

Non-Monotonicity of Branching Rules with respect to Linear Relaxations

Prachi Shah^{*1}, Santanu S. Dey^{†1}, and Marco Molinaro^{‡2}

¹School of Industrial and Systems Engineering, Georgia Institute of Technology

²Microsoft Research (Redmond) and Computer Science Department, PUC-Rio

Abstract

Modern mixed-integer programming solvers use the branch-and-cut framework, where cutting planes are added to improve the tightness of the linear programming (LP) relaxation, with the expectation that the tighter formulation would produce smaller branch-and-bound trees. In this work, we consider the question of whether adding cuts will always lead to smaller trees for a given fixed branching rule. We formally call such a property of a branching rule monotonicity. We prove that any branching rule which exclusively branches on fractional variables in the LP solution is non-monotonic. Moreover, we present a family of instances where adding a single cut leads to an exponential increase in the size of full strong branching trees, despite improving the LP bound. Finally, we empirically attempt to estimate the prevalence of non-monotonicity in practice while using full strong branching. We consider randomly generated multi-dimensional knapsacks tightened by cover cuts as well as instances from the MIPLIB 2017 benchmark set for the computational experiments. Our main insight from these experiments is that if the gap closed by cuts is small, change in tree size is difficult to predict, and often increases, possibly due to inherent non-monotonicity. However, when a sufficiently large gap is closed, a significant decrease in tree size may be expected.

1 Introduction

State-of-the-art mixed integer programming (MIP) solvers are based on the branch-and-cut framework. Classically, the cutting-plane method [14, 21] and the branch-and-bound method [23] were mostly explored independently. It was later discovered that combining them, by first adding cutting-planes to improve the formulation followed by applying branch-and-bound method produces good computational benefits. See [11, 26] for great expositions on the origins of the branch-and-cut framework, including discussion on landmark papers such as [12, 29, 28, 5, 6]. In recent years, there have been multiple studies understanding the trade-off between branching and the use of cutting-planes [7, 8, 22]. In this paper, we study another aspect of the interaction between cutting-planes and the branch-and-bound procedure.

Let us consider two formulations of a given MIP, where the second formulation is tighter than the first, that is the feasible region of the second Linear Programming (LP) relaxation is contained in that of the first relaxation. One could obtain the second relaxation, for example, by adding cutting-planes to the first relaxation. Suppose we have a branch-and-bound tree that solves the

*prachi.shah@gatech.edu

†santanu.dey@isye.gatech.edu

‡mmolinaro@microsoft.com

MIP using the first formulation. Then, it is clear that the same branch-and-bound tree, that is a tree with the same branching decisions, solves the MIP using the second formulation. Therefore, the branch-and-bound tree for the first formulation acts as a *branch-and-bound certificate* for the second formulation as well. Thus, one expects that the branch-and-bound tree needed to solve the MIP using the tighter formulation becomes smaller— this is the basis of the branch-and-cut framework. However, in order to use the branch-and-bound method, one must specify two rules: the node selection rule and the branching (variable selection) rule. Once these rules are specified, we formally obtain a *branch-and-bound algorithm*. The branching rule typically uses local node LP information to decide on which variable to branch. Therefore, it may be possible that a given branch-and-bound algorithm does not produce the same trees as certificates for the first and the second formulation - even worse it may produce a larger tree for the tighter formulation. The goal of this paper is to study the effect of branching rules on the size of branch-and-bound trees with respect to different formulations.

Formalizing the size of a tree for a given branching rule. Consider two formulations of a MIP: $P, P' \subseteq \mathbb{R}^n$ such that $P \cap \mathcal{X} = P' \cap \mathcal{X}$ where \mathcal{X} represents integrality constraints. In order to meaningfully compare the size of branch-and-bound trees generated by a given branching rule for these two polytopes, we should control for all other sources of variability of the branch-and-bound tree:

(1.) *Dual degeneracy:* If there are multiple optimal solutions of the LP at any node of the branch-and-bound tree, then we cannot control which optimal vertex is reported by the LP solver. Since branching rules often use local information, such as the list of fractional variables at a node, this can lead to different trees. An extreme case is when one of the optimal vertex is integral, but other vertices are not integral, see discussion in [16]. One way to resolve this situation is to compare branch-and-bound trees for P and P' , only for objective functions that do not cause any dual degeneracy at any node of any branch-and-bound tree for both P and P' . Let $C(P, P') \subseteq \mathbb{R}^n$ be the set of these objectives with no dual degeneracy. Note that $C(P, P')$ only discards finitely many objective functions.

(2.) *Node selection rule:* We will assume that nodes are processed in the best bound first order. Note that once we solve a problem with any node selection, one can always recover a subtree that solves the same instance, where the node selection rule for the subtree is the best-bound rule – that is the best-bound rule for node selection produces the smallest trees; see [15] for properties of this rule.

We let $T^r(P, c)$ be the branch-and-bound tree produced by the branching rule r when provided with relaxation P and objective function $c \in C(P, P')$ with the above assumptions. We define the size of the tree as the total number of its nodes and denote it as $|T^r(P, c)|$.

Our contributions. Given two linear relaxation P and P' for a MIP, that is $P' \cap \mathcal{X} = P \cap \mathcal{X}$, we say P' is tighter than P when $P' \subseteq P$. Motivated by the discussion above, we define the following:

Definition 1. A branching rule r is said to be monotonic, if for any MIP, given two linear relaxations P and P' with P' being tighter than P , for all $c \in C(P, P')$ we have that

$$|T^r(P', c)| \leq |T^r(P, c)|$$

Ideally, we may hope for “good” branching rules to be monotonic, which would be a theoretical justification of the branch-and-cut algorithm.

Our contributions can be summarized as follows:

1. We show that any rule that branches exclusively on fractional variables in the LP solution is non-monotonic. As a consequence, most standard branching rules in literature, including full strong branching (FSB), are not monotonic.
2. In particular, we prove that adding a single cut may lead to an exponential increase in the size of branch-and-bound trees for most of these branching rules.
3. Through computational experiments we attempt to estimate the prevalence of non-monotonicity in practice while using FSB. We do so by applying cover cuts on randomly generated multi-dimensional knapsacks as well as by considering cuts applied by SCIP [9] on MIPLIB 2017 benchmark set [20]. Our main insight from these experiments is that if the gap closed by cuts is small, change in tree size is difficult to predict, and often increases, possibly due to inherent non-monotonicity. However, when a sufficiently large gap is closed, a significant decrease in tree size may be expected.

The rest of the paper is organized as follows. In Section 2 we provide the background on branching rules in literature and present our theoretical contributions. Section 3 discusses the computational experiments and the results. Finally in Section 4 we conclude and discuss future directions of research.

