\}, \\ z_v \leq 1 - x_{i(v)}, \quad v \in V_0, \\ z_v \leq x_{i(v)}, \quad v \in V_1, \\ z_v \geq 1 - (1 - z_{p(v)}) - x_{i(v)}, \quad v \in V_0, \\ z_v \geq 1 - (1 - z_{p(v)}) - (1 - x_{i(v)}), \quad v \in V_1 \end{array} \right\}. \quad (10)$$

The following proposition establishes that B_{lin} is a valid extended formulation for B .

Proposition 3. *The projection of the set B_{lin} onto the $(x, [z_v]_{v \in L})$ space is equal to the set B ; that is,*

$$\text{proj}_{(x, [z_v]_{v \in L})}(B_{\text{lin}}) = B.$$

Proof. Consider the extended binary polynomial set

$$B_{\text{ext}} := \left\{ (x, z) \in \mathcal{X} \times \{0, 1\}^{|V|} : \begin{array}{l} z_r = 1, \\ z_v = \prod_{i \in \mathbb{0}_v} (1 - x_i) \cdot \prod_{i \in \mathbb{1}_v} x_i, \quad v \in V \setminus \{r\}, \end{array} \right\}.$$

This set extends B by introducing indicator variables z_v for all nodes $v \in V \setminus L$, along with their corresponding activation constraints. Since the z variables do not depend on each other, we have $\text{proj}_{(x, [z_v]_{v \in L})}(B_{\text{ext}}) = B$.

For any node $v \in V \setminus \{r\}$, the definition of z_v in B_{ext} can be expressed recursively in terms of its parent's indicator variable, $z_{p(v)}$. If $v \in V_0$, we have

$$z_v = z_{p(v)} \cdot (1 - x_i) \quad (11)$$

and, complementarily, if $v \in V_1$, we have

$$z_v = z_{p(v)} \cdot x_i. \quad (12)$$

We can thus rewrite B_{ext} using this recursive notation:

$$\bar{B}_{\text{ext}} = \left\{ (x, z) \in \mathcal{X} \times \{0, 1\}^{|V|} : \begin{array}{l} z_r = 1, \\ z_v = z_{p(v)} \cdot (1 - x_i), \quad v \in V_0, \\ z_v = z_{p(v)} \cdot x_i, \quad v \in V_1, \end{array} \right\}.$$

By linearizing each bilinear constraint of B_{ext} using the standard procedure of, e.g., [28], we obtain B_{lin} . Consequently, the equivalence $B_{\text{ext}} = B_{\text{lin}}$ holds. We may write

$$\text{proj}_{(x, [z_v]_{v \in L})}(B_{\text{lin}}) = \text{proj}_{(x, [z_v]_{v \in L})}(B_{\text{ext}}) = B.$$

□

We now further tighten the set B_{lin} using the disjunctive property of BB trees.

Proposition 4. *For any non-leaf node $v \in V \setminus L$, the equality*

$$z_v = z_{c_0(v)} + z_{c_1(v)} \quad (13)$$

is valid for the set B_{lin} . Furthermore, if we strengthen B_{lin} by adding Equation (13) for every $v \in V \setminus L$, then the inequalities

$$z_v \leq z_{p(v)}, \quad v \in V \setminus \{r\}, \quad (14)$$

$$z_v \geq 1 - (1 - z_{p(v)}) - x_{i(v)}, \quad v \in V_0, \quad (15)$$

$$z_v \geq 1 - (1 - z_{p(v)}) - (1 - x_{i(v)}), \quad v \in V_1, \quad (16)$$

become redundant and can be removed from the definition of B_{lin} .

Proof. First, we prove the validity of Equation (13). Let $v \in V \setminus L$ be an arbitrary non-leaf node. Recall that for $c_0(v)$ and $c_1(v)$ we have $x_{i(c_0(v))} = 0$ and $x_{i(c_1(v))} = 1$. We may write

$$z_{c_0(v)} = z_v \cdot (1 - x_{i(c_0(v))}),$$

$$z_{c_1(v)} = z_v \cdot x_{i(c_1(v))}.$$

Summing both inequalities and noting that $i(c_0(v)) = i(c_1(v))$ yields the desired result.

Second, we show that inequalities (14)–(16) are redundant when Equation (13) is added. Equation (13) states that $z_{p(v)} = z_v + z_{s(v)}$ for any node $v \in V \setminus \{r\}$. Since $z_{s(v)} \geq 0$, this equality implies $z_v \leq z_{p(v)}$, making Inequality (14) redundant.

To show that Inequality (15) is redundant, assume $v \in V_0$. Its sibling $s(v)$ must then be in V_1 . From the constraints of B_{lin} , we have $z_{s(v)} \leq x_{i(v)}$. Using Equation (13), we derive

$$z_v = z_{p(v)} - z_{s(v)} \geq z_{p(v)} - x_{i(v)}.$$

This is precisely Inequality (15). A symmetric argument holds for Inequality (16) when $v \in V_1$. □

Based on Proposition 4, we define the tightened, linearized binary polynomial set B_{tight} by adding the valid equalities (13) to B_{lin} and removing the now-redundant inequalities:

$$B_{\text{tight}} := \left\{ (x, z) \in \mathcal{X} \times \{0, 1\}^{|V|} : \begin{array}{l} z_r = 1, \\ z_v \leq 1 - x_{i(v)}, \quad \forall v \in V_0, \\ z_v \leq x_{i(v)}, \quad \forall v \in V_1, \\ z_v = z_{c_1(v)} + z_{c_2(v)}, \quad \forall v \in V \setminus L, \end{array} \right\}. \quad (17)$$

Remark 2. *The definition of B_{tight} in Equation (17) resembles a network design formulation [23]. The z variables characterise the flow in the network whereas the x variables model the design decisions.*

Remark 3. Proposition 4 implies the following. For any node $v \in V \setminus L$, we have

$$z_v = \sum_{u \in L_v} z_u,$$

where $L_v \subseteq L$ is the set of leaf nodes descending from v . As a result, all intermediate variables z_v for $v \in V \setminus L$ can be eliminated from the formulation of B_{tight} by substitution.

We now define the linearized and tightened branching-based outer approximation as:

$$O_{2,\text{lin}} := H \cap B_{\text{tight}}.$$

Finally, we compare the strength of the continuous relaxation $\hat{O}_{2,\text{lin}}$ of $O_{2,\text{lin}}$ with the disjunctive approximation O_1 .

Proposition 5. The projection of the set $\hat{O}_{2,\text{lin}}$ in the x space is weaker than the set O_1 . Specifically,

$$\text{proj}_x(\hat{O}_{1,\text{lin}}) = O_1 \subseteq \text{proj}_x(\hat{O}_{2,\text{lin}}).$$

Proof. We have $\hat{O}_{2,\text{lin}} = \hat{H} \cap \hat{B}_{\text{tight}}$. The projection of an intersection is a subset of the intersection of the projections, i.e., $\text{proj}_x(\hat{O}_{2,\text{lin}}) = \text{proj}_x(\hat{H}) \cap \text{proj}_x(\hat{B}_{\text{tight}})$. It suffices to show that $O_1 \subseteq \text{proj}_x(\hat{H})$ and $O_1 \subseteq \text{proj}_x(\hat{B}_{\text{tight}})$.

First, we show $O_1 \subseteq \text{proj}_x(\hat{H})$. We have

$$O_1 \subseteq \text{conv} \left(\bigcup_{v \in L} \{x \in \hat{\mathcal{X}} : c^\top x \geq l_v\} \right) \subseteq \{x \in \hat{\mathcal{X}} : c^\top x \geq \min_{v \in L} \{l_v\}\} \subseteq \text{proj}_x(\hat{H}),$$

where $\hat{\mathcal{X}}$ is the continuous relaxation of \mathcal{X} .

Second, we show $O_1 \subseteq \text{proj}_x(\hat{B}_{\text{tight}})$. For any leaf node $v \in L$, we introduce the modified atom

$$\bar{\mathcal{A}}_v = \left\{ x \in \hat{\mathcal{X}} : \begin{array}{ll} x_i = 0, & \text{for } i \in \mathbb{0}_v, \\ x_i = 1, & \text{for } i \in \mathbb{1}_v, \end{array} \right\},$$

which is a relaxation of the original atom \mathcal{A}_v defined in Equation (2), i.e.,

$$\mathcal{A}_v \subseteq \bar{\mathcal{A}}_v.$$

Let an arbitrary point $x \in \bar{\mathcal{A}}_v$ be selected. Next, we construct $z \in \{0, 1\}^{|V|}$ and show that it satisfies $z \in \hat{B}_{\text{tight}}$. We set $z_u = 1$ for any node $u \in V \setminus \{v\}$ located on the unique path between the root node r and the leaf node v , otherwise $z_u = 0$. We have $(x, z) \in \hat{B}_{\text{tight}}$. Consequently, any point $x \in \bigcup_{v \in L} \bar{\mathcal{A}}_v$ satisfies $x \in \text{proj}_x(\hat{B}_{\text{tight}})$. Finally, we can state the set of inclusions

$$O_1 \subseteq \text{conv} \left(\bigcup_{v \in L} \bar{\mathcal{A}}_v \right) \subseteq \text{conv}(\text{proj}_x(\hat{B}_{\text{tight}})) = \text{proj}_x(\text{conv}(\hat{B}_{\text{tight}})) = \text{proj}_x(\hat{B}_{\text{tight}}),$$

where the last equality holds because \hat{B}_{tight} is a convex set. \square

Proposition 5 states that the CGLP associated with $\hat{O}_{1,\text{lin}}$ necessarily produces valid inequalities that are at least as strong as those from the CGLP associated with $\hat{O}_{2,\text{lin}}$. However, the two CGLP do not scale in the same manner, as we further discuss.

