Projection heuristics for binary branchings between sum and product

Oliver Kullmann   Oleg Zaikin

Computer Science Department
Swansea University

SAT 2021
July 8, 2021

SAT solvers and branching trees I

- We consider SAT solvers based on either
  - Look-Ahead (LA)
  - Cube-and-Conquer (C&C) $\approx$ LA + CDCL ("old and new").
- Such solvers build a branching (backtracking) tree, with either solved instances at the leaves (LA), or with instances for the conquer-solver (C&C).
- One important heuristic target is to minimise tree-size.

SAT solvers and branching trees II

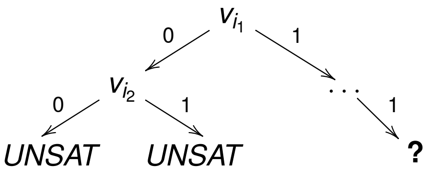
- Default branchings are binary (on boolean variables), but can be larger by taking more variables into account (each individual binary branching might be weak, but the whole might be strong).

A whole tree T can be considered
as a single branching (leaving out intermediate nodes).
We call this process "flattening".

To minimise tree size, the best branching
should be chosen for every step.

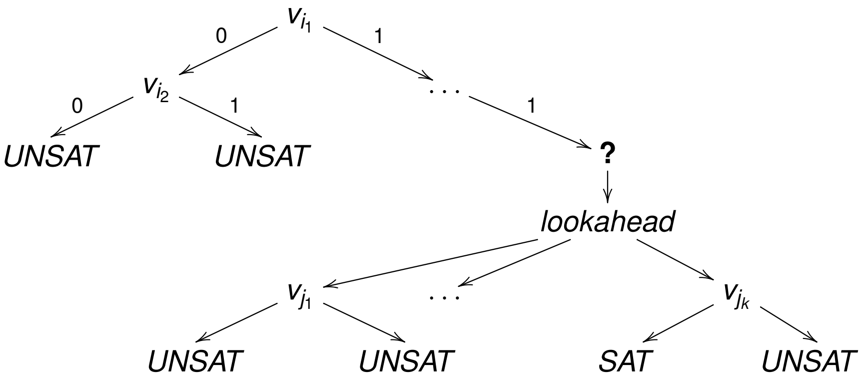
Branching tree for LA

- Branching variables are chosen via lookahead.
- Non-binary branchings are possible.
- Types of leaves: UNSAT, SAT.



Branching tree for LA

- Branching variables are chosen via lookahead.
- Non-binary branchings are possible.
- Types of leaves: UNSAT, SAT.

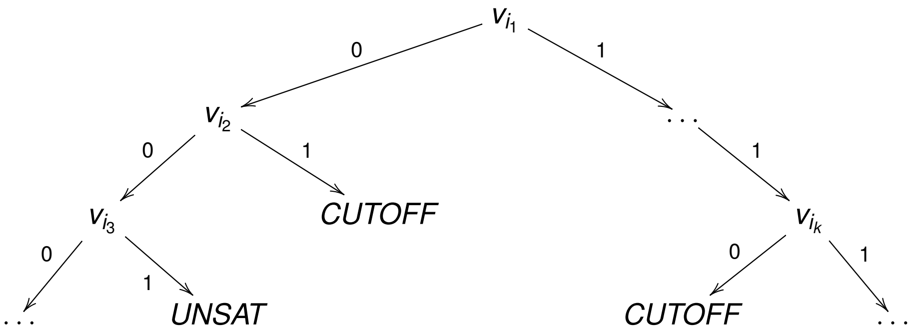


Branching tree for C&C

Cube cubes are produced by LA solver.

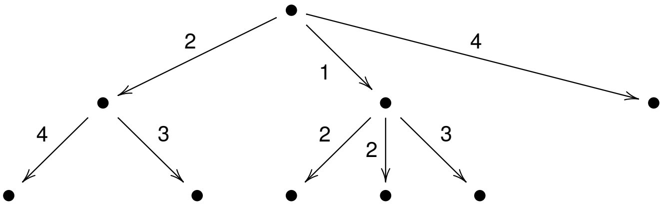
Conquer cubes are solved by specialised solver, by default CDCL.

