

The Strength of Root Cuts in an Extended Abstract Branch-and-Cut Model*

Boyang Han

Aleksandr M. Kazachkov

Abstract

The branch-and-cut paradigm has led to effective solvers for a wide range of mathematical optimization problems. Despite significant theoretical work on pure branching and pure cutting, less is known about their interaction. We study an extended abstract branch-and-cut model to investigate tradeoffs between branching and cutting, including the time to generate cuts. We prove that, under symmetric branching, some minimal-time tree has all cuts at the root, before the first branching node, and that there is an asymmetric setting where some minimal-time tree has no internal cuts, i.e., cuts both preceded and followed by branching. For other settings, although a root-cut tree may be suboptimal and differ substantially in structure from a minimal-time tree, it nevertheless provides a reliable approximation whose tree time does not deviate arbitrarily from the optimum. For constant cut strength and bounded time functions, we show that the best root-cut tree is asymptotically optimal as the target bound grows. In addition, under a single-cut-per-path restriction, we characterize when some minimal-time tree has an internal cut. Lastly, simulations indicate that the best root-cut tree is much closer to a minimal-time tree than our worst-case approximation guarantee suggests.

1 Introduction

Branch-and-cut (B&C), a hybrid version of the branch-and-bound (B&B) [17] and cutting plane (CP) methods [10–12], is a powerful tree-based framework that is a key ingredient of mathematical optimization solvers. Each node of the tree corresponds to a relaxation of the original optimization problem that is refined by either valid inequalities (*cuts*) or splitting the search space (*branching*). Solvers determine *where* in the tree to add rounds of cuts from empirical guidelines and typically focus on the *root node* of the tree. In this paper, we provide complementary theory on cut placement with provable guarantees on the computational cost of the resulting tree, within an extension of the *abstract B&C model* by Kazachkov, Le Bodic, and Sankaranarayanan [15, 16].

Our model assumes that key solution parameters are known: the bound that needs to be proved (e.g., an optimal solution is given, but we need to certify optimality via the tree), the evolution of the bound over iterated branching and cutting operations, and an estimated time to solve each node of the tree. Our perspective (when such information may be available) is a *post-optimality analysis* of the quality of the best-possible B&C tree for a problem, which can provide beneficial insights when solving the same problem repeatedly.

In the original abstract B&C model [16], branching yields fixed (constant) bound improvements, while cuts may have diminishing strength across rounds or increase node processing time. Under symmetric branching and constant cut strength, they prove that a *root-cut tree*, where all cut nodes are before the first branch node, is sufficient: there always exists a minimal-time root-cut tree, for any function defining node solving time [16, Theorem 23].

*Extension of paper from The 27th Conference on Integer Programming and Combinatorial Optimization [13].

We generalize this model to accommodate broader branching and cutting scenarios and show that a root-cut tree solving in minimal (or nearly minimal) time continues to exist. Even when the root-cut tree is suboptimal, i.e., when any minimal-time tree requires *local* cuts (deeper in the tree), we derive an upper bound on how much its processing time can deviate from the minimal-time tree, potentially offering guidance (or validation) for the root-cut dominant approach of solvers. Our contributions are summarized next.

- ▷ [Section 3.1](#): We prove that, under symmetric branching, a minimal-time root-cut tree always exists, even when simultaneously considering multiple cut and branching families, or node-dependent cut generation times.
- ▷ [Section 3.2](#): We prescribe how to best add a sequence of weak and strong cuts, when the weak cuts cause a relatively smaller increase in node solving time.
- ▷ [Section 3.3](#): We identify an asymmetric branching setting where some minimal-time tree has no internal cuts, i.e., all cuts are at the root or at the leaves.
- ▷ [Section 4.1](#): We bound the approximation error of the best root-cut tree with respect to the minimal-time tree.
- ▷ [Section 4.2](#): We analyze bounded time functions, for which the best root-cut tree is within a constant factor of a minimal-time tree, and for constant cuts, it is asymptotically optimal.
- ▷ [Section 5.1](#): We show by example that the minimal-time tree can be structurally complex for asymmetric settings: a local cut may appear at any depth.
- ▷ [Section 5.2](#): We characterize conditions that allow the construction of minimal-time trees featuring local cuts under a single-cut-per-path assumption.
- ▷ [Section 5.3](#): We study how cut generation time impacts number of cut nodes in a minimal-time tree, proving monotonicity for constant separation times.
- ▷ [Section 5.4](#): We empirically assess the quality of the root-cut approximation.

Related literature. Recent theory has been developed to analyze the strength of the pure B&B process [8, 5, 9, 18], pure CP [6], and comparing B&C, B&B, and CP [1, 2]. Basu, Conforti, Di Summa, and Jiang [1] show that B&C can be strictly superior to B&B and CP. In contrast, and counterintuitively, Shah, Dey, and Molinaro [21] prove that a single cut can make B&B exponentially worse. Related to our work, Jiang [14] identifies—within our model—the fraction of the target bound proved by cut nodes in a minimal-size B&C tree, as a function of how cumulative cut strength grows. Other papers adopt a learning-based perspective: Berthold, Francobaldi, and Hendel [3] propose methods for learning to use local cuts effectively within B&C; Cheng and Basu [4] establish generalization guarantees for robustness of learned B&C policies across problem instances. However, in all this research, the explicit tradeoffs and interplay between branching and cutting decisions remain poorly understood. Our high-level motivation is to systematically study the interaction between branching and cutting within a B&C framework in a stylized (but already nontrivial to analyze) model, through which we can progress towards theoretical guidance for cut placement and selection in solvers.

2 Preliminaries: Notation and Definitions

Abstractly, B&C involves solving a sequence of tightened subproblems in a tree structure. For example, in mixed-integer optimization, the B&C process starts from the *root node*, a convex relaxation of the original problem. The optimal value at the root node is a dual bound for the original problem. Then, the problem formulation (and dual bound) is strengthened by either: (1) partitioning the feasible space and creating two subproblems, or (2) adding a round of valid cuts that are violated by points in the current relaxation but satisfied by all feasible mixed-integer solutions. The B&C tree

is built by recursively selecting between these two operations until the tree *proves a (dual) bound* matching the primal optimal value (we assume this is given as Z).

We largely adopt the notions modeling a B&C tree and its processing time from Kazachkov, Le Bodic, and Sankaranarayanan [16, Definitions 1–5]. The authors define a *time function* [16, Definition 2] to model that cuts make formulations more complex, thus increasing the processing time of subsequent subproblems. In practice, solvers can choose among *different families of cuts* that have heterogeneous properties, and *cut generation time* also affects cut selection. We introduce two extended models of solving time: a *simple time function* (Definition 1) in which solving time of a node in the tree only depends on the number of cuts previously generated from each family, and a *generalized time function* (Definition 3) that also accounts for the time to compute cuts. We will assume that no cut family strictly dominates another (i.e., when a cut family yields a better bound improvement, it causes slower relaxations).

Let $\mathcal{V}_T$ denote the nodes of a B&C tree T . In our model, each node in the tree has a type $i \in \mathcal{I} := \{0, 1, \dots, m\}$, where type 0 represents a branch node, and type $i \in [m] := \{1, \dots, m\}$ represents a cut node from family i . We make the simplifying assumption that the bound improvement from cut family $i \in [m]$ only depends on the number of cuts previously added from that family. We represent the bound improvement from a cut node from family i via a (*worsening*) *cut function* $c_i \in \mathcal{F}_{\searrow}$, where $\mathcal{F}_{\searrow}$ denotes the set of nonincreasing functions $\mathbb{Z}_{>0} \rightarrow \mathbb{R}_{\geq 0}$ and $c_i(k)$ represents the bound improvement by the k th round of cuts from family i . The bound improvement from a branch node is given by constants ℓ (to the left) and r (to the right). We assume that $\ell \leq r$ (without loss of generality) and that cut and branch nodes are derived from the same information (e.g., disjunction), so $c_i(k) \leq \ell \leq r$ for every $k \in \mathbb{Z}_{>0}$ and cut family $i \in [m]$. For a node $v \in \mathcal{V}_T$, we denote by T_v the subtree of T rooted at v . Each node v also has an associated bound, denoted $z_T(v)$, defined as the total bound improvement along the path from the root of T to v , where the root of a tree has bound 0; see [16, Definition 1]. A tree proves a bound Z if the bound at all leaves is at least Z .

Let $\vec{h}(v) \in \mathbb{Z}_{\geq 0}^{\mathcal{I}}$ be the *history vector*, where $\vec{h}(v)_i$ is the number of nodes of type i applied before a node $v \in \mathcal{V}_T$. Let $e_i \in \mathbb{R}^{\mathcal{I}}$ denote the unit vector in direction i .

Definition 1 (Simple time function). Given cut functions $\{c_i\}_{i=1}^m$, a function $w : \mathbb{Z}_{\geq 0}^{\mathcal{I}} \rightarrow [1, \infty)$ is a *simple time function* if it is elementwise nondecreasing, $w(\mathbf{0}) = 1$, and if, for all history vectors $\vec{h} \in \mathbb{Z}_{\geq 0}^{\mathcal{I}}$, when the next cut from family i is stronger than one from family j , i.e., $c_i(\vec{h}_i + 1) \geq c_j(\vec{h}_j + 1)$, then $w(\vec{h} + e_i) \geq w(\vec{h} + e_j)$.

Though Definition 1 allows dependence on the number of branch nodes, in this work, we assume that if $\vec{h}_i = \vec{h}'_i$ for all $i \in \mathcal{I} \setminus \{0\}$, then $w(\vec{h}) = w(\vec{h}')$. As a result, for simplicity, we omit index 0 for a history vector input to w .

Definition 2 (Separation time function). A family of functions $\{s_i : i \in \mathcal{I}\}$ with $s_i : \mathbb{Z}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ for all $i \in \mathcal{I}$, is a *separation time function family* if $s_0 \equiv 0$ and $s_i(\vec{h}_i)$, $i \in [m]$, represents the time to generate one round of cuts from family i given $\vec{h}_i$ prior cut rounds from that same family.

Definition 3 (Generalized time function). Given a simple time function w and a separation time function family $\{s_i\}_{i=0}^m$, a function $\tilde{w} : \mathbb{Z}_{\geq 0}^{\mathcal{I}} \times \mathcal{I} \rightarrow [1, \infty)$ is a *generalized time function* if for all history vectors $\vec{h}$ and node types $i \in \mathcal{I}$, it holds that $\tilde{w}(\vec{h}, i) = w(\vec{h}) + s_i(\vec{h}_i)$. Given a B&C tree T and a node $v \in \mathcal{V}_T$ of type $i \in \mathcal{I}$ having a history vector $\vec{h}(v)$, we will also use $\tilde{w}(v) := \tilde{w}(\vec{h}(v), i)$.

Optimizing tree size, i.e., $w \equiv 1$, is studied by Le Bodic and Nemhauser [18] (for B&B) and Kazachkov, Le Bodic, and Sankaranarayanan [16] (for B&C). However, a smaller B&C tree does not guarantee a better solving time. Example 1 illustrates this phenomenon. In this work, we focus on

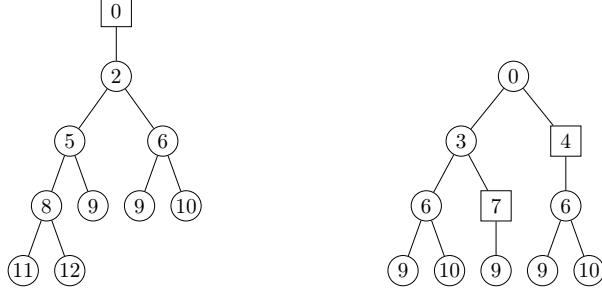


Figure 1: Two B&C trees proving $Z = 9$, with $c = 2$, $\ell = 3$, $r = 4$, $w(1) = 1.3$, and with $w(k) = M$ for $k \geq 2$ and sufficiently large $M \in \mathbb{R}_{\geq 0}$.

optimizing *tree time* $\tau_{\tilde{w}}(T) := \sum_{v \in \mathcal{V}_T} \tilde{w}(v)$ with respect to nontrivial time functions ($\tilde{w} \neq 1$). The resulting optimal structure is referred to as a *minimal-time tree*. We call a tree proving bound Z *cut-minimal* (with respect to a given time function) if it is a minimal-time tree and has the fewest cut nodes among all minimal-time trees proving Z . A cut-minimal tree always exists in our model. A *pure cutting tree* has no branch nodes and a *pure branching tree* has no cut nodes. For $z \in \mathbb{R}$, let $T(z)$ denote the (unique [18]) smallest pure branching tree proving bound z for fixed ℓ and r , in which a node is a branch node if and only if its bound is less than z .

Our focus is on the *structure* of the tree, meaning the relative placement of cut and branch nodes. To this end, we distinguish between *root cut* nodes (applied in the tree before any branch node) and *local cut* nodes (applied after the first branch node). Define a *leaf cut* node as a cut node without a branch node descendant (i.e., part of a path ending in a *leaf node*), and an *internal cut* node as a local (not root) cut that is not a leaf cut. Define a *branch-then-cut* node to be a branch node with at least one internal cut child.

Example 1. Figure 1 shows two B&C trees that prove bound $Z = 9$. In the left panel, the tree has 10 nodes and solves in tree time $1 + 9w(1) = 12.7$. The tree in the right panel has 11 nodes, but solves more quickly, with a tree time of $7 + 4w(1) = 12.2$, and is in fact the unique minimal-time tree with respect to w . In the right panel of Figure 1, the root is a branch-then-cut node because its child is an internal cut, while the node labeled 7 is a leaf cut.

3 When Root Cuts Are (Nearly) Sufficient

In this section, we identify settings, both symmetric (Section 3.1) and asymmetric (Section 3.3), in which there is a minimal-time tree without internal cuts, i.e., in which all cuts are at the root or at the leaves. We also show (Section 3.2) how to best order a sequence of weak and strong cuts, when weak cuts cause a comparatively smaller increase in node solving time.

3.1 Symmetric Branching

While the assumption of symmetric branching (when $\ell = r$) is restrictive in general, it is a natural theoretical starting point for our study. Basu, Conforti, Di Summa, and Jiang [1] (see also [16, Example 12]) provide a concrete construction demonstrating that adding cuts can lead to an exponential improvement in the B&C tree under a symmetric branching framework. Further, many combinatorial optimization problems, such as scheduling and set covering, have inherent algebraic or structural symmetries [19].

We first prove [Lemma 4](#), which states that no cut-minimal tree has a branch node whose children begin with the same path of cut nodes, unless both paths continue with cut nodes (from different families).¹ Ruling out this “redundant” substructure underlies our proofs of [Theorems 5](#) and [12](#).

Lemma 4. *Let $(\vec{c}, \ell, r) \in \mathcal{F}_{\searrow}^m \times \mathbb{R}_{>0} \times \mathbb{R}_{>0}$, and let T be a cut-minimal tree proving target bound Z with respect to a generalized time function $\tilde{w}$. Then T contains no branch node whose two child subtrees begin with a common sequence of $k \geq 1$ cut nodes (i.e., with the same families in the same order) that, in at least one of the two subtrees, is followed by a branch or leaf node.*

Proof. Suppose, for the sake of contradiction, that the children of a branch node $u \in \mathcal{V}_T$ begin with identical paths of k cut nodes, where at least one of these paths ends at a branch or leaf node.

Let T_L and T_R be the left and right child subtrees of u , whose first k cut nodes have the same ordered family labels. Denote the cut nodes in T_R by $v_1, \dots, v_k$, and let y be the child of v_k , which we may assume without loss of generality is a branch or leaf node (else, we apply an analogous argument to T_L). When y is a branch node, we denote its two children subtrees as $T_{y\ell}$ and T_{yr} ; when y is a leaf, we set $\tau_{\tilde{w}}(T_{y\ell}) = \tau_{\tilde{w}}(T_{yr}) = 0$ below. The subtree remaining after the first k cuts in T_L , denoted T_L^{sub} , may begin with further cuts. Let $\vec{h}(v)$ be the history vector at $v \in \mathcal{V}_T$ with respect to T . The first k cut nodes of T_L have the same families and cut histories as $v_1, \dots, v_k$, and hence the same node times.

