SAT Solving and Applications

Introduction to MaxSAT Solving

Prof. Dr. Wolfgang Küchlin
Rouven Walter, M.Sc.

University of Tübingen

25 July 2015

Contents

What is MaxSAT?

Example

$x_6 \vee x_2$ $\quad \neg x_6 \vee x_2$ $\quad \neg x_2 \vee x_1$ $\quad \neg x_1$

$\neg x_6 \vee x_8$ $\quad x_6 \vee \neg x_8$ $\quad x_2 \vee x_4$ $\quad \neg x_4 \vee x_5$

$x_7 \vee x_5$ $\quad \neg x_7 \vee x_5$ $\quad \neg x_5 \vee x_3$ $\quad \neg x_3$

Is this clause set satisfiable?

What is MaxSAT?

No!

$x_6 \vee x_2$ $\quad \neg x_6 \vee x_2$ $\quad \neg x_2 \vee x_1$ $\quad \neg x_1$

$\neg x_6 \vee x_8$ $\quad x_6 \vee \neg x_8$ $\quad x_2 \vee x_4$ $\quad \neg x_4 \vee x_5$

$x_7 \vee x_5$ $\quad \neg x_7 \vee x_5$ $\quad \neg x_5 \vee x_3$ $\quad \neg x_3$

Clause set is unsatisfiable!

MaxSAT:

- Find a valuation which **maximizes** the number of satisfied clauses
- In the above set, a maximum of **10 clauses** can be satisfied simultaneously
- There are several variants of the MaxSAT problem

Variants of MaxSAT — 1

MaxSAT

- All clauses are *soft*, i.e. need not necessarily be satisfied.
- Maximize number of satisfied soft clauses
- Minimize number of unsatisfied soft clauses

Partial MaxSAT

- There are *hard* clauses, which *must* be satisfied
- Minimize number of unsatisfied soft clauses

Application example: car configuration

- Hard clauses: certain options/features must be present in car
- Soft clauses: additional options/features with dependencies
- Question: how many (and which) options can be ordered additionally for the car?

Variants of MaxSAT — 2

Weighted MaxSAT

- All clauses are soft
- All clauses carry *weights* in addition
- Minimize the sum of weights in unsatisfied clauses

Weighted Partial MaxSAT

- There are *hard* clauses, which must be satisfied
- There are soft clauses which carry weights
- Minimize the sum of weights in the unsatisfied soft clauses

Application example: car configuration

- Hard clauses: the configuration constraints given by the manufacturer (technical, legal, sales, . . .)
- Soft clauses: options/features desired by the customer (typically unit clauses: a set of option codes desired to be all true), possibly with priorities or costs as weights
- Questions: optimal car configuration (sum of priorities maximized)

Notation

Notation: weighted clause

(c, w): weighted clause

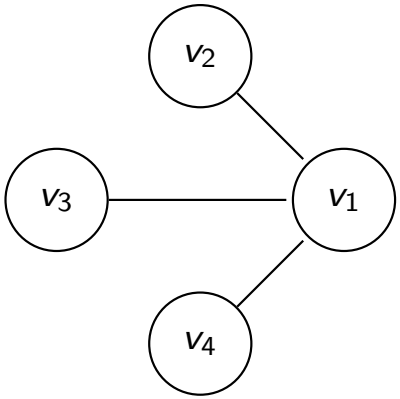
- c is a set of literals (a clause)
- w is a non-negative integer value or ∞ (or $\top$)
  - cost (penalty), if c is not satisfied

Notation: clause set

φ: set of weighted clauses

- Soft clauses: (c, w) with $w < \infty$
  - Cost, if c is not satisfied
- Hard clauses: (c, ∞)
  - Clause c must be satisfied

Modelling Example



Minimum Vertex Cover

- Vertex Cover $U \subseteq V$: For each edge $(v_i, v_j) \in E$ we have $v_i \in U$ or $v_j \in U$
- Minimum Vertex Cover: Vertex Cover U of minimal size

Example

Partial MaxSAT Encoding

- Variables: x_i for each vertex $v_i \in C$, with $x_i = 1$, *iff* $v_i \in U$
- Hard clauses: $(x_i \vee x_j)$ for each edge $(v_i, v_j) \in E$ *(... is Vertex Cover)*
- Soft clauses: $(\neg x_i)$ for each edge $v_i \in V$ *(... is minimal)*
  - the more variables are set to 0, the smaller the cover set

