

# Overview

Introduction

Tractable cases

DPLL algorithms

- Monien-Speckenmeyer algorithm

- Kullmann's method

- Algorithm of Zhang

CDCL solvers

Lookahead-based solvers

Probabilistic algorithms

Certification

Applications

# The basic DPLL algorithm

After *Davis, Putnam, Logemann and Loveland*, 1960

The basic DPLL Algorithm:

$DPLL(F, \alpha)$

if  $F\alpha = 0$  then return UNSAT

if  $F\alpha = 1$  then return  $\alpha$

pick  $x \in V(F\alpha)$

$\beta := DPLL(F, \alpha \cup [x := 0])$

if  $\beta \neq \text{UNSAT}$

then return  $\beta$

else return  $DPLL(F, \alpha \cup [x := 1])$

## A simple analysis

For a  $k$ -CNF formula  $F$ , iterate the following:

- ▶ pick a clause  $C$  in  $F$
- ▶ branch successively on the  $k$  variables in  $V(C)$

Of the  $2^k$  assignments to  $V(C)$ , one sets  $\alpha(C) = \alpha(F) = 0$ .

Thus:  $2^k - 1$  branching tree of height  $n/k$ .

Runtime is essentially the tree-size  $b_k^n$  for  $b_k := \sqrt[k]{2^k - 1} < 2$

$k$	3	4	5	6	7	8
$b_k$	1.91294	1.96799	1.98735	1.99477	1.99777	1.99903

# Unit propagation

A branching strategy (almost) always employed:

- ▶ pick  $x$  if  $x$  occurs in a unit clause  $a$ .

One branch immediately fails

$\rightsquigarrow$  just set  $[a := 1]$  instead of branching

Mostly realized as simplification step before branching:

`UnitProp( $F, \alpha$ )`

  while  $F\alpha$  contains unit clause  $a$

$\alpha := \alpha \cup [a := 1]$

  return  $\alpha$

## Pure literals and subsumption

Other simplification steps in original DPLL:

- ▶ elimination of pure literals
- ▶ deletion of subsumed clauses

PureLit( $F, \alpha$ )

```
while  $F\alpha$  contains pure literal  $a$   
   $\alpha := \alpha \cup [a := 1]$ 
```

Clause  $C$  subsumes  $D$  if  $C \subseteq D$

Subs( $F$ )

```
while  $F$  contains clauses  $C \subseteq D$   
   $F := F \setminus D$ 
```

# The general DPLL algorithm

DPLL( $F, \alpha$ )

  simplify( $F, \alpha$ )

  if  $F = 0$  then return UNSAT

  if  $F = 1$  then return  $\alpha$

  pick  $x \in V(F)$  and  $\epsilon \in \{0, 1\}$

$\beta := \text{DPLL}(F[x := \epsilon], \alpha \cup [x := \epsilon])$

  if  $\beta \neq \text{UNSAT}$

    then return  $\beta$

    else return  $\text{DPLL}(F[x := \bar{\epsilon}], \alpha \cup [x := \bar{\epsilon}])$

# Monien-Speckenmeyer algorithm - simple version

Branching strategy:

- ▶ pick literal from a shortest clause

Equivalently:

`simpleMS( $F, \alpha$ )`

`if  $F\alpha = 0$  then return UNSAT`

`if  $F\alpha = 1$  then return  $\alpha$`

`pick shortest clause  $C = a_1 \vee \dots \vee a_r$`

`for  $i := 1$  to  $r$  do`

`$\gamma_i := [a_1 := 0, \dots, a_{i-1} := 0, a_i := 1]$`

`$\beta := \text{simpleMS}(F, \alpha \cup \gamma_i)$`

`if  $\beta \neq \text{UNSAT}$  then return  $\beta$`

`return UNSAT`

## Analysis of simpleMS

At each node:  $r$ -fold branching for some  $r \leq k$ ,  
in  $i$ th branch  $n - i$  variables.

Recursion for the tree-size:

$$T(n) := T(n-1) + \dots + T(n-k)$$

Set  $T(n) := b^n$ , thus the basis  $b_k = b$  satisfies

$$b^k := b^{k-1} + \dots + b + 1$$

The solutions for small  $k$  are:

$k$	3	4	5	6	7	8
$b_k$	1.83929	1.92757	1.96595	1.98359	1.99197	1.99603

# Autarkies

An assignment  $\alpha$  is **autark** for  $F$  if  $F\alpha \subseteq F$ .

Generalizes pure literals:

- ▶ Assignment  $[a := 1]$  is autark for  $F$  iff  $a$  is pure in  $F$ .

## Property

If  $\alpha$  is autark for  $F$ ,  
then  $F$  is satisfiable iff there is  $\beta \supseteq \alpha$  with  $\beta \models F$ .

