Projection heuristics for binary branchings between sum and product

Oliver Kullmann* and Oleg Zaikin*

Swansea University, Swansea, UK
{O.Kullmann,O.S.Zaikin}@Swansea.ac.uk

Abstract. We consider a fundamental problem in the theory of branching heuristics for tree-based solvers, applicable e.g. to SAT, #SAT, CSP, #CSP. Such tree-based solvers are used as the cubing-part in the Cube-and-Conquer paradigm, and are thus of renewed interest for general (#)SAT solving. These solvers build at least implicitly a branching (backtracking) tree, with the goal to minimise tree-size. The heuristics are based on evaluating the progress made in a transition from an instance F to some "simplified" F' by a distance $d(F, F')$ (the bigger the more progress). When a branching $(F'_1, \ldots, F'_k)$ is to be chosen for F, for each possibility we consider its branching tuple t given by $t_i = d(F, F'_i)$, project it to a single number $\pi(t)$, and choose a branching with minimal $\pi(t)$. This paper investigates the choices for $\pi(t)$, in a theoretical framework. The general theory is reviewed, together with the theoretical result on the "canonical projection" $\pi(t) = \tau(t)$. Focusing then on binary branchings ($k = 2$, $t = (a, b)$), we analyse the asymptotics of $\tau(a, b)$, and reflect on the whole possible range of binary projections, arriving at first practical possibilities for dynamic heuristics.

1 Introduction

Historically, look-ahead solvers ([9,16]) were the first successful *complete* SAT solvers. With the rise of CDCL solvers ([17]), they went into oblivion, but the successful Cube-and-Conquer (C&C) framework ([11]) creates some space for them, since the efficient cubing (splitting) depends essentially on a good understanding of the branching process. Look-ahead solvers seem fundamentally tree-based: this restricts their power, but enables perfect parallelisation and a good understanding of the branching process (which needs to be aborted at some point – that's what C&C is doing, passing the task to the conquer-solver).

This paper starts a review of the fundamental theory of branching heuristics for tree-based solvers, looking especially at the question of what makes a "good projection": the heuristical measurement comes up with numbers for each branch, collected into a tuple of positive real numbers, a "branching tuple" t, and the "projection" $\pi(t)$ combines these numbers into a single number, which is to be minimised (maximised) to find the best branching variable. Obviously

* Supported by EPSRC grant EP/S015523/1.

the projection π is not unique, but what about the induced linear order? In the chapter [16] of the Handbook of Satisfiability on Branching Heuristics, it is outlined that under the assumption, that the branching width k is arbitrarily large, the induced linear ordering is indeed unique, given various natural axiomatic requirements on the consistency of the ordering. We review and extend these basic statements, and provide a proof for the existence and uniqueness of the canonical ordering, in a simplified setting. The canonical ordering is given by the canonical projection, the tau-function $\tau(t)$. Concentrating on binary branching, as in (#)SAT, i.e., $k = 2$, we analyse this fundamental function, and show how to (relatively) efficiently compute $\tau(a, b)$ (with high precision).

When only binary branchings are considered, then the uniqueness-result in fact does not apply (at least not directly). In practice an "approximation" to the tau-function is used, the product (i.e., $(a, b) \mapsto a \cdot b$, which is then to be maximised), improving on the earlier use of the sum. When standardised to form a (generalised) mean, then indeed τ lies in the interval given by the product- and the sum-projections. We consider a range of possible alternative projections in this interval. The final target is an understanding of the many parameters involved, so that they can be chosen offline and/or optimised online to yield efficient SAT solvers, based on proper dynamic heuristics.

Before we come to an outline of this paper, we review some recent literature. The basic method of trees with branching tuples (called "metric trees" in this paper), allowing the branching tuples to contain any positive *real* numbers, is a fundamental tool in the field of exponential upper bounds for algorithms; see [7] for an overview on that field, and see [8] for applications to bounds on circuit size and #SAT. Practical applications include solving vehicle routing problems ([19]), and solving the minimum latency problem ([5]). In [4,2] a similar but restricted branching theory was developed, only considering binary branchings with natural numbers as distances (which in turn allows stronger tools from the theory of recurrences), in the context of branch-and-bound algorithms for MIP.

An outline of the paper is as follows: Section 2 reviews the general theory, and provides a proof of the fundamental Lemma 1 on the composition of branching tuples, missing from the literature. The basic Theorem 1 on estimating tree sizes (known since [15]) gets a new proof, and we apply it to control the growth of trees based upon a single branching tuple. Section 3 proves Theorem 2 on the uniqueness of the order of branching tuples, given natural axioms. Corollary 1 gives a formulation suitable for restricted branching width (as used by SAT solvers). Section 4 then studies the binary tau-function, obtaining sharp asymptotic bounds and a method for fast computation. Section 5 finally considers the whole range of possibilities for binary projections, and we conclude in Section 6 by a summary and an outlook on future applications.

2 Branching tuples and distances

In this section we review the relevant elements of the theory of branching heuristics (for look-ahead solvers), as given in [16]. In Subsection 2.1 we speak about

the background, the (abstract) backtracking tree T of a look-ahead solver, whose size we want to minimise (by polytime means). "Minimising the size" refers to the construction of T, by recursively expanding a node v into its children $w_1, \dots, w_k$. This is guided by attaching numerical information to each branch, the "distances" $d(v, w_i) > 0$, a positive real number, the greater the more progress was made. A simple example for a distance is measuring the number of variables eliminated in the branch $v \rightsquigarrow w_i$. This numerical information is collected per node in the associated branching tuple $d(v)$. For the actual choice of the branching, in the context of the solving process (note that T is the final result of this process, while d records the progress measurements for the branching chosen by the solver), we assume that a list of branching tuples is given, and a selection is done by minimising the "projected value" of each branching tuple $d(v)$; that there is a canonical choice for a projection is shown in Section 3, based on the general theory on tree sizes reviewed in Subsection 2.2. A fundamental lemma is presented in Lemma 1, connecting the canonical projection of the branching tuples in a tree with the canonical projection of the flattening of the whole tree into a single branching tuple. This lemma is implicit in [16, Lemma 7.5.1], but there is no proof of it in the literature, and the concept of "flattening a tree" is a new device introduced here, simplifying certain aspects of the theory. For the basic Theorem 1 on bounds of tree sizes (which was introduced in [15]), we get in this way a new proof (the proof of [16, Theorem 7.4.8] is based on tree-probability distributions, which are not needed in this paper). This section is concluded by lemmas on expansions of branching tuples into trees, such that a strong handle on the tree sizes and other data are obtained.

2.1 Trees and distances

The basic object is a *rooted tree* T, which we treat as (special) directed (finite) acyclic graph (dag) with exactly one source. So as a digraph we have the set $V(T)$ of vertices, which we denote here as the set of nodes $\mathrm{nds}(T) := V(T)$. The unique source (the node with no in-neighbours) is denoted by $\mathrm{rt}(T) \in \mathrm{nds}(T)$. The special property of T, which makes it a rooted tree, is that from $\mathrm{rt}(T)$ there is exactly one path to every node of T. As usual we denote by $E(T) \subset \mathrm{nds}(T) \times \mathrm{nds}(T)$ the set of arcs (directed from the root towards the leaves).

