

SAT and SMT Solving

Sarah Winkler

SS 2018

Department of Computer Science
University of Innsbruck

Outline

- Summary of Last Week
- Maximum Satisfiability
- Algorithms for Maximum Satisfiability

1

Summary of Last Week

Definition (Implication Graph)

For derivation $\| F \xRightarrow{*}_B M \| F'$ **implication graph** is constructed as follows:

- ▶ add node labelled l for every decision literal l in M
- ▶ repeat until there is no change:
 - if \exists clause $l_1 \vee \dots \vee l_m \vee l'$ in F' such that there are already nodes l_1^c, \dots, l_m^c
 - ▶ add node l' if not yet present
 - ▶ add edges $l_i^c \rightarrow l'$ for all $1 \leq i \leq m$ if not yet present
 - if \exists clause $l'_1 \vee \dots \vee l'_k$ in F' such that there are nodes l_1^c, \dots, l_k^c
 - ▶ add conflict node labeled C
 - ▶ add edges $l_i^c \rightarrow C$

Definitions

- ▶ **cut** of implication graph has at least all decision literals on the left, and at least the conflict node on the right
- ▶ literal l in implication graph is **unique implication point (UIP)** if all paths from last decision literal to conflict node go through l

Lemma

if edges intersected by cut are $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$ then $F' \models l_1^c \vee l_k^c$

2

Backjump Clauses by Resolution

- ▶ set C_0 to conflict clause
- ▶ let l be last assigned literal such that l^c is in C_0
- ▶ while l is no decision literal:
 - ▶ C_{i+1} is resolvent of C_i and clause D that led to assignment of l
 - ▶ let l be last assigned literal such that l^c is in C_{i+1}

Observation

every C_i corresponds to cut in implication graph

3

Definition (DPLL with Learning and Restarts)

DPLL with learning and restarts \mathcal{R} extends system \mathcal{B} by following three rules:

- ▶ **learn** $M \parallel F \implies M \parallel F, C$
if $F \models C$ and all atoms of C occur in M or F
- ▶ **forget** $M \parallel F, C \implies M \parallel F$
if $F \models C$
- ▶ **restart** $M \parallel F \implies \parallel F$

Theorem (Termination)

any derivation $\parallel F \implies_{\mathcal{R}} S_1 \implies_{\mathcal{R}} S_2 \implies_{\mathcal{R}} \dots$ is finite if

- ▶ it contains no infinite subderivation of learn and forget steps, and
- ▶ restart is applied with increasing periodicity

Theorem (Correctness)

for $\parallel F \implies_{\mathcal{R}} S_1 \implies_{\mathcal{R}} S_2 \implies_{\mathcal{R}} \dots \implies_{\mathcal{R}} S_n$ with final state S_n :

- ▶ if $S_n = \text{FailState}$ then F is unsatisfiable
- ▶ if $S_n = M \parallel F'$ then F is satisfiable and $M \models F$

4

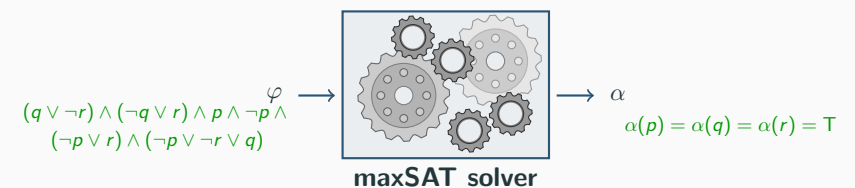
Maximum Satisfiability

maxSAT

maxSAT Problem

input: propositional formula φ in CNF

output: valuation α such that α satisfies maximal number of clauses in φ



Terminology

- ▶ **optimization problem** P asks to find “best” solution among all solutions
- ▶ **maxSAT encoding** transforms optimization problem P into formula φ such that “best” solution to P is obtained from maxSAT solution to φ

5

Remark

many real world problems have optimization component

Examples

- ▶ find **shortest** path/execution to goal state
 - ▶ planning, model checking
- ▶ find **smallest** explanation
 - ▶ debugging, configuration, ...
- ▶ find **least resource-consuming** schedule
 - ▶ scheduling, logistics, ...
- ▶ find **most probable** explanation
 - ▶ probabilistic inference, ...