We first consider replacing T_R in T by a modified tree T'_R , where the path of k cuts at the root of T_R is shifted down a “layer”. A visual aid for T_R and T'_R is in [Figure 2](#). The root of T'_R is a branch node u' . When y is a branch node, each child subtree of u' starts with a path of k cut nodes $v'_1, \dots, v'_k$, followed by a copy of $T_{y\ell}$ in the left child of u' and a copy of T_{yr} in the right child. When y is a leaf node, then each child of root u' in T'_R is a path of $k - 1$ cut nodes ending with a leaf node that we will call v'_k (in which case, unlike v_k , v'_k is not a cut node and has no generation cost). Note that T'_R proves a bound at least as good as T_R . Further, as the solving time for a node does not depend on the number of previous branch nodes, we define $\vec{h}'$ as a convenient modified history vector for T'_R , where $\vec{h}'(u') = \vec{h}(u)$ and $\vec{h}'(v'_i) = \vec{h}(v_i)$ for $i \in [k]$.

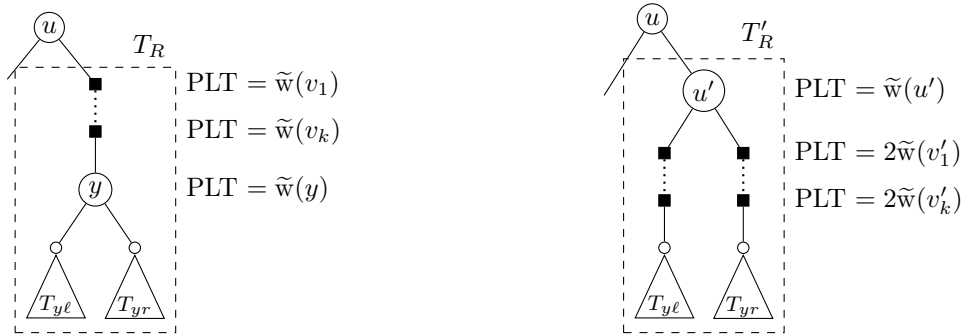


Figure 2: Illustration of T_R (left panel) and T'_R (right panel). PLT calculates the *Partial Layer Time*, calculated only on the nodes in the visualization.

¹The extra condition is needed for our proof, but we conjecture the statement holds more generally, i.e., that no cut-minimal tree has a branch node whose two children are cut nodes of the same family.

The (sub)tree times for T_R and T'_R are:

$$\begin{aligned}\tau_{\tilde{w}}(T_R) &= \sum_{i \in [k]} \tilde{w}(v_i) + \tilde{w}(y) + \tau_{\tilde{w}}(T_{y\ell}) + \tau_{\tilde{w}}(T_{yr}), \\ \tau_{\tilde{w}}(T'_R) &= \tilde{w}(u') + 2 \sum_{i \in [k]} \tilde{w}(v'_i) + \tau_{\tilde{w}}(T_{y\ell}) + \tau_{\tilde{w}}(T_{yr}) \\ &\leq \tilde{w}(u) + 2 \sum_{i \in [k]} \tilde{w}(v_i) + \tau_{\tilde{w}}(T_{y\ell}) + \tau_{\tilde{w}}(T_{yr}) \quad (\text{as } \tilde{w}(v'_k) \leq \tilde{w}(v_k)).\end{aligned}$$

As T is a minimal-time tree, we have that $\tau_{\tilde{w}}(T_R) \leq \tau_{\tilde{w}}(T'_R)$, and we derive

$$\tilde{w}(y) \leq \tilde{w}(u) + \sum_{i \in [k]} \tilde{w}(v_i). \quad (1)$$

Next, let T'' denote a tree obtained from T by shifting “up” a layer the path of k cut nodes from both children of u . Note that T'' proves the same bound as T . Let T_u be the subtree of T rooted at u , T_L^{sub} be the subtree of T_L rooted at the child of the k th cut node, and T''_u be the tree that starts with a path of k cut nodes $v''_1, \dots, v''_k$ followed by a branch node u'' , where the right child of u'' is the subtree T_y from T rooted at y and the left child is T_L^{sub} . Figure 3 depicts T_u and T''_u .

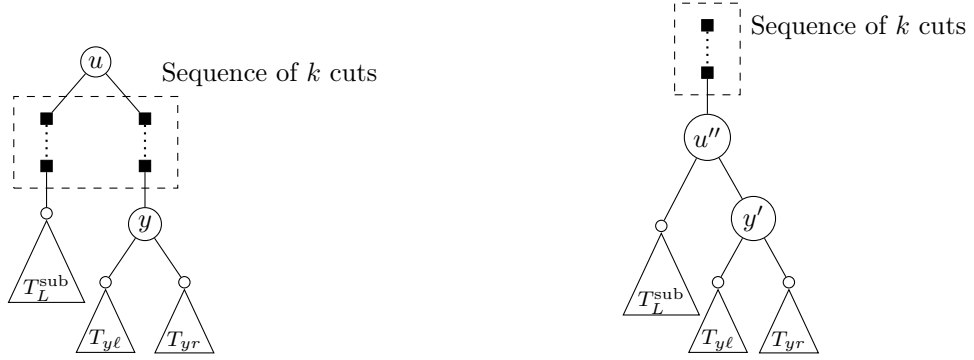


Figure 3: Illustration of the tree T_u (left panel) and T''_u (right panel).

Observe that

$$\begin{aligned}\tau_{\tilde{w}}(T_u) &= \tilde{w}(u) + 2 \sum_{i \in [k]} \tilde{w}(v_i) + \tilde{w}(y) + \tau_{\tilde{w}}(T_{y\ell}) + \tau_{\tilde{w}}(T_{yr}) + \tau_{\tilde{w}}(T_L^{\text{sub}}), \\ \tau_{\tilde{w}}(T''_u) &= \sum_{i \in [k]} \tilde{w}(v''_i) + \tilde{w}(u'') + \tilde{w}(y') + \tau_{\tilde{w}}(T_{y\ell}) + \tau_{\tilde{w}}(T_{yr}) + \tau_{\tilde{w}}(T_L^{\text{sub}}) \\ &= \sum_{i \in [k]} \tilde{w}(v_i) + 2\tilde{w}(y) + \tau_{\tilde{w}}(T_{y\ell}) + \tau_{\tilde{w}}(T_{yr}) + \tau_{\tilde{w}}(T_L^{\text{sub}}).\end{aligned}$$

Then $\tau_{\tilde{w}}(T''_u) - \tau_{\tilde{w}}(T_u) = \tilde{w}(y) - (\tilde{w}(u) + \sum_{i \in [k]} \tilde{w}(v_i))$ is always nonpositive by (1). Hence, the modified tree T'' has tree time at most that of T , while having k fewer cut nodes, contradicting the assumption that T is cut-minimal. $\square$

We next prove, in Theorem 5, that under symmetric branching, every cut-minimal tree has all cuts at the root, allowing multiple cut families and cut generation time. This generalizes a previous result [16, Theorem 23] that assumes only one cut family and constant cut strength.

Theorem 5. *Let $(\vec{c}, \ell, r) \in \mathcal{F}_{\searrow}^m \times \mathbb{R}_{>0} \times \mathbb{R}_{>0}$ such that $\ell = r$. For any target bound Z and generalized time function $\tilde{w}$, every cut-minimal B&C tree proving bound Z has only root cuts.*

Proof. Let T be a cut-minimal tree proving bound Z with respect to $\tilde{w}$, and suppose that a branch node u of T has a cut node v as a child. Let y be the other child of u . Since $\ell = r$, $z_T(v) = z_T(y)$ and $\vec{h}(v) = \vec{h}(y)$, so (as T is cut-minimal) T_v and T_y have the same tree time and contain the same number of cut nodes. Replacing T_y by a copy of T_v therefore yields a cut-minimal tree in which both child subtrees of u begin with the same maximal sequence of $k \geq 1$ cut nodes, with the same families in the same order, followed by a branch or leaf node, contradicting Lemma 4. Hence, no branch node of T has a cut node as a child, so every cut node of T lies on the path of cut nodes starting at the root, i.e., T has only root cuts. $\square$

We emphasize that symmetry plays a crucial role in this proof. In particular, symmetry of a tree will force cut nodes to move up, regardless of how the time function is defined. On the other hand, the proof does not rely on branching strength: the result can be extended even with multiple symmetric branch variable candidates, i.e., possibly different values throughout the tree but always equal left/right bound improvements.

3.2 Ordering Weak Versus Strong Cut Families

Restricting to only root-cut trees, whether optimal or not, leaves open questions such as the number of cuts to add (previously approximated in [16, Theorem 15]) and, with multiple cut families, their order. We consider this latter question for a path of cuts (which may not be at the root) in a minimal-time tree, when there are weak and strong (constant-strength) cut families. For a constant cut function, we use c_i to denote both the function and its value.

Theorem 6. *Let $(\vec{c}, \ell, r) \in \mathcal{F}_{\searrow}^m \times \mathbb{R}_{>0} \times \mathbb{R}_{>0}$ with constant cut functions c_i for $i \in [m]$. For any target bound Z and generalized time function $\tilde{w}$, there exists a minimal-time tree proving Z in which, along every path of cut nodes, the cut families appear in nondecreasing order of strength c_i .*

Proof. Let T be a minimal-time tree proving Z with respect to $\tilde{w}$, and consider a path of κ consecutive cut nodes in T . Let k_i denote the number of cuts from family $i \in [m]$ in this path. We will reorder these cut nodes by nondecreasing strength. Since the cut functions are constant, the reordered tree still proves Z (any permutation of the path increases the bound by $\sum_{i=1}^m k_i c_i$), and all nodes outside of the path keep their bounds and node times. For the nodes on the path, as $\tilde{w}$ is the sum of a simple time function w and separation times, we analyze its two components separately.

First, we show that total separation time is invariant under any reordering. Let $\vec{h}$ be the history vector at the first cut node in the path. The total separation time of the sequence is $\sum_{i=1}^m \sum_{h=\vec{h}_i}^{\vec{h}_i+k_i-1} s_i(h)$, which depends only on $\vec{h}$ and the counts k_i , not the order.

Next, for node processing time, label the positions of the cuts as $0, \dots, \kappa-1$, from start to end of the path, and let f_j be the index of the cut family at position j . Let v_{j_0} , at position j_0 , be a cut from a weakest family in the sequence, and consider moving v_{j_0} to position 0. This does not affect the nodes before position 0 or after position j_0 , so it suffices to analyze positions $0, \dots, j_0$. Let w_j and w'_j denote the node processing time at position j before and after the move. Then $w_0 = w'_0$. For $j \in [j_0]$, let $X_j := \vec{h} + \sum_{j'=0}^{j-2} e_{f_{j'}}$, from which the history vector at position j is $X_j + e_{f_{j-1}}$ before the move and $X_j + e_{f_{j_0}}$ after it. Since $c_{f_{j_0}} \leq c_{f_{j-1}}$, Definition 1 gives $w'_j = w(X_j + e_{f_{j_0}}) \leq w(X_j + e_{f_{j-1}}) = w_j$.

Recursively repeating this move of the next-weakest cut node, starting at position 1, then 2, and so on, orders the path by nondecreasing strength without increasing its tree time. Applying this to every maximal path of consecutive cut nodes in T (these sequences are disjoint, and each step leaves all other nodes unchanged) yields a minimal-time tree with the desired property. $\square$

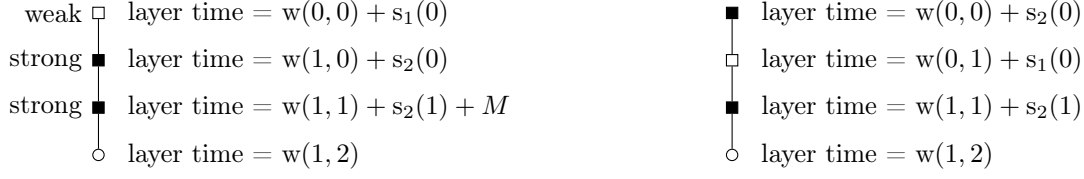


Figure 4: With more general separation time functions, placing weaker cuts before stronger ones, as in [Theorem 6](#), may be suboptimal. In the example in this figure, if generating two consecutive cuts from the same family incurs a cost $M > w(0, 1) - w(1, 0)$, the alternating structure in the right panel is cheaper than the weak-then-strong pattern in the left panel.

[Theorem 6](#) relies on the crucial assumption from [Definition 2](#) that separation time depends only on the number of previous cuts from the same family, so that separation time does not affect the order of consecutive cut nodes. [Figure 4](#) depicts one example where this assumption is relaxed: when there is a heavy penalty on consecutive nodes belonging to the same cut family, the minimal-time tree alternates between cut families instead of following a weak-then-strong order.

3.3 Root-Cut Trees Under Asymmetric Branching

In [Section 3.1](#), we showed that it is sufficient to consider root cuts in symmetric branching settings with multiple cut families. We now turn to the single cut-family, single branching-family B&C model with constant but nonidentical bound improvements ($\ell \neq r$), under a simple time function.

By [Lemma 4](#), symmetry in a tree shifts cut nodes upward toward the root. Consequently, a natural question is whether symmetry in the tree is necessary, i.e., might one always find, under an asymmetric branching regime, a combination of target bound and time function that forces the inclusion of local cuts within any minimal-time tree. We prove that this is not universally true: there exists an asymmetric setting ($c:\ell:r$ in ratio 1:2:4) in which a minimal-time tree without any internal cuts always exists, regardless of the specific form of the (simple) time function. Proving this involves several intermediate results that we present first.

Define a *branch-then-bothcut* node to be a branch node with both children being cut nodes from the same family. The following corollary, which follows directly from [Lemma 4](#), states that a branch-then-bothcut node cannot exist for a single cut family setting.²

Corollary 7. *For any $(c, \ell, r) \in \mathbb{R}_{>0}^3$, target bound Z , and generalized time function $\tilde{w}$, every cut-minimal tree proving Z has no branch-then-bothcut node.*

Proof. If a cut-minimal tree has a branch-then-bothcut node, then the two children of this node begin with the same sequence of $\min\{k_\ell, k_r\}$ cut nodes, where k_ℓ and k_r are the lengths of the maximal cut paths starting at each child. By maximality of the cut paths, at least one of these sequences is followed by a branch or leaf node, so [Lemma 4](#) implies the tree is not cut-minimal. $\square$

It remains unclear whether branch nodes with exactly one child as a cut node may be necessary. More importantly, if such branch nodes do appear in a minimal-time tree, it is not immediately clear whether the deepest such node has a left-child cut or a right-child cut. We call a branch-then-cut node a *branch-then-leftcut* node if its left child is an internal cut. We address this question (for internal cut nodes) in [Lemma 8](#). When $\ell = r$, [Theorem 5](#) implies that no cut-minimal tree has a branch-then-cut node, so we focus on $r = 2\ell$.

²It can be optimal for both children of a branch node to be cuts from different families.

Lemma 8. *Let $(c, \ell, r) \in \mathbb{R}_{>0}^3$ with $r = 2\ell$, w be a simple time function, and Z be a target bound. Among all cut-minimal trees proving Z , let T be one whose deepest branch-then-cut node (if any) is as deep as possible. Every deepest branch-then-cut node of T is a branch-then-leftcut node.*

Proof. Suppose that a deepest branch-then-cut node u of T has an internal cut node v as its right child. The left child of u is a branch node x — it is not a cut node, as otherwise u would be a branch-then-bothcut node, contradicting [Corollary 7](#), and it is not a leaf node, as $z_T(x) < z_T(v) < Z$, where the last inequality holds by minimality since v is not a leaf. Let y be the left child of x . Then $z_T(y) = z_T(u) + 2\ell = z_T(v)$, and v and y have the same cut history, so either of the subtrees T_v and T_y can replace the other, and minimality gives $\tau_w(T_v) = \tau_w(T_y)$. As T is cut-minimal, T_y has at least as many cut nodes as T_v , so replacing T_y by a copy of T_v yields a cut-minimal tree in which x is a branch-then-cut node deeper than u , contradicting the choice of T . $\square$

Unlike symmetric branching, where symmetry is maintained throughout the tree, any node-shifting operation can significantly alter the tree structure under an asymmetric branching setting. Therefore, it is crucial to understand how tree structure changes as the target bound varies. [Lemma 9](#) provides a useful pattern under the special setting of $c:\ell:r = 1:2:4$. For any $z \in \mathbb{R}$, let $p(z) := |T(z)|$ (the size of the smallest pure branching tree that proves bound z), with $p(z) = 1$ if $z \leq 0$, and let $\lambda(z)$ denote the number of leaf nodes of $T(z)$ with the smallest bound. If ℓ , r , and z are even, then $\lambda(z)$ equals the number of leaf nodes proving exactly z . Note also that $p(0) = \lambda(0) = 1$.