Encoding:

- $\varphi_H = \{(x_1 \vee x_2), (x_1 \vee x_3), (x_1 \vee x_4)\}$
- $\varphi_S = \{(\neg x_1), (\neg x_2), (\neg x_3), (\neg x_4)\}$

MaxSAT vs. MinUNSAT

MinUNSAT

- Find valuation which minimizes the number U of unsatisfied clauses
  - this valuation also maximizes the number of satisfied clauses
- $\mathrm{MinUNSAT}(\varphi) := U$.
- variants partial, weighted, and partial weighted are possible (like with MaxSAT)
  - Weighted MinUNSAT: minimize the weight of unsatisfied clauses

Relation

Let φ be a clause set, then:

$$|\varphi| = \mathrm{MaxSAT}(\varphi) + \mathrm{MinUNSAT}(\varphi)$$

Note: this relation holds for all variants

- Sometimes algorithms are given for MinUNSAT which can equally well be used for MaxSAT because of this relation

Example: MaxSAT vs. MinUNSAT

Example

- Let $\varphi = \mathit{Hard} \dot{\cup} \mathit{Soft}$ with

$$\begin{aligned} \mathit{Hard} &= \{\{x\}\} \\ \mathit{Soft} &= \{(\{\neg x\}, 2), (\{y\}, 6), (\{\neg x, \neg y\}, 5)\} \end{aligned}$$

- $\mathrm{PartialWeightedMaxSAT}(\varphi) = 6$
  Valuation: $x \mapsto 1$, $y \mapsto 1$
- $\mathrm{PartialWeightedMinUNSAT}(\varphi) = 7$
  Valuation: $x \mapsto 1$, $y \mapsto 1$
- We have:

$$13 = \mathrm{PartialWeightedMaxSAT}(\varphi) + \mathrm{PartialWeightedMinUNSAT}(\varphi)$$

Pseudo-Boolean Constraints & Optimization

Pseudo-Boolean Constraints

- Boolean variables: $x_1, \ldots, x_n$
- Linear inequalities

$$\sum_{i \in N} a_i \ell_i \geq b, \qquad \ell_i \in \{x_i, \bar{x}_i\}, x_i \in \{0,1\}, a_i, b \in \mathbb{N}_0^+$$

Pseudo-Boolean Optimization

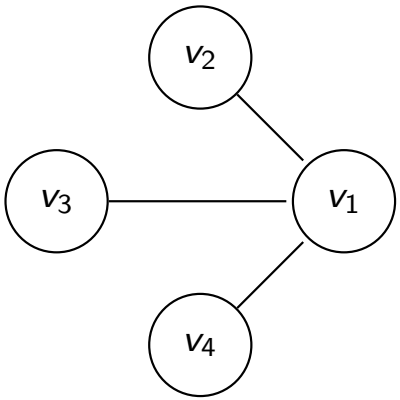
Minimize

$$\sum_{h \in N} w_h \cdot x_h$$

with respect to n PB constraints

$$\sum_{i \in N} a_{ij} \ell_i \geq b_j, \qquad j \in \{1 \ldots n\}, \ell_i \in \{x_i, \bar{x}_i\}, x_i \in \{0,1\}, a_{ij}, b_j \in \mathbb{N}_0^+$$

Modelling Example



Minimum Vertex Cover

- Vertex Cover $U \subseteq V$: for each edge $(v_i, v_j) \in E$ we have $v_i \in U$ or $v_j \in U$
- Minimum Vertex Cover: Vertex Cover U of minimal size

Encoding as PBO

- Variables: x_i for each vertex $v_i \in V$, with $x_i = 1$ if $v_i \in U$
- PB Constraints: $x_i + x_j \geq 1$ for each edge $(v_i, v_j) \in E$
- Goal function: minimize number of variables with value$= 1$

  $\Rightarrow$ i.e. minimize number of vertices in the Vertex Cover

Problem instance: $x_1 + x_2 \geq 1, x_1 + x_3 \geq 1, x_1 + x_4 \geq 1$
Goal function: $x_1 + x_2 + x_3 + x_4$

From (pure) MaxSAT to PBO

Starting from an unsatisfiable CNF formula φ:

1. Generate φ' from φ:
   - Replace each clause c_i with $c_i' = c_i \cup \{r_i\}$
   - where r_i is a new variable (*selector variable, blocking variable*)
   - Now the problem can be trivially solved by setting all r_i to 1
2. Minimize goal function $\sum r_i$