Now a leaf can be a cutoff-point.



Trees and distances

The **final** branching tree might look like



The edges (branches) are now labelled with **positive real numbers** ("distances"), showing progress.

- For each branching, a LA solver considers a list of possibilities, based on "looking-ahead and measuring".
- In general distances can be arbitrary positive real numbers (important for optimisation).

Branching tuples I

- To every potential branch, a **distance** is considered, a positive real number measuring **progress**.
- Branchings are compared by comparing their **branching tuples** (the tuple of distances).
- The previous tree contains the branching tuples $(2, 1, 4)$, $(4, 3)$, $(2, 2, 3)$. Flattening the whole tree yields $(6, 5, 3, 3, 4, 4)$.
- A major basic question is:

HOW TO COMPARE BRANCHING TUPLES?

Branching tuples II

- Choices for distances are:
  - number of variables eliminated
  - number of new clauses (weighted by clause-width).
- In this talk we will consider the distances as given:

  The theory works for ARBITRARY distances.

Branching tuples

A **branching tuple** is a tuple $a = (a_1, \ldots, a_k)$ of positive real numbers, with $k \geq 2$.

We use $|a| := k$ for the width.

The set of all branching tuples is

$$\mathcal{BT} := \bigcup_{k=2}^{\infty} (\mathbb{R}_{>0})^k.$$

Simple comparisons

The task is compare branching tuples,
finding out which is "better".

We write $\boldsymbol{a} \prec \boldsymbol{b}$ for "$a \in \mathcal{BT}$ is strictly better than $b \in \mathcal{BT}$".

Let's start simple:

1. $(1, 2, 3) \prec (1, 2, 2)$ — at least one component better.
2. $(3, 2, 1) \prec (1, 2, 2)$ — order doesn't matter.
3. $(1, 2) \prec (1, 2, 3)$ — widening always impairs.
4. $(2, 1) \prec (1, 2, 3)$ — again, order doesn't matter.
5. $(2, 2) \prec (1000, 2, 1)$ — combining both methods.

Trivially better

The order-relation $\boldsymbol{a} \lneq \boldsymbol{b}$

"a is trivially smaller than b"

holds for $a, b \in \mathcal{BT}$ if the following three conditions are fulfilled:

1. $|a| \le |b|$
2. there is a permutation a' of a with $\forall\, i \in \{1, \dots, |a|\} : a'_i \ge b_i$
3. either $|a| < |b|$ or there is $i \in \{1, \dots, |a|\}$ with $a'_i > b_i$.

So all the previous examples for "$a \prec b$" were examples for $a \lneq b$.

Reminders:

- "Smaller" here means "better" (smaller tree sizes).
- Smaller branchings have greater distances.
- Additional branches are "bad".

Basic axioms for comparing branching tuples

We are seeking to determine a total quasi-order $\preceq$ on $\mathcal{BT}$:

- $a \preceq b$ means "a is better or equal than b".
- "Total quasi-order" means:
  - $a \preceq a$
  - $a \preceq b \wedge b \preceq c \Rightarrow a \preceq c$
  - $a \preceq b \vee b \preceq a$.

As usual we define

- $a \simeq b$ if $a \preceq b$ and $b \preceq a$ ($\simeq$ is an equivalence relation on $\mathcal{BT}$),
- $a \prec b$ if $a \preceq b$ and $a \not\simeq b$.

The following two axioms should be indisputable (for all $a, b \in \mathcal{BT}$):

(S) Symmetry For a permutation b of a holds $a \simeq b$.

(T) Trivial comparison If $a \lneq b$ then $a \prec b$.

A harder case

What about comparing

$$(2,3) \text{ vs } (1,4) \text{ ?}$$

We could invoke here another principle:

"superlinear" growth.