3.3. Cut Generation. We now discuss the generation of inequalities parameterized in x and valid for the continuous relaxation $\hat{O}_{2,\text{lin}}$. The auxiliary variables z are projected out and the CGLP of Section 2.3 is used to generate valid inequalities.

After substituting the non-leaf indicator variables as described in Remark 3, the mathematical formulation of $\hat{O}_{2,\text{lin}}$ has $|L|$ auxiliary variables. We now count the number of linear constraints of $\hat{O}_{2,\text{lin}}$. First, the constraints necessary to bound the variables in $\hat{\mathcal{X}}$ equals $2|I_B| + o$ for the x variables, where we recall that o corresponds to the number of upper and lower bound constraints necessary for bounding the continuous variables. The number of constraints necessary to bound the z variables in $\hat{\mathcal{X}}$ equals $|L|$. Second, the set H introduces one constraint. Third, the set B_{tight} introduces

$$1 + |V_0| + |V_1| = |V| = 2|L| - 1 \quad (18)$$

constraints. The equality in Equation (18) holds because the branch-and-bound tree is assumed to have a full binary structure (see Section 2.1). For such a tree, the number of internal nodes and leaves are related by $|V| = 2|L| - 1$, and the branches are divided equally, so $|V_0| = |V_1| = (|V| - 1)/2$ [13].

This analysis, combined with the general discussion in Section 2.3 leads to the following observation.

Observation 2. *The CGLP associated with the extended formulation $\hat{O}_{2,\text{lin}}$ requires the construction of $\mathcal{O}(|L|)$ variables and $\mathcal{O}(|L| + |I_B| + o)$ constraints.*

A comparison between Observation 2 and Observation 1 reveals that the CGLP for $\hat{O}_{2,\text{lin}}$ scales more favorably compared to the CGLP of $\hat{O}_{1,\text{lin}}$ when the number of leaf nodes and original x variables increase. Nevertheless, this computational advantage comes at a price. As established in Proposition 5, the resulting inequalities are provably weaker than those derived from O_1 , though our numerical results in Section 6 indicate that the increased separation speed compensates this drawback.

4. A MIXING-SET BASED OUTER-APPROXIMATION

This section introduces a third outer approximation of the set F . We first define this outer approximation using a mixing set. We then prove that this new formulation is equivalent to the branching-based outer approximation from Section 3. Subsequently, we study the relationship between the continuous relaxations of these two sets. Finally, we analyze the size of the corresponding CGLP. This analysis reveals that the new approximation offers no computational advantage over the branching-based approach. This observation motivates the development presented in Section 5 where the mixing-set is taken advantage of to construct a family of valid inequalities separated in polynomial time.

4.1. Definition and Validity of the Outer Approximation. Let $\mu = \min_{v \in L} (l_v)$ be the minimum of the dual bounds over all leaf nodes of the BB tree. The value μ is a valid lower bound on the optimal objective value of Problem (1). We define the mixing set M as follows:

$$M := \left\{ (x, z) \in \mathcal{X} \times \{0, 1\}^{|V|} : c^\top x + (\mu - l_v)z_v \geq \mu, v \in V \right\}. \quad (19)$$

The set describes a *strengthened* mixing set in the sense of Luedtke et al. [40]. We skip the definition of the set O_3 and directly define the linearized outer approximation $O_{3,\text{lin}}$. It is constructed as the intersection of the set M and the set B_{tight} defined in Equations (19) and (17), such that

$$O_{3,\text{lin}} := B_{\text{tight}} \cap M.$$

The following proposition establishes the equivalence of $O_{3,\text{lin}}$ with $O_{2,\text{lin}}$.

Proposition 6. *The sets $O_{2,\text{lin}}$ and $O_{3,\text{lin}}$ are equivalent; that is,*

$$O_{2,\text{lin}} = O_{3,\text{lin}}.$$

Proof. Let $(x, z) \in B_{\text{tight}}$. We show that for any such point, the sets H and M are equal.

By assumption, the BB tree yields a disjoint partition of the feasible space (see Section 2.1). Let $u \in L$ be the unique leaf node for which $z_u = 1$. We also have $\sum_{v \in L} z_v = 1$.

First, consider the defining inequality for the set H given in Equation (7). Since $z_u = 1$ and $z_v = 0$ for all $v \in L \setminus \{u\}$, this inequality simplifies to

$$c^\top x \geq l_u. \quad (20)$$

Second, consider the inequalities defining M given in Equation (19). We analyze these inequalities by considering four cases.

Case 1. The considered node is $v = u$, which satisfies $z_u = 1$. The associated constraint simplifies to the same inequality as Inequality (20):

$$c^\top x \geq l_u. \quad (21)$$

Case 2. The considered node is $v \in L \setminus \{u\}$. We necessarily have that $z_u = 0$. The associated constraint reduces to

$$c^\top x \geq \mu.$$

This constraint is dominated by Inequality (21), since $l_u \geq \mu$ by definition.

Case 3. The considered node is $v \in V \setminus L$ and $z_v = 1$. The condition $z_v = 1$ implies that node v lies on the unique path from the root node to the leaf node u . In that case, the associated constraint reduces to

$$c^\top x \geq l_v$$

and is dominated by Equation (21) because $l_u \geq l_v$ always holds by monotonicity of the dual bounds of BB trees.

Case 4. The considered node is $v \in V \setminus L$ and $z_v = 0$. As in Case 2, the constraint becomes

$$c^\top x \geq \mu$$

and is dominated by Equation (21). \square

We now compare the continuous relaxations $\hat{O}_{2,\text{lin}}$ and $\hat{O}_{3,\text{lin}}$.

Proposition 7. *For any point (x, z) in the continuous relaxation \hat{B}_{tight} , the inequality*

$$c^\top x \geq \sum_{v \in L} l_v z_v$$

dominates the inequality

$$c^\top x \geq \mu + (l_u - \mu)z_u$$

for any $u \in V$.

Proof. First, we rewrite the expression $\sum_{v \in L} l_v z_v$. Using the property $\sum_{v \in L} z_v = 1$ discussed in Remark 3, we have

$$\begin{aligned} \sum_{v \in L} l_v z_v &= \sum_{v \in L} (\mu + l_v - \mu) z_v \\ &= \mu \left(\sum_{v \in L} z_v \right) + \sum_{v \in L} (l_v - \mu) z_v \\ &= \mu + \sum_{v \in L} (l_v - \mu) z_v. \end{aligned} \quad (22)$$

Next, we show that for any node $u \in V$, the following inequality holds:

$$\sum_{v \in L} (l_v - \mu) z_v \geq (l_u - \mu) z_u. \quad (23)$$

The set L_u denote the set of leaf nodes in the subtree rooted at u . From Remark 3, we know that $z_u = \sum_{v \in L_u} z_v$. Moreover, the dual bounds are non-decreasing, so $l_v \geq l_u$ for all $v \in L_u$. We can therefore write

$$\begin{aligned} \sum_{v \in L} (l_v - \mu) z_v &\geq \sum_{v \in L_u} (l_v - \mu) z_v \\ &\geq \sum_{v \in L_u} (l_u - \mu) z_v \\ &= (l_u - \mu) \sum_{v \in L_u} z_v \\ &= (l_u - \mu) z_u. \end{aligned} \quad (24)$$

Combining the result from Equation (22) with Inequality (24) yields the desired dominance result. \square

Corollary 1. *The continuous relaxation $\hat{O}_{2,\text{lin}}$ is a subset of the continuous relaxation $\hat{O}_{3,\text{lin}}$; that is,*

$$\hat{O}_{2,\text{lin}} \subseteq \hat{O}_{3,\text{lin}}.$$

Proof. Let $(x, z) \in \hat{O}_{2,\text{lin}}$. By definition, (x, z) belongs to \hat{B}_{tight} and satisfies the inequality $c^\top x \geq \sum_{v \in L} l_v z_v$. Proposition 7 implies that this inequality dominates $c^\top x \geq \mu + (l_u - \mu) z_u$ for any $u \in V$. Therefore, (x, z) also satisfies all the defining inequalities for \hat{M} . It follows that $(x, z) \in \hat{B}_{\text{tight}} \cap \hat{M} = \hat{O}_{3,\text{lin}}$. \square

4.2. Cut Generation. We now discuss the generation of inequalities valid in the x space using the continuous relaxation $\hat{O}_{3,\text{lin}}$. Just like Section 2.3 and Section 3.3, the auxiliary variables z are projected out and the CGLP is used to generate valid inequalities. We analyze the size of this CGLP to demonstrate why $\hat{O}_{3,\text{lin}}$ is not a practical candidate for cut separation via such an approach.