2 Non-Monotonicity of Branching Rules

2.1 Background on branching rules

One of the most important branching rules is called the *full strong branching rule* (FSB) [4]. It is empirically known to produce one of the smallest trees as compared to all other branching rules [2]. Let Δ_i^0 and Δ_i^1 be the LP gains, i.e. the difference in the LP value of a node and its children when tentatively branched on variable i . Then Δ_i^0 and Δ_i^1 are aggregated to obtain a score for variable i and the variable with the highest score is selected for branching¹. The following functions are the most commonly used [2, 1, 25] due to their robust performances,

1. Product rule : $\text{score}(i) = \max\{\Delta_i^0, \epsilon\} \cdot \max\{\Delta_i^1, \epsilon\}$, where $\epsilon > 0$ is small
2. Linear rule : $\text{score}(i) = (1 - \mu) \min\{\Delta_i^0, \Delta_i^1\} + \mu \max\{\Delta_i^0, \Delta_i^1\}$ for some $\mu \in [0, 1]$. In this paper, we choose $\mu = 1/6$ following the recommendation in [2].
3. Le Bodic-Nemhauser ratio rule [24]: $\text{score}(i) = 1/\phi^*$, where ϕ^* is the unique root greater than 1 of the polynomial, $p(\phi) = \phi^{\max\{\Delta_i^0, \Delta_i^1\}} - \phi^{|\Delta_i^0 - \Delta_i^1|} - 1 = 0$.

2.2 Any rule that branches exclusively on fractional variables is non-monotonic

Most standard branching rules branch including strong branching, reliability branching, most fractional branching, and maximum separating distance branching define the set of candidate branching variables as those that have a fractional value in the LP optimal solution, whereas variables that have an integer value are eliminated from consideration or assigned a zero score. For our first theoretical result, we show that any branching rule which branches on fractional variables exclusively is non-monotonic, implying that all the above-mentioned rules are also non-monotonic in general.

Theorem 1. *Any branching rule that branches only on fractional variables in the optimal LP solution is non-monotonic.*

¹If a child node is infeasible, the corresponding LP gain is infinite. If multiple branching candidates lead to infeasible children, we branch on the variable with two infeasible nodes if such exists, else, maximize the gain on the feasible branch.

Proof. Consider the polytope Q formed by the convex hull of the union of the following sets of vertices,

$$V_1 = \{x \in \mathbb{R}^4 : x_1 = 1/2, (x_2, x_3) \in \{0, 1\}^2, x_4 = 0\}$$

$$V_2 = \{x \in \mathbb{R}^4 : x_2 = 1/2, (x_1, x_3) \in \{0, 1\}^2, x_4 = 1\}$$

$$V_3 = \{x \in \mathbb{R}^4 : x_3 = 1/2, (x_1, x_2) \in \{0, 1\}^2, x_4 = 1\}$$

Polytope Q is closely related to cross-polytope considered [10]. Further consider the polytope, P created by introducing an additional vertex $v_0 = (1, 1, 1, 1/2)$, that is, $P = \text{conv}(Q \cup v_0)$ and therefore it trivially follows that $Q \subset P$. In fact, Q can be obtained from P as the lift-and-project closure of P with respect to x_4 .

Now, consider the two pure binary programs maximizing the vector $c = (1, 1, 1, 1 - \epsilon)$ over P and Q for some $\epsilon \in (0, 1)$. Both problems are infeasible since P and Q do not contain any integral points, but the formulation with Q is tighter. Consider any branching rule that considers only the variables that are fractional in the optimal LP solution. We will now show that it generates a larger tree to prove infeasibility for the formulation with Q than that with P .

First, consider the formulation with P . The LP optimal solution at the root node is the vertex v_0 , where only x_4 is fractional. So the branching rule must branch on x_4 . On the branch where $x_4 = 0$, at any vertex, only x_1 is fractional and branching on it leads to infeasible nodes on both branches. Whereas, on the branch with $x_4 = 1$, x_1 is integral on all vertices and therefore isn't considered on branching. To prove infeasibility, any branching rule must branch on both x_2 and x_3 as illustrated in Fig. 1a leading to a tree with 11 nodes in total.

On the other hand, in the case of the formulation with Q , x_4 is integral on all vertices of the polytope and is never considered for branching. The branch-and-bound tree is, therefore, the same as that for the cross-polytope which needs to branch on all of the 3 variables, generating a complete tree in x_1, x_2, x_3 [13, 16] shown in Fig. 1b. This tree has 15 nodes, and the rule is therefore non-monotonic. □

2.3 Exponential increase in size of branch-and-bound tree

We next focus on full strong branching and show that not only it is non-monotonic but in the worst case it can generate exponentially larger trees when provided with a tighter formulation, for any of the scores discussed in Section 2.1.

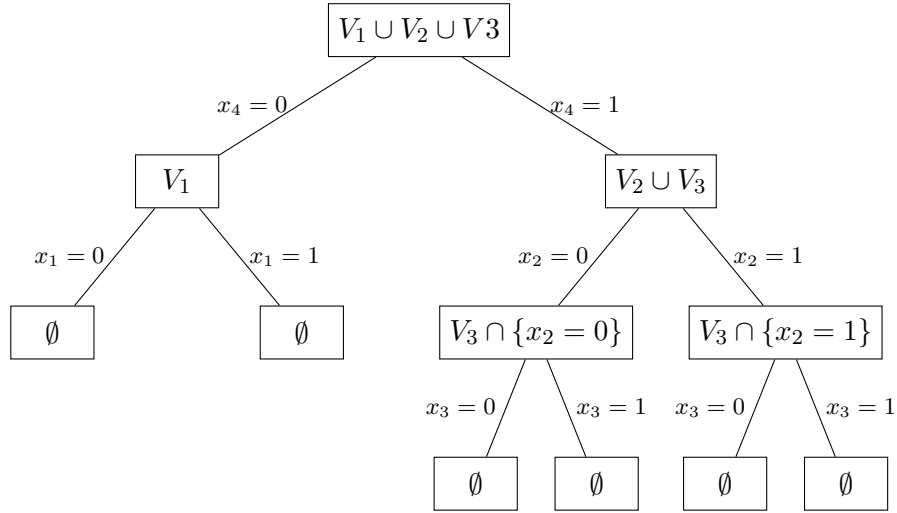
Theorem 2. *Adding a single cut can increase the size of the branch-and-bound tree exponentially when the tree is based on full strong branching with any one of the following scores: product score, linear score and ratio score.*

Proof. We prove this result for full strong branching with product score (FSB-P). The proof is identical for all the other scores. Consider the following polytopes,