Remember the numbers are changes (indeed reductions) in size (from the same (residual) instance):

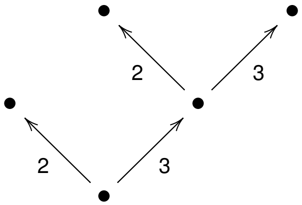
1. Having "linear" growth means exactly that all tuples with the same sum of entries would be equivalent.
2. "Superlinear" means that the bad branch here counts more than the good branch, and thus we should have

$$(2,3) \prec (1,4).$$

But what about $(2,2)$ vs $(1,4)$?

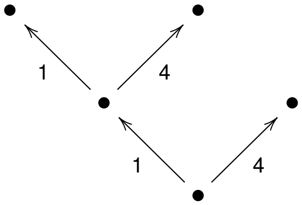
Making growth appearing I

Growth can be made explicit by *expansion* and subsequent *flattening*:
Expand $(2,3)$ to



$\rightsquigarrow (2,5,6)$.

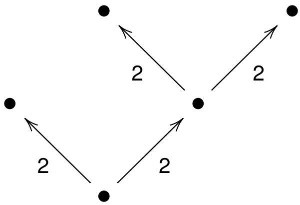
Expand $(1,4)$ to



$\rightsquigarrow (2,5,4)$.

Making growth appearing II

Expand $(2,2)$ to



$\rightsquigarrow (2,4,4)$.

So

- $(2,3) \rightsquigarrow (2,5,6)$
- $(1,4) \rightsquigarrow (2,5,4)$
- $(2,2) \rightsquigarrow (2,4,4)$

Since

$$(2,5,6) \lneq (2,5,4) \lneq (2,4,4)$$

we should have

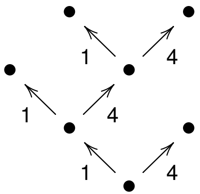
$$(2,3) \prec (1,4) \prec (2,2).$$

Expansions

An **expansion** of a branching tuple *a* is any branching tuple *b* obtained by

1. considering the tuple *a* as a branching constituting the root of a tree, with $|a|$ many leaves;
2. replacing leaves iteratively by branching *a* (arbitrarily often);
3. finally flattening the whole tree (with edges labelled by the numbers of *a*) into one branching tuple *b*.

We have already seen three (one-step) expansions. Here is a two-step expansion $(2, 6, 9, 4)$ of $(1, 4)$:



$$\rightsquigarrow (2, 5, 4) \rightsquigarrow (2, 5 + 1, 5 + 4, 4) = (2, 6, 9, 4).$$

The Expansion Axiom I

For the order $\preceq$ on $\mathcal{BT}$ we are seeking to axiomatically determine, we require now a third axiom (recall the basic two axioms, (S) (Symmetry) and (T) (Trivial comparison)):

(E) Expansion If b is an expansion of a, then $b \simeq a$.

The Expansion Axiom II

Theorem 2.1

For all $a, b \in \mathcal{BT}$ exactly one of the following four cases holds:

(i) *There are expansions a' of a and b' of b with $a' \lneq b'$.*

(ii) *Same as (i), but with $b' \lneq a'$.*

(iii) *Same as (i), but with a' being a permutation of b'.*

(iv) *For any expansions a', b' of a, b neither $a' \lneq b'$ nor $b' \lneq a'$ holds, nor is a' a permutation of b'.*

By (S), (T), (E) we get:

Case (i) $a \prec b$

Case (ii) $b \prec a$

Case (iii) $a \simeq b$.

The Density Axiom

The order $\preceq$ is not determined in Case (iv):

- We could just say that $a \simeq b$ holds for Case (iv).
- But we can indeed conclude this from another intuitive axiom.
- The expansions are somewhat similar to decimal expansions of a number: they get closer and closer to reveal the true “value” of a branching tuple.
- We can indeed capture this by requiring that between $a \prec b$ there is always some c with $a \prec c \prec b$, where $c = a - \varepsilon$ for some $\varepsilon \in \mathbb{R}_{>0}$ (meaning: subtract ε from every component of a).