Lemma 9. *If $c:\ell:r = 1:2:4$ and $Z \geq 4c$ is an even multiple of c , then*

- (1) $p(Z + 2c) + p(Z - 4c) = 2p(Z)$, and
- (2) $\lambda(Z + 2c) + \lambda(Z - 4c) = 2\lambda(Z)$, with $\lambda(Z) - \lambda(Z - 4c) = \lambda(Z - 2c) \geq 1$.

Proof. For $z > 0$, the root of $T(z)$ branches into copies of $T(z - 2c)$ and $T(z - 4c)$, so $p(z) = 1 + p(z - 2c) + p(z - 4c)$ [\[18\]](#). Moreover, for $z \geq 4c$, the leaves of $T(z)$ that prove exactly z are the union of the leaves of $T(z - 2c)$ and $T(z - 4c)$ that prove their respective (relative) target bounds, so $\lambda(z) = \lambda(z - 2c) + \lambda(z - 4c)$. Hence, $f \in \{p, \lambda\}$ satisfies $f(z) = a_f + f(z - 2c) + f(z - 4c)$ for $z \geq 4c$, with $a_p = 1$ and $a_\lambda = 0$. This gives $f(Z + 2c) - f(Z) = f(Z) - f(Z - 4c)$, which proves (1) and the first part of (2). Finally, $\lambda(Z) - \lambda(Z - 4c) = \lambda(Z - 2c) \geq 1$, as the leftmost leaf of $T(Z - 2c)$ proves exactly $Z - 2c$. $\square$

Next, [Lemma 10](#) states that, when $\ell \geq 2c$, it is (weakly) better to replace two consecutive leaf cuts (two cut nodes followed by a leaf) by a branch node with two leaf children, i.e., no leaf is preceded by two consecutive cut nodes.

Lemma 10. *For any $(c, \ell, r) \in \mathbb{R}_{>0}^3$ with $2c \leq \ell \leq r$, simple time function w , and target bound Z , every cut-minimal tree proving Z contains at most one leaf cut node on each root-to-leaf path.*

Proof. Suppose that a cut-minimal tree contains two consecutive cut nodes followed by a leaf, with h cut nodes above the first of them. Replacing these three nodes by a branch node with two leaf children yields a tree that still proves Z (as $r \geq \ell \geq 2c$), has weakly smaller tree time (as $3w(h) \leq w(h) + w(h + 1) + w(h + 2)$), and has two fewer cut nodes, a contradiction. $\square$

[Lemma 11](#) proves a property of a cut-minimal tree when $c:\ell:r = 1:2:4$: for a subtree T_g with no internal cuts, after we “remove” any leaf cuts (i.e., replace each leaf cut and its leaf child by a single leaf), we obtain the smallest pure branching tree T_G proving a bound Z_G that is an even multiple of c . Further, either $T_g = T_G$, or T_g is exactly T_G with an additional cut node inserted above each leaf v of T_G with $z_{T_g}(v) = Z_G$. Thus, the time of T_g is determined by Z_G and whether leaf cuts are used. The proof of [Theorem 12](#) will use an idea similar to our proof in [Lemma 4](#), of

shifting cuts up or down a candidate tree. Our analysis is complicated by the asymmetric setting; [Lemma 9](#) handles this, as our moves change Z_G by even multiples of c .

Lemma 11. *Suppose that $c : \ell : r = 1 : 2 : 4$, and let T be a cut-minimal tree proving a target bound Z with respect to a simple time function w . Let g be a branch node of T such that the subtree T_g contains no internal cuts, let T_G be the pure branching tree obtained from T_g by removing its leaf cuts, and let Z_G be the smallest bound at a leaf node of T_G , relative to g . Then*

- (1) Z_G is an even multiple of c with $Z_G \geq 2c$;
- (2) $T_G = T(Z_G)$, the smallest pure branching tree proving Z_G ; and
- (3) the nodes v of T_g with $z_{T_g}(v) = Z_G$ are either all leaves or all leaf cuts, and $z_{T_g}(v) = Z_G$ at every leaf cut v in T_g .

Proof. As $(\ell, r) = (2c, 4c)$, for all nodes v of T_G , $z_{T_G}(v)$ is an even multiple of c . Since g is a branch node, we have $Z_G \geq 2c$, proving (1). Let $R := Z - z_T(g)$ be the remaining bound at g , so that $z_{T_g}(v) \geq R$ for every leaf v of T_g . Let v be a leaf node of T_G . By minimality, if $z_{T_G}(v) \geq R$, then T_g has the exact same leaf (without a leaf cut inserted). Conversely, if $z_{T_G}(v) < R$, then a leaf cut is needed there in T_g , and, by [Lemma 10](#), a single leaf cut suffices. Thus, as Z_G is the smallest leaf bound in T_G , T_g contains leaf cuts if and only if $Z_G < R$. Moreover, $R \leq Z_G + c$, and, as all bounds are even multiples of c , every leaf cut u of T_g has $z_{T_g}(u) = Z_G$, proving the second half of (3).

Observe that, by construction, all branch nodes in T_G and T_g are identical. If T_g contains no leaf cuts, then $T_g = T_G$ and $R \leq Z_G$, so (2) follows from minimality of T , and all nodes of T_g with bound Z_G are leaves, completing (3). Otherwise, $Z_G < R$, and let v be a branch node of T_G with bound $z_v := z_{T_G}(v) (= z_{T_g}(v))$. For T_G to be the smallest pure branching tree proving Z_G , we need to show that $z_v < Z_G$. By minimality of T , as v is not a leaf, $z_v < R$ (implying $z_v \leq Z_G$). By the definition of Z_G , some leaf of T_G has bound $Z_G < R$, and, as argued above, the corresponding position in T_g is a leaf cut. If $z_v = Z_G$, the children of v are leaves with bound at least R , and T_v can replace the leaf cut and leaf in T_g without increasing tree time, contradicting cut-minimality of T . Thus, $z_v < Z_G$, proving (2). Finally, as $Z_G < R$, every leaf u of T_G with $z_{T_G}(u) = Z_G$ needs a leaf cut in T_g , and, by (2), these are the only nodes of T_g with bound Z_G , proving the first half of (3). $\square$

Finally, we prove [Theorem 12](#), the main result of this subsection: for (c, ℓ, r) satisfying a ratio $1 : 2 : 4$, there exists a minimal-time tree where all cuts are either at the root or at the leaves.

Theorem 12. *For $(c, \ell, r) \in \mathbb{R}_{>0}^3$ with $c : \ell : r = 1 : 2 : 4$, every target bound Z , and any simple time function w , there exists a minimal-time tree that does not contain any internal cuts.*

Proof. Suppose, for the sake of contradiction, that for some target bound Z and time function w , every minimal-time tree proving Z has an internal cut. Then every cut-minimal tree has a branch-then-cut node. Let u be the deepest branch-then-cut node of a tree $\hat{T}$ whose deepest branch-then-cut node is as deep as possible among all cut-minimal trees proving Z . Let T be the subtree rooted at u , with T_ℓ the left and T_r the right child subtree of u . By [Lemma 8](#), u is a branch-then-leftcut node. Let κ be the length of the maximal path of cut nodes starting at the root of T_ℓ . These cuts are internal, so this path is followed by a branch node, which we will denote by g , the root of subtree T_g .

The proof compares two ways of moving this path: up a layer above u , or down two layers below two branch nodes. As pure branching trees have a recursive structure in this setting ([Lemma 9](#)), these moves change tree time by exactly opposite amounts. As moving the cuts down cannot help in a minimal-time tree, moving them up is at least as good. This yields a tree with fewer cut nodes or a deeper branch-then-cut node, contradicting the choice of $\hat{T}$.

Let K be the number of cut nodes above u , and let $\bar{w}(h) := w(K + h)/w(K)$ for $h \in \mathbb{Z}_{\geq 0}$, which is again a simple time function with $\bar{w}(0) = 1$. As the roots of T , T_ℓ , and T_r all have K

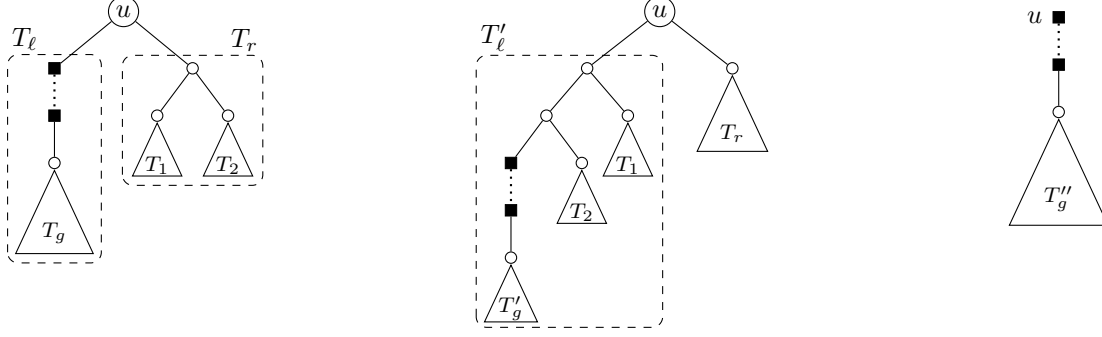


Figure 5: Left panel: tree T . Middle panel: tree T' , obtained by moving the κ cut nodes down two layers. Right panel: tree T'' , obtained by moving the κ cut nodes up one layer to u .

cut nodes above them, [16, Lemma 24] implies that they are minimal-time trees for their remaining bounds with respect to $\bar{w}$; for simplicity, we write w for $\bar{w}$ hereafter. Moreover, these trees are also cut-minimal for their remaining bounds, as otherwise replacing one of them by a minimal-time subtree with fewer cut nodes would contradict the cut-minimality of the whole tree.

Let $Z_u := Z - z_{\hat{T}}(u)$ be the remaining bound at u , and let $Z_\ell := Z_u - \ell$ be the remaining bound that T_ℓ needs to prove. By minimality and the need for an internal cut in T_ℓ , we have that $Z_\ell > \ell = 2c$. This implies that T_r is not a single leaf node, as it must prove $Z_\ell - 2c > 0$. The root of T_r is not a cut node, as otherwise u would be a branch-then-bothcut node, contradicting Corollary 7. Thus, the root of T_r is a branch node. Let T_1 and T_2 be its left and right child subtrees.

Since u is a deepest branch-then-cut node, there is no internal cut in T_r or in T_g . Let T_G and Z_G be as in Lemma 11 applied to g , and let $I_{\text{leaf}} \in \{0, 1\}$ indicate whether T_g contains a leaf cut. By Lemma 11, $Z_G \geq 2c$ is an even multiple of c and $T_G = T(Z_G)$ (statements (1) and (2)), and T_g is obtained from T_G by inserting a leaf cut above each of its $\lambda(Z_G)$ leaves v with $z_{T_g}(v) = Z_G$ if $I_{\text{leaf}} = 1$ (statement (3)). Thus, T_ℓ proves $\kappa c + Z_G + I_{\text{leaf}}c \geq Z_\ell$.

For an even multiple $\bar{Z} \geq 0$ of c , define $F(\bar{Z}, I_{\text{leaf}}) := w(\kappa)p(\bar{Z}) + I_{\text{leaf}}w(\kappa + 1)\lambda(\bar{Z})$, so that $F(Z_G, I_{\text{leaf}}) = \tau_w(T_g)$. The tree times of T_ℓ , T_r , and T are then

$$\begin{aligned} \tau_w(T_\ell) &= \sum_{i \in [0, \kappa-1]} w(i) + F(Z_G, I_{\text{leaf}}), \\ \tau_w(T_r) &= 1 + \tau_w(T_1) + \tau_w(T_2), \quad \text{and} \quad \tau_w(T) = 1 + \tau_w(T_\ell) + \tau_w(T_r). \end{aligned} \tag{2}$$

As F is a linear combination of p and λ , Lemma 9 implies that the identity

$$F(\bar{Z} + 2c, I_{\text{leaf}}) - F(\bar{Z}, I_{\text{leaf}}) = F(\bar{Z}, I_{\text{leaf}}) - F(\bar{Z} - 4c, I_{\text{leaf}}) \tag{3}$$

holds for all even multiples $\bar{Z} \geq 4c$ of c and $I_{\text{leaf}} \in \{0, 1\}$.

Case 1: $Z_G \geq 4c$. We compare T with two other trees, illustrated in Figure 5, that also prove Z_u . For the first tree, T' , we shift the κ cut nodes *down two layers* along the leftmost path of T_ℓ while keeping T_r unchanged. In T' , the left child of u is a branch node whose right child is a copy of T_1 (as both are reached from u by one left and one right branch, they have the same bound and history). The left child of this branch node is again a branch node, whose right child is a copy of T_2 (as branching twice left and once right is equivalent to two right branches when $r = 2\ell$), and whose left child is the path of κ cut nodes followed by T'_g , obtained from $T'_G := T(Z_G - 4c)$ by inserting (if $I_{\text{leaf}} = 1$) a leaf cut above each leaf v with $z_{T'_g}(v) = Z_G - 4c$. To see that T' proves Z_u , the only nontrivial case is the leftmost path to the leaves of T'_g . As all bounds are even multiples of c , every

leaf v of T'_G with $z_{T'_G}(v) \neq Z_G - 4c$ has $z_{T'_G}(v) \geq Z_G - 2c$, so T'_g proves $Z_G - 4c + I_{\text{leaf}}c$. Hence, every leaf v of T' in T'_g satisfies $z_{T'}(v) \geq 3\ell + \kappa c + (Z_G - 4c) + I_{\text{leaf}}c = \ell + \kappa c + Z_G + I_{\text{leaf}}c \geq \ell + Z_\ell = Z_u$.

Next, define tree T'' by shifting the κ cut nodes *up one layer*: T'' has a path of κ cut nodes followed by T''_g , analogously related to $T''_G := T(Z_G + 2c)$ as T'_g to T'_G . As with T'_g , every leaf v of T'' satisfies $z_{T''}(v) \geq \kappa c + (Z_G + 2c) + I_{\text{leaf}}c = \ell + \kappa c + Z_G + I_{\text{leaf}}c \geq Z_u$, so T'' also proves Z_u .

The subtree T'_ℓ of T' rooted at the left child of u consists of two branch nodes, copies of T_1 and T_2 , the κ cut nodes, and T'_g , so $\tau_w(T'_\ell) = 1 + \tau_w(T_r) + \sum_{i=0}^{\kappa-1} w(i) + F(Z_G - 4c, I_{\text{leaf}})$, using $\tau_w(T_r) = 1 + \tau_w(T_1) + \tau_w(T_2)$. The tree time of T'' is $\tau_w(T'') = \sum_{i=0}^{\kappa-1} w(i) + F(Z_G + 2c, I_{\text{leaf}})$. Combining these expressions with (2) and (3) gives us

$$\begin{aligned} \Delta &:= \tau_w(T) - \tau_w(T'') = 1 + \tau_w(T_r) - (F(Z_G + 2c, I_{\text{leaf}}) - F(Z_G, I_{\text{leaf}})) \\ &= 1 + \tau_w(T_r) - (F(Z_G, I_{\text{leaf}}) - F(Z_G - 4c, I_{\text{leaf}})) \\ &= \tau_w(T'_\ell) - \tau_w(T_\ell) = \tau_w(T') - \tau_w(T). \end{aligned} \tag{4}$$

Since T is minimal-time and both T' and T'' prove Z_u , we have $\tau_w(T') \geq \tau_w(T)$ and $\tau_w(T'') \geq \tau_w(T)$, i.e., $\Delta \geq 0$ and $\Delta \leq 0$ by (4). Thus, $\Delta = 0$, and T' and T'' are both minimal-time trees proving Z_u . Let $n(\cdot)$ denote the number of cut nodes of a tree. As T' contains T_r as well as copies of T_1 and T_2 , statement (2) of Lemma 9 gives

$$\begin{aligned} n(T') + n(T'') &= (\kappa + I_{\text{leaf}} \lambda(Z_G - 4c) + 2n(T_r)) + (\kappa + I_{\text{leaf}} \lambda(Z_G + 2c)) \\ &= 2(\kappa + I_{\text{leaf}} \lambda(Z_G) + n(T_r)) = 2n(T). \end{aligned}$$

As T is cut-minimal, $n(T') \geq n(T)$ and $n(T'') \geq n(T)$, so $n(T') = n(T'') = n(T)$, and replacing T by T' in $\hat{T}$ yields a cut-minimal tree. If $Z_G \geq 6c$, then the κ cut nodes of T' are followed by a branch node, so this cut-minimal tree has a branch-then-cut node two layers deeper than u , contradicting the choice of $\hat{T}$. If $Z_G = 4c$, then $T'_G = T(0)$ is a single leaf, so in T' the κ cut nodes, together with a leaf cut if $I_{\text{leaf}} = 1$, are followed by a leaf, which contradicts Lemma 10 unless $\kappa = 1$ and $I_{\text{leaf}} = 0$.