The formulation introduces one auxiliary variable for each node in the tree, yielding $|V| = 2|L| - 1$ variables [13]. Next, we count the constraints. As discussed in Section 3.3, the set \hat{X} and the set \hat{B}_{tight} introduce $2|I_B| + o + 2|L| - 1$ and $2|L| - 1$ constraints, respectively. The set \hat{M} contributes to $|V| = 2|L| - 1$ constraints [13]. This analysis, combined with the general discussion in Section 2.3 on the size of the CGLP for an extended formulation, leads to the following observation.

Observation 3. *The CGLP associated with the extended formulation $\hat{O}_{3,\text{lin}}$ requires the construction of $\mathcal{O}(|L|)$ variables and $\mathcal{O}(|L| + |I_B| + o)$ constraints.*

A comparison of Observation 3 with Observation 2 reveals that the CGLPs for $\hat{O}_{3,\text{lin}}$ and $\hat{O}_{2,\text{lin}}$ are of the same order of magnitude in size. In fact, a direct comparison of the constraint and variable counts shows that the CGLP for $\hat{O}_{3,\text{lin}}$ is larger. By combining this size analysis with the inclusion result from Corollary 3, we conclude that generating valid inequalities using $\hat{O}_{3,\text{lin}}$ offers no computational advantage over using $\hat{O}_{2,\text{lin}}$. This conclusion motivates the work in the subsequent section, where we introduce a new family of valid inequalities that can be separated more efficiently via a combinatorial algorithm.

5. THE STAR TREE INEQUALITIES

Motivated by the discussion of Section 4.2, we propose a novel family of valid inequalities for F . We coin them “Star Tree Inequalities” (STIs), as their construction combines the star inequalities presented in [35] and the inequalities that model the tree disjunctions in the set B_{tight} .

This section is organized as follows. First, we review the formulation of star inequalities, also known as mixing inequalities. Second, we introduce the STIs and prove their validity for the set $O_{3,\text{lin}}$. Subsequently, we analyze the closure of the STIs and establish that it contains the linear relaxation of the set $O_{2,\text{lin}}$. Finally, we demonstrate that the associated separation problem can be solved to optimality with a polynomial-time combinatorial algorithm.

The following theorem, adapted from Luedtke et al. [40], presents the star inequalities for the strengthened mixing set defined in Equation (19).

Theorem 2 (Luedtke et al. [40]). *Let ϕ be a permutation of the set V that satisfies $l_{\phi_1} \geq \dots \geq l_{\phi_{|V|}}$. The inequalities*

$$c^\top x \geq l_{t_{k+1}} + \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) z_{t_j}, \quad T = \{t_1, \dots, t_k\} \subseteq \{\phi_1, \dots, \phi_{|V|-1}\} \quad (25)$$

with $t_1 < \dots < t_k$ and $l_{t_{k+1}} := l_{\phi_{|V|}}$, are valid for M . Moreover, the inequalities (25) are facet-defining for $\text{conv}(M)$ if and only if $l_{t_1} = l_{\phi_1}$.

Star inequalities are studied extensively in the mathematical programming literature due to their favorable computational properties and the frequent appearance of mixing sets in integer programming models. Notably, the separation problem for the star inequalities (25) can be solved in polynomial time by reduction to a shortest-path problem on a graph [3, 35].

Remark 4. *The presentation of inequalities (25) in Luedtke et al. [40] differs slightly from our own. Equivalence can be established by substituting each binary variable z_v with its complement, $1 - z_v$.*

5.1. Definition and Validity of the STI. We now formally introduce the STIs. Their construction relies on the following lemma, which establishes a lower bound on the auxiliary variables z_v .

Lemma 1. *For any point $(x, z) \in B_{\text{tight}}$ and any node $v \in V$, the following inequality holds:*

$$z_v \geq 1 - \sum_{i \in \mathbb{1}_v} (1 - x_i) - \sum_{i \in \mathbb{0}_v} x_i. \quad (26)$$

Proof. We proceed by induction on the nodes of a BB tree.

Base Case. Consider the root node, $v = r$. By definition, the sets $\mathbb{1}_r$ and $\mathbb{0}_r$ are empty. Inequality (26) therefore simplifies to $z_r \geq 1$. This is valid because $z_r = 1$ is a defining constraint of the set B_{tight} , as specified in Equation (17).

Inductive step. Assume the inequality holds for a node $p(u)$. We show it holds for its child $u \in V \setminus \{r\}$. We assume w.l.o.g. that $u \in V_1$ as defined in Equation (9). From Proposition 4 it follows that any point in B_{tight} satisfies

$$\begin{aligned} z_u &\geq 1 - (1 - z_{p(u)}) - (1 - x_{i(v)}) \\ &\geq 1 - \sum_{i \in \mathbb{1}_{p(u)}} (1 - x_i) - \sum_{i \in \mathbb{0}_{p(u)}} x_i - (1 - x_{i(v)}) \\ &\geq 1 - \sum_{i \in \mathbb{1}_u} (1 - x_i) - \sum_{i \in \mathbb{0}_u} x_i. \end{aligned}$$

The case $u \in V_0$ is complementary. \square

The following proposition formally defines the STIs. Their validity is established by strengthening the star inequalities from Theorem 2 with the lower bound from Lemma 1.

Proposition 8 (Star-Tree Inequalities). *Let ϕ be a permutation of the set V that satisfies $l_{\phi_1} \geq \dots \geq l_{\phi_{|V|}}$. For any arbitrary set $T = \{t_2, \dots, t_k\} \subseteq \{\phi_2, \dots, \phi_{|V|-1}\}$ with $t_1 := \phi_1$, $t_2 < \dots < t_k$, and $t_{k+1} := \phi_{|V|}$, the inequality*

$$c^\top x \geq l_{t_1} - \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) \Delta_{t_j}(x), \quad (27)$$

where for any $v \in V$:

$$\Delta_v(x) := \min \left(1, \sum_{i \in \mathbb{1}_v} (1 - x_i) + \sum_{i \in \mathbb{0}_v} x_i \right),$$

is valid for the set $O_{3,\text{lin}}$.

Proof. From Theorem 2, the star inequality

$$c^\top x \geq l_{t_{|L|}} + \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) z_{t_j} \quad (28)$$

is valid for the set M . Consequently, it is also valid for the set $M \cap B_{\text{tight}} = O_{3,\text{lin}}$. For any node $v \in V$, Lemma 1 states that

$$z_v \geq 1 - \sum_{i \in \mathbb{1}_v} (1 - x_i) - \sum_{i \in \mathbb{0}_v} x_i \quad (29)$$

is valid for the set B_{tight} . By definition of z , we also have $z_v \geq 0$. The inequality

$$\begin{aligned} z_v &\geq \max \left(0, 1 - \sum_{i \in \mathbb{1}_v} (1 - x_i) - \sum_{i \in \mathbb{0}_v} x_i \right) \\ &= 1 - \min \left(1, \sum_{i \in \mathbb{1}_v} (1 - x_i) + \sum_{i \in \mathbb{0}_v} x_i \right). \end{aligned} \quad (30)$$

is valid for the set B_{tight} . Consequently, it is also valid for the set $M \cap B_{\text{tight}} = O_{3,\text{lin}}$.

Since the coefficients $(l_{t_j} - l_{t_{j+1}})$ are non-negative by construction, we can substitute the lower bound from Inequality (30) into Inequality (28) to derive the STI:

$$\begin{aligned} c^\top x &\geq l_{t_{|L|}} + \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) z_{t_j} \\ &\geq l_{t_{|L|}} + \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) \left(1 - \min \left(1, \sum_{i \in \mathbb{1}_{t_j}} (1 - x_i) + \sum_{i \in \mathbb{0}_{t_j}} x_i \right) \right) \\ &\geq l_{t_{|L|}} + (l_{t_1} - l_{t_{|L|}}) - \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) \min \left(1, \sum_{i \in \mathbb{1}_{t_j}} (1 - x_i) + \sum_{i \in \mathbb{0}_{t_j}} x_i \right) \\ &\geq l_{t_1} - \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) \min \left(1, \sum_{i \in \mathbb{1}_{t_j}} (1 - x_i) + \sum_{i \in \mathbb{0}_{t_j}} x_i \right), \end{aligned}$$

which terminates the proof. \square

Corollary 2. *Inequality (27) is valid for $O_{3,\text{lin}}$ and depends only on the x variables. Thus, STIs are valid for $\text{proj}_x(O_{3,\text{lin}})$ and, in particular, for the feasible region F .*

Remark 5. *Although Inequality (27) is nonlinear because of the Δ operator, it implies a family of linear inequalities. For any given point x , a linear inequality is obtained by replacing each Δ operator with either 1 or the corresponding sum over variables. The choice that yields the strongest cut is determined during separation, as formalized in Proposition 10.*

The following result establishes a relationship between the continuous relaxation $\hat{O}_{2,\text{lin}}$ and the closure of the STIs, which we define subsequently.