Every node $v \in \mathrm{nds}(T)$ has a set $\mathrm{chd}_T(v) \subset \mathrm{nds}(T)$ of children (the out-neighbours of v). The number of children of a node is its degree, denoted by $\deg_T(v) := |\mathrm{chd}(v)| \in \mathbb{N}_0$. The leaves of T are the sinks of T, and the set of leaves is denoted by $\mathrm{lvs}(T) := \{v \in \mathrm{nds}(T) : \mathrm{chd}(v) = \emptyset\} \subseteq \mathrm{nds}(T)$; obviously $v \in \mathrm{lvs}(T) \Leftrightarrow \deg_T(v) = 0$. An inner node of T is a node which is not a leaf, and we use $\mathrm{inds}(T) := \mathrm{nds}(T) \setminus \mathrm{lvs}(T)$ for the set of all inner nodes. To avoid technical difficulties we assume that T has no single-child nodes, that is, for every $v \in \mathrm{inds}(T)$ we have $\deg(v) \geq 2$. The main complexity measure here is $\#\mathrm{nds}(T) := |\mathrm{nds}(T)| \in \mathbb{N}$, the number of nodes of T. Sometimes it is more convenient to consider $\#\mathrm{lvs}(T) := |\mathrm{lvs}(T)| \leq \#\mathrm{nds}(T)$, the number of leaves. By definition holds $\#\mathrm{nds}(T) = \#\mathrm{lvs}(T) + |\mathrm{inds}(T)|$. In a full binary tree (i.e., all inner nodes have degree two) we have $\#\mathrm{nds}(T) = 2\,\#\mathrm{lvs}(T) - 1$ and

$|\mathrm{inds}(T)| = \#\mathrm{lvs}(T) - 1$. To be able to speak of the *tuple* of children of an inner node, we make the further assumption that we have *ordered* trees; we could avoid that by using multisets, however using tuples seems more natural in our context. Technically this is handled by considering one infinite linearly ordered set $(\mathcal{U}, \leq)$ (for example $\mathcal{U} = \mathbb{N}$ with the natural order) as the universe of vertices, that is, for every rooted tree T we have $V(T) \subset \mathcal{U}$. When we use now $\mathrm{chd}(v) = \{w_1, \ldots, w_k\}$, then we assume from now one that $w_1 < \cdots < w_k$ (and thus $k = \deg(v)$). As the induced linear order on $\mathrm{lvs}(T)$ we use the lexicographical order as given by the paths from the root to the leaves (that is, the order of the leaves as they show up in the inorder traversal of T); we could use any other linear order on the leaves, but this order lets us avoid certain abstractions here.

The main examples of such trees T are given by the branching trees (backtracking trees) of look-ahead solvers. At the leaves of T we might have the solved nodes (where either unsatisfiability was determined, or a (partial) satisfying assignment was found), or the nodes as given to a conquer-solver in the C&C setting. For #SAT-solving we naturally have to consider the *whole* tree (as is done in this paper), since we need to count all solutions. For SAT-solving, considering the whole tree means that the basis for the heuristic is the unsatisfiable case (without early abortion by finding a satisfying assignment). This is a natural point of view, since complete SAT-solvers are intrinsically unsatisfiability-driven, while the possible short-cuts for (just) finding a single satisfiable assignment are handled by the choice of the first branch; see [16, Section 7.9] and [9, Subsection 5.3.2] for information on this in the context of look-ahead solvers. We remark that we exclude single-child nodes, since they correspond to an actual reduction performed (no branching occurred), and can be contracted into the parent node.

The basic building block of the branching heuristic, from the theoretical side, is given by a **distance** on T, a map $d : E(T) \to \mathbb{R}_{>0}$ labelling every arc with a positive real number. The intuitive meaning of $d((v, w)) = d(v, w) > 0$ is that it measures the progress made in the transition from v to its child $w \in \mathrm{chd}(v)$. More generally, for any (undirected) graph G and a mapping $d : E(G) \to \mathbb{R}_{>0}$ we obtain a metric, which is a map $d : V(G)^2 \to \mathbb{R}_{\geq 0}$, by defining $d(v, w)$ for arbitrary $v, w \in V(G)$ as the minimum weighted length of a path between v and w. So a distance d on a (rooted, directed) tree T induces a metric, using the underlying (undirected) graph. Thus for $v, w \in \mathrm{nds}(T)$ we have $d(v, w) = d(v, c) + d(c, w)$ for the common ancestor c of v, w in T, and where $d(v, c) = d(c, v)$ resp. $d(c, w)$ is the sum of d-values of arcs on the unique path from c to v resp. w. In this paper we don't travel paths against the direction of the arcs, but the terminology of a "metric tree" for a pair (T, d) is nevertheless a natural choice:

Definition 1. *A* ***nontrivial rooted tree*** T *here is a dag* $T = (V, E)$ *with* $V \subset \mathcal{U}$ *and* $E \subseteq V \times V$ *with exactly one source, where the underlying graph is a tree (connected and acyclic), where every inner node has at least two children, and with* $\#\mathrm{nds}(T) = |V(T)| \geq 3$. *A* ***distance*** *on* T *is a map* $d : E(T) \to \mathbb{R}_{>0}$. *Pairs* (T, d) *are called* ***metric trees****, and the set of all metric trees is* $\boldsymbol{\mathcal{MT}}$. *For an inner node* v *with* $\mathrm{chd}(v) = \{w_1, \ldots, w_k\}$ *the* ***associated branching tuple***

$\boldsymbol{d(v)} := (d(v,w_1),\dots,d(v,w_k)) \in \mathbb{R}_{>0}^{\deg(v)}$ *is the tuple of distances from* v *to its children (using the given linear order on the children).*

As T is the final result of the run of the look-ahead solver, the branching tuple $d(v)$ only records the distances for the (final) choices as made by the algorithm for the branchings. In the process of recursively expanding v, the setting is that the solver actually sees for every possible branching a corresponding branching tuple, and the solver chooses one of them; the distance on T then records this choice, while for the solver a distance $d(F,F')$ between problem instances is needed to construct the branching tuples. The object of study of this paper is not d (which we assume is given), but the choice of "the best" branching tuple. For more information on distances d as used in (look-ahead) SAT solvers see [9, Subsection 5.3.1] and [16, Section 7.7],

So we consider now branching tuples in isolation (later in Subsection 2.2 we introduce the "canonical projection", which cans the branching tuple into a single number, the smaller the better). Let $\boldsymbol{\mathcal{BT}} := \bigcup_{k\geq 2}\mathcal{BT}_k$, $\boldsymbol{\mathcal{BT}_k} := \mathbb{R}_{>0}^k$ be the set of all branching tuples, i.e., all tuples of positive real numbers of width (length) $k \geq 2$. We use the following natural operations for $t \in \mathcal{BT}_k$:

- $|t| := k \in \mathbb{N}_{\geq 2}$ is the length of t;
- $\min(t), \max(t) \in \mathbb{R}_{>0}$ are the minimal resp. maximal values;
- $\Sigma(t) := \sum_{i=1}^{|t|} t_i \in \mathbb{R}_{>0}$ is the sum of all values.

We emphasise that branching tuples contain arbitrary positive *real* numbers, and so the theory is a generalisation of the theory of recurrences, which only consider branching tuples with natural numbers as entries.