In this remaining case, $1 = n(T'') = n(T) = 1 + n(T_r)$, so T_r has no cut nodes. Then T_ℓ consists of a cut node followed by $T_G = T(4c)$, which has 5 nodes, so $\tau_w(T_\ell) = 1 + 5w(1) \geq 6$. Moreover, $Z_\ell > 4c$, as otherwise $T(Z_\ell)$ has at most 5 nodes and time at most $5 < 6 \leq \tau_w(T_\ell)$, contradicting minimality. Hence, the right child of u must prove $Z_\ell - 2c \in (2c, 3c]$, so T_r , which has no cut nodes, is $T(4c)$ (with 5 nodes). The left child of the root of T_r has remaining bound in $(0, c]$ and two leaf children; replacing these three nodes by a cut node followed by a leaf yields a valid tree, so the minimality of T_r gives $3 \leq 1 + w(1)$, i.e., $w(1) \geq 2$. On the other hand, $\Delta = 1 + \tau_w(T_r) - w(1)(p(4c) - p(0)) = 6 - 4w(1) \geq 0$ gives $w(1) \leq 3/2$, which is a contradiction.

Case 2: $Z_G = 2c$. Here, we cannot define T' via the “two-layers-down” operation, as $T_G = T(2c)$ has only one branch node.

Case 2a: T_ℓ contains a leaf cut. T_ℓ consists of the κ cut nodes followed by a branch node with two children (a leaf cut and a leaf). Then $\tau_w(T_\ell) = \sum_{i=0}^{\kappa-1} w(i) + 3w(\kappa) + w(\kappa + 1)$. Replacing T_ℓ by $\kappa - 1$ cut nodes followed by $T(4c)$ (with 5 nodes) yields a subtree that proves the same bound, has tree time $\sum_{i=0}^{\kappa-2} w(i) + 5w(\kappa - 1) \leq \tau_w(T_\ell)$ since w is nondecreasing, and has two fewer cut nodes, contradicting cut-minimality of T . Figure 6 shows case $\kappa = 1$.

Case 2b: T_ℓ contains no leaf cuts. With no leaf cuts in T_ℓ , $\tau_w(T_\ell) = \sum_{i=0}^{\kappa-1} w(i) + 3w(\kappa)$. Analogous to the “two-layers-down” operation, we now employ a “one-layer-down” transformation, and we continue to denote the resulting tree by T' : the left child of u becomes a branch node whose right child is a copy of T_1 and whose left child is the path of κ cut nodes followed by a leaf, which suffices as T_ℓ proves $\kappa c + 2c$. We also continue to use T'' for the tree consisting of the κ cut nodes followed



Figure 6: Exceptional case when $Z_G = 2c$ and T_ℓ contains leaf cuts. The tree on the left panel is dominated by the tree on the right, with fewer cut nodes.

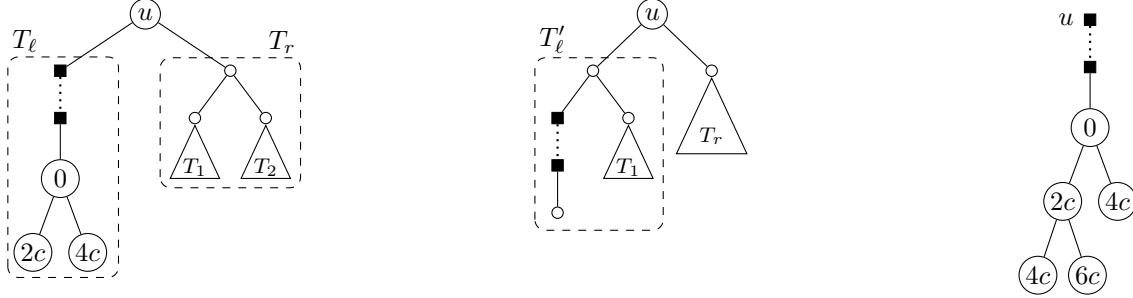


Figure 7: Case $Z_G = 2c$ with no leaf cuts. Left panel: tree T . Middle panel: tree T' , obtained by moving the κ cut nodes down a layer. Right panel: tree T'' , obtained by moving the κ cut nodes up a layer to u . Node labels are bounds relative to the node after the cut path.

by $T(4c)$ (with 5 nodes), which also proves Z_u , as in Case 1. Visualizations of these trees are in Figure 7. Denoting by T'_ℓ the subtree of T' rooted at the left child of u , the minimality of T gives

$$\tau_w(T'_\ell) = 1 + \tau_w(T_1) + \sum_{i=0}^{\kappa} w(i) \geq \sum_{i=0}^{\kappa-1} w(i) + 3w(\kappa) = \tau_w(T_\ell),$$

which implies $1 + \tau_w(T_1) \geq 2w(\kappa)$. Comparing the tree times of T and T'' , and using $\tau_w(T_r) = 1 + \tau_w(T_1) + \tau_w(T_2) > \tau_w(T_1)$, we obtain

$$\begin{aligned} \tau_w(T) - \tau_w(T'') &= \left(1 + \tau_w(T_r) + \sum_{i=0}^{\kappa-1} w(i) + 3w(\kappa) \right) - \left(\sum_{i=0}^{\kappa-1} w(i) + 5w(\kappa) \right) \\ &= (1 + \tau_w(T_r)) - 2w(\kappa) > (1 + \tau_w(T_1)) - 2w(\kappa) \geq 0, \end{aligned}$$

contradicting the minimality of T .

As every case leads to a contradiction, this completes the proof. $\square$

Theorem 12 demonstrates that the importance of root-cut structure extends beyond symmetric branching. For this special asymmetric setting, not using internal cuts remains optimal uniformly over arbitrary target bounds and time functions. What, in particular, is special about this setting? While we do not have a precise answer, computationally, we could not find any instances requiring internal cuts when $r = 2\ell$, but the current proof does not directly extend to this more general case.

The results derived for the single cut-family, single branching-family model may not easily extend to settings with multiple branching variables or cut families. For instance, while internal cuts are unnecessary under the individual settings $(c_1, \ell_1, r_1) = (1, 2, 4)$ (by Theorem 12) and

$(c_2, \ell_2, r_2) = (1, 3, 3)$ (by [Theorem 5](#)), the existence of an internal-cuts-free minimal-time tree is no longer guaranteed when the two settings are combined, as [Example 2](#) shows.

Example 2. Consider a single cut family with constant strength $c = 1$ and two branching variables, with bound improvements $(2, 4)$ and $(3, 3)$, so that each branch node may use either pair. Let $Z = 8$ and $w = (1, a, M, M, \dots)$, with $a \in (1, 4/3)$ and M sufficiently large, so that at most one cut is used on any root-to-leaf path. The tree in [Figure 8](#) branches at the root using $(2, 4)$; its left child is followed by three branch nodes using $(3, 3)$ and four leaves, and its right child is an internal cut followed by a branch node using $(3, 3)$ with two leaves. Its tree time is $9 + 3a$, and it is the unique minimal-time tree, whereas the best tree without internal cuts has tree time $\min\{1 + 11a, 13\}$, attained by a root cut followed by pure branching or by pure branching.

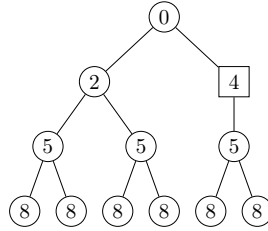


Figure 8: The unique minimal-time tree in [Example 2](#), for $c = 1$, branching improvements $(2, 4)$ or $(3, 3)$ at each branch node, $Z = 8$, and $w = (1, a, M, \dots)$ with $a \in (1, 4/3)$. The root uses $(2, 4)$ and all other branch nodes use $(3, 3)$; the cut node with bound 4 is an internal cut. Node labels are bounds, and square nodes are cut nodes.

4 How Close Are Root-Cut Trees to Optimal?

[Section 3](#) studies settings in which some minimal-time tree has no internal cuts, but generally, minimal-time trees may need local cuts, possibly at any depth ([Section 5.1](#)). In this section, we quantify how much is lost by restricting cuts to the root: [Section 4.1](#) bounds the ratio between the tree time of the best root-cut tree and that of a minimal-time tree for any fixed target bound, and [Section 4.2](#) shows that, as the target bound grows, the best root-cut tree grows at the same rate as a minimal-time tree and, for constant cuts, is asymptotically optimal.

Throughout this section, we consider a single cut family setting $(c, \ell, r) \in \mathcal{F}_\downarrow \times \mathbb{R}_{>0} \times \mathbb{R}_{>0}$ and a generalized time function $\tilde{w}$ with simple time function w and separation time function s (of the cut family). Thus, a branch or leaf node with h cut nodes above it has time $w(h)$, and a cut node with h cut nodes above it has time $w(h) + s(h)$. Let $C_k := \sum_{h=1}^k c(h)$ denote the cumulative bound improvement of k cuts. For a target bound Z , let $\tau_{\tilde{w}}^*(Z)$ denote the tree time of a minimal-time tree proving Z with respect to $\tilde{w}$, with no restriction on where cuts are placed, and let $t^*(Z)$ denote the size of a minimal-size tree proving Z . We use $p(z) = |T(z)|$ for $z \in \mathbb{R}$, as in [Section 3.3](#).

For $k \in \mathbb{Z}_{\geq 0}$, let $\pi(k)$ be the tree time of a path of k cut nodes then a leaf:

$$\pi(k) := \sum_{i \in [0, k-1]} (w(i) + s(i)) + w(k).$$

A root-cut tree with k cut nodes proves Z if and only if its branching component proves $Z - C_k$. All of its branch and leaf nodes have time $w(k)$, so the minimal-time such tree uses $T(Z - C_k)$ and

$$\tau_{\text{root}}(k) := \sum_{i=0}^{k-1} (w(i) + s(i)) + w(k) p(Z - C_k) = \pi(k) + w(k) (p(Z - C_k) - 1). \quad (5)$$

The minimal time of a root-cut tree proving Z is $\tau_{\text{root}}^*(Z) := \min_{k \in \mathbb{Z}_{\geq 0}} \tau_{\text{root}}(k)$, and we denote by $k_{\text{root}}(Z)$ a minimizer. We refer to a root-cut tree attaining $\tau_{\text{root}}^*(Z)$ as a *best root-cut tree*.

4.1 Root-Cut Approximation for a Fixed Target Bound

We first bound the tree time of any tree from below, in terms of the maximum number of cut nodes along a root-to-leaf path.

Lemma 13. *Let T be any tree proving Z , and let k be the maximum number of cut nodes on a root-to-leaf path of T . Then $|T| \geq k + p(Z - C_k)$ and*

$$\tau_{\tilde{w}}(T) \geq \pi(k) + p(Z - C_k) - 1.$$

Proof. Deleting the cut nodes of T leaves a pure branching tree B . Each leaf of B has at most k cut nodes above it in T , which improve its bound by at most C_k , so B proves $Z - C_k$ and $|B| \geq p(Z - C_k)$. As T has at least k cut nodes, $|T| \geq k + |B|$, which proves the first claim. For the second, a root-to-leaf path with k cut nodes has time $\pi(k)$, and each of the remaining $|T| - k - 1$ nodes has time at least 1. $\square$

Lemma 13 uses the argument of [16, Lemma 7] and [14, Lemma 4.1], which move all cut nodes to the root without increasing the size of a tree. With this, we bound the root-cut approximation for a minimal-time tree. For a minimal-time tree T^* proving Z , let k^* denote the maximum number of cut nodes on a root-to-leaf path of T^* , and let $B^* := p(Z - C_{k^*}) - 1$.

Proposition 14. *For every target bound Z ,*

$$\pi(k^*) + B^* \leq \tau_{\tilde{w}}^*(Z) \leq \tau_{\text{root}}^*(Z) \leq \min\{p(Z), \pi(k^*) + w(k^*) B^*\}.$$

Consequently,

$$\frac{\tau_{\text{root}}^*(Z)}{\tau_{\tilde{w}}^*(Z)} \leq \min \left\{ \frac{p(Z)}{\pi(k^*) + B^*}, \frac{\pi(k^*) + w(k^*) B^*}{\pi(k^*) + B^*} \right\} \leq w(k^*).$$

Proof. The first inequality is Lemma 13 applied to T^* , and the second holds as root-cut trees are a subset of all trees. For the third, (5) gives $\tau_{\text{root}}(0) = p(Z)$, as $\pi(0) = w(0) = 1$, and $\tau_{\text{root}}(k^*) = \pi(k^*) + w(k^*) B^*$. The last inequality follows from $\pi(k^*) \leq w(k^*) \pi(k^*)$. $\square$

The factor $w(k^*)$ is the bound of [13, Theorem 11]. Proposition 14 improves it to $(\pi(k^*) + w(k^*) B^*) / (\pi(k^*) + B^*)$ and extends it to worsening cut functions and separation time. The bound is exact when $w \equiv 1$ and $s \equiv 0$, and when $B^* = 0$, in which case the pure cutting tree with k^* cut nodes is a minimal-time tree. For constant cuts and converging separation time, the bound tends to 1 as Z grows (Theorem 17). We revisit this factor via simulation in Section 5.4.

4.2 Asymptotic Optimality of Root-Cut Trees

Proposition 14 bounds how far the best root-cut tree can be from a minimal-time tree for a fixed target bound. We now ask how this comparison behaves as the target bound grows. Le Bodic and Nemhauser [18] study the asymptotic growth of the size of a minimal B&B tree, as the target bound Z tends to infinity. Jiang [14] extends this analysis to the abstract B&C model, focusing on size and the proportion of the bound proved by cut nodes. We further generalize these results from tree size to tree time, including the time to generate cuts, and show that as the target bound increases, the best root-cut tree grows at the same rate as a minimal-time tree (Lemma 15) and becomes asymptotically optimal for constant cuts (Theorem 17).

Throughout this subsection, we assume that w and r are both bounded from above. The sequence $\{w(i)\}$ is then convergent, as it is nondecreasing and upper bounded. Define $W_{\max} := \lim_{i \rightarrow \infty} w(i) \geq 1$ and $S_{\max} := \sup_{h \in \mathbb{Z}_{\geq 0}} s(h) \geq 0$. Let $\mu^* > 0$ be the unique solution of $e^{-\mu^* \ell} + e^{-\mu^* r} = 1$, which governs the exponential growth rate of pure branching trees [18, 14]. Jiang [14, Theorem 4.3] shows that minimal-size trees grow at a rate determined by how C_k grows relative to $\log k$. Lemma 15 shows that, under bounded node and separation times, the same holds for the best root-cut tree and for a minimal-time tree. A technical point is that we allow the limit γ in Lemma 15 to be ∞ , i.e., $C_k / \log k \rightarrow \infty$.

Lemma 15. *For every target bound Z ,*

$$t^*(Z) \leq \tau_w^*(Z) \leq \tau_{\text{root}}^*(Z) \leq (W_{\max} + S_{\max}) t^*(Z).$$

Further, if the limit $\gamma := \lim_{k \rightarrow \infty} C_k / \log k$ exists in $[0, \infty]$, then

$$\lim_{Z \rightarrow \infty} \frac{\log(\tau_{\text{root}}^*(Z))}{Z} = \lim_{Z \rightarrow \infty} \frac{\log(\tau_w^*(Z))}{Z} = \lim_{Z \rightarrow \infty} \frac{\log(t^*(Z))}{Z} = \frac{\mu^*}{1 + \gamma \mu^*},$$

where the right-hand side of the last equality is taken as 0 when $\gamma = \infty$.