(D) Density For $a \prec b$ there is $\varepsilon > 0$ such that $a - \varepsilon \in \mathcal{BT}$ and $a - \varepsilon \prec b$.

The Canonical Branching order

Call a total quasi-order $\preceq$ on $\mathcal{BT}$ fulfilling (S), (T), (E), (D) a

canonical branching order.

Theorem 2.2

There is exactly one canonical branching order.

We now turn to concretely determine the canonical branching order.

(This also will show that comparisons $a \prec b$, $a \preceq b$ and $a \simeq b$ are decidable in polynomial time for a, b made from algebraic numbers.)

Measuring branching tuples

We define $\boldsymbol{\tau} : \mathcal{BT} \to \mathbb{R}_{>1}$ by

- $\tau(a, b)$ is the unique $x > 1$ with $x^{-a} + x^{-b} = 1$.
- $\tau(a, b, c)$ is the unique $x > 1$ with $x^{-a} + x^{-b} + x^{-c} = 1$.
- And so on.

This generalises the so-called "characteristic polynomial" of difference and differential equations.
The tau-function evaluates branching tuples (the smaller the tau-value the better the branching tuple), and the derived ordering of branching tuples is the canonical branching order:

Theorem 2.3

For $a, b \in \mathcal{BT}$ holds $a \preceq b \Leftrightarrow \tau(a) \leq \tau(b)$.

Variants of the tau-function

For numeric purposes **log-tau $\mathrm{l}\boldsymbol{\tau}$** $: \mathcal{BT} \to \mathbb{R}_{>0}$ is more appropriate:

$$\mathrm{l}\tau(a) := \ln(\tau(a)).$$

Obviously $\tau(a) \le \tau(b) \Leftrightarrow \mathrm{l}\tau(a) \le \mathrm{l}\tau(b)$.

In a sense log-tau computes a sort of "mean value" of a branching tuple — we get indeed a proper form of a mean value, called **mean-tau $\mathfrak{T}$** $: \mathcal{BT} \to \mathbb{R}_{>0}$ by taking the reciprocal value and scaling:

$$\mathfrak{T}(a) := \frac{\ln(|a|)}{\mathrm{l}\tau(a)}.$$

Binary tau

Currently for SAT most important is binary log-tau:

$$\mathrm{l}\tau(x,y) := \mathrm{l}\tau((x,y)).$$

We get a very good handle on $\mathrm{l}\tau(x,y)$ by reducing it to **w-tau** $\mathbf{w\tau} : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$, a function with just one argument:

$$\begin{aligned} \mathrm{w}\tau(x) &:= \mathrm{l}\tau(1, \frac{1}{x}) \\ \mathrm{l}\tau(x,y) &= \frac{1}{x}\,\mathrm{w}\tau(\frac{x}{y}). \end{aligned}$$

W-tau is asymptotically equal to the Lambert-W function, which in turn is asymptotically equal to the logarithm:

$$\begin{aligned} |\mathrm{w}\tau(x) - W(x)| &= o(1) \\ |W(x) - \ln(x)| &= O(1) \end{aligned}$$

Warning: These approximations are bad for (relevant) "small" x.

Alternatives to binary tau?

Given a binary branching tuple $(x, y) \in \mathcal{BT}_2 := (\mathbb{R}_{>0})^2$, we determine its “value” by $\tau(x, y)$ (or $\mathrm{l}\tau(x, y)$):

The proof that this is the only way
hinges on having arbitrarily large trees
containing (only) (x, y).

But this is not applicable to inputs (or residual instances) of finite size — can (should?) we take this into account?!?
For the remainder our projections (maps from $\mathcal{BT}_2$ to $\mathbb{R}_{>0}$) are to be maximised (the larger the better).

Historically

1. first the “projection” $(x, y) \mapsto x + y$ was used (sum rule)
2. which was soon replaced by $(x, y) \mapsto x \cdot y$ (product rule).