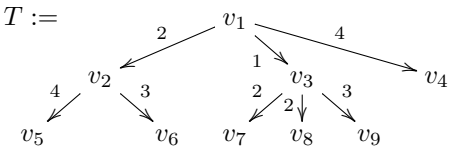
From metric trees we extract the set of branching tuples, and for a set of branching tuples we consider the set of all metric trees using only these tuples:

Definition 2. *For* $(T,d) \in \mathcal{MT}$ *let* $\boldsymbol{\mathcal{BT}(T,d)} := \{d(v) : v \in \mathrm{inds}(T)\} \subset \mathcal{BT}$ *be the (finite) set of branching tuples associated with the inner nodes. And for* $B \subseteq \mathcal{BT}$ *let* $\boldsymbol{\mathcal{MT}(B)} := \{T \in \mathcal{MT} : \mathcal{BT}(T) \subseteq B\}$ *be the (infinite) set of metric trees, where all associated branching tuples are in* B*.*

A tree with a distance can be flattened to a single branching tuple, forgetting the branching structure:

Definition 3. *For* $T \in \mathcal{MT}$ *let the branching tuple* $\boldsymbol{\mathrm{fl}(T)} \in \mathcal{BT}_{\#\mathrm{lvs}(T)}$ *contain for each leaf* $v \in \mathrm{lvs}(T)$ *the sum of distances from* $\mathrm{rt}(T)$ *to* v*, that is, for* $\mathrm{lvs}(T) = \{w_1,\dots,w_{\#\mathrm{lvs}(T)}\}$ *we set* $\mathrm{fl}(T) := (d(\mathrm{rt}(T),w_1),\dots,d(\mathrm{rt}(T),w_{\#\mathrm{lvs}(T)}))$.

Example 1. For the metric tree

$T :=$ 

we have

1. $\#\mathrm{nds}(T) = 9$, $\#\mathrm{lvs}(T) = 6$, $|\mathrm{inds}(T)| = 3$.
2. $\mathcal{BT}(T) = \{(2,1,4),(4,3),(2,2,3)\}$.
3. $\mathrm{fl}(T) = (2+4, 2+3,\ 1+2, 1+2, 1+3,\ 4) = (6,5,3,3,4,4)$.

2.2 The tau-function and bounds on tree sizes

Generalising the notion of a root of a "characteristic polynomial" from the theory of recurrences, the "tau-function" is a fundamental tool:

Definition 4. *For $t \in \mathcal{BT}$ we define the characteristic function $\chi_t : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ by $\chi(t)(x) := \sum_{i=1}^{|t|} x^{-t_i}$. Since $\chi(t)$ is strictly decreasing with $\chi(t)(1) = |t| \geq 2$ and $\lim_{x\to\infty} \chi(t)(x) = 0$, there is exactly one $x_0 \in \mathbb{R}_{>1}$ with $\chi(t)(x_0) = 1$, and we define $\tau(t) := x_0$.*

So $\tau(t) \lessgtr x$ for $x > 0$ iff $\chi(t)(x) \lessgtr 1$. The basic intuition behind $\tau : \mathcal{BT} \to \mathbb{R}_{>1}$ can be grasped by considering the "main" solution to a difference equation: One seeks a function $f : \mathbb{N}_0 \to \mathbb{R}_{\geq 0}$ satisfying for some given $a \in \mathbb{N}^k$ a recurrence $\forall\, n \in \mathbb{N}_0, n \geq a_i : f(n) = \sum_{i=1}^k f(n - a_i)$. For example the Fibonacci recurrence $f(n) = f(n-1) + f(n-2)$ is given by $k = 2$ and $a = (1, 2)$. Now $f(n) := \tau(a)^n$ fulfils the recurrence, since $\sum_{i=1}^k \tau(a)^{-a_i} = 1$, and multiplying both sides with $\tau(a)^n$ yields $\tau(a)^n = f(n) = \sum_{i=1}^k \tau(a)^{n-a_i} = \sum_{i=1}^k f(n - a_i)$. Basic properties for $a \in \mathcal{BT}$, $\lambda \in \mathbb{R}_{>0}$ and the restriction $\tau_k := \tau \mid \mathcal{BT}_k$ are (all with easy proofs):

1. τ_k is symmetric (invariant under permutation).
2. τ_k is strictly decreasing in each component.
3. $\tau_k(1, \ldots, 1) = k$.
4. If t is a proper prefix of t' then $\tau(t) < \tau(t')$.
5. $\tau(\lambda \cdot a) = \tau(a)^{1/\lambda}$, $\tau(a)^{\min(a)} \leq |a| \leq \tau(a)^{\max(a)}$.

An important property of all τ_k is that they are strictly convex, that is for all $a, b \in \mathcal{BT}_k$ and $0 < \lambda < 1$ holds $\tau((1-\lambda)a + \lambda b) < (1-\lambda)\tau(a) + \lambda\tau(b)$ (this requires more work to check). This implies for example that $\tau(2, 2) < \tau(1, 3)$, using $a := (1, 3)$, $b := (3, 1)$ and $\lambda := 0.5$. Strict convexity is a generalisation of the "penalty" given to "imbalanced" branching tuples, due to the inherent exponential growth: Comparing e.g. (x, x) with $(x-\varepsilon, x+\varepsilon)$, the former is better since the loss in the branch $x - \varepsilon$ is bigger than the win in the branch $x + \varepsilon$, due to the convexity of exponential functions. A simple sufficient criterion for deriving $\tau(a) \leq \tau(b)$ is obtained by using symmetry, monotonicity and the prefix condition; we condense this into the following order relation:

Definition 5. *The order-relation $a \precsim b$ ("a is trivially smaller than b") holds for $a, b \in \mathcal{BT}$ if the following three conditions are fulfilled: (1) $|a| \leq |b|$; (2) there is a permutation a' of a, such that for all $i \in \{1, \ldots, |a|\}$ holds $a_i \geq b_i$; (3) either $|a| < |b|$ or there is $i \in \{1, \ldots, |a|\}$ with $a'_i > b_i$.*

Some simple properties of the trivially-smaller-relation ("trivially better") are:

1. $\precsim$ is a strict partial order on $\mathcal{BT}$ (i.e., irreflexive ($a \not\precsim a$) and transitive ($a \precsim b \wedge b \precsim c \Rightarrow a \precsim c$)), without minimal or maximal elements.
2. If $a \precsim b$ then $\tau(a) < \tau(b)$.
3. Sufficient criterion for $a \precsim b$ are:
   (a) $|a| \leq |b| \wedge \min(a) > \max(b) \Rightarrow a \precsim b$.
   (b) $|a| < |b| \wedge \min(a) \geq \max(b) \Rightarrow a \precsim b$.

A fundamental lemma is that the tau-value for a flattened metric tree is in the interval given by minimum and maximum tau-values over the inner nodes; for a set of branching tuples B we use $\tau(B) := \{\tau(t) : t \in B\}$ in the usual way:

Lemma 1. *For $T \in \mathcal{MT}$ holds $\min \tau(\mathcal{BT}(T)) \leq \tau(\mathrm{fl}(T)) \leq \max \tau(\mathcal{BT}(T))$.*