Proposition 9. *Let ϕ be a permutation of the set V that satisfies $l_{\phi_1} \geq \dots \geq l_{\phi_{|V|}}$. For any point (x, z) inside the continuous relaxation \hat{B}_{tight} and any $T = \{t_2, \dots, t_k\} \subseteq \{\phi_2, \dots, \phi_{|V|-1}\}$, the inequality*

$$c^\top x \geq \sum_{v \in L} l_v z_v$$

dominates the inequality

$$c^\top x \geq l_{t_{k+1}} + \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) z_{t_j},$$

where $t_1 = \phi_1$ and $t_{k+1} = \phi_{|V|}$.

Proof. From Equation (22) in the the proof of Proposition 7 we know that since $(x, z) \in \hat{B}_{\text{tight}}$,

$$\sum_{v \in L} l_v z_v = \mu + \sum_{v \in L} (l_v - \mu) z_v \quad (31)$$

holds. It remains to show that for any arbitrary $T = \{t_2, \dots, t_k\} \subseteq \{\phi_2, \dots, \phi_{|L|-1}\}$, the inequality

$$l_{t_{k+1}} + \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) z_{t_j} \leq \mu + \sum_{v \in V} (l_v - \mu) z_v \quad (32)$$

holds. We first introduce some notation. Recall that $T \subset V$. We introduce the set of leaf nodes $L(T)$ that are descendants of the nodes selected inside T . Mathematically speaking, we have

$$L(T) := \bigcup_{j=1}^k L_{t_j},$$

where $L_{t_j} \subseteq L$ is the set of leaves that descend from the node t_j . Second, for any node $v \in V$, we introduce the set of ancestor nodes $A(v)$ of v that are also included in the set T . Mathematically speaking, we have

$$A(v) := T \cap P(r, v),$$

where P models all the nodes located on the unique path between the root node r and node v , including v itself.

We proceed to show the validity of Inequality (32). We use the facts that $l_{k+1} = \mu$, that $z_{t_j} = \sum_{v \in L_{t_j}} z_v$ for $(x, z) \in \hat{B}_{\text{tight}}$ (see Remark 3), and that $L(T) \subseteq L$:

$$\begin{aligned}
l_{t_{k+1}} + \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) z_{t_j} &= \mu + \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) z_{t_j}, \\
&= \mu + \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) \left(\sum_{v \in L_{t_j}} z_v \right), \\
&= \mu + \sum_{v \in L(T)} z_v \left(\sum_{t_j \in A(v)} (l_{t_j} - l_{t_{j+1}}) \right), \\
&\leq \mu + \sum_{v \in L} z_v \left(\sum_{t_j \in A(v)} (l_{t_j} - l_{t_{j+1}}) \right). \tag{33}
\end{aligned}$$

Furthermore, for any leaf node $v \in L$, the inequality

$$\sum_{t_j \in A(v)} (l_{t_j} - l_{t_{j+1}}) \leq l_v - \mu \tag{34}$$

is valid. Validity follows because v is a descendant of any $t_j \in A(v)$ so that $l_v \geq \max_{t_j \in A(v)} (l_{t_j})$ and because μ is the smallest possible dual bound in the tree so that $\mu = \min_{u \in V} (l_u) \leq \min_{t_j \in A(v)} (l_{t_{j+1}})$.

Combining Inequality (33), Inequality (34), and Equation (31) yields the desired result. \square

The dominance result in Proposition 9 allows us to compare the STI closure with the projection of the continuous relaxation $\hat{O}_{2,\text{lin}}$. With some abuse of notation, let \mathcal{G} denote the information collected from the BB tree. The STI closure for \mathcal{G} is defined as

$$\mathcal{C}(\mathcal{G}) := \bigcap_{T \in \{\phi_2, \dots, \phi_{|V|-1}\}} \left\{ x \in \mathbb{R}^n : c^\top x \geq l_{t_1} - \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) \Delta_{t_j}(x) \right\},$$

where the intersection is over all valid sets $T = \{t_2, \dots, t_k\}$. We can now state the following inclusion result.

Corollary 3. *The projection of the continuous relaxation $\hat{O}_{2,\text{lin}}$ onto the x -space is a subset of the STI closure $\mathcal{C}(\mathcal{G})$; that is,*

$$\text{proj}_x(\hat{O}_{2,\text{lin}}) \subseteq \mathcal{C}(\mathcal{G}).$$

Proof. By Proposition 9, for any arbitrary subset $T \in \{\phi_2, \dots, \phi_{|V|-1}\}$ the inequality

$$c^\top x \geq l_{t_{k+1}} + \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) z_{t_j},$$

is satisfied if $(x, z) \in \hat{O}_{2,\text{lin}} = \hat{B}_{\text{tight}} \cap \hat{M}$. By Lemma 1, the inequality

$$z_v \geq 1 - \sum_{i \in \mathbb{1}_v} (1 - x_i) - \sum_{i \in \mathbb{0}_v} x_i.$$

is satisfied if $(x, z) \in \hat{O}_{2,\text{lin}} = \hat{B}_{\text{tight}} \cap \hat{M}$. Combining both inequalities as is done in the proof of validity of the STI demonstrates that any x for which there exists a z such that $(x, z) \in \hat{O}_{2,\text{lin}}$ also satisfies any STI and is hence part of the closure $\mathcal{C}(\mathcal{G})$. \square

Corollary 3 shows that no STI can cut off a point in the set $\text{proj}_x(\hat{O}_{2,\text{lin}})$. Although this may seem discouraging, it is only one aspect to consider. The practical efficacy of a family of inequalities also depends on the computational cost of their separation. The next section shows that a polynomial-time algorithm exists to separate the STIs, motivating their use in our numerical study.

5.2. Cut Generation. We now study the separation problem for the STIs. In contrast to the approaches discussed in Sections 2.3, 3.3, and 4.2, we do not use a CGLP. The following proposition shows that separation can be done with an efficient combinatorial algorithm.

Proposition 10. *The separation problem for STIs admits a polynomial-time combinatorial algorithm.*

Proof. The separation problem for a given point \bar{x} requires us to find a sequence $T = (t_2, \dots, t_k)$ that defines the most violated STI. This is equivalent to minimizing the right-hand side of Inequality (27).

Given a point \bar{x} , the values $\Delta_v(\bar{x})$ are computed in linear time. We show how this is carried out. Let $v \in V$ be an arbitrary node of the BB tree. We assume w.l.o.g. that the last branching carried out to reach v is $x_{i(v)} = 1$. Then, the following series of equalities hold,

$$\begin{aligned} \Delta_v(x) &= \min \left(1, \sum_{i \in \mathbb{1}_v} (1 - x_i) + \sum_{i \in \mathbb{0}_v} x_i \right), \\ &= \min \left(1, \sum_{i \in \mathbb{1}_{p(v)}} (1 - x_i) + (1 - x_{i(v)}) + \sum_{i \in \mathbb{0}_{p(v)}} x_i \right), \\ &= \min \left(1, \Delta_{p(v)}(x) + (1 - x_{i(v)}) \right). \end{aligned}$$

Hence, the term $\Delta_v(\bar{x})$ can be computed from the value of $\Delta_{p(v)}(\bar{x})$ which associates to its parent node $p(v)$. A straightforward algorithm starts at the root node and iteratively computes $\Delta_v(\bar{x})$ for the remaining child nodes until no more nodes are left. This algorithm needs a maximum of $\mathcal{O}(|V|)$ operations. Then, the value of Δ_v implies if the minimum operator is replaced by 1 or $\sum_{i \in \mathbb{1}_v} (1 - x_i) + \sum_{i \in \mathbb{0}_v} x_i$ to obtain a linear inequality which answers the point made in Remark 5.

With these pre-computed $\Delta_v(\bar{x})$ values, the separation problem reduces to finding a sequence $T = (t_2, \dots, t_k)$ that maximizes

$$l_{t_1} - \sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) \Delta_{t_j}(\bar{x}) - c^\top \bar{x}.$$

This is equivalent to minimizing the sum $\sum_{j=1}^k (l_{t_j} - l_{t_{j+1}}) \Delta_{t_j}(\bar{x})$. This optimization problem can be modeled as a shortest-path problem on a directed acyclic graph, just like the original mixing inequalities [35, 40]. It runs in $\mathcal{O}(|V|^2)$ iterations. \square

While a CGLP is theoretically powerful tool, its practical application for separating inequalities requires constructing and solving an auxiliary linear program that can be prohibitively large. For example, the CGLP for the set $\hat{O}_{1,\text{lin}}$ has $\mathcal{O}(|L|n)$ constraints, which can make the separation problem much larger than the original MBP. Modern solvers typically rely on specialized algorithms to generate cuts from specific families of inequalities. These algorithms often exploit the problem structure to generate cuts efficiently. This context emphasizes the practical value of families of inequalities, like the STIs, that admit tailored, polynomial-time separation algorithms. The trade-off between the theoretical strength of inequalities and the

practical cost of their separation is a central theme that we revisit in the numerical analysis in Section 6.