# Monien-Speckenmeyer algorithm

$MS(F, \alpha)$

if  $F\alpha = 0$  then return UNSAT

if  $F\alpha = 1$  then return  $\alpha$

pick shortest clause  $C = a_1 \vee \dots \vee a_r$

for  $i := 1$  to  $r$  do

$\gamma_i := [a_1 := 0, \dots, a_{i-1} := 0, a_i := 1]$

    if  $\gamma_i$  autark for  $F\alpha$  then return  $MS(F, \alpha \cup \gamma_i)$

for  $i := 1$  to  $r$  do

$\beta := MS(F, \alpha \cup \gamma_i)$

    if  $\beta \neq \text{UNSAT}$  then return  $\beta$

return UNSAT

# Analysis of Monien-Speckenmeyer

$T(n)$ : tree-size for formulas in  $n$  variables

$T'(n)$ : tree-size for formulas in  $n$  variables  
with a clause  $C$  of width  $w(C) \leq k - 1$

$$T(n) = \max(T(n-1), T'(n-1) + \dots + T'(n-k))$$

$$T'(n) = \max(T(n-1), T'(n-1) + \dots + T'(n-k+1))$$

## Lemma

$T(n) \leq T'(n) + T'(n-1)$  for every  $n$ .

Setting  $T'(n) = b^n$  yields:  $b^{k-1} = b^{k-2} + \dots + b + 1$

and  $T(n) = O(b^n)$  for  $b = b_k$ :

$k$	3	4	5	6	7	8
$b_k$	1.61804	1.83929	1.92757	1.96595	1.98359	1.99197

## Branching tuples

For a vector  $(d_1, \dots, d_m)$  with  $m \geq 1$  and  $d_i > 0$   
let  $\tau(d_1, \dots, d_m)$  be the unique positive solution to:

$$x^{-d_1} + \dots + x^{-d_m} = 1$$

For a tree  $T$  with edge valuation  $d$ :

- ▶ for a vertex  $v$  with children  $w_1, \dots, w_m$ :

$$d(v) = (d_1, \dots, d_m) \text{ where } d_i = d(v, w_i).$$

- ▶ branching number  $\tau(v) := \tau(d(v))$
- ▶  $\tau(T) := \max_v(\tau(v))$  over the inner vertices  $v$
- ▶ for a path  $p = v_1, \dots, v_\ell$  let  $d(p) := \sum_{i=1}^{\ell-1} d(v_i, v_{i+1})$
- ▶  $d(T) := \max_p d(p)$  over all paths from the root to a leaf

# Analysing DPLL with branching tuples

## Theorem

*The number of leaves in  $T$  is at most  $\tau(T)^{d(T)}$ .*

To analyse a DPLL algorithm:

find distance  $d$  function for the recursion tree s.t.

- ▶  $d(T) \leq n$
- ▶  $\tau(T)$  is minimized

## simpleMS revisited

To analyse simpleMS for 3-SAT, define:

$d(v, w) :=$  number of variables set.

$d(p) \leq n$  for every path.

Inner vertex  $v$  has 3 children of distance 3, 2, 1 resp.

Thus:  $\tau(v) = \tau(3, 2, 1)$  is the solution of

$$x^{-3} + x^{-2} + x^{-1} = 1$$

Multiplying by  $x^3$  yields  $x^3 = x^2 + x + 1$  as before

Thus  $\tau(T) = \tau(v) = 1.83929\dots$

# Analysing Monien-Speckenmeyer

There are three types of inner vertices:

- ▶ Type 1: autark assignment:

$v$  has one child of distance 1:

$$\tau(v) = \tau(1) = 1$$

- ▶ Type 2: after non-autark assignment:

$v$  has 2 children of distance 2 and 1:

$$\tau(v) = \tau(2, 1) = 1.61804 \dots$$

- ▶ Type 3: after autark assignment:

$v$  has 3 children of distance 3, 2 and 1:

$$\tau(v) = \tau(3, 2, 1) = 1.83929 \dots$$

↪ no improvement over simpleMS.

# Analysing Monien-Speckenmeyer

Vertex  $v$  of type 3 only as child of  $v'$  of type 1.

Idea: merge  $v$  and  $v'$  together to one vertex  $w$   
 $w$  has 3 children of distance 4, 3 and 2:

$$\tau(w) = \tau(4, 3, 2) = 1.46558\dots$$

Thus  $\tau(T) = \tau(2, 1) = 1.61804\dots$

Alternative preserving tree structure: redefine distances

- ▶ for type 1:  $d(v, v') = 1 - \epsilon$
- ▶ for type 3:  $d(v, w_i) = i + \epsilon$  for  $i = 1, 2, 3$

$d(p)$  remains unchanged for every path  $p$ .

Now for  $\epsilon = 0.5$  we have for  $v$  of type 3:

$$\tau(v) = \tau(3.5, 2.5, 1.5) = 1.59074\dots < \tau(2, 1).$$

# Properties of branching numbers

- ▶ If  $e > d_1$ , then  $\tau(e, d_2, \dots, d_m) < \tau(d_1, \dots, d_m)$ .
- ▶ If  $d_1 + d_2 = e_1 + e_2$ , and  $\min(d_1, d_2) \geq \min(e_1, e_2)$ , then
$$\tau(d_1, \dots, d_m) \leq \tau(e_1, e_2, d_3, \dots, d_m)$$
where equality only holds if it holds in the premise.
- ▶ Let  $d = (d_1, \dots, d_m)$  and  $e := (e_1, \dots, e_n)$ and define  $d^* := (d_1 + e_1, \dots, d_1 + e_n, d_2, \dots, d_m)$ .Then:
  - ▶ if  $\tau(d) \leq \tau(e)$ , then  $\tau(d) \leq \tau(d^*) \leq \tau(e)$ .
  - ▶ if  $\tau(d) \geq \tau(e)$ , then  $\tau(d) \geq \tau(d^*) \geq \tau(e)$ .

Both inequalities are strict if those in the premise are.