An axiomatic framework for binary projections

Starting from the (generalised) mean $\mathfrak{T} : \mathcal{BT} \to \mathbb{R}_{>0}$, we consider the following seven axioms for binary projections $m : \mathcal{BT}_2 \to \mathbb{R}_{>0}$, now using "the larger the better" (for all $x, y, x', y' \in \mathbb{R}_{>0}$):

(i) $m(x, y) = m(y, x)$ (symmetry)

(ii) m strictly increasing in each component

(iii) $\min(x, y) \leq m(x, y) \leq \max(x, y)$ (consistency)

(iv) $\lambda > 0 \Longrightarrow m(\lambda \cdot x, \lambda \cdot y) = \lambda \cdot m(x, y)$ (homogeneity – scale invariance)

(v) $0 \leq \lambda \leq 1 \Longrightarrow m(\lambda \cdot x + (1-\lambda) \cdot x', \lambda \cdot y + (1-\lambda) \cdot y') \geq \lambda \cdot m(x, y) + (1-\lambda) \cdot m(x', y')$ (concavity)

(vi) $\lim_{x \to \infty} m(x, y) = \infty$ (∞-dominated)

(vii) $\lim_{x \to 0} m(x, y) = 0$ (0-dominated).

$\mathfrak{T}_2$ fulfils these seven axioms.

Power means

The first four axioms are fulfilled by the power means $m_p : \mathcal{BT}_2 \to \mathbb{R}_{>0}$, defined for $-\infty \leq p \leq +\infty$ as follows:

1. $m_{-\infty}(x, y) = \min(x)$ (the "most pessimistic choice")
2. $m_{+\infty}(x, y) = \max(x)$ (the "most optimistic choice")
3. $m_0(x, y) = \sqrt{x \cdot y}$ (the geometric mean)
4. otherwise $m_p(x, y) = (\frac{1}{2}(x^p + y^p))^{1/p}$.

m_1 is the arithmetic mean, m_{-1} is the harmonic mean.

The choice of m_1 corresponds to the "sum-rule", the choice of m_0 to the "product rule".

- m_p is concave iff $p \leq 1$.
- m_p is ∞-dominated iff $p \geq 0$.
- m_p is 0-dominated iff $p \leq 0$.

So m_0 is the only power means fulfilling all seven axioms.

Reducing binary means to their kernels

For comparing means $m : \mathcal{BT}_2 \to \mathbb{R}_{>0}$, thanks to symmetry and homogeneity we can restrict attention to their "kernels"

$$\overline{m} : \mathbb{R}_{\geq 1} \to \mathbb{R}_{\geq 1}$$

defined by

$$\overline{m}(x) := m(1, x).$$

$\overline{m}(x)$ says how much imbalanced branching tuples are penalised — the large $\overline{m}(x)$ the less penalisation.

For $0 \leq p \leq 1$:

1. $\overline{m_1}(x) = \frac{1}{2} + \frac{x}{2}$ penalises the least.
2. $\overline{m_0}(x) = \sqrt{x}$ penalises the most.

$\overline{\mathfrak{T}}(x) \sim \frac{x}{\log_2(x)}$ penalises less than m_0, and more than m_p for $p \leq 0.307$.

Future research

1. Develop a dynamic binary projection, which takes the changing scales into account: more penalisation towards the root, less towards the leaves.
2. Generalise the results for binary-tau to non-binary tau.
3. Make use of the numeric values of the tau-function (not just using it for comparison) — the tau-value yields a global evaluation of the current state of the search.

End

(references on the remaining slides).

For my papers see
http://cs.swan.ac.uk/~csoliver/papers.html.

Bibliography I

[1] Oliver Kullmann. Fundaments of branching heuristics. In 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*, chapter 7, pages 205–244. IOS Press, February 2009. ISBN 978-1-58603-929-5. doi:10.3233/978-1-58603-929-5-205.