6. NUMERICAL ANALYSIS

Our numerical experiments are designed to achieve four primary objectives.

- (1) First, we empirically validate the theoretical inclusion hierarchy established in Proposition 5 and Corollary 3, namely, that

$$\text{proj}_x(O_{1,\text{lin}}) \subseteq \text{proj}_x(\hat{O}_{2,\text{lin}}) \subseteq \mathcal{C}(\mathcal{G}).$$

- (2) Second, we empirically compare the computational effort required for separation over each outer approximation. Let $\text{sep}(S)$ denote the time required to separate over a set S . We expect the separation time hierarchy to mirror the inclusion hierarchy, such that

$$\text{sep}(\mathcal{C}(\mathcal{G})) \leq \text{sep}(\text{proj}_x(\hat{O}_{2,\text{lin}})) \leq \text{sep}(\text{proj}_x(O_{1,\text{lin}})).$$

as discussed in Sections 2.3, 3.3, and 5.2. Moreover, such a comparison allows us to evaluate the tractability of each formulation and offers guidance on selecting an appropriate separation procedure.

- (3) One of the practical motivations of this paper is the use of the outer approximation to generate non-trivial valid inequalities that may be used when solving a problem with a similar structure but different problem coefficients. We want to show that there is information that can be salvaged from a BB tree and has value in a computational context. In addition, we want this information to be reused in an efficient manner, which is valuable for practical applications.
- (4) Fourth, we investigate the influence of the BB tree’s structure on the quality of the outer approximation. Specifically, we test the hypothesis that larger or more detailed trees yield tighter approximations relative to the feasible set F . Moreover, we also aim to provide insights into how the considered tree influences separation time.

To achieve these objectives, our experimental design is inspired by the “shooting” experiment of [38], in which the authors measure polyhedral approximations of a stability region. Our approach consists of a two-phase process. In the first phase, we solve a MBP instance, termed the “original instance”, and store the complete BB tree that certifies its optimality. In the second phase, we use this BB tree to generate valid inequalities for a “perturbed instance”. This perturbed instance is the same instance but with a perturbed set of objective coefficients.

We employ a pure cutting plane approach as used in, for example, [8, 30, 14]. This iterative procedure involves repeatedly solving the continuous relaxation of the perturbed instance and separating the most violated valid inequality until no more cuts can be found. We repeat this experiment for five perturbed objective vectors for each original instance. Furthermore, to study the impact of tree size, we replicate these experiments using truncated versions of the original BB tree.

6.1. Problem Instances and Computational Environment. Our numerical experiments are conducted on two classes of problems: multi-dimensional knapsack problems (MKP) and set-covering problems (SCP). For each class, we use instances with $n = 10, 20, 40$, and 60 variables, which have $n/2$ constraints, respectively. The instances were generated according to the procedure described in [17], with further details provided in Appendix A. All experiments were performed on a machine with two Intel(R) Xeon(R) Gold 6226 CPUs (12 cores each) and 384 GB of RAM. Each experiment was executed on a single thread. We use CPLEX 22.1 as the linear programming solver, and all algorithms are implemented in

Python. The source code and data for all experiments are publicly available at <https://gitlab.com/branchandboundtreeclosure/bnbtc>.

6.2. Implementation Details. As previously mentioned, our evaluation method consists of two phases. We now provide more details for the execution of each.

6.2.1. Branch-and-Bound on the Original Instance. We execute a branch-and-bound algorithm to solve the original instance to optimality. For each node in the resulting search tree, we record its feasibility status and, if feasible, the objective value of its linear relaxation. Unfortunately, open-source and commercial MIP solvers do not reliably provide all the information required for our analysis. For instance, Gurobi does not provide any access to the tree structure. On the other hand, while CPLEX, Xpress, and SCIP provide branching information, they only give partial information about explored nodes. Specifically, when a node is pruned, it is not possible to distinguish whether the pruning is due to the node bound being sub-optimal or due to the node being infeasible. Therefore, we implemented a custom branch-and-bound algorithm.

To ensure that clear patterns can be identified, our implementation is intentionally minimalist. The two main algorithmic components are: (i) node selection, which is performed using the best-first search, and (ii) branching, which is performed using the most-fractional variable rule. That is, we branch on the variable that has the most fractional value in the solution of the current node’s relaxation and has not been branched upon in the current path from the root to the node. Dual bounds are obtained by solving the continuous relaxation at each node. Valid inequalities are not added when solving the original instance.

6.2.2. Cut Generation on the Perturbed Instance. We apply a pure cutting plane approach to the perturbed instance. We compare the performance of valid inequalities generated from the three proposed outer approximations: the classical disjunctive approximation $\text{proj}_x(O_{1,\text{lin}})$, the network-design reformulation $\text{proj}_x(\hat{O}_{2,\text{lin}})$ obtained after reformulating and tightening the binary polynomial set, and the closure of the star-tree inequalities $\mathcal{C}(\mathcal{G})$. As a baseline, we also consider a trivial outer approximation formed by the single objective cut $c^\top x \geq l_{\text{opt}}$, where l_{opt} is the tightest valid dual bound available in the considered tree. The value of l_{opt} , equals the optimal objective value if the BB tree certifies optimality for the original instance. This baseline, which we refer to as **Obj**, demonstrates what is achievable without the proposed outer approximations.

The perturbed instances are created by modifying the objective coefficients of the original instance. Each coefficient c_i is perturbed by sampling from a normal distribution with mean c_i and standard deviation $0.1 \cdot |c_i|$. The cut generation process for each perturbed instance is subject to a strict 10-minute time limit. This limit includes all computations: constructing the CGLP, solving the CGLP, extracting and adding the new cut, and re-solving the tightened linear relaxation of the perturbed instance. The CGLP for a given tree is constructed only once and is reused in all subsequent iterations. For $\mathcal{C}(\mathcal{G})$, we implement the separation procedure from Proposition 10, which does not require a CGLP. This entire process is repeated for five different perturbed versions of each original instance. Furthermore, each experiment is replicated using truncated BB trees. We truncate a tree by retaining only the nodes up to a certain depth d , defined as

$$d = r_{\text{depth}} d_{\text{max}},$$

where d_{max} is the maximum depth of the full tree and the depth ratio r_{depth} is varied in $\{0.25, 0.5, 0.75, 1.0\}$.

6.3. Results. Tables 1 and 2 summarize the results for the MKP and SCP instances, respectively. Each table reports the following metrics: instance type (Instance), instance size (Size), the outer approximation used (Approx), the relative tree depth (Depth), the final dual bound tightness (Gap), the total time for the cutting plane procedure (Time), the number of instances (out of 5) that reach the cut generation time limit (Timeout) and the number of cuts generated (Cuts).

The dual bound tightness is measured by the relative gap with respect to the true optimal value of the perturbed instance,

$$\text{Gap} = \frac{l_{\text{dual}} - l_{\text{opt}}}{l_{\text{opt}}},$$

where l_{dual} is the dual bound obtained after the cutting plane procedure terminates and l_{opt} is the optimal objective value of the perturbed instance. The reported values for the optimality gap, computation time, and number of cuts are averages computed over the five perturbed instances.

The results in Tables 1 and 2 are consistent with the inclusion hierarchy established previously. For instances where no cutting plane method timed out, we observe that Obj , $\mathcal{C}(\mathcal{G})$, $\text{proj}_x(\hat{O}_{2,\text{lin}})$, and $\text{proj}_x(O_{1,\text{lin}})$ yield progressively tighter relative gaps, which confirms our first objective.

Second, the separation times mirror the inverse of the inclusion hierarchy. On instances where no method timed out, the total separation times for Obj , $\mathcal{C}(\mathcal{G})$, $\text{proj}_x(\hat{O}_{2,\text{lin}})$, and $\text{proj}_x(O_{1,\text{lin}})$ are progressively longer for the same tree depth. Also, when $\text{proj}_x(O_{1,\text{lin}})$ is used to approximate large problems, more time-outs occur due to the high computational cost associated. This confirms our second objective. We note that while separating the STI can be time-consuming, as is the case for MKP 60, for instance, the separation algorithm used in the experiments is a straightforward Python implementation. Significant speed-ups can be achieved by implementing this procedure in a compiled language such as C++.