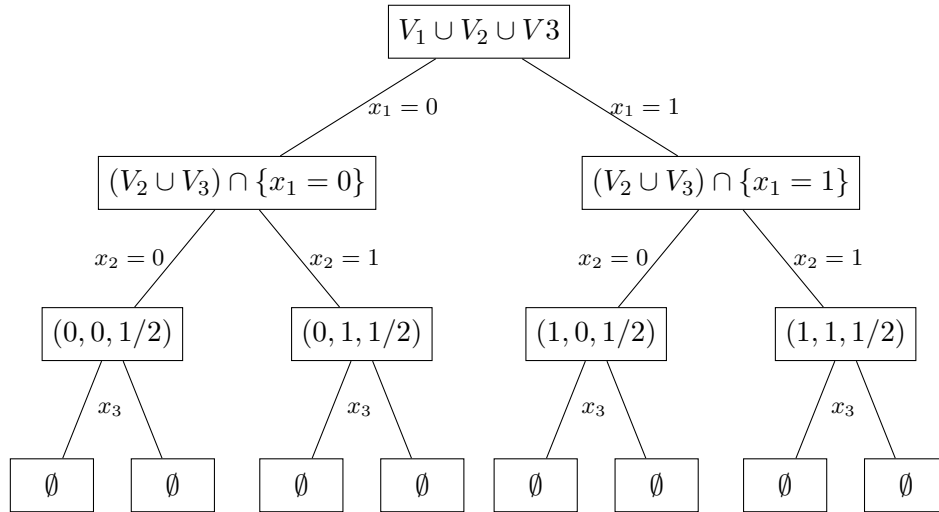
$$P = \{(x, y) \in [0, 1]^2 \mid -7x + y \leq 0.3, 5x + 8y \leq 8.5, 3x + 2y \leq 3.7\}, \quad (1)$$

$$P' = \{(x, y) \in P \mid 13x + 10y \leq 14\}, \quad (2)$$

where $P \cap \mathbb{Z}^2 = P' \cap \mathbb{Z}^2$. The polytope P is illustrated in Fig. 2 as $ABCFED$ and the polytope P' as $A'B'CFED$. Observe that $P' \subset P$, is obtained by adding a single inequality to P that is valid



(a) Tree given formulation P for any rule considering only fractional variables



(b) Tree given formulation Q for any rule considering only fractional variables

Figure 1: Illustration of branch-and-bound trees for the example in Theorem 1.

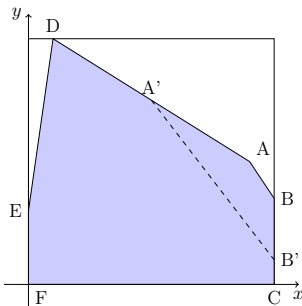


Figure 2: Polytopes P and P'

Vertex	x	y	Obj val
A	0.9	0.5	7.9
B	1	0.35	7.75
A'	0.5	0.75	6.75
B'	1	0.1	6.5
C	1	0	6
D	0.1	1	5.6
E	0	0.3	1.5
F	0	0	0

Table 1: Vertices of the polytopes

for all integer points in P . Now, consider the 2-dimensional MIP with formulations,

$$\max_{x,y} \{6x + 5y \mid (x,y) \in P \cap \mathbb{Z}^2\} \quad (3)$$

$$\max_{x,y} \{6x + 5y \mid (x,y) \in P' \cap \mathbb{Z}^2\} \quad (4)$$

and note the formulation with P' is tighter. Neither of these formulations has dual degeneracy. The objective values of all vertices of P and P' are presented in Table 1. Observe that A is the optimal solution over the LP relaxation P and C is the optimal solution of the MIP.

Next, consider the following MIP, referred to as Q_n

$$\max \quad 6 \sum_{i=1}^n x_i + 5 \sum_{i=1}^n y_i \quad (5a)$$

$$\text{s.t.} \quad (x_i, y_i) \in P \cap \mathbb{Z}^2 \quad \text{for } i = 1, \dots, n \quad (5b)$$

$$13x_i + 10y_i \leq z \quad \text{for } i = 1, \dots, n \quad (5c)$$

$$z \leq 16.7, z \in \mathbb{R}. \quad (5d)$$

Notice that since z does not appear in the objective, and is only present in constraints (5c) and (5d), these constraints imply $13x_i + 10y_i \leq 16.7$ for all $i \in [n]$ and are therefore redundant. Thus, the problem can be decomposed into n independent MIPs where each subproblem is (3). Consequently, the optimal solution to the LP relaxation is $(x_i, y_i) = (0.9, 0.5)$, i.e. vertex A for all $i \in [n]$.

Let $T(n)$ be the size of the tree for solving Q_n using FSB-P rule.

At the root node, by symmetry and the independence of subproblems, the LP gains and consequently the FSB-P scores of all variables are identical to their counterparts in MIP (3). Consider tentatively branching on x at the root node of the two-dimensional MIP using FSB-P. The optimal LP solution for the face $x = 0$ is E , and that of face $x = 1$ is B . Similarly, when tentatively branching on y , the optimal solution of the LP relaxation on face $y = 0$ is C , and that on face $y = 1$ is D . The scores are then calculated as,

$$\text{score}(x) = \Delta_x^0 \cdot \Delta_x^1 = (z_A - z_E) \cdot (z_A - z_B) = 0.96.$$

$$\text{score}(y) = \Delta_y^0 \cdot \Delta_y^1 = (z_A - z_C) \cdot (z_A - z_D) = 4.37.$$

Therefore, at the root node of Q_n , $\text{score}(x_i) = 0.96$ and $\text{score}(y_i) = 4.37$ for all $i \in [n]$. Ties may be broken arbitrarily and without loss of generality, FSB-P rule branches on y_n .

Let's first consider the node $y_n = 0$. Here, $x_n = 1$ in the LP optimal solution and therefore x_n is not considered for branching any further. This holds true for any descendent node. Thus the

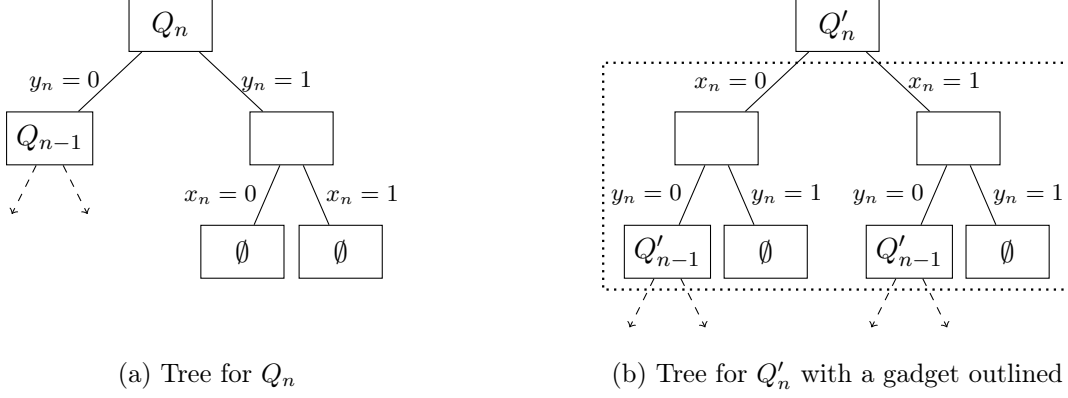


Figure 3: Illustration of strong branching trees for the example in Theorem 2.

problem at this node is equivalent to Q_{n-1} and the size of the subtree rooted at this node is at most $T(n-1)$.