Proof. Let $r := \mathrm{rt}(T)$. First consider the special case $|\mathrm{inds}(T)| = 2$. Let $a := d(r)$, and for the other inner node $v \in \mathrm{inds}(T) \setminus \{r\}$ let $b := d(v)$, while $c := \mathrm{fl}(T)$. W.l.o.g. we can assume that v is the first child of r, and thus, using $p := \deg(r)$ and $q := \deg(v)$ we have $c = (a_1 + b_1, \ldots, a_1 + b_q, a_2, \ldots, a_p)$. W.l.o.g. we further assume that $\tau(a) \leq \tau(b)$, and thus $\chi(b)(\tau(a)) \geq 1$ and $\chi(a)(\tau(b)) \leq 1$. So we have to show $\tau(a) \leq \tau(c) \leq \tau(b)$. The first inequality follows by

$$\begin{aligned}\chi(c)(\tau(a)) &= \sum_{i=1}^{q} \tau(a)^{-a_1-b_i} + \sum_{i=2}^{p} \tau(a)^{-a_i} = \tau(a)^{-a_1} \cdot \sum_{i=1}^{q} \tau(a)^{-b_i} + \sum_{i=2}^{p} \tau(a)^{-a_i} \\ &= \tau(a)^{-a_1} \cdot \chi(b)(\tau(a)) + \sum_{i=2}^{p} \tau(a)^{-a_i} \geq \tau(a)^{-a_1} + \sum_{i=2}^{p} \tau(a)^{-a_i} = 1,\end{aligned}$$

and the second inequality follows by

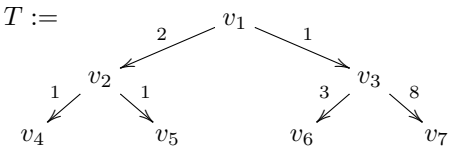
$$\begin{aligned}\chi(c)(\tau(b)) &= \sum_{i=1}^{q} \tau(b)^{-a_1-b_i} + \sum_{i=2}^{p} \tau(b)^{-a_i} = \tau(b)^{-a_1} \cdot \sum_{i=1}^{q} \tau(b)^{-b_i} + \sum_{i=2}^{p} \tau(b)^{-a_i} \\ &= \tau(b)^{-a_1} + \sum_{i=2}^{p} \tau(b)^{-a_i} = \chi(a)(\tau(b)) \leq 1.\end{aligned}$$

Now we prove the statement by induction over $n := |\mathrm{inds}(T)|$. For $n = 1$ we have $\mathrm{fl}(T) = d(r)$ and $\mathcal{BT}(T) = \{d(r)\}$, and thus the assertion trivially holds. It remains the case $n \geq 2$. Consider a node $v \in \mathrm{inds}(T)$ with $\mathrm{chd}(v) \subseteq \mathrm{lvs}(T)$. Let T' be the subtree of T with the leaves of v removed, that is, $V(T') = V(T) \setminus \mathrm{chd}(v)$ and $E(T') = E(T) \setminus \{(v, w) : w \in \mathrm{chd}(T)\}$. So $|\mathrm{inds}(T)| = n - 1$, and we can apply the induction hypothesis to T', that is, we have

$$\min \tau(\mathcal{BT}(T')) \leq \tau(\mathrm{fl}(T')) \leq \max \tau(\mathcal{BT}(T')).$$

We note that $\mathcal{BT}(T) = \mathcal{BT}(T') \cup \{d(v)\}$. Let T'' be the flattening of T' as a tree with one inner node, and so we have $\mathrm{fl}(T'') = \mathrm{fl}(T')$. We assume $\mathrm{rt}(T'') = r$ and $v \in \mathrm{lvs}(T'')$, and thus for the metric tree S obtained by attaching the branching of v in T to T'' we have $\mathrm{fl}(S) = \mathrm{fl}(T)$ and $\mathcal{BT}(S) = \{\mathrm{fl}(T'), d(v)\}$. We can apply the above special case to S and obtain $\min \tau(\mathcal{BT}(S)) \leq \tau(\mathrm{fl}(S)) \leq \max \tau(\mathcal{BT}(S))$. Finally we have $\min \tau(\mathcal{BT}(T)) = \min(\tau(\mathcal{BT}(T')) \cup \{d(v)\}) \leq \min(\{\tau(\mathrm{fl}(T'))\} \cup \{d(v)\}) = \min \tau(\mathcal{BT}(S))$, and similarly holds $\max \tau(\mathcal{BT}(S)) \geq \max(\tau(\mathcal{BT}(T)))$. □

Example 2. For the metric tree

$T :=$ 

we have

1. $\mathcal{BT}(T) = \{(2,1),(1,1),(3,8)\}$.
2. $\mathrm{fl}(T) = (2+1, 2+1,\ 1+3, 1+8) = (3,3,4,9)$.
3. $\tau(\mathcal{BT}(T)) = \{1.618\ldots,\ 2,\ 1.1461\ldots\}$, $\tau(\mathrm{fl}(T)) = 1.4147\ldots$.

The global meaning of the tau-values of inner nodes over a tree with given distance is that they yield upper (and lower) bounds on the number of leaves, which is expressed by the following theorem (for which we give an alternative, simpler proof here):

Theorem 1. *[16, Theorem 7.4.8] For $T \in \mathcal{MT}$ holds*

$$(\min \tau(\mathcal{BT}(T)))^{\min \mathrm{fl}(T)} \le \#\mathrm{lvs}(T) \le (\max \tau(\mathcal{BT}(T)))^{\max \mathrm{fl}(T)}.$$

Proof. Lemma 1 yields $(\min \tau(\mathcal{BT}(T)))^{\min \mathrm{fl}(T)} \le \tau(\mathrm{fl}(T))^{\min \mathrm{fl}(T)} \le |\mathrm{fl}(T)| = \#\mathrm{lvs}(T) \le \tau(\mathrm{fl}(T))^{\max \mathrm{fl}(T)} \le (\max \tau(\mathcal{BT}(T)))^{\max \mathrm{fl}(T)}$. □

Theorem 1 gives a global meaning to the target for the branching heuristics to choose branchings with associated branching tuples t minimising $\tau(t)$. Of course, this only makes sense for a "sensible" distance d, with $\max \mathrm{fl}(T)$ being a reasonable parameter of the input. Our main application of Theorem 1 is to show that branching tuples t, t' with $\tau(t) < \tau(t')$ can be expanded so that the tau-relation becomes trivial:

Definition 6. *An **expansion** of $t \in \mathcal{BT}$ is any $\mathrm{fl}(T) \in \mathcal{BT}$ for $T \in \mathcal{MT}(\{t\})$.*

So for any expansion t' of t we have $\tau(t') = \tau(t)$.

Lemma 2. *For $t \in \mathcal{BT}$ and $K \in \mathbb{R}_{\ge \max(t)}$ there exists an expansion t' of t with $\min(t') > K - \max(t)$, $\max(t') \le K$, and $\tau(t)^{K-\max(t)} < |t'| \le \tau(t)^K$.*

Proof. Consider any metric tree T with $\mathcal{BT}(T) = \{t\}$, such that $\max \mathrm{fl}(T) \le K$, while T can not be expanded further (by expanding any leaf) without violating the bound $\max \mathrm{fl}(T) \le K$; obviously such T exist, since one can start with t itself, and expand leaves as long as one stays below K. Let $t' := \mathrm{fl}(T)$. Then we have $\min(t') > K - \max(t)$, and by Theorem 1 we get $\tau(t)^{K-\max(t)} < |t'| \le \tau(t)^K$. □

Lemma 3. *For all $a, b \in \mathcal{BT}$ with $\tau(a) < \tau(b)$ there are expansions a' of a and b' of b with $a' \lneq b'$.*

Proof. Let $\alpha := \frac{\ln(\tau(b))}{\ln(\tau(a))}$ (thus $\alpha > 1$). Choose any $K_b \ge \frac{\max(a) + \alpha \cdot \max(b)}{\alpha - 1} > 0$. Thus for $\beta := K_b + \max(a)$ and $\gamma := \alpha \cdot (K_b - \max(b))$ holds $\beta \le \gamma$, and we can choose (any) K_a with $\beta \le K_a \le \gamma$. By Lemma 2 there are expansions a', b' of a, b with

- $\min(a') > K_a - \max(a)$ and $|a'| \le \tau(a)^{K_a}$;
- $\max(b') \le K_b$ and $\tau(b)^{K_b - \max(b)} < |b'|$.