Regarding our third objective, in the context of reoptimization, the proposed outer approximations effectively reduce the optimality gap. While the improvement over the Obj method is marginal in a few cases (e.g., for SCP 10 with a tree depth of 0.25), the approximations always perform better, often reducing the gap by a quarter and in some cases by half. It is difficult to conclude whether the methods are more effective for MKP or SCP instances, as performance depends on multiple factors. We remark that, although we have shown that there is potential to implement the proposed cutting planes in a reoptimization fashion, carrying this out in practice is a very complicated task that requires deep knowledge of modern optimization solvers, substantial engineering effort, and extensive testing. Therefore, we believe this to be beyond the scope of this paper and deserving of its own research endeavor. Instead, our work offers a good starting point for identifying which outer approximation should be used in specific reoptimization contexts.

Finally, we address our fourth objective concerning the impact of tree size. A clear trend is observed: outer approximations constructed using deeper trees yield smaller optimality gaps. Interestingly, using larger trees does not uniformly increase the total separation time. For instance, with the MKP 20 instance, separation using the full-depth tree (depth ratio 1.0) is faster than using a tree with a depth ratio of 0.75. This highlights that tree selection is a critical factor influencing the computational tractability of the approach. A more striking example is the SCP 60 instance with $\text{proj}_x(O_{1,\text{lin}})$, where the total separation time drops dramatically from 120 to 0.13 seconds when increasing the tree depth ratio from 0.50 to 1.00. This suggests that larger, more informative trees can sometimes lead to deeper cuts such that the overall separation process becomes faster.

TABLE 1. Results on MKP instances

Instance	Size	Approx	Depth	Gap	Time	Timeout	Cuts
			0.50	7.12 %	0.00	0	1.0
			0.75	6.18 %	0.00	0	1.0
			1.00	2.33 %	0.00	0	1.0
		$\mathcal{C}(\mathcal{G})$	0.25	7.87 %	0.00	0	1.2
			0.50	6.86 %	0.00	0	1.8
			0.75	5.73 %	0.01	0	5.2
			1.00	1.70 %	0.01	0	4.6
		$\text{proj}_x(\hat{O}_{2,\text{lin}})$	0.25	7.85 %	0.00	0	1.0
			0.50	6.83 %	0.00	0	2.4
			0.75	5.52 %	0.01	0	10.2
			1.00	1.29 %	0.01	0	9.0
		$\text{proj}_x(O_{1,\text{lin}})$	0.25	7.85 %	0.00	0	1.2
			0.50	6.80 %	0.01	0	2.2
			0.75	5.44 %	0.02	0	3.8
			1.00	1.24 %	0.03	0	3.4
	20	Obj	0.25	2.96 %	0.00	0	1.0
			0.50	2.73 %	0.00	0	1.0
			0.75	2.48 %	0.00	0	1.0
			1.00	1.25 %	0.00	0	1.0
		$\mathcal{C}(\mathcal{G})$	0.25	2.84 %	0.00	0	2.4
			0.50	2.52 %	0.02	0	3.4
			0.75	2.25 %	0.08	0	6.6
			1.00	1.20 %	0.07	0	4.6
		$\text{proj}_x(\hat{O}_{2,\text{lin}})$	0.25	2.83 %	0.00	0	1.8
			0.50	2.42 %	0.02	0	10.0
			0.75	2.05 %	0.08	0	24.0
			1.00	1.15 %	0.07	0	16.6
		$\text{proj}_x(O_{1,\text{lin}})$	0.25	2.83 %	0.01	0	1.8
			0.50	2.41 %	0.30	0	5.2
			0.75	2.02 %	120.57	1	6366.2
			1.00	1.12 %	121.55	1	6612.0
	40	Obj	0.25	1.56 %	0.00	0	0.8
			0.50	1.36 %	0.00	0	1.0
			0.75	1.26 %	0.00	0	1.0
			1.00	1.02 %	0.00	0	1.0
		$\mathcal{C}(\mathcal{G})$	0.25	1.54 %	0.07	0	3.0
			0.50	1.31 %	1.56	0	4.6
			0.75	1.17 %	3.11	0	4.8
			1.00	1.02 %	1.85	0	2.6
		$\text{proj}_x(\hat{O}_{2,\text{lin}})$	0.25	1.50 %	0.02	0	4.2
			0.50	1.14 %	120.72	1	4263.0
			0.75	1.01 %	240.51	2	7587.0
			1.00	1.00 %	120.77	1	2958.4
		$\text{proj}_x(O_{1,\text{lin}})$	0.25	1.50 %	2.50	0	2.8
			0.50	1.17 %	484.82	4	3925.4
			0.75	1.04 %	600.00	5	5970.2
			1.00	1.01 %	510.38	4	3381.8
	60	Obj	0.25	1.05 %	0.00	0	1.0
			0.50	0.93 %	0.00	0	1.0
			0.75	0.84 %	0.00	0	1.0
			1.00	0.76 %	0.00	0	1.0
		$\mathcal{C}(\mathcal{G})$	0.25	1.00 %	10.41	0	2.8
			0.50	0.88 %	43.97	0	4.4
			0.75	0.80 %	35.51	0	3.2
			1.00	0.76 %	23.47	0	2.4
		$\text{proj}_x(\hat{O}_{2,\text{lin}})$	0.25	0.92 %	241.18	2	4160.4
			0.50	0.78 %	251.99	2	1935.8
			0.75	0.76 %	251.36	2	1963.0
			1.00	0.75 %	42.14	0	38.0
		$\text{proj}_x(O_{1,\text{lin}})$	0.25	1.14 %	480.03	4	0.6
			0.50	1.14 %	600.00	5	2045.2
			0.75	1.13 %	600.00	5	0.2
			1.00	1.13 %	600.00	5	0.2

TABLE 2. Results on SCP instances

Instance	Size	Approx	Depth	Gap	Time	Timeout	Cuts
SCP	10	obj	0.25	10.34 %	0.00	0	0.0
			0.50	8.07 %	0.00	0	0.2
			0.75	8.07 %	0.00	0	0.2
			1.00	3.66 %	0.00	0	0.6
		$\mathcal{C}(\mathcal{G})$	0.25	10.34 %	0.00	0	0.0
			0.50	8.05 %	0.00	0	0.2
			0.75	8.05 %	0.00	0	0.2
			1.00	2.74 %	0.00	0	0.6
		$\text{proj}_x(\hat{O}_{2,\text{lin}})$	0.25	10.34 %	0.00	0	0.0
			0.50	8.05 %	0.00	0	0.2
			0.75	8.05 %	0.00	0	0.2
			1.00	2.74 %	0.00	0	0.6
		$\text{proj}_x(O_{1,\text{lin}})$	0.25	10.34 %	0.00	0	0.0
			0.50	8.05 %	0.00	0	0.2
			0.75	8.05 %	0.00	0	0.2
			1.00	2.74 %	0.00	0	0.6
	20	obj	0.25	12.09 %	0.00	0	0.2
			0.50	10.92 %	0.00	0	0.6
			0.75	10.26 %	0.00	0	0.8
			1.00	7.45 %	0.00	0	1.0
		$\mathcal{C}(\mathcal{G})$	0.25	11.57 %	0.00	0	0.2
			0.50	9.82 %	0.00	0	0.8
			0.75	9.35 %	0.00	0	1.2
			1.00	5.62 %	0.00	0	1.6
		$\text{proj}_x(\hat{O}_{2,\text{lin}})$	0.25	11.57 %	0.00	0	0.2
			0.50	9.82 %	0.00	0	1.0
			0.75	9.16 %	0.00	0	1.2
			1.00	4.88 %	0.00	0	1.8
		$\text{proj}_x(O_{1,\text{lin}})$	0.25	11.57 %	0.00	0	0.2
			0.50	9.82 %	0.00	0	0.8
			0.75	9.16 %	0.00	0	1.0
			1.00	4.88 %	0.00	0	1.6
	40	obj	0.25	14.18 %	0.00	0	0.6
			0.50	11.78 %	0.00	0	0.8
			0.75	8.89 %	0.00	0	0.8
			1.00	5.02 %	0.00	0	1.0
		$\mathcal{C}(\mathcal{G})$	0.25	13.71 %	0.00	0	1.0
			0.50	10.85 %	0.00	0	2.6
			0.75	8.07 %	0.01	0	4.4
			1.00	3.77 %	0.00	0	1.8
		$\text{proj}_x(\hat{O}_{2,\text{lin}})$	0.25	13.62 %	0.00	0	1.0
			0.50	10.55 %	0.00	0	2.0
			0.75	7.38 %	0.01	0	3.2
			1.00	3.58 %	0.01	0	3.0
		$\text{proj}_x(O_{1,\text{lin}})$	0.25	13.62 %	0.00	0	1.0
			0.50	10.55 %	0.02	0	2.0
			0.75	7.18 %	0.05	0	4.0
			1.00	3.44 %	0.05	0	3.2
	60	obj	0.25	24.94 %	0.00	0	1.0
			0.50	21.46 %	0.00	0	1.0
			0.75	16.75 %	0.00	0	1.0
			1.00	13.16 %	0.00	0	1.0
		$\mathcal{C}(\mathcal{G})$	0.25	23.65 %	0.00	0	2.2
			0.50	20.20 %	0.01	0	4.4
			0.75	15.88 %	0.01	0	4.6
			1.00	12.75 %	0.01	0	1.8
		$\text{proj}_x(\hat{O}_{2,\text{lin}})$	0.25	23.39 %	0.00	0	1.6
			0.50	19.55 %	0.01	0	6.2
			0.75	15.40 %	0.01	0	5.6
			1.00	12.64 %	0.01	0	3.4
		$\text{proj}_x(O_{1,\text{lin}})$	0.25	23.39 %	0.01	0	1.8
			0.50	19.43 %	120.06	1	3933.0
			0.75	15.19 %	0.18	0	6.4
			1.00	12.63 %	0.13	0	3.0

7. CONCLUSION

This paper introduced three a-posteriori outer approximations of the feasible region of mixed-binary programs derived from a branch-and-bound (BB) tree, together with a new family of valid inequalities, the Star Tree Inequalities (STIs). We established an inclusion hierarchy among the approximations (disjunctive programming-based, branching-based, and mixing-set based), as well as a reverse separation time hierarchy. This hierarchy theoretically formalizes the trade-off between tightness and computational tractability of the proposed approaches. The STIs, while theoretically weakest in closure, stand out practically due to a polynomial-time combinatorial separation algorithm. Our computational study on multi-dimensional knapsack and set-covering problems empirically confirms the hierarchy and demonstrates that cuts generated a posteriori from BB trees can reduce optimality gaps in perturbed instances.