Now consider the node where $y_n = 1$. If this node is not pruned by bound, then since x_n is fractional in the LP solution at this node and branching on it generates two infeasible nodes leading to an infinite score for x_n . On the other hand, the scores of x_i, y_i for all $i \neq n$ remain finite. Thus, FSB-P chooses to branch on x_n to create two infeasible nodes as depicted in Fig. 3a. Thus, we have,

$$T(n) \leq 4 + T(n-1) \quad (6)$$

Applying the inequality recursively, and using $T(0) = 1$, we have $|T^{\text{FSB}}(Q_n)| = T(n) \leq 4n + 1$.

Next, consider the case where we add a single valid cut $z \leq 14$. The constraints (5c) and (5d), are now equivalent to a $13x_i + 10y_i \leq 14$ for all i , which is exactly the cut that generates P' from P . We refer to this problem as Q'_n and observe that it can be decomposed into n independent MIPs where each subproblem is (4). It is therefore a tighter formulation of Q_n . The optimal solution to the LP relaxation of Q'_n is $(x_i, y_i) = (0.5, 0.75)$, i.e. vertex A' for all $i \in [n]$, with optimal objective value $z_{LP}^* = 6.75n$. The optimal objective value of the MIP is $z_{IP}^* = 6n$ and therefore the additive integrality gap is $G = 0.75n$.

Following the same arguments as used in the analysis of Q_n , the FSB-P scores of all variables can be computed by considering the two-dimensional subproblems individually. At the root node of (4), the scores for x and y can be similarly computed as follows,

$$\begin{aligned} \text{score}(x) &= (z_{A'} - z_E) \cdot (z_{A'} - z_{B'}) = 1.3125 \\ \text{score}(y) &= (z_{A'} - z_C) \cdot (z_{A'} - z_D) = 0.8625 \end{aligned}$$

Therefore, due to the symmetry and independence of subproblems, $\text{score}(x_i) = 1.3125$ and $\text{score}(y_i) = 0.8625$ for all $i \in [n]$ at the root node of Q'_n . Thus, FSB-P rule branches on x_n , breaking ties arbitrarily without loss of generality. At both of the nodes thus created by fixing x_n , further fixing $y_n = 1$ leads to infeasibility, giving y_n an infinite score whereas the scores of variables in other subproblems remain unchanged. Thus, both of these nodes branch on y_n if not pruned by bound. More generally, any node created by branching on any x_i , if not pruned, must branch on y_i .

Once both x_n and y_n are fixed, the restricted MIP is equivalent to solving Q'_{n-1} , allowing the branching pattern to repeat. We call this repeating block of nodes a gadget and it is indicated in Fig. 3b. In general, a gadget corresponding to subproblem i branches on x_i and y_i in sequence and includes 2 internal nodes in addition to the following 4 nodes in the lowest level: (i) $(x_i, y_i) = (0, 0)$ with LP gain $z_{A'} - z_F = 6.75$, (ii) $(x_i, y_i) = (0, 1)$ which is infeasible, (iii) $(1, 0)$ with LP gain $z_{A'} - z_C = 0.75$, and (iv) $(x_i, y_i) = (1, 1)$ which is infeasible.

As the tree is composed of repetitions of possibly partial gadgets, we will estimate the size of the tree by counting the number of complete gadgets. It is straightforward to see that a gadget is incomplete only due to pruning by bound. A node may be pruned by bound only if the gap closed by the tree at the node exceeds the integrality gap of G . Consider any feasible node at depth $2d$, where the root node is at depth 0 by convention. The maximum LP gain by a gadget at any feasible node is $g = 6.75$. Therefore, the gap closed by the tree at any feasible node at depth $2d$ is at most gd . This implies that at any feasible node till depth $2 \lfloor G/g \rfloor$, the gap closed is no more than G and there is no pruning by bound till this depth.

Using the fact that the number of gadgets doubles with an increase in depth by 2, the number of complete gadgets in the FSB-P tree is at least $2^{\lfloor G/g \rfloor} - 1$. Thus, the number of nodes in the tree is at least,

$$|T^{\text{FSB}}(Q'_n)| \geq 6 (2^{\lfloor G/g \rfloor} - 1) = 6 (2^{\lfloor \frac{0.75n}{6.75} \rfloor} - 1).$$

□

Remark 1. *The above result holds even if we do not assume the best-bound node selection rule, since in estimating an upper bound on $|T^{\text{FSB}}(Q_n)|$ we did not consider pruning by bound.*

3 Computational Experiments

In this section, our goal is to conduct computational experiments to ascertain how common is it for the strong branching rule to exhibit non-monotonicity. We focus our attention on the product rule for FSB (denoted as FSB-P) since it has been shown to be computationally superior [1] to many other rules.

3.1 Cover Cuts for Multi-dimensional Knapsack Problem (MKP)

Instances. Consider an instance \mathcal{I} of the multi-dimensional knapsack problem (MKP), which can be formulated as follows,

$$\max \left\{ \sum_{i=1}^n c_i x_i \mid \sum_{i=1}^n a_i^j x_i \leq b^j, \forall j \in [m], x \in \{0, 1\}^n \right\}. \quad (7)$$

We randomly generated 100 instances with 20 variables and 50 constraints. Weights a_i^j are independent and set to 0 with probability 0.25, else uniformly picked from the set $\{1, \dots, 1000\}$. Capacity b^j is set to $\lfloor 0.5 \cdot \sum_{i=1}^n a_i^j \rfloor$. Prices c_i are independently picked from the uniform distribution on the interval $[0, 1)$. By randomly sampling objective coefficients from a continuous distribution, we ensure that almost surely the instances do not exhibit dual degeneracy.

Experiments. For each knapsack constraint in the instance, we try to find cover cuts separating the root LP solution. We solve the following separation problem,

$$\begin{aligned} d_j^* = \max \quad & \sum_{i=1}^n x_i^* w_i - \sum_{i=1}^n w_i + 1 \\ \text{s.t.} \quad & \sum_{i=1}^n a_i^j w_i \geq b^j + 1 \\ & w_i \in \{0, 1\} \quad \text{for all } i = 1, \dots, n \end{aligned} \quad (8)$$

If the above cut-generating IP is infeasible, there are no cover cuts for the knapsack constraint j and the constraint is in fact redundant. If the cut-generating IP is feasible, let \hat{w}^j be the optimal solution and set C^j be the support of \hat{w}^j . Then a valid cover cut is given by,

$$\sum_{i \in C^j} x_i \leq |C^j| - 1 \quad (9)$$

Moreover, the cut separates x^* from the integer hull of the MKP if and only if $d_j^* > 0$. Let $\mathcal{C}_{\mathcal{I}}$ be the set of all violated cover inequalities discovered. For the first set of experiments, we add each cover cut individually. That is for every $C \in \mathcal{C}_{\mathcal{I}}$, we solve,

$$\max \left\{ \sum_{i=1}^n c_i x_i \mid \sum_{i=1}^n a_i^j x_i \leq b^j \forall j \in [m], \sum_{i \in C} x_i \leq |C| - 1, x \in \{0, 1\}^n \right\},$$

using branch-and-bound with FSB-P rule. For the next experiment, we add all separating cover cuts found for the instance. That is for every \mathcal{I} , we solve,

$$\max \left\{ \sum_{i=1}^n c_i x_i \mid \sum_{i=1}^n a_i^j x_i \leq b^j \forall j \in [m], \sum_{i \in C} x_i \leq |C| - 1 \forall C \in \mathcal{C}_{\mathcal{I}}, x \in \{0, 1\}^n \right\}.$$