Example

- CNF formula φ:

$$\varphi = \{\{x_1, \neg x_2\}, \{x_1, x_2\}, \{\neg x_1\}\}$$

- Modified formula φ':

$$\varphi = \{\{x_1, \neg x_2, r_1\}, \{x_1, x_2, r_2\}, \{\neg x_1, r_3\}\}$$

- PB Constraints: $x_1 + \bar{x_2} + r_1 \geq 1, x_1 + x_2 + r_2 \geq 1, \bar{x_1} + r_3 \geq 1$
- Minimize goal function: $r_1 + r_2 + r_3$

From Partial (Weighted) MaxSAT to PBO

Starting from a Partial (Weighted) MaxSAT instance with φ_H and φ_S

Generate PBO instance:

$$\text{minimize } \sum w_i r_i, \text{ such that } \varphi_T \text{ holds, with}$$

- $\varphi_T = \varphi'_H \cup \varphi'_S$
- Each hard clause (c, ∞) is mapped to a clause c in φ'_H
- Each soft clause (c, w) is mapped to a clause $(c_i \vee r_i)$ and the term $w_i r_i$ is added to the goal function

Example

- Original problem: $(\{x, y, \neg z\}, \infty), (\{x, \neg y\}, 4), (\{\neg x\}, 8), (\{x, z\}, 2)$
- $\varphi_T = \underbrace{(x, y, \neg z)}_{\varphi'_H}, \underbrace{(x, \neg y, r_1), (\neg x, r_2), (x, z, r_3)}_{\varphi'_S}$
- PB constraints: $x + y + \bar{z} \geq 1, x + \bar{y} + r_1 \geq 1, \bar{x} + r_2 \geq 1, x + z + r_3 \geq 1$
- Minimize goal function: $4r_1 + 8r_2 + 2r_3$

Application: Software Package Upgrades

- Available software packages: $P = \{p_1, \ldots, p_n\}$
- Variable x_i for each package $p_i \in P$. $x_i = 1$, *iff* p_i is installed
- Constraints for each package p_i: (p_i, D_i, C_i)
  - D_i: Dependencies (required packages) when installing p_i
  - C_i: Conflicts (disallowed packages) when installing p_i
- Example problem: Maximum Installability
  - Maximal number of packages which can be installed
  - Package Constraints are hard clauses
  - Each individual package is a soft clause

Example

Package Constraints

$(p_1, \{p_2 \vee p_3\}, \{p_4\})$
$(p_2, \{p_3\}, \{p_4\})$
$(p_3, \{p_2\}, \emptyset)$
$(p_4, \{p_2 \wedge p_3\}, \emptyset)$

MaxSAT Encoding

$$\begin{aligned} \varphi_H &= \{(\neg x_1 \vee x_2 \vee x_3), (\neg x_1 \vee \neg x_4), \\ &\quad (\neg x_2 \vee x_3), (\neg x_2 \vee \neg x_4), (\neg x_3 \vee x_2), \\ &\quad (\neg x_4 \vee x_2), (\neg x_4 \vee x_3) \\ \varphi_S &= \{(x_1), (x_2), (x_3), (x_4)\} \end{aligned}$$

Cardinality Constraints

Question: How do we handle Cardinality Constraints?

- General form: $\sum_{j=1}^{n} x_j \bowtie k$ with $\bowtie \in \{<, \leq, =, \geq, >\}$
- In particular AtMost1 constraints: $\sum_{j=1}^{n} x_j \leq 1$

Solution 1

Use a special PB Solver

- Hard to keep up with progress in SAT Solving
- For SAT/UNSAT the best solvers already encode in CNF
  - such as Minisat+, QMaxSat, MSUnCore, (W)PM2

Solution 2

- encode cardinality constraints in CNF
- use a SAT solver

Equals, AtLeast1 & AtMost1 Constraints

Special treatment of constraints with right-hand side $= 1$ — occurring frequently:

- Car must have exactly on motor ...
- Exactly one graphics driver must be selected ...
- At most one SAT solver may be installed in Eclipse ...