6

Variants of Maximal Satisfiability

Consider CNF formula φ as set of clauses, denote number of clauses by $|\varphi|$.

Maximal Satisfiability (maxSAT)

instance: CNF formula φ

question: what is maximal $|\psi|$ such that $\psi \subseteq \varphi$ and $\bigwedge_{C \in \psi} C$ is satisfiable?

Partial Maximal Satisfiability (pmaxSAT)

instance: CNF formulas χ and φ

question: what is maximal $|\psi|$ such that $\psi \subseteq \varphi$ and $\chi \wedge \bigwedge_{C \in \psi} C$ is satisfiable?

Example

$$\varphi = \{ \begin{matrix} 6 \vee 2, & \bar{6} \vee 2, & \bar{2} \vee 1, & \bar{1}, & \bar{6} \vee 8, & 6 \vee \bar{8}, \\ 2 \vee 4, & 4 \vee 5, & 7 \vee 5, & \bar{7} \vee 5, & 3, & 5 \vee 3 \end{matrix} \}$$

$$\chi = \{ \bar{1} \vee 2, \quad \bar{2} \vee \bar{3}, \quad \bar{5} \vee 1, \quad 3 \}$$

- ▶ $\text{maxSAT}(\varphi) = 10$, e.g. for valuation $\bar{1} 2 \bar{3} 4 5 6 \bar{7} 8$
- ▶ $\text{pmaxSAT}(\chi, \varphi) = 8$, e.g. for valuation $\bar{1} \bar{2} 3 4 \bar{5} 6 7 8$

7

Weighted Maximal Satisfiability (maxSAT_w)

instance: CNF formula φ with weight $w_C \in \mathbb{Z}$ for all $C \in \varphi$

question: what is maximal $\sum_{C \in \psi} w_C$ for $\psi \subseteq \varphi$ and $\bigwedge_{C \in \psi} C$ satisfiable?

Weighted Partial Maximal Satisfiability (pmaxSAT_w)

instance: CNF formulas φ and χ , with weight $w_C \in \mathbb{Z}$ for all $C \in \varphi$

question: what is maximal $\sum_{C \in \psi} w_C$ for $\psi \subseteq \varphi$ and $\chi \wedge \bigwedge_{C \in \psi} C$ satisfiable?

Notation

- ▶ write $\text{maxSAT}(\varphi)$ and $\text{maxSAT}_w(\varphi)$ for solution to (weighted) maximal satisfiability problem for φ
- ▶ write $\text{pmaxSAT}(\chi, \varphi)$ and $\text{pmaxSAT}_w(\chi, \varphi)$ for solution to (weighted) partial maximal satisfiability problem for hard clauses χ and soft clauses φ

Example

$$\varphi = \{ (\neg x, 2), \quad (y, 4), \quad (\neg x \vee \neg y, 5) \}$$

$$\chi = \{ x \}$$

- ▶ $\text{maxSAT}_w(\varphi) = 11$ e.g. for valuation $v(x) = F$ and $v(y) = T$
- ▶ $\text{pmaxSAT}_w(\chi, \varphi) = 5$, e.g. for valuation $v(x) = T$ and $v(y) = F$

8

Minimum Unsatisfiability (minUNSAT)

instance: CNF formula φ

question: what is minimal $|\psi|$ such that $\psi \subseteq \varphi$ and $\bigwedge_{C \in \psi} \neg C$ is satisfiable?

Notation

write $\text{minUNSAT}(\varphi)$ for solution to minimal unsatisfiability problem for φ

Lemma

$$|\varphi| = |\text{minUNSAT}(\varphi)| + |\text{maxSAT}(\varphi)|$$

Example

$$\varphi = \{ \neg x, \quad x \vee y, \quad \neg y \vee \neg z, \quad x, \quad y \vee \neg z \}$$

using $v(x) = v(y) = T$ and $v(z) = F$ have

- ▶ $\text{maxSAT}(\varphi) = 4$
- ▶ $\text{minUNSAT}(\varphi) = 1$

Remark

maxSAT and minUNSAT are equivalent

9

Application: Automotive Configuration (1)