Let the LP relaxation of the original formulation have objective value z and let the size of strong branching tree for this formulation be T . Define \hat{z} and \hat{T} similarly for a formulation strengthened by cuts. Moreover, let z_{IP} be the value of the integer optimal solution. We compute the gap closed by the cuts ΔG and the change in tree size ΔT as follows,

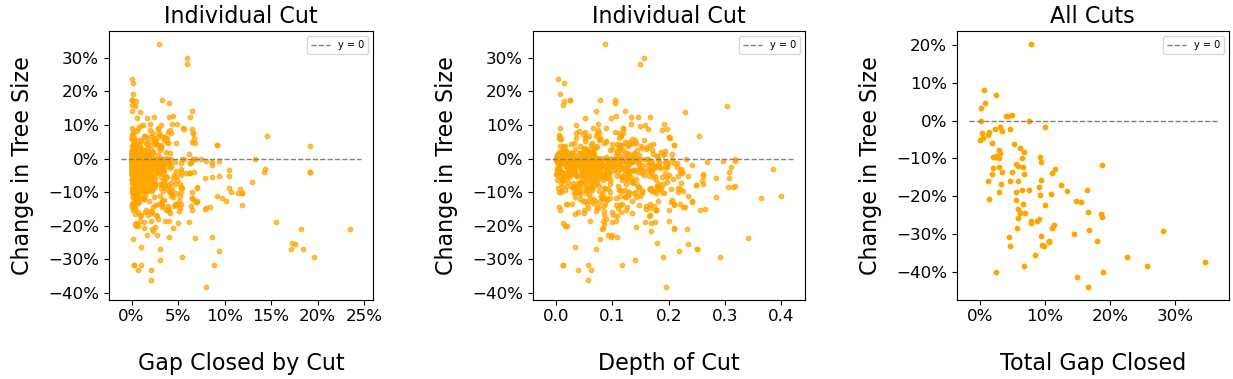
$$\Delta G = \frac{z - \hat{z}}{z - z_{IP}}, \quad \Delta T = \frac{\hat{T} - T}{T}. \quad (10)$$

Lastly, the depth of cut, Δd is the Euclidean distance between the LP solution and the separating hyperplane.

The experiments were conducted on a Python 3.10-based implementation of the naive branch-and-bound framework to ensure a controlled experiment, unaffected by other sophisticated processes of modern solvers. Gurobi 10.0.1 was used as the LP solver. FSB-P was implemented purely as an oracle returning a branching variable, without influencing any other process within branch-and-bound, thus allowing fair node counting [18]. Nodes were processed in the best-bound first order. Finally, neither presolve, nor additional cutting planes besides the cover cuts mentioned above were employed.

Results. The results of the experiments are plotted in Fig. 4. For the first experiment, Fig. 4a and 4b show the change in tree size due to adding a single cut against the percentage gap closed and depth of cut respectively, for all separating cover cuts across all instances. Note that the points above the $y = 0$ line denote an increase in the tree size while points below the line indicate a decrease in tree size. Contrary to expectation, 19.8% of the cuts led to an increase in tree size. Moreover, the correlation of the change in tree size with the gap closed by the cut as well as the depth of cut was weak.

For the second experiment, all separating cuts found for the instance were simultaneously added. Fig. 4c presents the change in tree size against the gap closed by the cuts, where every point now represents an instance. Even when several cuts were added simultaneously, tree sizes for 8 out of 100 instances increased. However, observe that the total gap closed by cuts in all of these instances was at most 10%.



(a) ΔT versus ΔG when an individual cut is added. (b) ΔT versus Δd when an individual cut is added. (c) ΔT versus ΔG when all cuts are added.

Figure 4: Impact of adding cover cuts on the size of FSB-P trees for MKP instances.

Fig. 4c suggests that if the gap closed is small (also happens when just one cut is added), it is difficult to predict the direction of change in tree size. However, after enough gap is closed, there is a clear decreasing trend, and we may expect the size of the FSB-P tree to have reduced. With this hypothesis in mind, we conduct the next set of experiments, where we study the change in tree size as more rounds of cuts are added.

3.2 Experiments on MIPLIB

The experiments in Section 3.1 allowed us to completely control all sources of variability in tree size, ensuring that any increase in tree size is an example of non-monotonicity. However, a shortcoming of these experiments is that these instances do not represent the diversity of problems encountered in practice. With this in mind, we next consider instances from the Benchmark Set of MIPLIB 2017 [20] and cuts applied by solvers in practice.

The goal of these experiments is to understand the effects of cutting planes on FSB-P tree sizes, and how that may vary as more gap is closed. In particular, the number of rounds of cuts applied at the root node is varied. After each additional round of cuts, the FSB-P tree size is computed. The underlying hypothesis is that if enough gap is closed, the size of trees must decrease as the increased effectiveness of pruning by bounds outweighs possibly worse branching decisions. While these experiments may be more relevant practically, we can no longer control dual degeneracy or variation in tree size due to it. We therefore run every instance with 3 different seeds.

The experiments in this section are conducted using SCIP 8.0 [9] as the MIP solver and PySCIPOpt [27] as the API. All computations were done on a Linux based cluster. Whenever SCIP is called, presolve and propagation are disabled, cuts are allowed only at the root node, and branching variable is selected by the *vanilla full strong branching* rule while disabling strong branching side-effects. For every instance, the ordering of variables remains the same for a fixed seed. Moreover, independence of the branch-and-bound trees from node selection rules is ensured by providing the optimal MIP value to the solver. In the absence of dual degeneracy, configuring SCIP to these settings eliminates all sources of performance variability in the number of nodes to the best of our understanding, and enables us to isolate the impact of any change in branching decisions.

Instances. Due to the expensive nature of full strong branching, we first restrict the set to “easy” instances with at most 10,000 variables and 500 integer or binary variables. We also exclude infeasible instances. This set of candidates consists of 35 instances. These instances are then solved while restricting cuts to one round of separation at the root node. Those instances not solved in 72 hours are discarded. In addition, 1 instance was discarded since it was solved at the root node. This resulted in a final set of 21 instances that were used in the experiments discussed in the rest of this section.

Experiments. In these experiments, we allow SCIP to apply its default cuts at the root node, but limit the number of rounds of cuts it applies. Every instance and seed combination is solved 11 times, while the upper limit on the rounds of cuts is increased from 0 to 10. Tree size as well as the dual bounds at the root node are saved for comparison. We then compute the gap closed by cuts and change in tree size with respect to the formulation without cuts (run with the same seed), using Eq.(10).