Encodings

- $\sum_{j=1}^{n} x_j = 1$: encode as $(\sum_{j=1}^{n} x_j \leq 1) \wedge (\sum_{j=1}^{n} x_j \geq 1)$
- $\sum_{j=1}^{n} x_j \geq 1$: encode as $(x_1 \vee x_2 \vee \cdots \vee x_n)$
- $\sum_{j=1}^{n} x_j \leq 1$: encode as:
  - Pairwise encoding:
    - clauses: $\mathcal{O}(n^2)$; no auxiliary variables
  - Sequential counter
    - clauses: $\mathcal{O}(n)$; auxiliary variables: $\mathcal{O}(n)$
  - Bitwise encoding:
    - clauses: $\mathcal{O}(n \log n)$; auxiliary variables $\mathcal{O}(\log n)$

General Cardinality Constraints

General form: $\sum_{j=1}^{n} x_j \leq k$ (or $\sum_{j=1} x_j \geq k$)

Encodings

- Sequential counter
  - clauses/variables: $\mathcal{O}(nk)$
- BDDs
  - clauses/variables: $\mathcal{O}(nk)$
- Sorting networks
  - clauses/variables: $\mathcal{O}(n \log^2 n)$
- Cardinality networks
  - clauses/variables: $\mathcal{O}(n \log^2 k)$

Pseudo-Boolean Constraints

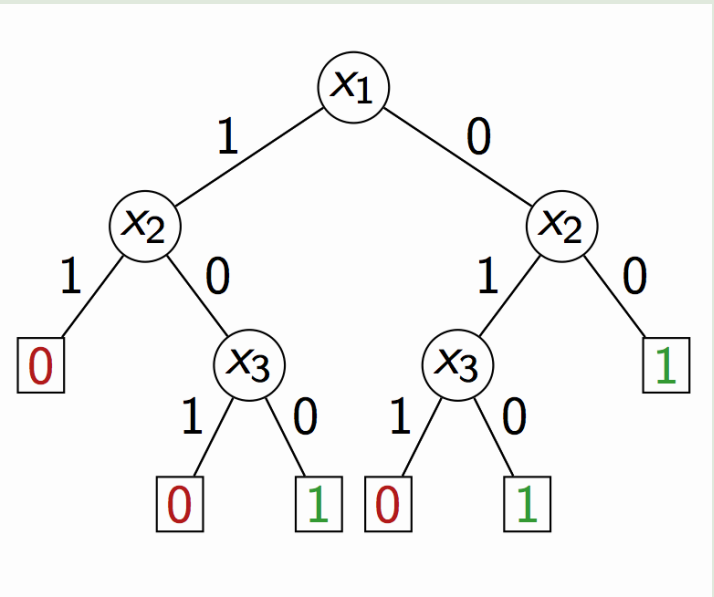
General form: $\sum_{j=1}^{n} a_j x_j \leq b$

- Encoding e.g. with BDDs (worst case: exponential number of clauses!)

Example

$$3x_1 + 3x_2 + x_3 \leq 3$$

- Encode a BDD, i.e. analyze variables by subtracting coefficients
- Convert BDD to CNF
- Simplify



Pseudo-Boolean Constraints

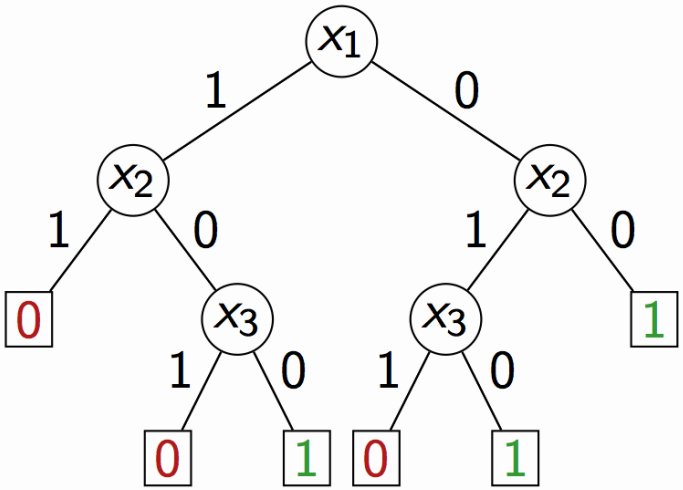
General form: $\sum_{j=1}^{n} a_j x_j \leq b$

- Encoding e.g. with BDDs (worst case: exponential number of clauses!)