From $K_a \ge \beta$ we get $K_a - \max(a) \ge K_b$, and thus $\min(a') > \max(b')$. And from $K_a \le \gamma$ we get $\tau(a)^{K_a} \le \tau(b)^{K_b - \max(b)}$, and thus $|a'| < |b'|$. □

3 The canonical order of branching tuples

As explained in Subsection 2.2, the canonical projection $\tau(t)$ has a natural *global meaning*, namely it yields an upper bound on the number of leaves in the branching tree. Thus it makes sense to use $\tau(t)$ for projection, and observing its value should be also useful to monitor the state of the search. Here now we give its *local meaning*: the order on branching tuples as stipulated by τ is uniquely given by natural axioms on how we want the projection to behave when comparing different branching tuples. More precisely, recall that a total (or linear) quasi-order on a set X is a binary relation $\leq$ which is reflexive ($x \leq x$), transitive ($x \leq y \wedge y \leq z \Rightarrow x \leq z$) and total ($x \leq y \vee y \leq x$). By defining $t \leq_\tau t' :\Leftrightarrow \tau(t) \leq \tau(t')$ we obtain a total quasi-order on $\mathcal{BT}$ (the smaller the "better"). The task of this section is to give an intrinsic characterisation of this order. This is achieved in Theorem 2, which is equivalent to [16, Theorem 7.5.3], but there is no proof there, while here we give a complete proof. Based on the notion of expansions, we can also provide a natural formulation for the case of restricted width of branching tuples.

What are now the axioms for comparing branching tuples? Consider a total quasi-order $\preceq$ on $\mathcal{BT}$. As usual we define $t \simeq t'$ if $t \preceq t'$ and $t' \preceq t$ ($\simeq$ is an equivalence relation on $\mathcal{BT}$), and $t \prec t'$ if $t \preceq t'$ and $t \not\simeq t'$. We call it a **canonical branching order** if it fulfils the following four properties for all $t, t' \in \mathcal{BT}$:

(S) Symmetry For a permutation t' of t holds $t \simeq t'$.
(E) Expansion If t' is an expansion of t then $t' \simeq t$.
(T) Trivial comparison If $t \lneq t'$ then $t \prec t'$.
(D) Density For $t \prec t'$ there is $\varepsilon > 0$ such that $t - \varepsilon \in \mathcal{BT}$ and $t - \varepsilon \prec t'$.

For (D) we used $t - \varepsilon := (t_1 - \varepsilon, \ldots, t_{|t|} - \varepsilon)$.

Theorem 2. *There is exactly one canonical branching order, namely $\leq_\tau$.*

Proof. $\leq_\tau$ is a canonical branching order by Lemma 1 and continuity of τ (for fixed branching-width). Now consider any canonical branching order $\preceq$ and $a, b \in \mathcal{BT}$, where we assume w.l.o.g. $a \preceq b$. In case of $\tau(a) < \tau(b)$ by Lemma 3 there are expansions a', b' with $a' \lneq b'$. Thus by (E) and (T) we get $a \simeq a' \prec b' \simeq b$. It remains the case $\tau(a) = \tau(b)$. If $a \simeq b$, then we are done, so assume w.l.o.g. $a \prec b$. By (D), (T) there is $\varepsilon > 0$ with $a \prec a - \varepsilon \prec b$. We have $\tau(a) < \tau(a - \varepsilon)$, and thus $\tau(b) < \tau(a - \varepsilon)$, whence by the first part $b \prec a - \varepsilon$, a contradiction. □

Corollary 1. *For $k \in \mathbb{N}$, $k \geq 2$, the canonical branching order $\leq_\tau$ restricted to $\mathcal{BT}_k$ is uniquely determined by the conditions (S), (T), (D) and*

($\mathbf{E_k}$) *If for $a, b \in \mathcal{BT}_k$ there are expansions a', b' with $a' \lneq b'$, then $a \prec b$.*

Proof. The proof of Theorem 2 works as well. □

4 Analysis and numerics of binary tau

In the remainder of this paper we focus on binary branchings ($k = 2$). In this section we first concentrate $\tau(a, b)$ to its essential core $\mathrm{w}\tau(x)$, and show in Lemma 4, that this core is asymptotically very close to a well-known special function, the Lambert-W function. This enables us in Theorem 4 to give good lower and upper bounds for $\mathrm{w}\tau(x)$ by elementary functions. Using these bounds as starting points for the Newton-Raphson algorithm, only very few iterations are needed to compute $\mathrm{w}\tau(x)$ (and thus $\tau(a, b)$) with full precision (the observed worst-case precision for `double` is two ulp's, that is, nearly precise in the last machine-digit).

At least for numerical reasons it seems better to consider the logarithm of the tau-function, denoted by $\mathrm{l}\tau : \mathcal{BT} \to \mathbb{R}_{>0}$, and defined by $\mathrm{l}\tau(t) := \ln(\tau(t))$ (using the natural logarithm). This replaces the computation of arbitrary powers $(x, y) \mapsto x^y$ by the computation of the exponential function $(x, y) \mapsto \exp(x \cdot y)$. It seems indeed best to compute $\mathrm{l}\tau(t)$ directly, and then to use $\tau(t) = \exp(\mathrm{l}\tau(t))$: $\mathrm{l}\tau(t)$ is the unique $x \in \mathbb{R}_{>0}$ such that $\sum_{i=1}^{|t|} \exp(-t_i \cdot x) = 1$. We now have $\mathrm{l}\tau(\lambda \cdot t) = \frac{1}{\lambda} \cdot \mathrm{l}\tau(t)$ and $\mathrm{l}\tau_k(1, \ldots, 1) = \ln(k)$.

We see that $\mathfrak{T}(t) := \frac{\ln(|t|)}{\mathrm{l}\tau(t)}$ fulfils $\mathfrak{T}(\lambda \cdot t) = \lambda \cdot \mathfrak{T}(t)$ and $\min(t) \le \mathfrak{T}(t) \le \max(t)$. Indeed $\mathfrak{T} : \mathcal{BT} \to \mathbb{R}_{>0}$ has further properties of a (general) "mean", as shown in [16, Section 7.3.3]. We will discuss some further properties of such means in Subsection 5.1. Here for us only the bounds on $\mathfrak{T}(t)$ from [16, Theorem 7.3.4] are relevant. In general we have $\mathfrak{T}(t) \le \frac{\Sigma(t)}{|t|}$, that is mean-tau is at most the arithmetic mean. We remark that $\mathfrak{T}(t)$ for fixed $|t|$ is strictly concave.

In the remainder of this paper we concentrate on $\mathrm{l}\tau_2 : \mathcal{BT}_2 \to \mathbb{R}_{>0}$, as this is at least currently most important for SAT solving. So $\mathrm{l}\tau(a, b)$ is the unique $x > 0$ with $\exp(-a \cdot x) + \exp(-b \cdot y) = 1$. It is an elementary exercise to show that $\mathfrak{T}(a, b) \ge \sqrt{a \cdot b}$ holds, that is, binary mean-tau is at least the geometric mean. Indeed it seems fastest and most accurate to not compute $\mathrm{l}\tau(a, b)$ directly, but to eliminate one argument of $\mathrm{l}\tau(a, b)$; we use the form which yields a strictly increasing function:

Definition 7. *For $x \in \mathbb{R}_{>0}$:* $\mathbf{w\tau(x)} := \mathrm{l}\tau(1, \frac{1}{x}) \in \mathbb{R}_{>0}$.