Manufacturer's Constraints on Components

component family	components limit	premise	conclusion
engine	$E_1, E_2, E_3 = 1$	G_1	$E_1 \vee E_2$
gearbox	$G_1, G_2, G_3 = 1$	$N_1 \vee N_2$	D_1
control unit	$C_1, \dots, C_5 = 1$	N_3	$D_2 \vee D_3$
dashboard	$D_1, \dots, D_4 = 1$	$AC_1 \vee AC_3$	$D_1 \vee D_2$
navigation system	$N_1, N_2, N_3 \leq 1$	AS_1	$D_2 \vee D_3$
air conditioner	$AC_1, AC_2, AC_3 \leq 1$	$R_1 \vee R_2 \vee R_5$	$D_1 \vee D_4$
alarm system	$AS_1, AS_2 \leq 1$		
radio	$R_1, \dots, R_5 \leq 1$		

Component dependencies

Component families with limitations

Encoding

- ▶ for every component c use variable x_c which is assigned T iff c is used
- ▶ require manufacturer's constraints φ_{car} by adding respective clauses

Problem 1: Validity of Configuration

- ▶ is desired configuration valid? SAT encoding
- e.g. $E_1 \wedge G_1 \wedge C_5 \wedge (D_2 \vee D_3) \checkmark$ $E_3 \wedge G_1 \wedge C_5 \wedge D_2 \vee AC_1 \times$ 10

Algorithms for Maximum Satisfiability

Application: Automotive Configuration (2)

Problem 2: Maximization of Chosen Components

- ▶ find maximal valid subset of configuration c_1, \dots, c_n partial maxSAT
- ▶ possibly with priorities p_i for component c_i weighted partial maxSAT

$$\underbrace{\varphi_{car}}_{\text{hard clauses}} \wedge \underbrace{x_{c_1} \wedge \dots \wedge x_{c_n}}_{\text{soft clauses}}$$

Problem 3: Minimization of Costs

- ▶ given cost q_i for each component c_i , find cheapest valid configuration weighted partial maxSAT encoding

$$\underbrace{\varphi_{car}}_{\text{hard clauses}} \wedge \underbrace{(c_1, -q_1) \wedge \dots \wedge (c_n, -q_n)}_{\text{soft clauses}}$$

Result

collaboration with BMW: evaluated on configuration formulas of 2013 product line 11

Branch & Bound

Idea

- ▶ gets list of clauses φ as input return $\text{minUNSAT}(\varphi)$
- ▶ explores assignments in depth-first search

Ingredients

- ▶ **UB** is minimal number of unsatisfied clauses found so far (best solution)
- ▶ φ_x is formula φ with all occurrences of x replaced by T
- ▶ $\varphi_{\bar{x}}$ is formula φ with all occurrences of x replaced by F
- ▶ for list of clauses φ , function $\text{simp}(\varphi)$
 - ▶ replace $\neg T$ by F and $\neg F$ by T
 - ▶ drops all clauses which contain T
 - ▶ removes F from all remaining clauses
- ▶ \square denotes empty clause and $\#\text{empty}(\varphi)$ number of empty clauses in φ

Example

$$\begin{aligned} \varphi &= y \vee \neg F, & x \vee y \vee F, & F, & x \vee \neg y \vee T, & x \vee \neg z \\ \text{simp}(\varphi) &= & x \vee y, & \square, & & x \vee \neg z \end{aligned}$$

Algorithm (Branch & Bound)

```

function BnB( $\varphi$ , UB)
   $\varphi$  = simp( $\varphi$ )
  if  $\varphi$  contains only empty clauses then
    return #empty( $\varphi$ )
  if #empty( $\varphi$ )  $\geq$  UB then
    return UB
  x = selectVariable( $\varphi$ )
  UB := min(UB, BnB( $\varphi_x$ , UB))
  return min(UB, BnB( $\varphi_{\bar{x}}$ , UB))
  
```

- ▶ number of clauses falsified by any valuation is $\leq |\varphi|$
- ▶ start by calling $\text{BnB}(\varphi, |\varphi|)$
- ▶ **idea:** #empty(φ) is number of clauses falsified by current valuation

13

Algorithm (Branch & Bound, improved)

```

function BnB'( $\varphi$ , UB)
   $\varphi$  = simp( $\varphi$ )
  if  $\varphi$  contains only empty clauses then
    return #empty( $\varphi$ )
  LB = #empty( $\varphi$ ) + underapproximate( $\varphi$ )
  if LB  $\geq$  UB then
    return UB
  x = selectVariable( $\varphi$ )
  UB = min(UB, BnB'( $\varphi_x$ , UB))
  return min(UB, BnB'( $\varphi_{\bar{x}}$ , UB))
  