Example

$$3x_1 + 3x_2 + x_3 \leq 3$$

- Encode a BDD, i.e. analyze variables by subtracting coefficients
- Convert BDD to CNF
- Simplify



- Conversion formula at node x: $F \equiv (\neg x \vee F|_{x=1}) \wedge (x \vee F|_{x=0})$
- yields: $(\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$
- simplifies to: $(\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3)$

Some Practical Algorithms

- Branch & Bound
- Iterative (pure) SAT Solving
  - Step-wise approach from below (pure, simple, inefficient)
  - Leaping approach from above: LeBerre´s Algorithm (for MaxSAT)
  - Binary Search (for MaxSAT)
- Iterative SAT Solving using the unsat core extension
  - Fu and Malik Algorithm for Partial MaxSAT
  - PM2 for Partial MaxSAT
  - WPM1 for Partial Weighted MaxSAT

MaxSAT by Iterative SAT Solving

Reducing MaxSAT Solving to SAT Solving

- Idea:
  - Iteratively call a SAT Solver, switching off clauses through blocking variables
- Advantage:
  - immediately use all modern efficient SAT Solving techniques (Unit propagation with Watched Literals, Clause Learning with non-chronological backtracking, etc) which do not (or only partially) apply to MaxSAT directly.
- Two approaches:
  - Reduction to (pure) SAT-Solving
  - Reduction to SAT-Solving with unsat core extension

Basic idea: step-wise approach from below

- Introduce a new blocking variable for each constraint: C_i becomes $C_i \vee b_i$.
- Introduce additional cardinality constraint on the blocking variables (blocked clauses): $\sum_{i=1}^{m} b_i \leq k$
- Iteratively solve $SAT((\bigcup C_i) \cup CNF(\sum_{i=1}^{m} b_i \leq k))$ für $k = 0, 1, \ldots$ until enough (MinUNSAT many) clauses have been switched off (blocked)

Iterative MaxSAT: Leaping approach from above

The algorithm of Daniel LeBerre

Algorithm 1: LeBerre(φ)

Input: MaxSAT instance $\varphi = \{C_1, \ldots, C_m\}$
$BV = \{b_1, \ldots, b_m\}$
$\varphi \leftarrow \{C_1 \vee b_1, \ldots, C_m \vee b_m\}$ `// Add blocking variables`
$ub \leftarrow m$
while $\mathrm{SAT}(\varphi \cup \mathrm{CNF}(\sum_{i=1}^{m} b_i < ub))$ **do**
  `// Are blocking variables used at all?`
  if $\#\mathrm{satisfiedBlockingVariables} > 0$ **then**
    $ub \leftarrow \#\mathrm{satisfiedBlockingVariables}$
  else
    return *0*
return *ub*

Literature: Biere *et al.* Handbook of Satisfiability, 19.6, S.625. IOS Press, 2009.

Unsatisfiable Core

Unsatisfiable Core

Sei φ eine Menge von Klauseln.

- Eine unerfüllbare Teilmenge von φ heißt *unsatisfiable core* (kurz: unsat core)
- Ein unsatisfiable core φ_c heißt *minimal unsatisfiable core*, falls jede echte Teilmenge von φ_c erfüllbar ist. In einem MUC m ist MinUNSAT(m)=1.

Beispiel

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

- φ ist unsat core, aber kein minimal unsat core (z.B. $\{\{y\}, \{\neg y\}\}$ unerfüllbar)
- $\{\{y\}, \{\neg y\}\}$ ist ein minimal unsat core (global)
- $\{\{x\}, \{y, \neg z\}, \{\neg y\}, \{\neg x, z\}\}$ ist ein minimal unsat core (lokal)

- Ein unsat core u kann mehrere MUC enthalten. Diese können disjunkt sein oder gemeinsame Klauseln enthalten. Es ist MinUNSAT(u) ≥ 1.
- Liefert ein SAT Solver einen unsat core, so kann diese Information genutzt werden für MaxSAT (Core Guided Algorithms)

Fu & Malik Algorithmus für Partial MaxSAT — Idee

Idee des Fu & Malik Algorithmus

- Prüfe iterativ mit Hilfe eines SAT Solvers (mit unsat core Unterstützung), ob Instanz erfüllbar ist
- Falls ja, so ist Optimum gefunden (alle Klauseln erfüllbar)
- Falls nein, so kann **mindestens eine** Klausel aus dem gelieferten unsat core **nicht** erfüllt werden! (Min core: genau eine) Welche genau, ist nicht bekannt.
  - Füge jeder soft Klausel eine (frische) blocking Variable hinzu
  - Füge cardinality constraint (Equals) über die eingeführten blocking Variablen hinzu, so dass genau eine Variable erfüllt sein muss. D.h. eine soft Klausel im unsat core wird bei der nächsten Iteration ausgeschlossen.
  - Erhöhe Kosten um 1
- Spezialfall: Befindet sich im unsat core keine soft Klausel, so sind die hard Klauseln bereits unerfüllbar, d.h. es gibt kein Optimum

- Literatur: Fu, Malik. On Solving the Partial MAX-SAT Problem. SAT2006. Biere et al. Handbook of Satisfiability, 19.6, S.625–626. IOS Press, 2009.