Proof. Since every node time is at least 1, the tree time of any tree is at least its size, so $t^*(Z) \leq \tau_w^*(Z)$, and $\tau_w^*(Z) \leq \tau_{\text{root}}^*(Z)$ holds as root-cut trees are a subset of all trees. Let T be a minimal-size tree proving Z , and let k be the maximum number of cut nodes on a root-to-leaf path of T . By Lemma 13, the root-cut tree consisting of k cut nodes followed by $T(Z - C_k)$, which proves Z , has size at most $|T|$, so it is also a minimal-size tree (as in [16, Lemma 7] and [14, Lemma 4.1]). As every node time is at most $W_{\max} + S_{\max}$, its tree time is at most $(W_{\max} + S_{\max}) t^*(Z)$, which proves the last inequality. When γ exists, the limit of $\log(t^*(Z))/Z$ exists and equals $\mu^*/(1 + \gamma \mu^*)$ by [14, Theorem 4.3]. Taking logarithms in the first claim, dividing by Z , and using $\log(W_{\max} + S_{\max})/Z \rightarrow 0$ completes the proof. $\square$

Moreover, by Proposition 14, the best root-cut tree is always within a constant factor of a minimal-time tree: $\tau_{\text{root}}^*(Z) \leq w(k^*) \tau_w^*(Z) \leq W_{\max} \tau_w^*(Z)$, even when the separation time is unbounded. When γ exists, the best root-cut tree, a minimal-time tree, and a minimal-size tree have the same exponential growth rate in the target bound. This lemma is particularly informative when tree size grows exponentially, i.e., when $\gamma < \infty$, such as for harmonically worsening cuts $c(h) = c_0/h$ ($\gamma = c_0$) or when $\sum_{h \geq 1} c(h) < \infty$ ($\gamma = 0$). In contrast, tree size grows linearly under a constant cut function $c > 0$, and the minimal tree size satisfies $t^*(Z) \leq \lceil Z/c \rceil + 1$, so the limits in Lemma 15 equal 0 (this is captured by $\gamma = \infty$). In this case, Theorem 17 gives a sharper result: the best root-cut tree is asymptotically optimal, and its root cuts prove asymptotically all of the target bound.

The key ingredient is the pure cutting tree, the root-cut tree with no branch nodes. For a constant cut function $c > 0$ and a target bound Z , let $K_Z := \lceil Z/c \rceil$, let $T_{\text{cut}}(Z)$ denote the pure cutting tree proving Z , consisting of K_Z cut nodes followed by a leaf, and let

$$\tau_{\text{cut}}(Z) := \tau_w(T_{\text{cut}}(Z)) = \pi(K_Z)$$

denote its tree time, which we compute asymptotically in Lemma 16. For Lemma 16 and Theorem 17, we also assume that the separation time converges, $s(h) \rightarrow s_{\infty}$ as $h \rightarrow \infty$. This holds, e.g., for a constant separation time $s \equiv \sigma$, with $s_{\infty} = \sigma$, or for no separation time, with $s_{\infty} = 0$.

Lemma 16. *Suppose that $s(h) \rightarrow s_{\infty}$ as $h \rightarrow \infty$. Then π is strictly increasing and $\lim_{k \rightarrow \infty} \pi(k)/k = W_{\max} + s_{\infty}$. Hence, for a constant cut function $c > 0$,*

$$\lim_{Z \rightarrow \infty} \frac{\tau_{\text{cut}}(Z)}{Z} = \frac{W_{\max} + s_{\infty}}{c}.$$

Proof. For every k , $\pi(k+1) - \pi(k) = s(k) + w(k+1) \geq 1$. Since $w(i) + s(i) \rightarrow W_{\max} + s_{\infty}$, the averages $\frac{1}{k} \sum_{i=0}^{k-1} (w(i) + s(i))$ converge to the same limit, and $w(k)/k \leq W_{\max}/k \rightarrow 0$, so $\pi(k)/k \rightarrow W_{\max} + s_{\infty}$. The last claim follows from $\tau_{\text{cut}}(Z) = \pi(K_Z)$ and $K_Z/Z \rightarrow 1/c$. $\square$

Theorem 17. *Let $c > 0$ be a constant cut function, and suppose that $s(h) \rightarrow s_{\infty}$ as $h \rightarrow \infty$. Then*

$$\lim_{Z \rightarrow \infty} \frac{\tau_{\text{root}}^*(Z)}{Z} = \lim_{Z \rightarrow \infty} \frac{\tau_w^*(Z)}{Z} = \frac{W_{\max} + s_{\infty}}{c}, \quad \text{so} \quad \lim_{Z \rightarrow \infty} \frac{\tau_{\text{root}}^*(Z)}{\tau_w^*(Z)} = 1.$$

Moreover, $Z - ck_{\text{root}}(Z) \in O(\log Z)$ and, for every choice of minimal-time tree T^* proving Z , $Z - ck^* \in O(\log Z)$. In particular, the root cuts of a best root-cut tree, and the cuts on some root-to-leaf path of a minimal-time tree, prove a fraction of the target bound that tends to 1.

Proof. Let T^* be any minimal-time tree proving Z , with k^* the maximum number of cut nodes on a root-to-leaf path of T^* . Applying Lemma 13 to T^* , and since $T_{\text{cut}}(Z)$ is a root-cut tree,

$$\pi(k^*) + B^* = \pi(k^*) + p(Z - ck^*) - 1 \leq \tau_w^*(Z) \leq \tau_{\text{root}}^*(Z) \leq \pi(K_Z). \quad (6)$$

As π is strictly increasing (Lemma 16) and $p(z) \geq 1$ for all z , this implies $k^* \leq K_Z$. Every node of $T(z)$ at depth less than z/r has bound less than z , so $T(z)$ contains a complete binary tree of depth $\lceil z/r \rceil$, and $p(z) \geq 2^{z/r}$. Hence, by (6) and $\pi(K_Z) \leq (W_{\max} + S_{\max})K_Z + W_{\max}$, we have $Z - ck^* \leq r \log_2 \pi(K_Z) \in O(\log Z)$. Similarly, a best root-cut tree has time $\pi(k_{\text{root}}) + w(k_{\text{root}})(p(Z - ck_{\text{root}}) - 1) \leq \pi(K_Z)$, so $k_{\text{root}} \leq K_Z$ (as $w \geq 1$) and $Z - ck_{\text{root}} \in O(\log Z)$.

Therefore, $Z - O(\log Z) \leq ck^* \leq cK_Z < Z + c$, so, as $Z \rightarrow \infty$, we have $k^* \rightarrow \infty$ (from the lower bound) and $ck^*/Z \rightarrow 1$. Since $k^* \rightarrow \infty$, Lemma 16 gives $\pi(k^*)/k^* \rightarrow W_{\max} + s_{\infty}$, so, as $Z \rightarrow \infty$,

$$\frac{\pi(k^*)}{Z} = \frac{\pi(k^*)}{k^*} \cdot \frac{k^*}{Z} \rightarrow \frac{W_{\max} + s_{\infty}}{c},$$

and likewise $\pi(K_Z)/Z \rightarrow (W_{\max} + s_{\infty})/c$. Dividing (6) by Z proves both limits, and hence that their ratio tends to 1. $\square$

Theorem 17 shows that, for constant cuts, restricting cuts to the root is asymptotically optimal and the root cuts prove all but a logarithmic portion of the target bound. For $w \equiv 1$, no separation time, and constant cuts ($\gamma = \infty$), Theorem 17 recovers [14, Corollary 4.4], which states that the fraction of the target bound proved by the root cuts of a minimal-size root-cut tree tends to 1, and extends it to every minimal-size tree, in that the cuts on some root-to-leaf path prove this fraction.

The proof of Theorem 17 implies that the factor $(\pi(k^*) + w(k^*)B^*)/(\pi(k^*) + B^*)$ from Proposition 14 tends to 1,³ whereas the factor $w(k^*)$ of [13, Theorem 11] tends to $W_{\max}$. We emphasize that using node times rather than tree sizes alone is what makes the bound in Proposition 14 asymptotically tight. Although larger separation time may cause fewer cut nodes for a fixed target bound (see Section 5.3), Theorem 17 shows that root cuts still asymptotically suffice for constant cuts.

5 Internal Cuts Can Be Necessary

In this section, we explore general asymmetric branching settings with a single constant-strength cut family in which internal cuts can be necessary in any minimal-time tree. This is the common practice in modern solvers, where cuts generated beyond the root node play a critical role in reducing solving time and improving performance [3].

³By (6), $B^* \leq \pi(K_Z) - \pi(k^*) \leq (W_{\max} + S_{\max})(K_Z - k^*) + W_{\max}$, which is in $O(\log Z)$, while $\pi(k^*)$ grows linearly.

5.1 Minimal-Time Trees Can Be Complex

Unlike the symmetric branching setting or the special case analyzed in [Section 3.3](#), where a minimal-time tree exists with cuts only in one of two locations (at the root or at a leaf), general asymmetric branching settings together with special choices of time functions can yield far more complex optimal structures, making prescribing best cut locations nontrivial. We construct an intuitive example in [Lemma 18](#) to illustrate the complexity of minimal-time trees under asymmetric branching settings.

Lemma 18. *Let $(c, \ell, r) = (2, 3, 4)$. For any depth $d \in \mathbb{Z}_{\geq 0}$, there exists a simple time function w such that any minimal-time tree proving bound $Z := 3d + 5$ will have a cut node at depth d along the leftmost root-to-leaf path.*

Proof. For $d \in \mathbb{Z}_{\geq 0}$, let $\bar{w}_d := 2 - 1/p(3d + 9)$, pick any $w(1) \in [\bar{w}_d, 2)$, and set $w(k) = M$ for all $k \geq 2$ with M sufficiently large that every minimal-time tree has at most one cut node on each root-to-leaf path. We proceed by induction on d . For $d = 0$ and $Z = 5$, a root cut followed by a branch node with two leaves has tree time $1 + 3w(1)$. As $w(1) < 2$, this is less than the time 7 of $T(5)$, the time $3 + 2w(1)$ of a branch node whose children are both leaf cuts, and the time $5 + w(1)$ of a branch node with one leaf-cut child, so every minimal-time tree has a cut node at its root.

Now suppose that the lemma holds for some $d \geq 0$ and every $w(1) \in [\bar{w}_d, 2)$, and consider $Z = 3d + 8$ and $w(1) \in [\bar{w}_{d+1}, 2)$. In a tree whose root is a cut node, the remaining nodes form a pure branching tree proving $Z - 2$, each with time $w(1)$, so its tree time is at least $1 + w(1)p(Z - 2) > 2p(Z - 2)$, as $w(1) \geq \bar{w}_{d+1} > 2 - 1/p(Z - 2)$. On the other hand, $p(Z) = 1 + p(Z - 3) + p(Z - 4) \leq 2p(Z - 2)$, as p is nondecreasing and $p(Z - 4) < p(Z - 2)$ (the leaf of $T(3d + 4)$ with bound $3d + 4$ is a branch node of $T(3d + 6)$). Hence, $T(Z)$ is faster than every tree with a root cut, so the root of every minimal-time tree is a branch node. By [\[16, Lemma 24\]](#), the left subtree of the root is a minimal-time tree proving $3d + 5$, and, as $w(1) \geq \bar{w}_{d+1} \geq \bar{w}_d$, the induction hypothesis gives a cut node at depth d on its leftmost path, i.e., at depth $d + 1$ on the leftmost path of the whole tree. $\square$

In the asymmetric setting of [Lemma 18](#), minimal-time trees have arbitrarily deep internal cuts. However, in the construction, the internal cut is always two layers above the bottom of the tree. In other words, while the depth is arbitrary in this setting, the placement of cut nodes is not.

5.2 Adding At Most One Cut Node Per Root-To-Leaf Path

Typically, solvers limit the number of cut rounds, as successive rounds often yield diminishing bound improvements [\[7\]](#) while substantially increasing problem size and solution time. Moreover, excessive cuts can harm numerical stability and lead to ill-conditioned subproblems that degrade solver performance [\[22, 20\]](#). In this subsection, we investigate necessary and sufficient conditions under which some minimal-time tree has an internal cut. We restrict our focus to the *one-cut* setting, defined as a configuration where at most one round of cuts is added in each root-leaf path. This assumption simplifies the analysis while preserving key structural insights.

Note that if $w(1) > 2$ under the one-cut setting, no cut nodes would appear in any minimal-time tree, as branch nodes provide both better bound improvements and smaller processing time when $w(1)$ is large. Meanwhile, when $w(1) = 1$, there always exists a minimal-time root-cut tree. Accordingly, we focus our analysis on $w(1) \in (1, 2)$. Under this assumption, a branch node in a minimal-time tree cannot have two cut node children.

Other than the time function, the relationship among Z , c , ℓ , and r also determines whether cut nodes will help. For instance, if $c = 1$ and all other parameters are even numbers, cut nodes would not appear in any minimal-time tree, for any value of $w(1)$. In [Lemma 19](#), we show that cuts

are strictly beneficial if and only if the pure branching tree has a node with at least 1 and at most c bound left to prove.

Lemma 19. *Given a one-cut setting $(Z, c, \ell, r) \in \mathbb{Z}_{>0}^4$ and $w(1) \in (1, 2)$, every minimal-time tree proving Z has a cut node if and only if there exist $k_1, k_2 \in \mathbb{Z}_{\geq 0}$ such that $k_1\ell + k_2r \in \{Z - c, \dots, Z - 1\}$.*

Proof. Suppose first that such a pair (k_1, k_2) exists, and let u be the node of $T(Z)$ reached from the root by k_1 left and k_2 right branches. This node exists and is a branch node, as its bound $k_1\ell + k_2r$ and those of its ancestors are less than Z , and its children are leaves, as $k_1\ell + k_2r + \ell \geq Z - c + \ell \geq Z$. Replacing u and its two children by a cut node followed by a leaf yields a tree that still proves Z , as $k_1\ell + k_2r + c \geq Z$, and whose tree time is smaller by $3 - (1 + w(1)) > 0$. As $T(Z)$ is the smallest pure branching tree, no pure branching tree is minimal-time.

Conversely, if no such pair exists, let v be a cut node of a tree T proving Z . Any leaf below v is reached from the root by some k_1 left and k_2 right branches, and, as v is the only cut on its path, its bound is $k_1\ell + k_2r + c \geq Z$. Hence, $k_1\ell + k_2r \geq Z - c$, so $k_1\ell + k_2r \geq Z$ by assumption, as all parameters are integers. Thus, removing v (connecting its child to its parent) yields a strictly faster tree that still proves Z , so T is not minimal-time. $\square$

When the condition of Lemma 19 holds, two trees serve as extreme cases. The *one-root-cut tree* is a cut node followed by $T(Z - c)$, with tree time $1 + p(Z - c)w(1)$. The *minimal leaf-cut tree* branches as $T(Z - c)$ and adds a cut above each of its $b_{\text{leaf}}(Z)$ leaves with remaining bound in $[c]$, with tree time $p(Z - c) + b_{\text{leaf}}(Z)w(1)$. As $b_{\text{leaf}}(Z) < p(Z - c)$, the first is faster for small $w(1)$ and the second for large $w(1)$. We write b_{leaf} for $b_{\text{leaf}}(Z)$ when the target bound is clear.

We next prove that Lemma 19 constrains the *relative* placement of cuts in sibling subtrees. Concretely, in a one-cut setting, if one child of a branch-then-cut node begins with an internal cut (a one-root-cut subtree), the other child will also contain a node where a cut saves time — so the sibling cannot be cut-free. Lemma 20 formalizes this observation.

Lemma 20. *Given a one-cut setting $(Z, c, \ell, r) \in \mathbb{Z}_{>0}^4$ and $w(1) \in (1, 2)$, a minimal-time tree cannot contain a branch-then-cut node where one child is a pure branching subtree.*

Proof. Suppose, for the sake of contradiction, that a minimal-time tree T contains a branch-then-cut node u whose one child is an internal cut node while the other is a branch node followed by a pure branching subtree. Without loss of generality, assume the left branch T_ℓ (rooted at the left child of u) follows a cut-then-branch framework while the right branch T_r (rooted at the right child of u) is a pure branching subtree. Let $\bar{Z} := Z - z_T(u) - \ell$. As there is no cut above u in the one-cut setting, [16, Lemma 24] implies that T_ℓ and T_r are minimal-time with respect to the same w , for target bounds $\bar{Z}$ and $\bar{Z} + \ell - r$, respectively. In particular, T_ℓ is a cut node followed by $T(\bar{Z} - c)$, and $\bar{Z} > c$ as this cut is internal. By Lemma 19, the existence of the cut node in T_ℓ implies that there exists a pair (k_1, k_2) satisfying $k_1\ell + k_2r \in \{\bar{Z} - c, \dots, \bar{Z} - 1\}$.