We have the following easy properties:

1. $\mathrm{w}\tau(x) = \mathrm{l}\tau(1, \frac{1}{x}) = x \cdot \mathrm{l}\tau(1, x)$, $\mathrm{w}\tau(x^{-1}) = x^{-1}\,\mathrm{w}\tau(x)$.
2. $\mathrm{w}\tau$ is strictly increasing with $\lim_{x\to 0} \mathrm{w}\tau(x) = 0$ and $\lim_{x\to+\infty} \mathrm{w}\tau(x) = +\infty$.
3. $\mathrm{w}\tau(1) = \ln(2)$, $\mathrm{w}\tau(2) = 2\ln(\frac{1+\sqrt{5}}{2})$, $\mathrm{w}\tau(\frac{1}{2}) = \ln(\frac{1+\sqrt{5}}{2})$.
4. $\mathrm{l}\tau(a, b) = \frac{1}{a}\,\mathrm{l}\tau(1, \frac{b}{a}) = \frac{1}{a}\,\mathrm{w}\tau(\frac{a}{b})$.
5. The characteristic equation for $\mathrm{w}\tau(a)$, $a \in \mathbb{R}_{>0}$, and $x \in \mathbb{R}_{>0}$ is
$$\exp(-x) + \exp(-\frac{x}{a}) = 1.$$

The above mentioned bounds, that $\mathfrak{T}(a, b)$ lies between the geometric and the arithmetic mean, yields elementary bounds for $\mathrm{w}\tau(x)$:

$$\frac{\ln 4}{1 + x^{-1}} = \ln(4)\frac{x}{x+1} \le \mathrm{w}\tau(x) \le \ln(2)\sqrt{x}.$$

We will see that asymptotically $\mathrm{w}\tau(x) \sim \ln(x)$, and thus these bounds are very bad for large x. The best bounds on $\mathrm{w}\tau(x)$ for larger x seem to be obtained by using the principal branch of the Lambert-W function (see [18, Section 4.13]). This is a function $W : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, defined for $a \in \mathbb{R}_{\geq 0}$ as the unique $x \in \mathbb{R}_{\geq 0}$ with $x \cdot \exp(x) = a$. Thus $W(a) \cdot \exp(W(a)) = a$, $\exp(W(a)) = \frac{a}{W(a)}$, $W(a) = \frac{a}{\exp(W(a))}$, and $\exp(-W(a)) = \frac{W(a)}{a}$ for $a > 0$. Simple values are $W(a) = 0$ iff $a = 0$, $W(e) = 1$, and more generally $W(x \cdot \exp(x)) = x$.

Before we can show the close relation between $\mathrm{w}\tau(x)$ and $W(x)$, we remind at a few elementary properties:

1. For all $x \in \mathbb{R}$: $\exp(x) \geq 1 + x$.
2. Thus for all $x \in \mathbb{R}_{>-1}$: $\exp(-x) \leq \frac{1}{1+x}$.
3. For all $x \in \mathbb{R} \setminus \{0, -1\}$: $\frac{1}{1+x} + \frac{1}{1+\frac{1}{x}} = 1$.

The following relation was pointed out in the discussion [13]:

Lemma 4. *For all $x \in \mathbb{R}_{>0}$ holds $W(x) \leq \mathrm{w}\tau(x) \leq \ln(\exp(W(x)) + 1)$.*

Proof. We show lower and upper bound by substitution into the characteristic equation for $\mathrm{w}\tau$. First we have $\exp(-W(x)) + \exp(-\frac{1}{x}W(x)) = \frac{W(x)}{x} + \exp(-\frac{1}{x}W(x)) \geq \frac{W(x)}{x} + (1 + -\frac{1}{x}W(x)) = 1$, and thus $W(x) \leq \mathrm{w}\tau(x)$. The upper bound follows similarly:

$$\begin{aligned}
&\exp(-\ln(\exp(W(x)) + 1)) + \exp(-\frac{1}{x}\ln(\exp(W(x)) + 1)) = \\
&\qquad \frac{1}{1+\exp(W(x))} + \exp(-\frac{1}{x}\ln(\exp(W(x)) + 1)) \leq \\
&\qquad \frac{1}{1+\exp(W(x))} + \frac{1}{1+\frac{1}{x}\ln(\exp(W(x)) + 1)} \leq \\
&\frac{1}{1+\exp(W(x))} + \frac{1}{1+\frac{1}{x}\ln(\exp(W(x)))} = \frac{1}{1+\exp(W(x))} + \frac{1}{1+\frac{1}{x}W(x)} = \\
&\qquad \frac{1}{1+\exp(W(x))} + \frac{1}{1+\frac{1}{\exp(W(x))}} = 1.
\end{aligned}$$

□

So $\mathrm{w}\tau(x)$ is asymptotically equal to $W(x)$. Indeed not just $\lim_{x\to\infty} \frac{\mathrm{w}\tau(x)}{W(x)} = 1$, but also $\lim_{x\to\infty} \mathrm{w}\tau(x) - W(x) = 0$. The best bounds on $W(x)$ seem to be as follows:

Theorem 3 ([12, Theorems 2.5, 2.7]). *For $x \in \mathbb{R}_{>1}$ holds:*

$$(\ln x - \ln\ln x + 1)\frac{\ln x}{1+\ln x} \leq W(x).$$

While for $x \in \mathbb{R}_{\geq e}$ holds:

$$W(x) \leq \ln x - \ln\ln x + \frac{e}{e-1}\frac{\ln\ln x}{\ln x}.$$

Thus $\lim_{x\to\infty} \frac{W(x)}{\ln(x)} = 1$, and so also $\lim_{x\to\infty} \frac{\mathrm{w}\tau(x)}{\ln(x)} = 1$. We summarise the elementary bounds for $\mathrm{w}\tau(x)$:

Theorem 4. *For $x \in \mathbb{R}_{>1}$ holds*

$$\max\big(\frac{\ln 4}{1+x^{-1}},\ (\ln x - \ln\ln x + 1)\frac{\ln x}{1+\ln x}\big) \le \mathrm{w}\tau(x) \le \min\big(\ln(2)\sqrt{x},\ \ln(\frac{x\cdot(1+\ln x)}{(\ln x - \ln\ln x + 1)\ln x} + 1)\big).$$

The first parts of the min- and max are used in the bounds only for very small x (less than 3; but for these x they are crucial), while for large x the second parts are (much) better (indeed very accurate). Using the lower bound of Theorem 4 to compute $\mathrm{w}\tau(x)$ via Newton-Raphson iteration (which will converge monotonically from below to $\mathrm{w}\tau(x)$, with quadratic convergence), yields very good results: The hardest cases are for say $x \le 1000$, with the maximum observed iterations until fixed-point (typically with one ulp precision, observed never more than two ulp) being six iterations, while the average case for usual SAT-solving seems to be around 3.5 iterations (always for full precision). Not using the bounds obtained by Lambert-W means for x in usual SAT ranges hundreds of iterations, and for very large x thousands of iterations.

We conclude by mentioning that in [14] an algorithm for computing the canonical order of branching tuples is stated, which aims at avoiding to compute the (l)tau-value explicitly, by using the characteristic functions to only perform Newton-Raphson iteration when needed. If this does not start with a good bound, then however this takes, as with the (l)tau-computation, many iterations. And for binary branching tuples, as we have seen above, the number of iterations is now, with the very good bounds, very low anyway, and thus such an approach seems only needed for non-binary branching tuples. For the best branching tuple found, one may want to compute the (l)tau-value anyway, due to its global value, making it possible to monitor the overall progress of the search.