These results support the idea that BB trees contain structural information about the underlying optimization problem that is valuable beyond a single solve. Reusing this information can be leveraged in multiple settings, such as reoptimization, restarts, decomposition, or sensitivity analyses of problem parameters. More generally, this paper highlights a novel perspective on data-driven optimization: rather than treating each solve as a one-shot experiment, one can view the entire process as a sequence where information accumulates and can be systematically reused. From this perspective, a central trade-off emerges between exploration and exploitation. Building richer trees and tighter approximations might require more time at first, but may provide substantial speed-ups later on as the available BB information is exploited to generate strong cuts. This dilemma motivates research on new design principles for optimization solvers.

ACKNOWLEDGMENTS

The support of SCALE-AI through its Research Chairs program, IVADO, the Canada First Research Excellence Fund (Apogée/CFREF), as well as the computational infrastructure provided by Compute Canada are gratefully acknowledged.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest that could have influenced the work reported in this paper.

REFERENCES

- [1] C. Ahmed, A. Forel, A. Parmentier, and T. Vidal. “DistrictNet: Decision-aware learning for geographical districting.” In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 128574–128602.
- [2] C. Artigues, S. Demassey, and E. Neron. *Resource-constrained project scheduling: models, algorithms, extensions and applications*. John Wiley & Sons, 2013.
- [3] A. Atamtürk, G. L. Nemhauser, and M. W. Savelsbergh. “The mixed vertex packing problem.” In: *Mathematical Programming* 89 (2000), pp. 35–53. DOI: [10.1007/s101070000154](https://doi.org/10.1007/s101070000154).
- [4] E. Balas. *Disjunctive programming*. Springer, 2018.
- [5] E. Balas. “Disjunctive programming and a hierarchy of relaxations for discrete optimization problems.” In: *SIAM Journal on Algebraic Discrete Methods* 6.3 (1985), pp. 466–486. DOI: [10.1137/0606047](https://doi.org/10.1137/0606047).

- [6] E. Balas, S. Ceria, and G. Cornuéjols. “A lift-and-project cutting plane algorithm for mixed 0–1 programs.” In: *Mathematical Programming* 58.1 (1993), pp. 295–324. DOI: [10.1007/BF01581273](https://doi.org/10.1007/BF01581273).
- [7] E. Balas, S. Ceria, and G. Cornuéjols. “Mixed 0-1 programming by lift-and-project in a branch-and-cut framework.” In: *Management Science* 42.9 (1996), pp. 1229–1246. DOI: [10.1287/mnsc.42.9.1229](https://doi.org/10.1287/mnsc.42.9.1229).
- [8] E. Balas and A. Saxena. “Optimizing over the split closure.” In: *Mathematical Programming* 113.2 (2008), pp. 219–240. DOI: [10.1007/s10107-006-0049-5](https://doi.org/10.1007/s10107-006-0049-5).
- [9] A. Basu, M. Conforti, M. Di Summa, and H. Jiang. “Complexity of branch-and-bound and cutting planes in mixed-integer optimization.” In: *Mathematical Programming* 198.1 (2023), pp. 787–810. DOI: [10.1007/s10107-022-01789-5](https://doi.org/10.1007/s10107-022-01789-5).
- [10] B. Becu, S. S. Dey, F. Qiu, and A. S. Xavier. *Approximating the Gomory Mixed-Integer Cut Closure Using Historical Data*. 2024. arXiv: [2411.15090](https://arxiv.org/abs/2411.15090) [math.OC].
- [11] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach. “Learning with Differentiable Perturbed Optimizers.” In: *Advances in neural information processing systems* 33 (2020), pp. 9508–9519.
- [12] S. Bolusani, M. Besançon, K. Bestuzheva, A. Chmiela, J. Dionísio, T. Donkiewicz, J. van Doornmalen, L. Eifler, M. Ghannam, A. Gleixner, C. Graczyk, K. Halbig, I. Hedtke, A. Hoen, C. Hojny, R. van der Hulst, D. Kamp, T. Koch, K. Kofler, J. Lentz, J. Manns, G. Mexi, E. Mühmer, M. E. Pfetsch, F. Schlösser, F. Serrano, Y. Shinano, M. Turner, S. Vigerske, D. Weninger, and L. Xu. *The SCIP Optimization Suite 9.0*. 2024. arXiv: [2402.17702](https://arxiv.org/abs/2402.17702) [math.OC].
- [13] J. A. Bondy, U. S. R. Murty, et al. *Graph theory with applications*. Vol. 290. Macmillan London, 1976.
- [14] D. Cattaruzza, M. Labbé, M. Petris, M. Roland, and M. Schmidt. *On Multidimensional Disjunctive Inequalities for Chance-Constrained Stochastic Problems with Finite Support*. 2025.
- [15] B. Chen, S. Küçükyavuz, and S. Sen. “A computational study of the cutting plane tree algorithm for general mixed-integer linear programs.” In: *Operations research letters* 40.1 (2012), pp. 15–19. DOI: [10.1016/j.orl.2011.10.009](https://doi.org/10.1016/j.orl.2011.10.009).
- [16] B. Chen, S. Küçükyavuz, and S. Sen. “Finite disjunctive programming characterizations for general mixed-integer linear programs.” In: *Operations Research* 59.1 (2011), pp. 202–210. DOI: [10.1287/opre.1100.0882](https://doi.org/10.1287/opre.1100.0882).
- [17] P. C. Chu and J. E. Beasley. “A genetic algorithm for the multidimensional knapsack problem.” In: *Journal of Heuristics* 4.1 (1998), pp. 63–86. DOI: [10.1023/A:1009642405419](https://doi.org/10.1023/A:1009642405419).
- [18] V. Chvátal. “Hard knapsack problems.” In: *Operations Research* 28.6 (1980), pp. 1402–1411. DOI: [10.1287/opre.28.6.1402](https://doi.org/10.1287/opre.28.6.1402).
- [19] F. Clautiaux and I. Ljubić. “Last fifty years of integer linear programming: A focus on recent practical advances.” In: *European Journal of Operational Research* 324.3 (2025), pp. 707–731. DOI: [10.1016/j.ejor.2024.11.018](https://doi.org/10.1016/j.ejor.2024.11.018).
- [20] M. Conforti, G. Cornuéjols, G. Zambelli, M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer programming models*. Springer, 2014.
- [21] C. Contardo, A. Lodi, and A. Tramontani. “Cutting planes from the branch-and-bound tree: Challenges and opportunities.” In: *INFORMS Journal on Computing* 35.1 (2023), pp. 2–4. DOI: [10.1287/ijoc.2022.1248](https://doi.org/10.1287/ijoc.2022.1248).
- [22] G. Cornuéjols and Y. Dubey. “Branch-and-Bound versus Lift-and-Project Relaxations in Combinatorial Optimization: G. Cornuéjols, Y. Dubey.” In: *Mathematical Programming* (2025), pp. 1–12. DOI: [10.1007/s10107-025-02248-7](https://doi.org/10.1007/s10107-025-02248-7).