We first claim that some such pair has $k_2 \geq 1$. Otherwise, as $c \leq \ell$, exactly one node x of $T(\bar{Z})$ has remaining bound in $\{1, \dots, c\}$, namely the node on its leftmost path with bound $k_1\ell$, and x is not the root as $\bar{Z} > c$. Thus, $T(\bar{Z} - c)$ has exactly one fewer branch node than $T(\bar{Z})$, so $p(\bar{Z} - c) = p(\bar{Z}) - 2$, and $\tau_w(T_\ell) = 1 + w(1)(p(\bar{Z}) - 2)$. Replacing the branching at x in $T(\bar{Z})$ by a cut node followed by a leaf instead yields a tree proving $\bar{Z}$ with tree time $p(\bar{Z}) - 2 + w(1)$, which is smaller than $\tau_w(T_\ell)$ by $(w(1) - 1)(p(\bar{Z}) - 3) > 0$, as $p(\bar{Z}) \geq 5$, contradicting the minimality of T_ℓ .

Thus, some pair has $k_2 \geq 1$, and the node of T_r reached from its root by $k'_1 = k_1 + 1$ left branches and $k'_2 = k_2 - 1$ right branches is a branch node whose children are both leaves, as $k'_1\ell + k'_2r = k_1\ell + k_2r + \ell - r \in \{\bar{Z} + \ell - r - c, \dots, \bar{Z} + \ell - r - 1\}$. Replacing this branch node and its two leaves in T_r by a cut node followed by a leaf yields a tree T'_r that proves the same bound, with

$\tau_w(T'_r) - \tau_w(T_r) = 1 + w(1) - 3 < 0$ as $w(1) < 2$, contradicting the minimality of T_r . The case in which T_r follows the cut-then-branch framework and T_ℓ is a pure branching tree is analogous, with the roles of left and right branches exchanged (using $c \leq r$ and the rightmost path). $\square$

Specifically, [Lemma 20](#) implies that, if u is a deepest branch-then-cut node in a minimal-time tree in a one-cut setting, then one child subtree starts with a root cut and the other must have leaf cuts. Both children have remaining bound greater than c : the internal cut is followed by a branch node, and if the other child had remaining bound at most c , replacing its subtree by a cut node then a leaf would be faster. Hence, the remaining bound at u is greater than $r + c$.

We now turn our attention to characterizing the structure of a minimal-time tree and, subsequently, the placement of the cut nodes. [Lemma 21](#) characterizes how the structure of a minimal-time tree evolves as $w(1)$ grows.

Lemma 21. *In a one-cut setting $(Z, c, \ell, r) \in \mathbb{Z}_{>0}^4$ satisfying the condition in [Lemma 19](#), the values of $w(1) \in (1, 2)$ for which some minimal-time tree is the one-root-cut tree, has an internal cut, or is a leaf-cut tree form three (possibly empty) intervals that appear in this order and overlap only at their endpoints. In particular, as $w(1)$ increases, the minimal-time structure can only change from one-root-cut to internal-cut to leaf-cut, possibly skipping a stage.*

Proof. Let T be a tree that is minimal-time for some $w(1) \in (1, 2)$. By [Lemma 19](#), T has a cut node, and its tree time is $a + bw(1)$, where b is the number of nodes below a cut and $a \geq 1$ is the number of other nodes. The time of T is at most the time of the one-root-cut tree, so $b \leq p(Z - c)$, with equality only if $a = 1$, i.e., T is the one-root-cut tree.

In T , a node with no cut above it and remaining bound in $\{1, \dots, c\}$ is a cut node followed by a leaf, as otherwise it is a branch node with two leaves, and replacing these three nodes by a cut node followed by a leaf is faster when $w(1) < 2$. For each of the b_{leaf} leaves of $T(Z - c)$ with remaining bound in $\{1, \dots, c\}$, follow the same branches from the root of T , skipping over any cut node; each node passed has remaining bound greater than c , or greater than 0 if a cut was passed, so none is a leaf. If a cut was passed, the node reached is a leaf below a cut; otherwise, it is a cut node followed by a leaf. This gives b_{leaf} distinct leaves below cuts, so $b \geq b_{\text{leaf}}$. If T has an internal cut, its child is a further node below a cut, so $b > b_{\text{leaf}}$. Otherwise, every cut node of T is followed by a leaf, so no node with remaining bound greater than c is a cut node, and T branches exactly at these nodes; thus, T is the minimal leaf-cut tree, with $b = b_{\text{leaf}}$.

Hence, the one-root-cut tree has slope $p(Z - c)$, the minimal leaf-cut tree has slope b_{leaf} , and every other tree that is minimal-time for some $w(1) \in (1, 2)$ has an internal cut and slope strictly in between. If trees with times $a_1 + b_1 w(1)$ and $a_2 + b_2 w(1)$ are minimal-time at $\omega_1 < \omega_2$, respectively, then adding $a_1 + b_1 \omega_1 \leq a_2 + b_2 \omega_1$ and $a_2 + b_2 \omega_2 \leq a_1 + b_1 \omega_2$ gives $(b_2 - b_1)(\omega_2 - \omega_1) \leq 0$, so the slope of a minimal-time tree is nonincreasing in $w(1)$. Thus, the one-root-cut tree is minimal-time exactly on an initial interval of $(1, 2)$, the minimal leaf-cut tree exactly on a final interval, and trees with internal cuts only in between. $\square$

Let $\alpha(Z)$ be the value of $w(1)$ at which the one-root-cut and minimal leaf-cut trees proving Z require the same processing time (for fixed (c, ℓ, r)). Concretely,

$$\alpha(Z) := \frac{p(Z - c) - 1}{p(Z - c) - b_{\text{leaf}}(Z)}.$$

For example, with $(c, \ell, r) = (2, 3, 4)$, $\alpha(5) = 2$ in the base case of [Lemma 18](#), where the one-root-cut tree is minimal-time for all $w(1) \in (1, 2)$. [Lemma 21](#) suggests that this threshold plays a crucial role in determining the architecture of the minimal-time tree, which [Corollary 22](#) makes precise.

Corollary 22. *Given a one-cut setting $(Z, c, \ell, r) \in \mathbb{Z}_{\geq 0}^4$ with $\alpha(Z) \in (1, 2)$, if some minimal-time tree proving Z has an internal cut for some $w(1) \in (1, 2)$, then so does some minimal-time tree proving Z for $w(1) = \alpha(Z)$.*

Proof. Let $\alpha := \alpha(Z)$. Define $R(\omega) := 1 + p(Z - c)\omega$ and $L(\omega) := p(Z - c) + b_{\text{leaf}}\omega$ for the tree times of the one-root-cut and minimal leaf-cut trees at $w(1) = \omega$, so that $R(\alpha) = L(\alpha)$, and let $I(\omega) = a + b\omega$ be the tree time of an internal-cut tree that is minimal-time at $w(1) = \omega_0 \in (1, 2)$, so $I(\omega_0) \leq \min\{R(\omega_0), L(\omega_0)\}$. By the proof of Lemma 21, $b_{\text{leaf}} < b < p(Z - c)$, so $I - R$ is decreasing and $I - L$ is increasing. If $\alpha \geq \omega_0$, then $I(\alpha) - R(\alpha) \leq I(\omega_0) - R(\omega_0) \leq 0$, and otherwise, $I(\alpha) - L(\alpha) \leq I(\omega_0) - L(\omega_0) \leq 0$. In both cases, $I(\alpha) \leq R(\alpha) = L(\alpha)$. Every one-root-cut tree has time at least $R(\alpha)$, every leaf-cut tree has time at least $L(\alpha)$ (as $\alpha < 2$), and no pure branching tree is minimal-time by Lemma 19, so at $w(1) = \alpha$, either this internal-cut tree is minimal-time, or a faster tree exists, which then has an internal cut. $\square$

Together, Lemma 21 and Corollary 22 give the following picture. If, for different values of $w(1) \in (1, 2)$, there are minimal-time trees of all three types (one-root-cut, internal-cut, and leaf-cut), then there exist $\omega_1 \leq \alpha(Z) \leq \omega_2$ such that a minimal-time one-root-cut tree exists for $w(1) \in (1, \omega_1]$, a minimal-time internal-cut tree exists for $w(1) \in [\omega_1, \omega_2]$, and a minimal-time leaf-cut tree exists for $w(1) \in [\omega_2, 2)$.

Finally, by combining Lemmas 19 to 21 and Corollary 22, we can determine whether some value of $w(1)$ yields a minimal-time tree with internal cuts.

Theorem 23. *Given a one-cut setting $(Z, c, \ell, r) \in \mathbb{Z}_{\geq 0}^4$ with $\alpha(Z) \in (1, 2)$, there exists $w(1) \in (1, 2)$ for which some minimal-time tree proving Z has an internal cut if and only if there are $k_1, k_2 \in \mathbb{Z}_{\geq 0}$ with $Z' := Z - k_1\ell - k_2r > r + c$ such that*

$$\alpha(Z) \leq \max\{\alpha(Z' - \ell), \alpha(Z' - r)\}.$$

In this case, some minimal-time tree proving Z has an internal cut for $w(1) = \alpha(Z)$.

Proof. For convenience, define $\alpha := \alpha(Z)$ and, for the value of Z' under consideration, $\alpha_\beta := \alpha(Z' - \beta)$ for $\beta \in \{\ell, r\}$. Suppose first that some minimal-time tree proving Z has an internal cut for some $w(1) \in (1, 2)$. By Corollary 22, some minimal-time tree T for $w(1) = \alpha$ has an internal cut, so T has a branch-then-cut node. Let u be a deepest branch-then-cut node of T , reached from the root by k_1 left and k_2 right branches. As u has a cut below it, there is no cut above it, so its remaining bound is $Z' = Z - k_1\ell - k_2r$, and $Z' > r + c$ by the discussion after Lemma 20. Let $\beta \in \{\ell, r\}$ be such that the child of u with remaining bound $Z' - \beta$ is an internal cut node. By [16, Lemma 24], the subtree rooted at this child is a minimal-time tree proving $Z' - \beta$, and, as it is a cut node followed by a pure branching tree, it is the one-root-cut tree proving $Z' - \beta$. By minimality, the time of this subtree is at most the time of the minimal leaf-cut tree proving $Z' - \beta$, showing that $\alpha \leq \alpha_\beta \leq \max\{\alpha_\ell, \alpha_r\}$.

Conversely, suppose that such k_1 and k_2 exist, and let $w(1) = \alpha$. As the one-root-cut and minimal leaf-cut trees then have the same tree time, some minimal-time tree T has a branch node at its root (the minimal leaf-cut tree, if the one-root-cut tree is minimal-time). Follow k_1 left and k_2 right branches from the root of T , passing through any cut node, to reach a node u . Each node on the way has remaining bound at least $Z' > r + c$, or greater than r if a cut was passed, so none is a leaf. If one of these nodes, including u , is a cut node, then it is not the root, and its child is a branch node, as it has positive remaining bound and is not a second cut on the path; thus, T has an internal cut. Otherwise, u is a branch node with no cut above it and remaining bound Z' . Let $\beta \in \{\ell, r\}$ with $\alpha \leq \alpha_\beta$, and let T_β be the subtree rooted at the child of u with remaining bound

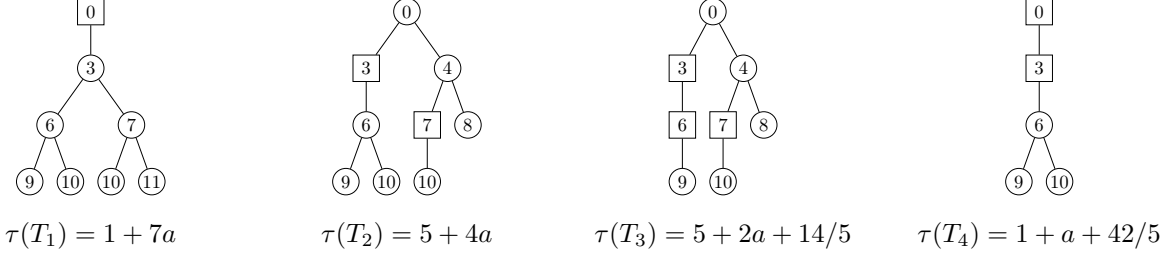


Figure 9: Four trees from [Example 3](#) proving target bound $Z = 8$ under the setting $(c, \ell, r) = (3, 3, 4)$ and $w = (1, a, 14/5, \infty, \dots)$. The tree time is shown below each tree, where $14/5 = w(2)$ for T_3 , and $42/5 = 3w(2)$ for T_4 .

$Z' - \beta$, which is a minimal-time tree proving $Z' - \beta$ by [\[16, Lemma 24\]](#). As $\alpha_\beta \geq \alpha > 1$, the minimal leaf-cut tree proving $Z' - \beta$ has more than one cut, so the condition of [Lemma 19](#) holds for $Z' - \beta$, and, by [Lemma 21](#) and its proof, T_β has an internal cut, is the one-root-cut tree, or is the minimal leaf-cut tree proving $Z' - \beta$. If T_β has an internal cut, so does T . If T_β is the one-root-cut tree, its cut is an internal cut of T , as it is followed by $T(Z' - \beta - c)$, whose root is a branch node as $Z' - \beta - c > r - \beta \geq 0$. If T_β is the minimal leaf-cut tree, then, its time is at most that of the one-root-cut tree T'_β proving $Z' - \beta$, so $\alpha \geq \alpha_\beta$. Hence, $\alpha = \alpha_\beta$ and T'_β has the same tree time as T_β . Replacing T_β by T'_β yields a minimal-time tree in which, as before, the cut at the child of u is internal. Thus, for $w(1) = \alpha$, some minimal-time tree proving Z has an internal cut. $\square$

In [Theorem 23](#), Z' is the remaining bound at the node reached from the root by k_1 left and k_2 right branches, and α_ℓ and α_r are the values of $w(1)$ at which the one-root-cut and minimal leaf-cut trees for its two children have the same tree time. The condition says that, at $w(1) = \alpha(Z)$, the one-root-cut tree has time that is no worse than that of the minimal leaf-cut tree for one of these children. The forward direction also gives a lower bound: if u is a deepest branch-then-cut node of a minimal-time tree with respect to $w(1) = \alpha(Z)$, then its child that is not a cut node has a subtree with a cut (by [Lemma 20](#)) but no internal cut, so, by the proof of [Lemma 21](#), this subtree is the minimal leaf-cut tree for its target bound, and the same comparison shows that $\alpha(Z)$ is at least the corresponding threshold. Hence, for such k_1 and k_2 , we have $\min\{\alpha_\ell, \alpha_r\} \leq \alpha(Z) \leq \max\{\alpha_\ell, \alpha_r\}$.

Lastly, we end this subsection with an example showing that similar monotonicity of structure does not follow when more than one cut per root-to-leaf path is allowed.

Example 3. Consider a single constant cut family with $c = \ell = 3$, $r = 4$, target bound $Z = 8$, and $w = (1, a, 14/5, \infty, \dots)$ with $a \in [1, 14/5]$, in which at most two cuts are used on any root-to-leaf path. [Figure 9](#) depicts four trees proving Z . Tree T_1 has one root cut. Tree T_2 has an internal cut in the left subtree and a leaf cut in the right subtree. Tree T_3 has three leaf cuts. Tree T_4 has two root cuts. As $a = w(1)$ increases, the unique minimal-time tree goes from T_1 to T_2 at $a = 4/3$, then T_3 at $a = 7/5$, and finally T_4 at $a = 8/5$. Thus, in contrast to the one-cut setting, cuts first move from the root toward the leaves and then back to the root as $w(1)$ increases.