5 On binary projections

Practical experience shows that in most cases the maximum-projection (i.e., target is to maximise) $(a,b) \in \mathcal{BT}_2 \mapsto a \cdot b$ is much better than $(a,b) \in \mathcal{BT}_2 \mapsto a+b$. Why is this the case? [16, Section 7.6] offers two explanations: On the one hand, based on Theorem 1 it is argued, that maximising the sum is like maximising a lower bound on the tree-size, while maximising the product is like minimising an upper bound on tree-size, which is intrinsically more meaningful; this argument makes sense, but is purely qualitatively. On the other hand, [16, Lemma 7.6.1] states that the approximation of $\mathfrak{T}(a,b)$ by $\sqrt{a\cdot b}$ is better than the approximation by $\frac{a+b}{2}$, measured in terms of differences; this does not take into account the scale of numbers, and is thus not a very precise argument. We have also the fact that the arithmetic mean is linear (convex and concave), while the geometric mean is concave, which fits better to the expected superlinear

growth. In this final section now we want to develop a more precise tool given by the "kernel" of a mean, and we show alternatives to sum, product and tau, by considering the p-mean for $0 \leq p \leq 1$ as some form of continuum between product and sum. The main argument why Corollary 1 possibly is not the final truth about binary projections is that the trees for concrete instances are not arbitrarily large (which is implicitly assumed in the proof of Corollary 1).

From the outset, one considers total quasi-orders $\prec$ on $\mathcal{BT}_2$ fulfilling (S), (T) and (D). By Corollary 1 we know that $\prec$ is $\leq_\tau$ (restricted to binary branchings) iff $\prec$ is compatible with every trivial comparison obtain by expansion, i.e., (E_2) holds. We now consider whether there are interesting such orders $\prec$ without (E_2). To get a handle on different orderings, we consider maps $m : \mathcal{BT}_2 \to \mathbb{R}_{>0}$, such that $m(a, b)$ is to be *maximised*. In order to be able to compare the numerical values of such projections m, we standardise them to form "means", as we will discuss now (that's why we consider maximisation here instead of minimisation, which was needed for the tau-function).

5.1 On means in general

The minimum requirements for a mean $m : \mathcal{BT}_2 \to \mathbb{R}_{>0}$ are:

1. $m(a, b) = m(b, a)$ (symmetry).
2. m is strictly increasing in each component.
3. $\min(a, b) \leq m(a, b) \leq \max(a, b)$ (consistency).

We additionally assume homogeneity here ("scale invariance"), that is, for $\lambda > 0$ holds $m(\lambda \cdot a, \lambda \cdot b) = \lambda \cdot m(a, b)$. As a further requirement, concavity is of importance, assuming that tree-growth is super-linear. We have mentioned three means already:

1. $m_0(a, b) := \sqrt{a \cdot b}$, the geometric mean: this is the default for SAT-solving (note that maximising $m_0(a, b)$ is equivalent to maximising $a \cdot b$).
2. $m_1(a, b) := \frac{a+b}{2}$, the arithmetic mean: this was the older default heuristic, and is typically still used for tie-braking.
3. $\mathfrak{T}(a, b) = \frac{\ln(2)}{\ln \tau(a,b)}$; we have $m_0(a, b) \leq \mathfrak{T}(a, b) \leq m_1(a, b)$.

A natural generalisation of m_0, m_1, as already considered in [16, Section 7.3.3] for obtaining bounds, are the p-means m_p for $p \in \mathbb{R}$. Since the sum is already bad enough (in most cases), in this initial study we do not go beyond m_1 — and indeed m_p is strictly convex iff $p > 1$. We also don't go below the geometric mean, as that seems not fruitful in general (though on selected families of benchmarks this might be different). So we restrict our attention to $0 \leq p \leq 1$:

1. $m_p(a, b) := (\frac{a^p+b^p}{2})^{1/p}$ for $0 < p \leq 1$.
2. $m_1(a, b)$ is the above arithmetic mean.
3. $m_p(a, b) \geq m_{p'}(a, b)$ for $p \geq p'$.
4. $\lim_{p \to 0} m_p(a, b) = m_0(a, b)$.

5.2 Comparing the various means by their kernels

Due to homogeneity, we can reduce means with two arguments to their "kernels", which are functions in just one argument:

Definition 8. *For a mean $m : \mathcal{BT}_2 \to \mathbb{R}_{>0}$ let the **kernel** $\overline{\mathrm{m}} : \mathbb{R}_{\geq 1} \to \mathbb{R}_{\geq 1}$ be defined as $\overline{\mathrm{m}}(x) := m(1,x)$ for $x \geq 1$.*

We assume in the following w.l.o.g. $0 < a \leq b$ (using symmetry). Due to homogeneity we have $m(a,b) = a \cdot \overline{\mathrm{m}}(\frac{b}{a})$. From $m \geq m'$ follows $\overline{\mathrm{m}} \geq \overline{\mathrm{m}}'$, and thus we have $\overline{\mathrm{m}}_1 \geq \overline{\mathrm{m}}_p \geq \overline{\mathrm{m}}_0$ as well as $\overline{\mathrm{m}}_1 \geq \overline{\mathfrak{T}} \geq \overline{\mathrm{m}}_0$. The meaning of the kernel is as follows:

- We measure the imbalance of $(a,b) \in \mathcal{BT}_2$ by the quotient $x := \frac{b}{a} \geq 1$ (the larger x, the greater the imbalance).
- $\frac{\overline{\mathrm{m}}(x)}{\overline{\mathrm{m}}(1)} = \overline{\mathrm{m}}(x)$ is the "reward" given for having $(1,x)$ instead of just $(1,1)$.
- Due to the standardisation via using means, we can indeed compare the kernel-values for different means p, q: if $\overline{p}(x) \geq \overline{q}(x)$, then the mean p gives a greater reward to x than the mean q.

The kernels of our means are as follows; we use the symbol $\sim$ here to denote asymptotic equality (i.e., the quotient approaches 1 as x goes to infinity):

1. $\overline{\mathrm{m}}_1(x) = \frac{1}{2} + \frac{x}{2} \sim \frac{x}{2}$.
2. $\overline{\mathrm{m}}_p(x) = (\frac{1}{2} + \frac{x^p}{2})^{1/p} \sim \frac{x}{2^{1/p}}$.
3. $\overline{\mathfrak{T}}(x) = \frac{\ln(2)}{\mathrm{w}\tau(1/x)} = \ln(2)\frac{x}{\mathrm{w}\tau(x)} \sim \ln(2)\frac{x}{W(x)} \sim \ln(2)\frac{x}{\ln(x)} = \frac{x}{\log_2(x)}$.
4. $\overline{\mathrm{m}}_0(x) = \sqrt{x}$.

m_1 gives the greatest reward, m_0 the smallest, while $m_p, \mathfrak{T}$ are incomparable in general, but there is a clear picture: There is a threshold value $p_0 \approx 0.307$ (the infimum of $0 \leq p \leq 1$ with $\overline{\mathfrak{T}} \leq \overline{\mathrm{m}}_p$), where for $p > p_0$ the mean m_p always gives a greater reward than $\mathfrak{T}$, while for $p < p_0$ first (i.e., small x) the reward given by m_p is smaller, and then greater than the reward given by $\mathfrak{T}$ (and the smaller p, the larger the first realm, being the whole range finally for $p = 0$).