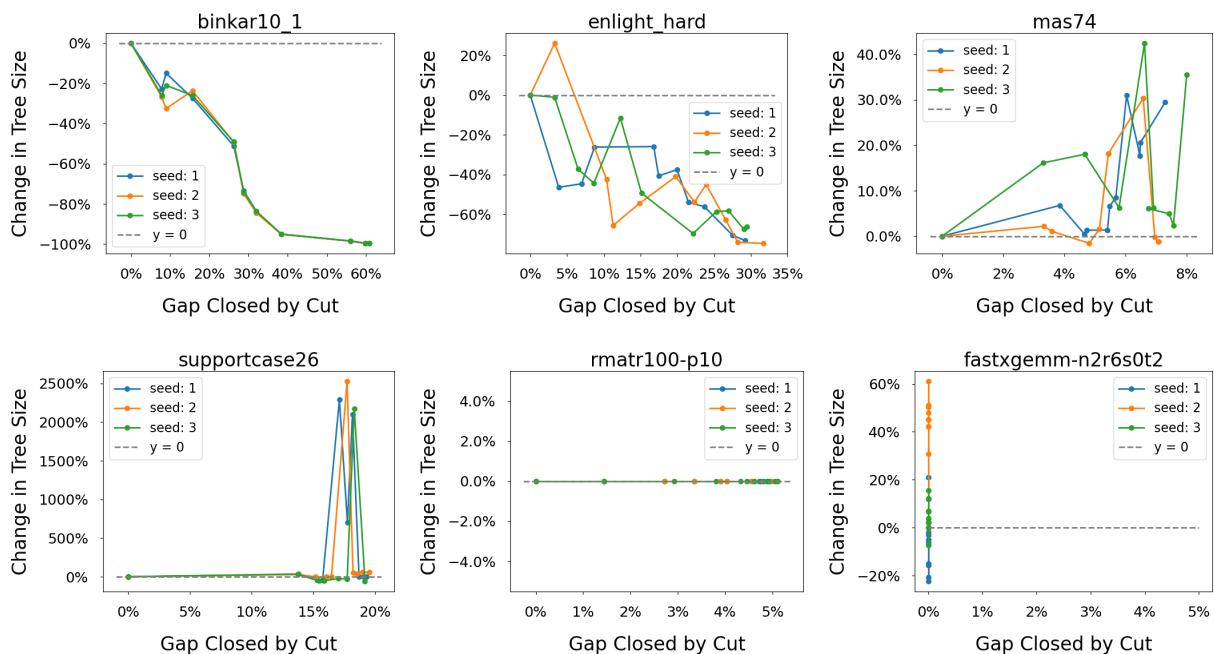


Figure 5: Progression of change in tree size and gap closed as the number of rounds of cuts is increased from 0 to 10.

Results. For the first set of results, Fig.5 presents the progression of tree size and the gap closed as more rounds of cuts are added for a subset of instances. These instances have been selected to cover different patterns observed and the results for all others are included in Appendix A.

The first striking observation is that non-monotonicity was encountered in most of the instances and appears to be surprisingly common in practice. Instances in the set showed varying sensitivity of tree size to a round of cuts and the trends in tree sizes could be categorized by the total gap closed in 10 rounds.

- When the total gap closed was large, (ΔG greater than 50%), in line with expectation, instances binkar10_1, mik-250-20-75-4 and n5-3 showed a steady decrease in tree size with increasing gap closed.

- For instances enlight_hard and seymour1, where a moderate gap was closed (ΔG between 20% and 50%) the tree size had a generally decreasing trend with some intermittent rises.
- In most instances, the total gap closed was small (ΔG at most 20%). These instances did not have a decreasing trend. Instances from mas, gen-ip and neos, istanbul-no-cutoff and ran14x18-disj-8 often had large spikes and drops with every round of cut. The largest jump in tree size was seen in the case of supportcase26 where, for all of the 3 seeds, the tree size increased by more than 20 times within a single round. On the other hand, rmatr100-p10 and glass-sc showed no change in tree size in spite of closing some gap.
- Finally, in the case of fastxgemm-n2r6s0t2, pk1, markshare_4_0 and mad, the gap closed is 0 even after 10 rounds of cuts, but the tree sizes changed significantly in both directions. It must be noted that these instances clearly have high dual degeneracy which also contributes to variability in size.

An inference that can be drawn is that if the gap closed is small, change in tree size is difficult to predict, and often increases, possibly due to non-monotonicity. However, when a large enough gap is closed, a significant decrease in tree size may be expected. This is seen clearly in Fig. 6 where there are no data points with an increase in tree size when the gap closed exceeds 20%. Note that a very similar pattern is also seen in Fig. 4c for the randomly generated MKP instances.

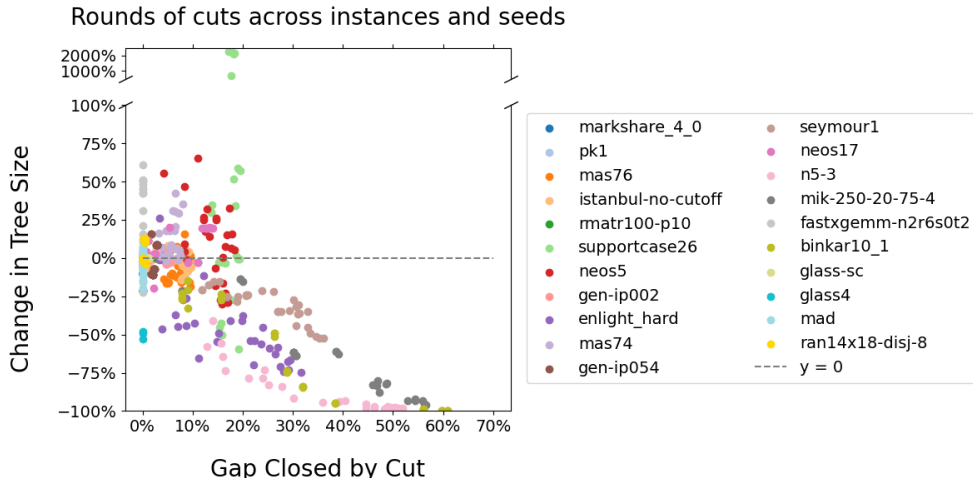


Figure 6: Change in tree size and the gap closed across all instances, seeds and limits on rounds of cuts.

4 Conclusion

The branch-and-cut method is the most efficient framework for solving general MIPs in practice. The objective of this paper is not to claim otherwise, but to highlight some of the fundamental challenges in this approach. Sometimes in practice, it is experienced that adding cuts increases the number of nodes. Through this work, we show that such aberrant behaviour may not always be because of engineering reasons like the addition of cuts leading to a change in some default parameters or other random tie-breaking, but the problem may in fact be mathematical in nature. In particular, as shown in Section 2, standard branching rules may sometimes produce larger trees when provided with a tighter relaxation and are in general not monotonic. In Section 3, we show

through computational experiments, that non-monotonicity is not merely a theoretical possibility, but is surprisingly prevalent in practice. When considering default cuts applied by SCIP on instances from the MIPLIB, non-monotonicity was encountered in most of the instances that were evaluated. We would like to emphasize here that these experiments are by no means an evaluation of SCIP, and the results do not reflect on its performance but only indicate inherent non-monotonicity in the branch-and-cut approach.