```

Underapproximation (Wallace and Freuder)

- ▶ $\text{ic}(x)$ is number of unit clauses x in φ inconsistency count
- ▶ $\text{underapproximate}(\varphi) = \sum_{x \text{ in } \varphi} \min(\text{ic}(x), \text{ic}(\neg x))$

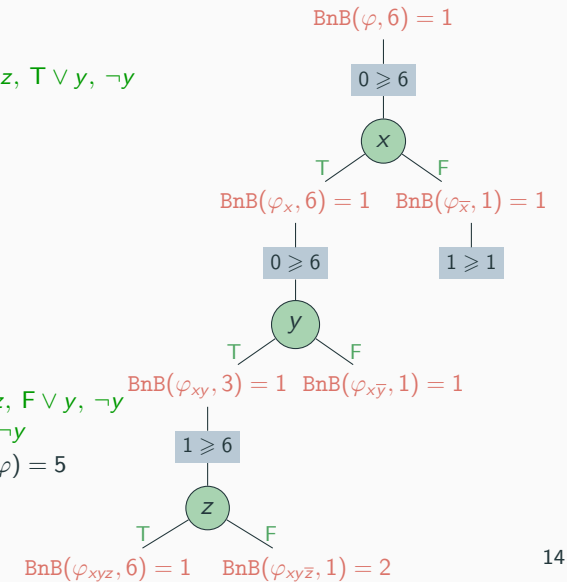
Theorem

$\text{BnB}(\varphi, |\varphi|) = \text{BnB}'(\varphi, |\varphi|) = \text{minUNSAT}(\varphi)$

15

Example

- ▶ $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call $\text{BnB}(\varphi, 6)$
- ▶ $\text{simp}(\varphi) = \varphi$
- ▶ $\varphi_x = T, \neg T \vee y, z \vee \neg y, T \vee z, T \vee y, \neg y$
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$
- ▶ $\varphi_{xy} = T, z \vee \neg T, \neg T$
 $\text{simp}(\varphi_{xy}) = z, \square$
- ▶ $\varphi_{xyz} = T, \square$
 $\text{simp}(\varphi_{xyz}) = \square$
- ▶ $\varphi_{xy\bar{z}} = F, \square$
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
- ▶ $\varphi_{x\bar{y}} = F, z \vee \neg F, \neg F$
 $\text{simp}(\varphi_{x\bar{y}}) = \square$
- ▶ $\varphi_{\bar{x}} = F, \neg F \vee y, z \vee \neg y, F \vee z, F \vee y, \neg y$
 $\text{simp}(\varphi_{\bar{x}}) = \square, z \vee \neg y, z, y, \neg y$
- ▶ $\text{minUNSAT}(\varphi) = 1, \text{maxSAT}(\varphi) = 5$
- ▶ $v(x) = v(y) = v(z) = T$



14

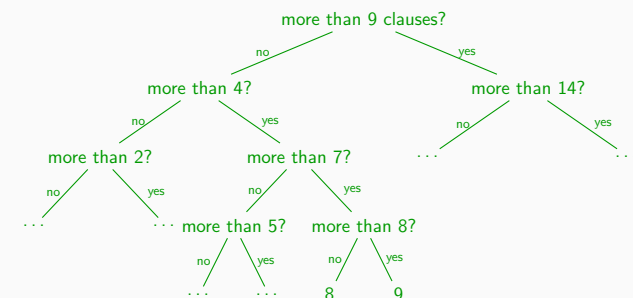
Binary Search

Idea

- ▶ gets list of clauses φ as input and returns $\text{minUNSAT}(\varphi)$
- ▶ repeatedly call SAT solver in binary search fashion

Example

Suppose given formula with 18 clauses. Can we satisfy ...