A curious consequence of [Example 3](#) is that increasing $w(1)$ can lead to a minimal-time tree having strictly more root cuts. The underlying reason is that, in our model, cuts affect the time of all nodes below them, so increasing $w(1)$ penalizes having many nodes with exactly one cut above them. The possible remedies are to either move cuts down the tree or to add cut nodes (to reduce overall tree size), and the latter becomes increasingly attractive as $w(1)$ approaches $w(2)$.

5.3 Effect of Cut Generation Time

[Theorems 5](#) and [6](#) hold for generalized time functions, so cut generation time does not affect the sufficiency of root cuts under symmetric branching or the best way to order consecutive cut nodes. However, it may affect the number of cut nodes in a minimal-time tree and, under asymmetric branching, where they are placed.

[Proposition 24](#) proves the intuitive fact that, as a cut family $i \in [m]$ with constant separation time becomes increasingly costly to generate, minimal-time trees proving the same target bound use weakly fewer cut nodes from that family, and the remaining tree time, excluding the time to generate these cuts, weakly increases. For any tree T and family $i \in [m]$, let $n_i(T)$ denote the number of cut nodes from family i in T , and define the tree time without the effect of generating cuts from family i as $\tau_{-i}(T) := \tau_{\tilde{w}}(T) - \sigma_i n_i(T)$, where σ_i is the separation time of family i under $\tilde{w}$. Since each cut node of family i contributes exactly σ_i to $\tau_{\tilde{w}}(T)$, $\tau_{-i}(T)$ does not depend on σ_i ; that is, $\tau_{\tilde{w}}(T) = \tau_{-i}(T) + \sigma_i n_i(T)$ for every value of σ_i .

Proposition 24. *Fix $i \in [m]$ as a cut family with constant separation time. For $\sigma \geq 0$, let $\tilde{w}_\sigma$ denote a generalized time function in which family i has separation time σ , with all other components fixed. Let $\sigma_i < \sigma'_i$, and let T and T' be minimal-time trees proving Z with respect to $\tilde{w}_{\sigma_i}$ and $\tilde{w}_{\sigma'_i}$, respectively. Then $n_i(T') \leq n_i(T)$ and $\tau_{-i}(T') \geq \tau_{-i}(T)$. Moreover, if $n_i(T') = n_i(T)$, then $\tau_{-i}(T') = \tau_{-i}(T)$, and both T and T' are minimal-time trees with respect to $\tilde{w}_\sigma$ for every $\sigma \in [\sigma_i, \sigma'_i]$.*

Proof. By minimality, $\tau_{\tilde{w}_{\sigma_i}}(T) \leq \tau_{\tilde{w}_{\sigma_i}}(T')$ and $\tau_{\tilde{w}_{\sigma'_i}}(T') \leq \tau_{\tilde{w}_{\sigma'_i}}(T)$, i.e.,

$$\begin{aligned} \tau_{-i}(T) + \sigma_i n_i(T) &\leq \tau_{-i}(T') + \sigma_i n_i(T') \quad \text{and} \\ \tau_{-i}(T') + \sigma'_i n_i(T') &\leq \tau_{-i}(T) + \sigma'_i n_i(T). \end{aligned}$$

Adding these yields $(\sigma'_i - \sigma_i)(n_i(T') - n_i(T)) \leq 0$, so $n_i(T') \leq n_i(T)$, and the first inequality then gives $\tau_{-i}(T) - \tau_{-i}(T') \leq \sigma_i(n_i(T') - n_i(T)) \leq 0$.

Now suppose that $n_i(T) = n_i(T') = n$. The two inequalities become $\tau_{-i}(T) \leq \tau_{-i}(T')$ and $\tau_{-i}(T') \leq \tau_{-i}(T)$, so $\tau_{-i}(T) = \tau_{-i}(T')$, and hence $\tau_{\tilde{w}_\sigma}(T) = \tau_{\tilde{w}_\sigma}(T')$ for all σ . As T is minimal-time for σ_i and T' for σ'_i , both are minimal-time for σ_i and for σ'_i . Finally, let $\sigma \in (\sigma_i, \sigma'_i)$ and let S be a minimal-time tree for σ . Applying the first part of the proposition to the tree pairs (T, S) and (S, T') gives $n \geq n_i(S) \geq n$, so $n_i(S) = n$. The argument above applied to (T, S) gives $\tau_{-i}(T) = \tau_{-i}(S)$. Hence, $\tau_{\tilde{w}_\sigma}(T) = \tau_{\tilde{w}_\sigma}(S) = \tau_{\tilde{w}_\sigma}(T')$, so T and T' are minimal-time for σ . $\square$

We next consider where cuts are placed when an increase in separation time strictly decreases the number of cut nodes. One might expect cuts to be pushed towards the root, but this does not hold in general: using fewer cut nodes can change where cuts are placed in either direction. A single root cut may replace several local cuts ([Example 4](#)), but root cuts may also be replaced by fewer, deeper cuts ([Example 5](#)). We write $w(k) = M$ for a value M large enough that using k or more cuts is never optimal.

Example 4. Let $(c, \ell, r) = (1, 2, 4)$, $Z = 5$, $w = (1, 1.4, M, \dots)$, for a single constant-strength cut family with constant separation time $s_1 \equiv \sigma$. Tree T_0 in the left panel of [Figure 10](#) has two leaf cuts and tree time $5 + 2w(1) + 2\sigma = 7.8 + 2\sigma$, while tree T_1 in the right panel has one root cut and tree time $1 + 5w(1) + \sigma = 8 + \sigma$. The unique minimal-time tree is T_0 when $\sigma < 1/5$ and T_1 when $1/5 < \sigma < 1$. For $\sigma > 1$, the pure branching tree $(T(5))$, with 9 nodes) is optimal.

Example 5. Let $(c, \ell, r) = (3, 3, 7)$, $Z = 12$, $w = (1, 1.4, 1.9, 2, 3.4, M, \dots)$, for a single constant-strength cut family with constant separation time $s_1 \equiv \sigma$. The pure cutting tree, with four cuts,



Figure 10: The trees T_0 (left) and T_1 (right) from [Example 4](#). With separation time σ per cut node, the minimal-time tree switches from T_0 to T_1 at $\sigma = 1/5$.

has time $9.7 + 4\sigma$. The tree with two root cuts followed by a branch node, whose left child is a leaf cut and right child is a leaf, has time $10.1 + 3\sigma$. The unique minimal-time tree is the former when $\sigma < 2/5$, the latter when $2/5 < \sigma < 49/30$, and pure branching for larger σ , so increasing separation time replaces two root cuts by a single leaf cut.

[Proposition 24](#) requires family i to have constant separation time. When s_i depends on the number of prior rounds, two trees with the same cut counts can have different separation times, and, in contrast to the second part of [Proposition 24](#), the minimal-time tree can change in structure even when the number of cut nodes does not.

5.4 Empirical Evaluation of the Root-Cut Approximation

In this section, we empirically assess (via simulations in our abstract model, using the one-cut setting of [Section 5.2](#)) the strength of a root-cut tree and the tightness of the approximation bound $\frac{\tau_{\text{root}}^*(Z)}{\tau_w^*(Z)}$ in [Proposition 14](#), where $\tau_{\text{root}}^*(Z)$ represents the tree time of the best root-cut tree and $\tau_w^*(Z)$ represents the tree time of a minimal-time tree. In the one-cut setting, the best root-cut tree has either no cut or a single root cut, so $\tau_{\text{root}}^*(Z) = \min\{p(Z), 1 + w(1)p(Z - c)\}$, and we compute $\tau_w^*(Z)$ exactly with the recursion in [Section A.1](#). We evaluate $w(1) \in [1, 2]$ in increments of $1/50$, as well as the value of $w(1)$ at which the ratio peaks, discussed below.⁴

We test settings with $Z = k_1\ell + k_2r + c$ for some $k_1, k_2 \in \mathbb{Z}_{\geq 0}$ with $k_1 + k_2 \geq 1$, so the condition of [Lemma 19](#) holds. We refer to $\lceil Z/\ell \rceil$, the depth of $T(Z)$, as the depth of a setting. We consider three groups of settings, comprising over 55 million parameter combinations: (A) all $c \leq \ell \leq r$ in $[50]$ with $k_1, k_2 \in [0, 50]$; (B) the nearly symmetric settings $(c, 200, r)$ with $r \in \{201, 202\}$, $c \in [100]$, and $k_1, k_2 \in [0, 400]$; and (C) for each $d \in \mathcal{D} := \{5, 10, 50, 100, 500, 1000\}$, all settings (c, ℓ, r) with $\ell \in \{25, 50, 100, 200\}$, $r - \ell \in \{1, 2\}$, $c \leq \min\{\ell, 100\}$, and depth exactly d . We focus on nearly symmetric settings in (B) and (C) because, in (A), the largest ratio is attained with $r = \ell + 1$ and does not increase with $r - \ell$, while the ratio is always 1 when $\ell = r$.

[Figure 11](#) shows, for each $d \in \mathcal{D}$, the setting of depth d with the worst (largest) approximation ratio among all settings tested. Following [Lemma 21](#), the approximation ratio exhibits a stable-increase-decrease pattern as $w(1)$ gradually increases, maximized at $w(1) = \frac{p(Z)-1}{p(Z-c)}$, which is when the pure branching tree and the one-root-cut tree have the same tree time. [Figure 12](#) shows how the worst-case ratio grows with depth, from 1.141 at depth 5 to 1.191 at depth 100 and 1.202 at depth 1000. The bound of [Proposition 14](#) provides a conservative estimate of the actual ratio: at the peak, it is between 1.41 and 1.49 for the settings in [Figure 11](#). Hence, a root-cut tree may be a practically-effective (and simpler, computationally) alternative to a minimal-time tree in our model.

⁴Tree times are computed in exact integer arithmetic when they fit in 64 bits, and in double precision otherwise. Precision (not time) is what limits computation at large depths.

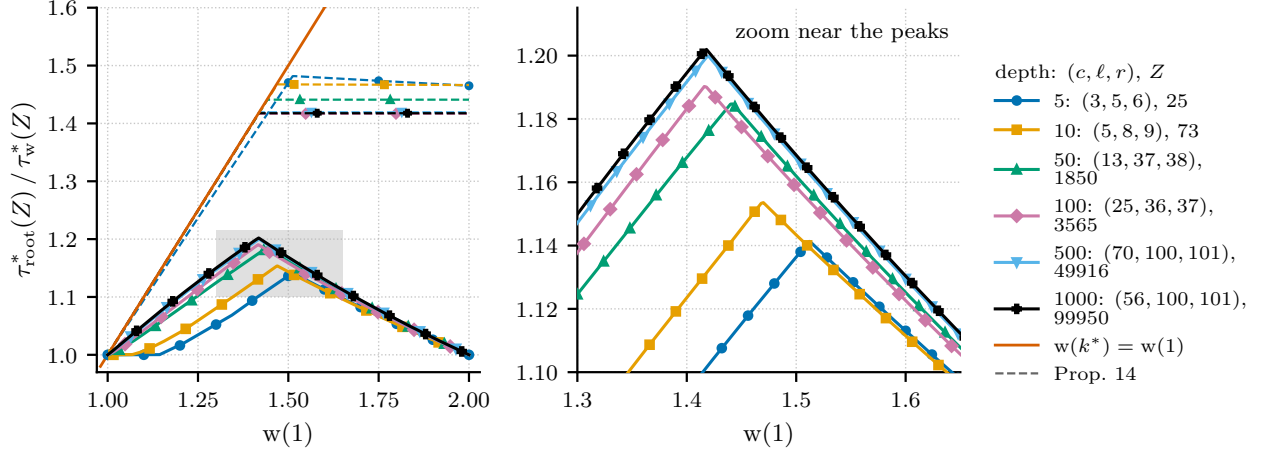


Figure 11: Setting at each depth with worst-case approximation ratio as $w(1)$ varies. The diagonal line is the bound $w(k^*)$, and the dashed lines represent the bound of Proposition 14.

Lastly, as $w(1)$ grows, by Lemma 21, the minimal-time structure shifts from root-cut to internal-cut to leaf-cut. Intuitively, cuts get pushed down in a minimal-time tree. *Does this shift progress one layer at a time?* The answer is *not always*. Empirically, the shallowest internal cut among all minimal-time trees may jump from the root to a much deeper layer as $w(1)$ grows, as depicted in Figure 13. Specifically, these plots show the depth of the shallowest internal cut over all minimal-time trees, for four settings and incrementing $w(1)$ by thousandths. A value of 0 indicates that no minimal-time tree has an internal cut, i.e., all cuts are at the root or at the leaves.

6 Conclusions

This paper highlights the structural importance of root cuts within our abstract B&C framework. In particular, we identify settings, both symmetric and asymmetric, in which some minimal-time tree has no internal cuts, even when accounting for cut generation time. More generally, we bound how much the best root-cut tree can deviate from a minimal-time tree, and we prove that, for constant cuts, the best root-cut tree is asymptotically optimal. In addition, we characterize when internal cuts are necessary in a one-cut setting, and we show that cut generation time can move cuts either toward or away from the root.

While this model captures a phenomenon commonly observed in state-of-the-art solvers, that cuts are used aggressively at the root, it intentionally abstracts away complex solver interactions. Specifically, cutting and branching interact solely through bound improvement and the slowdown in node processing time, leaving out other mechanisms such as pruning, restarts, primal heuristics, and the sequence of fractional solutions that a solver encounters. Consequently, preliminary computational experiments evaluating cut placement only exhibit noise-level differences in performance. This motivates generalizations of the current model. For example, as branching decisions can be significantly affected by cuts in practice, a natural extension is to incorporate cut-dependent branching decisions.

Acknowledgments. We appreciate the constructive feedback from IPCO reviewers that helped improve our paper. Based on initial results by the authors, Claude’s Fable 5.1 and Opus 5.5 models were used to check and improve proofs, correct grammar, fix notational inconsistencies, and

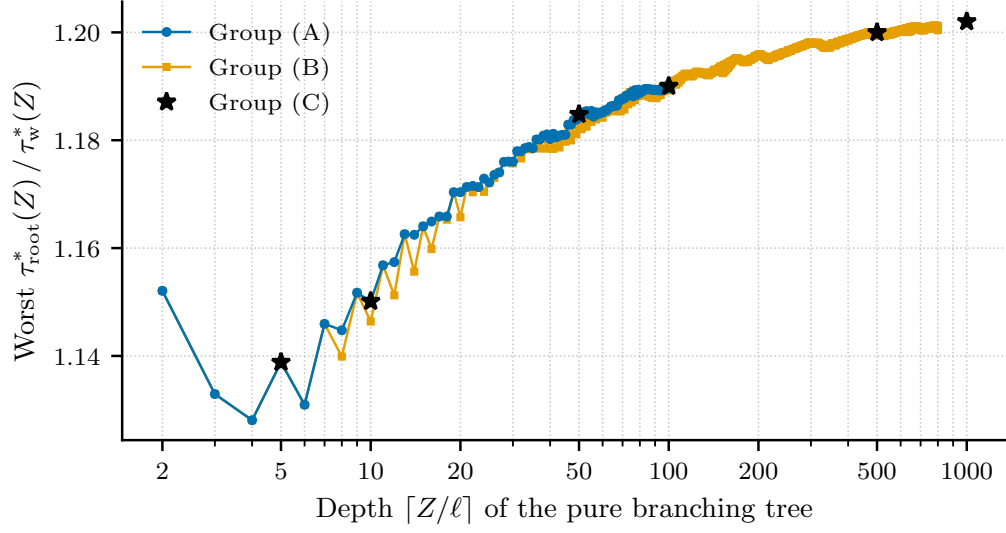


Figure 12: Largest approximation ratio at each depth $\lceil Z/\ell \rceil$. Groups (A) and (B) are shown up to depth 101 and 801, beyond which only settings with $r \gg \ell$ reach that depth.