This paper concentrates on the basic theory, but we can report on the very first experimental results. As look-ahead solver the simplest SAT-algorithm, the DLL algorithm ([6]) was used, with a modern implementation and branching heuristic as given by the `tawSolver` ([1]). The projections $\mathfrak{T}$ and m_p for $0 \leq p \leq 1$ were run on uniform random 3-SAT benchmarks, and compared by average size of the corresponding backtracking trees. The optimal p obtained was $p \approx 0.26$, which was still clearly worse here than $\mathfrak{T}$ (by 10%), while slighly better than m_0 (by 1%), and much better than m_1 (by 20%). Clearly the runtime for $\mathfrak{T}$ is higher, up to a total runtime twice as much compared to m_0, but for a stronger look-ahead solver (DLL indeed is the "zero-look-ahead look-ahead-solver"), which spends much more time on each variable, the tau-computation would contribute far less to the total runtime, while its influence could be much larger. Even for the `tawSolver`, on selected combinatorial benchmarks the effects (positive and negative) on tree-sizes can be much higher.

Our first general hypothesis on the (dynamic) choice of a good projection for binary branchings is, phrased in the terminology of means-functions: In general $\mathfrak{T}$ is best, but for large reductions (distances) or for nodes closer towards the leaves the penalty for imbalance can be reduced (there is no "exponential growth" anymore), moving towards m_1, while possibly closer to the root, or when only small reductions are achievable, it is conceivable that an increase in the penalty, moving closer to m_0, might be beneficial (the situation might get "out of control", and thus one needs to be more "cautious").

6 Summary and outlook

We provided a review of the general theory of branching heuristics for look-ahead-like solvers. Via flattening of metric trees, we gave a new simple proof of the main theorem on bounding tree sizes, and we were able to provide concise proofs showing the uniqueness of the canonical branching order. Turning to binary branchings, the core of the binary tau-function has been condensed into the function $\mathrm{w}\tau(x)$, where strong lower and upper bounds are given, showing $\mathrm{w}\tau(x) \sim \ln(x)$. That "core" (in a variation) yielded the kernel of generalised means, which enabled us to make precise comparisons between alternatives to standard projections for binary branchings. We derived a first general hypothesis on a better *dynamic* numerical control of the branching process.

Perhaps the most important future application of this whole approach is in improving Cube-and-Conquer (see [10] for a high-level overview). In general, having strong methods for splitting is vital here. The Cube-and-Conquer solver needs to use a strong branching scheme in the cubing-phase, creating a tree, where the leaves are the problems given to the conquer-solver. This scheme relies on the analysis of the tree created, with the main target of minimising tree size. More general than a lookahead SAT solver, the leaves here are not those nodes where the residual instances were "solved", but where they are "easy enough" for the conquer-solver. The target is still to minimise tree sizes, and for that the theory outlined in the paper is the very basis. The higher cost of computing $\tau(t)$ should be less relevant in this context.

A further aspect, which should become important in the future, is that for a branching-tuple t by $\tau(t)$ we obtain a *global* evaluation on the goodness of t ("global" in the sense of being comparable in principle over the whole course of computation). Monitoring these values should be valuable to gauge "success" or "failure" of the current strategy, possibly triggering a switch of methods, e.g., cutting off the cube-computation at the current node and switching to the conquer-solver (somewhat similar to random restarts for CDCL solvers).

References

1. Tanbir Ahmed, Oliver Kullmann, and Hunter Snevily. On the van der Waerden numbers $\mathrm{w}(2;3,t)$. *Discrete Applied Mathematics*, 174:27–51, September 2014. doi:10.1016/j.dam.2014.05.007.

2. Daniel Anderson, Pierre Le Bodic, and Kerri Morgan. Further results on an abstract model for branching and its application to mixed integer programming. *Mathematical Programming*, August 2020. doi:10.1007/s10107-020-01556-4.
3. Armin Biere, Marijn J.H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, February 2009.
4. Pierre Le Bodic and George L. Nemhauser. An abstract model for branching and its application to mixed integer programming. *Math. Program.*, 166(1-2):369–405, 2017. doi:10.1007/s10107-016-1101-8.
5. Teobaldo Bulhões, Ruslan Sadykov, and Eduardo Uchoa. A branch-and-price algorithm for the minimum latency problem. *Comput. Oper. Res.*, 93:66–78, 2018. doi:10.1016/j.cor.2018.01.016.
6. Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, July 1962. doi:10.1145/368273.368557.
7. Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010. doi:10.1007/978-3-642-16533-7.
8. Alexander Golovnev, Alexander S. Kulikov, Alexander V. Smal, and Suguru Tamaki. Gate elimination: Circuit size lower bounds and #SAT upper bounds. *Theoretical Computer Science*, 719:46–63, 2018. doi:10.1016/j.tcs.2017.11.008.
9. Marijn J. H. Heule and Hans van Maaren. Look-ahead based SAT solvers. In Biere et al. [3], chapter 5, pages 155–184. doi:10.3233/978-1-58603-929-5-155.
10. Marijn J.H. Heule and Oliver Kullmann. The science of brute force. *Communications of the ACM*, 60(8):25–34, August 2017. doi:10.1145/3107239.
11. Marijn J.H. Heule, Oliver Kullmann, and Armin Biere. Cube-and-conquer for satisfiability. In Youssef Hamadi and Lakhdar Sais, editors, *Handbook of Parallel Constraint Reasoning*, chapter 2, pages 31–59. Springer, 2018. doi:10.1007/978-3-319-63516-3_2.
12. Abdolhossein Hoorfar and Mehdi Hassani. Inequalities on the Lambert W function and hyperpower function. *Journal on Inequalities in Pure and Applied Mathematics*, 9(2):1–5, 2008. URL: https://www.emis.de/journals/JIPAM/article983.html.
13. Iosif Pinelis (https://mathoverflow.net/users/36721/iosif pinelis). A certain generalisation of the golden ratio. MathOverflow. URL: https://mathoverflow.net/q/320595.
14. Donald E. Knuth. *The Art of Computer Programming, Volume 4: Satisfiability (Fascicle 6)*. Addison-Wesley, 2015. ISBN-13 978-0134397603.
15. Oliver Kullmann. Obere und untere Schranken für die Komplexität von aussagenlogischen Resolutionsbeweisen und Klassen von SAT-Algorithmen. Master's thesis, Johann Wolfgang Goethe-Universität Frankfurt am Main, April 1992. (Upper and lower bounds for the complexity of propositional resolution proofs and classes of SAT algorithms (in German); Diplomarbeit am Fachbereich Mathematik).
16. Oliver Kullmann. Fundaments of branching heuristics. In Biere et al. [3], chapter 7, pages 205–244. doi:10.3233/978-1-58603-929-5-205.
17. Joao Marques-Silva, Ines Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Biere et al. [3], chapter 4, pages 131–153. doi:10.3233/978-1-58603-929-5-131.

18. Frank W.J. Olver, Daniel W. Lozier, Ronald F. Boisvert, and Charles W. Clark, editors. *NIST Handbook of Mathematical Functions*. NIST and Cambridge University Press, 2010. ISBN 978-0-521-19225-5.
19. Diego Pecin, Artur Alves Pessoa, Marcus Poggi, and Eduardo Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. *Math. Program. Comput.*, 9(1):61–100, 2017. doi:10.1007/s12532-016-0108-8.