16

Cardinality Constraints

Definitions

- ▶ **cardinality constraint** has form $\sum_{x \in X} x \bowtie N$ where \bowtie is $=, <, >, \leq,$ or $\geq,$ X is set of propositional variables and $N \in \mathbb{N}$
- ▶ valuation v satisfies $\sum_{x \in X} x \bowtie N$ iff $k \bowtie N$ where k is number of variables $x \in X$ such that $v(x) = T$

Remarks

- ▶ cardinality constraints are **expressible in CNF**
 - ▶ enumerate all possible subsets $O(2^{|X|})$
 - ▶ BDDs $O(N \cdot |X|)$
 - ▶ sorting networks $O(|X| \cdot \log^2(|X|))$
- ▶ write CNF($\sum_{x \in X} x \bowtie N$) for CNF encoding
- ▶ cardinality constraints occur very frequently! (n -queens, Minesweeper, ...)

Example

- ▶ $x + y + z = 1$ satisfied by $v(x) = v(y) = F, v(z) = T$
- ▶ $x_1 + x_2 + \dots + x_8 \leq 3$ satisfied by $v(x_1) = \dots = v(x_8) = F$

17

Algorithm (Binary Search)

```
function BinarySearch( $\{C_1, \dots, C_m\}$ )
   $\varphi := \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ 
  return search( $\varphi, 0, m$ )
```

b_1, \dots, b_m are fresh variables

```
function search( $\varphi, L, U$ )
  if  $L \geq U$  then
    return U
  mid :=  $\lfloor \frac{L+U}{2} \rfloor$ 
  if SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq \text{mid})$ ) then
    return search( $\varphi, L, \text{mid}$ )
  else
    return search( $\varphi, \text{mid} + 1, U$ )
```

Theorem

$\text{BinarySearch}(\varphi) = \text{minUNSAT}(\varphi)$

18

Example

$$\varphi = \{ 6 \vee 2 \vee b_1, \quad \bar{6} \vee 2 \vee b_2, \quad \bar{2} \vee 1 \vee b_3, \quad \bar{1} \vee b_4, \quad \bar{6} \vee 8 \vee b_5, \\ 6 \vee \bar{8} \vee b_6, \quad 2 \vee 4 \vee b_7, \quad \bar{4} \vee 5 \vee b_8, \quad 7 \vee 5 \vee b_9, \quad \bar{7} \vee 5 \vee b_{10}, \\ \bar{3} \vee b_{11}, \quad \bar{5} \vee 3 \vee b_{12} \}$$

- ▶ $L = 0, U = 12, \text{mid} = 6$ SAT($\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 6)$)? ✓
- ▶ $L = 0, U = 6, \text{mid} = 3$ SAT($\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 3)$)? ✓
- ▶ $L = 0, U = 3, \text{mid} = 1$ SAT($\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 1)$)? ✗
- ▶ $L = 2, U = 3, \text{mid} = 2$ SAT($\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 2)$)? ✓
- ▶ $L = 2, U = 2$ return 2

19

Cardinality Constraints in Z3

```
from z3 import *

xs = [ Bool("x"+str(i)) for i in range (0,10)]
ys = [ Bool("y"+str(i)) for i in range (0,10)]

def sum(ps):
    return reduce(lambda s,x: s + If(x, 1, 0), ps, 0)

solver = Solver()
solver.add(sum(xs) == 5, sum(ys) > 3, sum(ys) <= 4)

if solver.check() == sat:
    model = solver.model()
    for i in range(0,10):
        print xs[i], "=", model[xs[i]], ys[i], "=", model[ys[i]]
```

20

Definition

FP^{NP} is class of functions computable in polynomial time with access to **NP oracle**

Theorem

maxSAT is FP^{NP} -complete

Remarks

- ▶ FP^{NP} allows polynomial number of oracle calls (which is e.g. SAT solver)
- ▶ other members of FP^{NP} are travelling salesperson and Knapsack

-  Rouven Walter, Christoph Zengler and Wolfgang Küchlin.
Applications of MaxSAT in Automotive Configuration.
Proc. International Configuration Workshop 2013, pp. 21-28, 2013.
-  André Abramé and Djamel Habet.
ahmaxsat: Description and Evaluation of a Branch and Bound Max-SAT Solver.
Journal on Satisfiability, Boolean Modeling and Computation 9, pp. 89-128, 2015.
-  Chu-Min Li and Felip Manyà.
MaxSAT, hard and soft constraints.
In: Handbook of Satisfiability, IOS Press, pp. 613-631, 2009.
-  Zhaohui Fu and Sharad Malik.
On solving the partial MAX-SAT problem.
In Proc. Theory and Applications of Satisfiability Testing, pp. 252-265, 2006