This work gives rise to many open questions in the area of the design of branching rules. Consider a branching rule that fixes the branching decisions a priori without using local information from the linear relaxation. It is then clear that this rule is monotonic, even though it may generate large trees. On the other extreme, optimal branching [16] which considers global information is monotonic and produces small trees, but is $\#$ P-hard to compute [19] and is therefore impractical. It is not clear whether a good, efficient (polynomial-time complexity at each node) and monotonic rule even exists.

The results of this paper may also be viewed from the perspective of cutting-plane selection [3, 17]. Firstly, the depth of individual cuts is not a good measure of how much the tree size improves as seen in Fig. 4b. Also as we have seen, the empirical evidence from the computational experiments in Section 3 indicates that if the gap closed by the group of cuts is small, we cannot be assured that adding them will result in smaller trees since their impact on branching decisions is hard to model due to possible non-monotonicity. However, when a substantial gap is closed, the size of the branch-and-bound tree decreases. In particular, empirical evidence suggests that crossing the 20% gap-closed threshold via cuts could be a good rule of thumb for deciding whether the FSB-P tree size will decrease.

Acknowledgements. We would like to thank Avinash Bhardwaj for his feedback on preliminary computational results. We would also like to thank the support from AFOSR grant # F9550-22-1-0052.

References

- [1] Tobias Achterberg. *Constraint integer programming*. PhD thesis, 2007.
- [2] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.
- [3] Giuseppe Andreello, Alberto Caprara, and Matteo Fischetti. Embedding $\{0, 1/2\}$ -cuts in a branch-and-cut framework: A computational study. *INFORMS Journal on Computing*, 19(2):229–238, 2007.
- [4] David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. *Finding cuts in the TSP (A preliminary report)*, volume 95. Citeseer, 1995.
- [5] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42(9):1229–1246, 1996.
- [6] Egon Balas, Sebastian Ceria, Gérard Cornuéjols, and N Natraj. Gomory cuts revisited. *Operations Research Letters*, 19(1):1–9, 1996.
- [7] Amitabh Basu, Michele Conforti, Marco Di Summa, and Hongyi Jiang. Complexity of branch-and-bound and cutting planes in mixed-integer optimization-ii. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 383–398. Springer, 2021.

- [8] Amitabh Basu, Michele Conforti, Marco Di Summa, and Hongyi Jiang. Complexity of branch-and-bound and cutting planes in mixed-integer optimization. *Mathematical Programming*, 198(1):787–810, 2023.
- [9] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. Technical report, Optimization Online, December 2021.
- [10] Merve Bodur, Sanjeeb Dash, and Oktay Günlük. Cutting planes from extended lp formulations. *Mathematical Programming*, 161:159–192, 2017.
- [11] William Cook. Fifty-plus years of combinatorial integer programming. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 387–430, 2010.
- [12] Harlan Crowder, Ellis L Johnson, and Manfred Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983.
- [13] Daniel Dadush and Samarth Tiwari. On the complexity of branching proofs. In *Proceedings of the 35th Computational Complexity Conference*, pages 1–35, 2020.
- [14] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.
- [15] Santanu S Dey, Yatharth Dubey, and Marco Molinaro. Branch-and-bound solves random binary ips in polytime. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 579–591. SIAM, 2021.
- [16] Santanu S Dey, Yatharth Dubey, Marco Molinaro, and Prachi Shah. A theoretical and computational analysis of full strong-branching. *Mathematical Programming*, pages 1–34, 2023.
- [17] Santanu S Dey and Marco Molinaro. Theoretical challenges towards cutting-plane selection. *Mathematical Programming*, 170:237–266, 2018.
- [18] Gerald Gamrath and Christoph Schubert. Measuring the impact of branching rules for mixed-integer programming. In *Operations Research Proceedings 2017: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Freie Universität Berlin, Germany, September 6-8, 2017*, pages 165–170. Springer, 2018.
- [19] Max Gläser and Marc E Pfetsch. On computing small variable disjunction branch-and-bound trees. *Mathematical Programming*, pages 1–29, 2023.
- [20] Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp M. Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, Marco Lübbecke, Hans D. Mittelmann, Derya Ozyurt, Ted K. Ralphs, Domenico Salvagnin, and Yuji Shinano. MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. *Mathematical Programming Computation*, 2021.

- [21] Ralph E Gomory. Outline of an algorithm for integer solutions to linear programs. *Bull. Amer. Math. Soc.*, 1958.
- [22] Aleksandr M Kazachkov, Pierre Le Bodic, and Sriram Sankaranarayanan. An abstract model for branch and cut. *Mathematical Programming*, pages 1–28, 2023.
- [23] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.
- [24] Pierre Le Bodic and George Nemhauser. An abstract model for branching and its application to mixed integer programming. *Mathematical Programming*, 166(1-2):369–405, 2017.
- [25] Jeff T Linderoth and Martin WP Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, 1999.
- [26] Andrea Lodi. Mixed integer programming computation. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 619–645, 2010.
- [27] Stephen Maher, Matthias Miltenberger, João Pedro Pedroso, Daniel Rehfeldt, Robert Schwarz, and Felipe Serrano. PySCIPOpt: Mathematical programming in python with the SCIP optimization suite. In *Mathematical Software – ICMS 2016*, pages 301–307. Springer International Publishing, 2016.
- [28] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.
- [29] Tony J Van Roy and Laurence A Wolsey. Solving mixed integer programming problems using automatic reformulation. *Operations Research*, 35(1):45–57, 1987.

A MIPLIB Results

The plots for all instances considered in Section. 3.2 are presented in Fig. 7.