- [23] T. G. Crainic, M. Gendreau, and B. Gendron. “Fixed-charge network design problems.” In: *Network design with applications to transportation and logistics*. Springer, 2020, pp. 15–28.
- [24] G. Dalle, L. Baty, L. Bouvier, and A. Parmentier. *Learning with Combinatorial Optimization Layers: a Probabilistic Approach*. 2022. arXiv: [2207.13513 \[stat.ML\]](#).
- [25] S. S. Dey, Y. Dubey, and M. Molinaro. “Branch-and-bound solves random binary ips in polytime.” In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2021, pp. 579–591. DOI: [10.1137/1.9781611976465.35](#).
- [26] S. S. Dey, Y. Dubey, and M. Molinaro. “Lower bounds on the size of general branch-and-bound trees.” In: *Mathematical Programming* 198.1 (2023), pp. 539–559. DOI: [10.1007/s10107-022-01781-z](#).
- [27] S. S. Dey, Y. Dubey, M. Molinaro, and P. Shah. “A theoretical and computational analysis of full strong-branching.” In: *Mathematical Programming* 205.1 (2024), pp. 303–336. DOI: [10.1007/s10107-023-01977-x](#).
- [28] S. Elloumi and Z. Verchère. “Efficient linear reformulations for binary polynomial optimization problems.” In: *Computers & Operations Research* 155 (2023), p. 106240. DOI: [10.1016/j.cor.2023.106240](#).
- [29] A. N. Elmachtoub and P. Grigas. “Smart “predict, then optimize”.” In: *Management Science* 68.1 (2022), pp. 9–26. DOI: [10.1287/mnsc.2020.3922](#).
- [30] M. Fischetti and A. Lodi. “Optimizing over the first Chvátal closure.” In: *Mathematical Programming* 110.1 (2007), pp. 3–20. DOI: [10.1007/s10107-006-0054-8](#).
- [31] M. Fischetti and M. Monaci. “Backdoor branching.” In: *INFORMS Journal on Computing* 25.4 (2013), pp. 693–700. DOI: [10.1287/ijoc.1120.0531](#).
- [32] N. Fleming, M. Göös, R. Impagliazzo, T. Pitassi, R. Robere, L.-Y. Tan, and A. Wigderson. *On the Power and Limitations of Branch and Cut*. 2021. arXiv: [2102.05019 \[cs.CC\]](#).
- [33] G. Gamrath, B. Hiller, and J. Witzig. “Reoptimization Techniques for MIP Solvers.” In: *Experimental Algorithms*. Ed. by E. Bampis. Cham: Springer International Publishing, 2015, pp. 181–192. DOI: [10.1007/978-3-319-20086-6_14](#).
- [34] M. Gläser and M. E. Pfetsch. “On computing small variable disjunction branch-and-bound trees.” In: *Mathematical Programming* 206.1 (2024), pp. 145–173. DOI: [10.1007/s10107-023-01968-y](#).
- [35] O. Günlük and Y. Pochet. “Mixing mixed-integer inequalities.” In: *Mathematical Programming* 90 (2001), pp. 429–457. DOI: [10.1007/PL00011430](#).
- [36] R. G. Jeroslow. “Trivial integer programs unsolvable by branch-and-bound.” In: *Mathematical Programming* 6.1 (1974), pp. 105–109. DOI: [10.1007/BF01580225](#).
- [37] M. Kılınç, J. Linderoth, J. Luedtke, and A. Miller. “Strong-branching inequalities for convex mixed integer nonlinear programs.” In: *Computational Optimization and Applications* 59.3 (2014), pp. 639–665. DOI: [10.1007/s10589-014-9690-8](#).
- [38] F. Kılınç-Karzan, A. Toriello, S. Ahmed, G. Nemhauser, and M. Savelsbergh. “Approximating the stability region for binary mixed-integer programs.” In: *Operations Research Letters* 37.4 (2009), pp. 250–254. DOI: [10.1016/j.orl.2009.04.001](#).
- [39] A. H. Land and A. G. Doig. “An Automatic Method of Solving Discrete Programming Problems.” In: *Econometrica* 28.3 (1960), pp. 497–520. DOI: [10.2307/1910129](#).

- [40] J. Luedtke, S. Ahmed, and G. L. Nemhauser. “An integer programming approach for linear programs with probabilistic constraints.” In: *Mathematical Programming* 122.2 (2010), pp. 247–272. DOI: [10.1007/s10107-008-0247-4](https://doi.org/10.1007/s10107-008-0247-4).
- [41] A. Mahajan and T. Ralphs. “On the complexity of selecting disjunctions in integer programming.” In: *SIAM Journal on Optimization* 20.5 (2010), pp. 2181–2198. DOI: [10.1137/080737587](https://doi.org/10.1137/080737587).
- [42] G. Muñoz, J. Paat, and Á. S. Xavier. “Compressing branch-and-bound trees.” In: *Mathematical Programming* 210.1 (2025), pp. 669–694. DOI: [10.1007/s10107-024-02080-5](https://doi.org/10.1007/s10107-024-02080-5).
- [43] J. H. Owen and S. Mehrotra. “A disjunctive cutting plane procedure for general mixed-integer linear programs.” In: *Mathematical Programming* 89.3 (2001), pp. 437–448. DOI: [10.1007/PL00011407](https://doi.org/10.1007/PL00011407).
- [44] U. Sadana, A. Chenreddy, E. Delage, A. Forel, E. Frejinger, and T. Vidal. “A survey of contextual optimization methods for decision-making under uncertainty.” In: *European Journal of Operational Research* (2024). DOI: [10.1016/j.ejor.2024.03.020](https://doi.org/10.1016/j.ejor.2024.03.020).
- [45] W. Van Ackooij, I. Danti Lopez, A. Frangioni, F. Lacalandra, and M. Tahanan. “Large-scale unit commitment under uncertainty: an updated literature survey.” In: *Annals of Operations Research* 271.1 (2018), pp. 11–85. DOI: [10.1007/s10479-018-3003-z](https://doi.org/10.1007/s10479-018-3003-z).
- [46] J. Witzig. “Reoptimization techniques in MIP solvers.” MA thesis. 2014.

APPENDIX A. INSTANCE GENERATION

This section describes the procedure used to generate the test instances for the numerical experiments presented in the main text.

Multi-Knapsack Problem: Given a number of variables n , we construct instances of the multi-knapsack problem in the following form:

$$\max\{c^T x : Ax \leq b\},$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^{n/2}$ and $A \in \mathbb{R}^{(n/2) \times n}$. The instance data is generated as follows:

- Each entry of the cost vector c is drawn independently from the uniform distribution $\mathcal{U}(1, 2)$.
- Each entry of the matrix A is drawn independently from $\mathcal{U}(0, 1)$.
- Each entry of b is defined as

$$b_i = 0.9 \sum_{j=1}^n A_{ij}, \quad i = 1, \dots, n/2.$$

Set-Covering Problem: Given a number of variables n , representing the number of subsets, and a number of constraints m , representing the elements of the ground set, we consider a density parameter $q \in (0, 1)$ that specifies the probability that a given element belongs to a subset. An instance of the Set-Covering Problem can then be generated as

$$\max\{c^T x : Ax \geq 1\},$$

where $c \in \mathbb{R}^n$ and $A \in \{0, 1\}^{m \times n}$. Each entry of A takes the value 1 independently with probability q , and the cost vector c is drawn independently from the uniform distribution $\mathcal{U}(1, 2)$.

To simulate a change in the objective function, we generate a perturbed instance from each original instance by modifying the cost vector. Let c be the cost vector of the original instance. The perturbed cost vector \tilde{c} is defined as:

$$\tilde{c} = c + \varepsilon,$$

where each component ε_i is sampled independently from the normal distribution $\mathcal{N}(0, 0.1c_i)$, for $i = 1, \dots, n$.

(Marius Roland) (A) UNIV. LILLE, INRIA, CNRS, CENTRALE LILLE, UMR 9189 CRISTAL, LILLE, F-59000, FRANCE; (B) POLYTECHNIQUE MONTRÉAL, ANDRÉ-AISENSTADT PAVILLON, 2920 TOUR ROAD, MONTREAL, QUEBEC H3T 1N8, CANADA

Email address: `mmmroland@gmail.com`

(Nagisa Sugishita) UNIVERSITÉ DE MONTRÉAL, ANDRÉ-AISENSTADT PAVILLON, 2920 TOUR ROAD, MONTREAL, QUEBEC H3T 1N8, CANADA

Email address: `nagisa.sugishita@umontreal.ca`

(Alexandre Forel) POLYTECHNIQUE MONTRÉAL, ANDRÉ-AISENSTADT PAVILLON, 2920 TOUR ROAD, MONTREAL, QUEBEC H3T 1N8, CANADA, *The work was conducted prior to joining Amazon.*

Email address: `alex.forel@gmail.com`

(Youssef Emine) POLYTECHNIQUE MONTRÉAL, ANDRÉ-AISENSTADT PAVILLON, 2920 TOUR ROAD, MONTREAL, QUEBEC H3T 1N8, CANADA

Email address: `youssef.emine@gmail.com`

(Ricardo Fukasawa) UNIVERSITY OF WATERLOO, 200 UNIVERSITY AVENUE WEST, WATERLOO, ONTARIO N2L3G1, CANADA

Email address: `rfukasawa@uwaterloo.ca`

(Thibaut Vidal) POLYTECHNIQUE MONTRÉAL, ANDRÉ-AISENSTADT PAVILLON, 2920 TOUR ROAD, MONTREAL, QUEBEC H3T 1N8, CANADA

Email address: `thibaut.vidal@polymtl.ca`