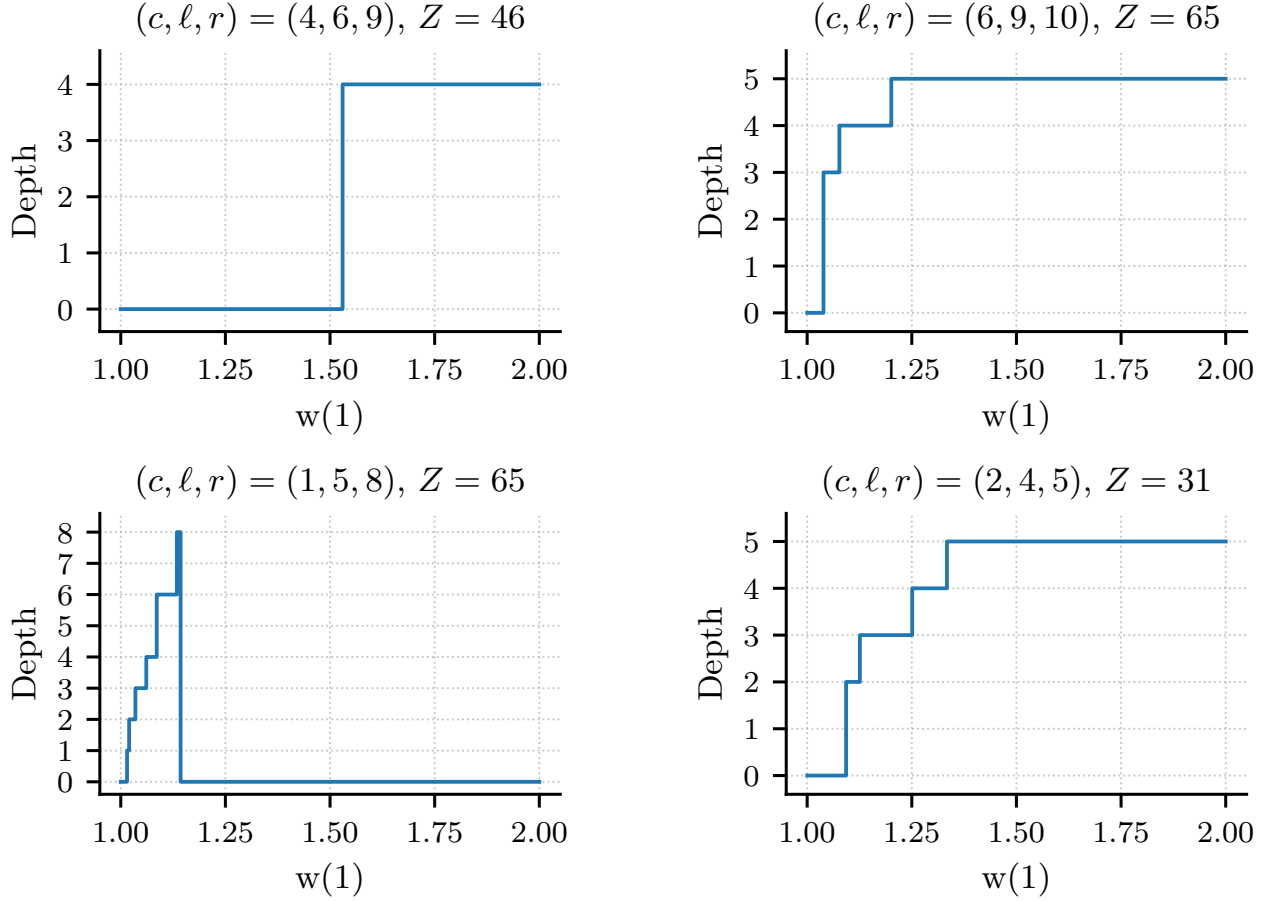


Figure 13: Depth of shallowest internal cut over all minimal-time trees as $w(1)$ varies. A value of 0 indicates that no minimal-time tree has an internal cut.

create better figures. Afterwards, these models helped simplify some results and generalize others, including an initial version of the asymptotic analysis at the authors’ prompting, to merge what was previously two sections into one (Section 4) and to precisely consider the effect of separation time (Section 5.3). Finally, the same models were used to generate the code for the simulation experiments and reproducible figures, replacing our previous slower mixed-integer optimization approach.

Funding. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-23-1-0340.

Code availability. The code for the simulations in Section 5.4 is available at <https://github.com/akazachk/abstract-branch-and-cut>.

References

- [1] Amitabh Basu, Michele Conforti, Marco Di Summa, and Hongyi Jiang. Complexity of branch-and-bound and cutting planes in mixed-integer optimization – II. *Combinatorica*, 42(1):971–996, 2022. URL <https://doi.org/10.1007/s00493-022-4884-7>.
- [2] Amitabh Basu, Michele Conforti, Marco Di Summa, and Hongyi Jiang. Complexity of branch-and-bound and cutting planes in mixed-integer optimization. *Math. Program.*, 198(1):787–810, 2023. URL <https://doi.org/10.1007/s10107-022-01789-5>.
- [3] Timo Berthold, Matteo Francobaldi, and Gregor Hendel. Learning to use local cuts. *Math. Program. Comput.*, 17(3):437–450, 2025. URL <https://doi.org/10.1007/s12532-025-00278-y>.
- [4] Hongyu Cheng and Amitabh Basu. Generalization guarantees for learning score-based branch-and-cut policies in integer programming. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 118669–118699. Curran Associates, Inc., 2025. URL <http://dx.doi.org/10.52202/085713-3959>.
- [5] Daniel Dadush and Samarth Tiwari. On the complexity of branching proofs. In *35th Computational Complexity Conference*, volume 169 of *LIPICs. Leibniz Int. Proc. Inform.*, pages 1–35. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2020. ISBN 978-3-95977-156-6. URL <https://doi.org/10.4230/LIPICs.CCC.2020.34>.
- [6] Santanu S. Dey and Marco Molinaro. Theoretical challenges towards cutting-plane selection. *Math. Program.*, 170(1, Ser. B):237–266, 2018. URL <https://doi.org/10.1007/s10107-018-1302-4>.
- [7] Santanu S. Dey, Aleksandr M. Kazachkov, Andrea Lodi, and Gonzalo Muñoz. Cutting plane generation through sparse principal component analysis. *SIAM J. Optim.*, 32(2):1319–1343, 2022. URL <https://doi.org/10.1137/21M1399956>.
- [8] Santanu S. Dey, Yatharth Dubey, and Marco Molinaro. Lower bounds on the size of general branch-and-bound trees. *Math. Program.*, 198(1):539–559, 2023. ISSN 0025-5610,1436-4646. URL <https://doi.org/10.1007/s10107-022-01781-z>.
- [9] Santanu S. Dey, Yatharth Dubey, Marco Molinaro, and Prachi Shah. A theoretical and computational analysis of full strong-branching. *Math. Progra.*, 205(1):303–336, 2024. URL <https://doi.org/10.1007/s10107-023-01977-x>.

- [10] Ralph E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bull. Amer. Math. Soc.*, 64:275–278, 1958.
- [11] Ralph E. Gomory. Solving linear programming problems in integers. In Richard Bellman and Marshall Hall, Jr., editors, *Combinatorial Analysis*, number 10 in Proceedings of Symposia in Applied Mathematics, pages 211–215. American Mathematical Society, 1960. doi: <https://doi.org/10.1090/psapm/010/0114670>. Proceedings of the Tenth Symposium in Applied Mathematics, Columbia University, April 24–26, 1958.
- [12] Ralph E. Gomory. An algorithm for integer solutions to linear programs. In Robert L. Graves and Philip Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, New York, 1963.
- [13] Boyang Han and Aleksandr M. Kazachkov. The strength of root cuts in an extended abstract branch-and-cut model. In Santanu S. Dey, Marco Di Summa, and Domenico Salvagnin, editors, *Integer Programming and Combinatorial Optimization*, volume 16588 of *Lecture Notes in Comput. Sci.*, pages 474–490, Cham, 2026. Springer. URL https://doi.org/10.1007/978-3-032-28691-8_31.
- [14] Hongyi Jiang. Sharp logarithmic thresholds for cut schedules in an abstract branch-and-cut model, 2026. URL <https://doi.org/10.48550/arXiv.2607.06343>.
- [15] Aleksandr M. Kazachkov, Pierre Le Bodic, and Sriram Sankaranarayanan. An abstract model for branch-and-cut. In Karen Aardal and Laura Sanità, editors, *Integer Programming and Combinatorial Optimization*, Lecture Notes in Comput. Sci., pages 333–346, Cham, 2022. Springer. URL https://doi.org/10.1007/978-3-031-06901-7_25.
- [16] Aleksandr M. Kazachkov, Pierre Le Bodic, and Sriram Sankaranarayanan. An abstract model for branch and cut. *Math. Program.*, 206:175–202, July 2024. URL <https://doi.org/10.1007/s10107-023-01991-z>.
- [17] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960. URL <https://doi.org/10.2307/1910129>.
- [18] Pierre Le Bodic and George Nemhauser. An abstract model for branching and its application to mixed integer programming. *Math. Program.*, 166(1-2):369–405, 2017. URL <https://doi.org/10.1007/s10107-016-1101-8>.
- [19] François Margot. Symmetry in integer linear programming. In Michael Jünger, Thomas Liebling, Denis Naddef, George Nemhauser, William Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence Wolsey, editors, *50 Years of Integer Programming 1958–2008*, pages 647–686. Springer-Verlag Berlin Heidelberg, 2010. URL http://dx.doi.org/10.1007/978-3-540-68279-0_17.
- [20] Matthias Miltenberger, Ted Ralphs, and Daniel E. Steffy. Exploring the numerics of branch-and-cut for mixed integer linear optimization. In Natalia Klierer, Jan Fabian Ehmke, and Ralf Borndörfer, editors, *Operations Research Proceedings 2017*, Oper. Res. Proc., pages 151–157. Springer, Cham, 2018. URL https://doi.org/10.1007/978-3-319-89920-6_21.
- [21] Prachi Shah, Santanu S. Dey, and Marco Molinaro. Non-monotonicity of branching rules with respect to linear relaxations. *INFORMS J. Comput.*, 38(1):53–66, 2026. URL <https://doi.org/10.1287/ijoc.2024.0709>.

- [22] Arrigo Zanette, Matteo Fischetti, and Egon Balas. Lexicography and degeneracy: can a pure cutting plane algorithm work? *Math. Program.*, 130(1, Ser. A):153–176, 2011. URL <http://dx.doi.org/10.1007/s10107-009-0335-0>.

Appendix

A Exact Approaches to Computing Minimal-Time One-Cut Trees

In this section, we show how to compute a minimal-time one-cut tree in two ways, first with a recursive formula, and then with a mixed-integer linear program.

A.1 Exact Recursion

In the one-cut setting, the minimal tree time can be computed by a recursion on the remaining bound. Consider a node with remaining bound z and no cut above it. If it is a cut node, then no further cut can be added below it, so its subtree is the cut node followed by $T(z - c)$, with tree time $1 + w(1)p(z - c)$. If it is a branch node, then, by [16, Lemma 24], its two subtrees are minimal-time trees for remaining bounds $z - \ell$ and $z - r$. Hence, the minimal tree time $\tau_w^*(z)$ satisfies

$$\tau_w^*(z) = \begin{cases} 1 & \text{if } z \leq 0, \\ 1 + \min\{\tau_w^*(z - \ell) + \tau_w^*(z - r), w(1)p(z - c)\} & \text{otherwise,} \end{cases}$$

where $p(z) = 1$ for $z \leq 0$ and $p(z) = 1 + p(z - \ell) + p(z - r)$ otherwise. The best root-cut tree has either no cut or a single root cut, so its tree time is $\min\{p(Z), 1 + w(1)p(Z - c)\}$, and the approximation ratio in Figure 11 is this value divided by $\tau_w^*(Z)$. Recording which term attains the minimum at each z recovers a minimal-time tree, and keeping all terms that attain it recovers all minimal-time trees; for example, the depth of the shallowest internal cut over all minimal-time trees, as in Figure 13, is computed by the same recursion.

Every remaining bound that appears in the recursion has the form $Z - k_1\ell - k_2r$, so, with memoization, τ_w^* and p are evaluated at most Z times each at positive arguments, and the recursion takes $O(Z)$ arithmetic operations for each value of $w(1)$. Moreover, $\tau_w^*(z)$ does not depend on Z , so a single pass of the recursion for a fixed (c, ℓ, r) gives the minimal tree time for every target bound of that triple at once. Writing $w(1) = k/50$ for an integer k , all tree times scaled by 50 are integers, which we compute exactly whenever they fit in 64 bits, so that ties between minimal-time trees, such as at $w(1) = \alpha(Z)$ or $w(1) = 2$, are detected exactly rather than up to a solver tolerance. As p grows exponentially (for example, $p(1000)$ has 210 digits when $(\ell, r) = (1, 2)$), larger values are computed in double precision, which was needed for about 73% of the settings in Section 5.4, all of them in groups (A) and (B). The reported worst-case settings and their peak ratios are rechecked with exact integer and rational arithmetic. Evaluating all values $w(1) \in \{1.00, 1.02, \dots, 2.00\}$ at once, the full set of 55,359,582 settings in Section 5.4 took 224 seconds with a Python implementation using 10 parallel workers on an Apple M1 Ultra processor.

A.2 Mixed-Integer Linear Programming Formulation

Another way to compute the trees in Section 5.4 is by solving a mixed-integer linear program over a complete binary tree of candidate nodes. The input is a one-cut setting (Z, c, ℓ, r) and the value $w(1)$. Let $\mathcal{V}$ be the node set of a complete binary tree with root 0 and depth $\lceil Z/\ell \rceil$, which is the

maximum necessary depth when considering only branch nodes. For $v \in \mathcal{V} \setminus \{0\}$, let $\text{par}(v)$ denote the parent of v , and let $\mathcal{V}_L$ and $\mathcal{V}_R$ be the sets of left and right children in $\mathcal{V}$. For $v \in T$, let $P(v)$ denote the nodes on the path from the root to v , including v .

For each $v \in \mathcal{V}$, define a binary variable y_v to indicate whether v is a node of the tree, and a binary variable x_v for whether a cut node is placed immediately above v , i.e., between v and its parent, or above the root if $v = 0$. Let $z_v \in \mathbb{R}_{\geq 0}$ be the variable for the bound at v . The binary variable q_v indicates whether a cut is applied on the path from the root to v , in which case v has time $w(1)$, and the binary variable m_v linearizes the product $q_v y_v$. The formulation is

$$\begin{aligned}
\min_{x,y,q,m,z} \quad & \sum_{v \in \mathcal{V}} (x_v + y_v + (w(1) - 1) m_v) \\
z_v = z_{\text{par}(v)} + \ell + c x_v & \quad v \in \mathcal{V}_L, & (7a) \\
z_v = z_{\text{par}(v)} + r + c x_v & \quad v \in \mathcal{V}_R, & (7b) \\
z_0 = c x_0, \quad y_0 = 1, & & (7c) \\
y_v \geq x_v & \quad v \in \mathcal{V}, & (7d) \\
y_v = y_{v'} & \quad \text{siblings } v, v' \in \mathcal{V}, & (7e) \\
y_{\text{par}(v)} \geq y_v \geq 1 - z_{\text{par}(v)}/Z & \quad v \in \mathcal{V} \setminus \{0\}, & (7f) \\
\sum_{u \in P(v)} x_u \leq 1 & \quad v \in \mathcal{V}, & (7g) \\
q_v \geq \sum_{u \in P(v)} x_u & \quad v \in \mathcal{V}, & (7h) \\
m_v \leq q_v, \quad m_v \leq y_v, \quad m_v \geq q_v + y_v - 1 & \quad v \in \mathcal{V}, & (7i) \\
x_v, y_v, q_v, m_v \in \{0, 1\}, \quad z_v \in \mathbb{R}_{\geq 0} & \quad v \in \mathcal{V}.
\end{aligned}$$

The objective is the tree time. Each node of the tree contributes 1, plus $w(1) - 1$ if a cut lies on its path from the root, and each cut node contributes 1, as it has no other cut above it in the one-cut setting. Constraints (7a)–(7c) compute the bound at each node, where a cut above v adds c . By (7d), a cut is only placed above a node of the tree, and by (7e), either both children of a node are in the tree or neither is. By (7f), a node is in the tree only if its parent is, and the children of a node must be in the tree whenever its bound is less than Z , so every leaf proves Z . Constraint (7g) allows at most one cut on each root-to-leaf path, and (7h) sets $q_v = 1$ when there is such a cut above v . Finally, (7i) are the McCormick inequalities for $m_v = q_v y_v$; as $w(1) > 1$, an optimal solution has $m_v = 1$ exactly when v is in the tree and has a cut above it. For the best root-cut tree, we also fix $x_v = 0$ for all $v \in \mathcal{V} \setminus \{0\}$.