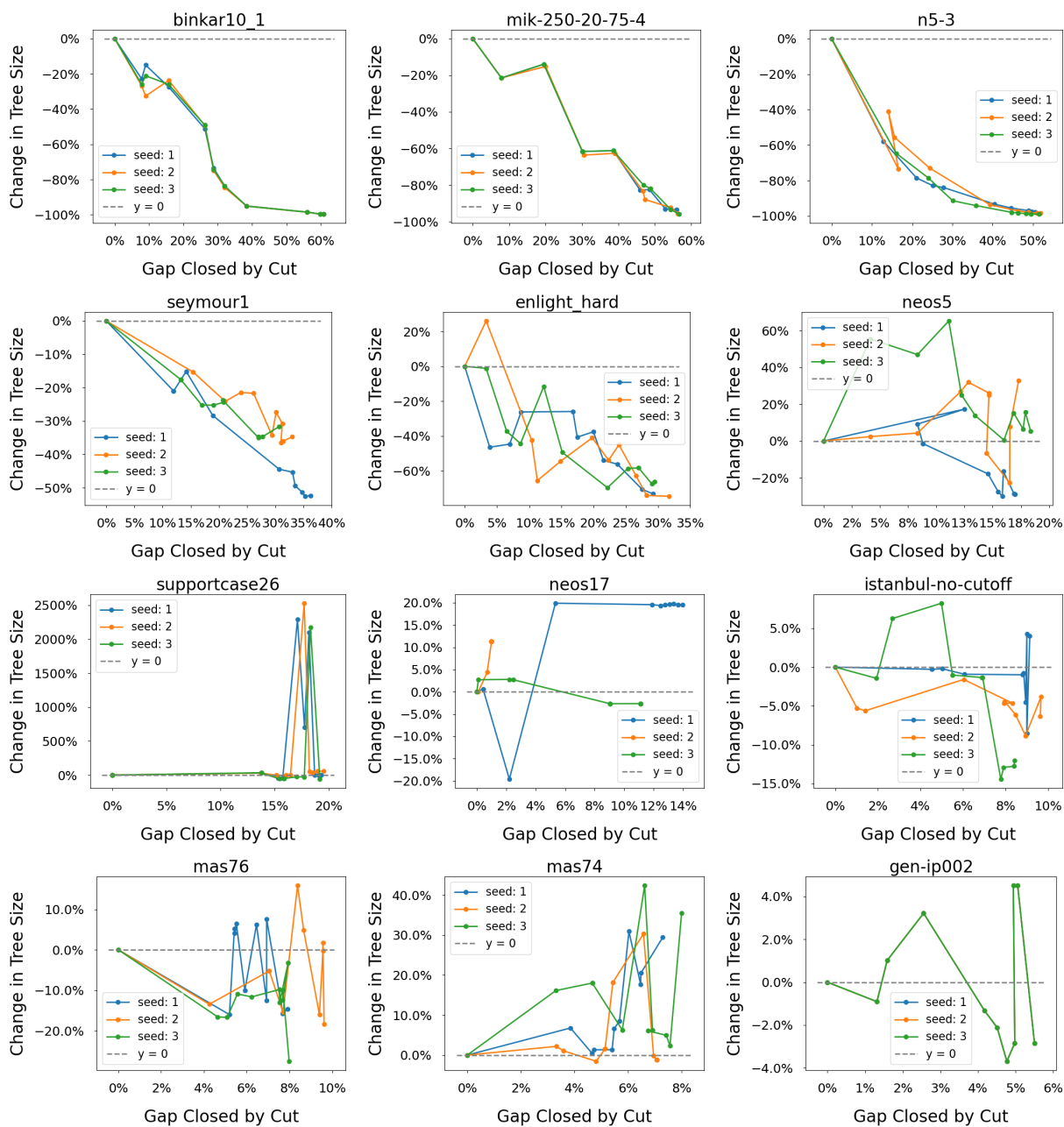


Figure 7: Progression of change in tree size and gap closed as number of rounds of cuts are increased from 0 to 10. (cont.)

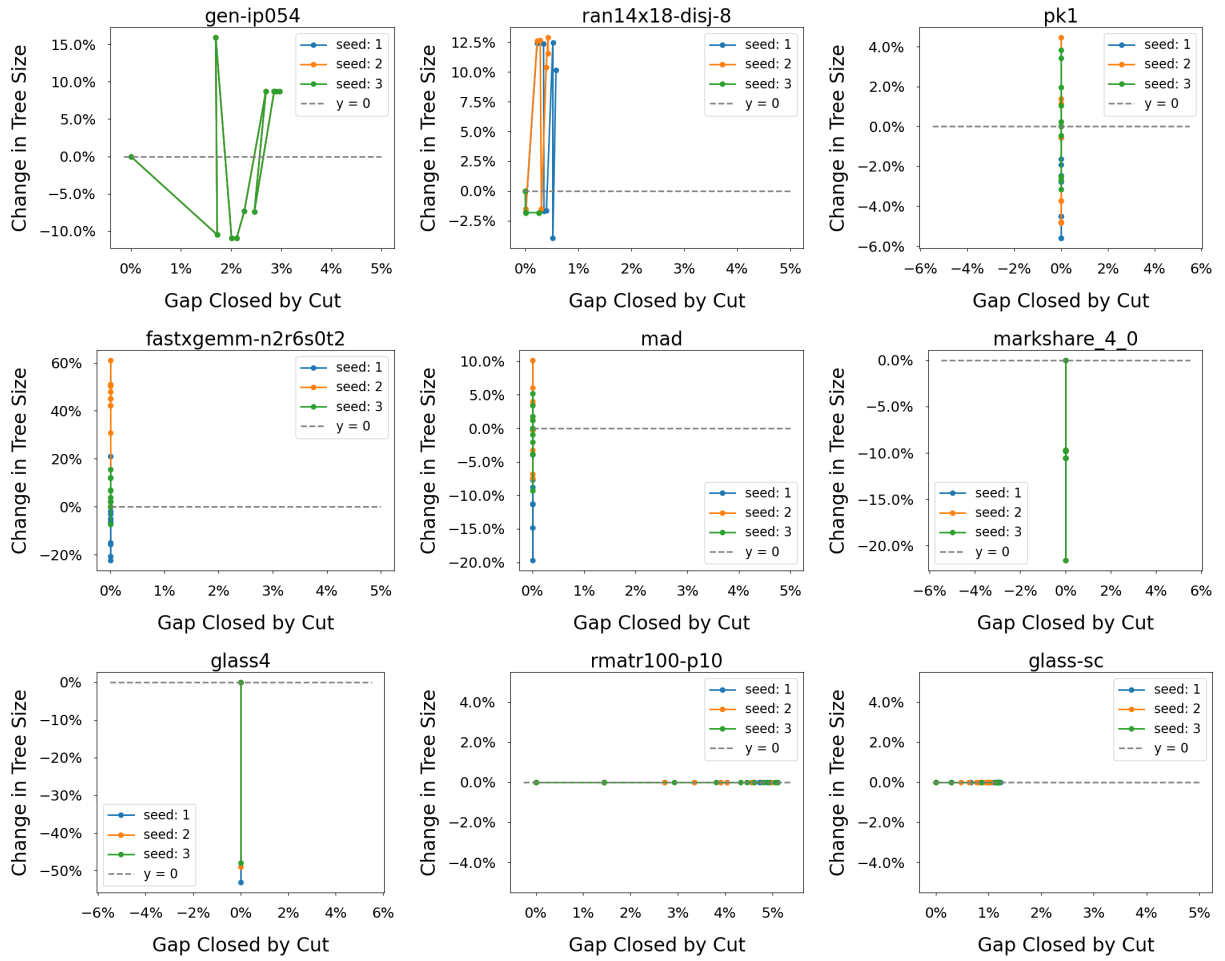


Figure 7: Progression of change in tree size and gap closed as number of rounds of cuts are increased from 0 to 10.