Fu & Malik Algorithmus für Partial MaxSAT

Der Algorithmus

Algorithm 2: Fu&Malik(φ)

Input: Partial MaxSAT Instanz φ

$cost \leftarrow 0$

while *true* **do**

  $(st, \varphi_c) \leftarrow \mathrm{SAT}(\varphi)$ // SAT solver call

  if $st = SAT$ **then return** *cost*

  $BV \leftarrow \emptyset$ // Set of blocking variables

  foreach $C \in \varphi_c$ **do**

    if C *is soft* **then**

      $b \leftarrow$ new blocking variable

      $\varphi \leftarrow \varphi \setminus \{C\} \cup \{C \vee b\}$ // Add blocking variable

      $BV \leftarrow BV \cup \{b\}$

  if $BV = \emptyset$ **then return** None

  $\varphi \leftarrow \varphi \cup \mathrm{CNF}(\sum_{b \in BV} b = 1)$ // Cardinality constraint is hard

  $cost \leftarrow cost + 1$

Fu & Malik Algorithmus für Partial MaxSAT — Beispiel

Beispiel

- $\varphi = \mathit{Hard} \,\dot\cup\, \mathit{Soft}$ mit

$$\begin{aligned} \mathit{Hard} &= \{\{x, y\}\} \\ \mathit{Soft} &= \{\{\neg x\}, \{x\}, \{\neg y\}\} \end{aligned}$$

  Hinweis: φ enthält unter anderem die unsat cores:
  - $\{\{\neg x\}, \{x\}\}$
  - $\{\{\neg x\}, \{x, y\}, \{\neg y\}\}$
- 1. Iteration: $\mathrm{SAT}(\varphi) = \mathit{false}$
  $\varphi_c \leftarrow \{\{\neg x\}, \{x\}\}$
  $\varphi \leftarrow \{\{x, y\}\} \cup \{\{\neg x, b_1^1\}, \{x, b_2^1\}, \{\neg y\}\} \cup \mathrm{CNF}\left(\sum_{i=1}^{2} b_i^1 = 1\right)$
  $\mathit{cost} \leftarrow 1$
- 2. Iteration: $\mathrm{SAT}(\varphi) = \mathit{true}$
  Belegung: $x \mapsto 1$, $y \mapsto 0$, $b_1^1 \mapsto 1$, $b_2^1 \mapsto 0$
  Ergebnis: $\mathit{cost} = 1$.
- Zweiter unsat core wird nicht betrachtet, da durch „blockierte“ Klausel $\{\neg x\}$ alle weiteren unsat cores erfüllbar sind.

PM2 für Partial MaxSAT — Idee

Idee des PM2 Algorithmus

- Nachteil des Algorithmus von Fu & Malik:
  - Jede soft Klausel innerhalb eines unsat cores erhält neue blocking Variable
  - Kommt eine Klausel in mehreren unsat cores vor, erhält sie mehrere blocking Variablen
  - Suchraum für SAT Solver wird vergrößert!
- Idee: Füge jeder soft Klausel nur eine blocking Variable hinzu und limitiere Erfüllbarkeit der blocking Variablen durch cardinality constraints
- Vorteil: Nur eine blocking Variable pro Klausel
- Nachteil: Anzahl der cardinality constraints wird (je nach Codierung) sehr groß, d.h. sehr viele neue Klauseln

- Literatur: Ansótegui, Bonet, Levy. On Solving MaxSAT Through SAT. POS-10. Pragmatics of SAT.

PM2 für Partial MaxSAT

Der Algorithmus

Algorithm 3: $\mathrm{PM2}(\varphi)$

Input: Partial MaxSAT Instanz $\varphi = \mathit{Hard} \dot{\cup} \mathit{Soft}$ mit $\mathit{Soft} = \{C_1, \ldots, C_m\}$

$BV \leftarrow \{b_1, \ldots, b_m\}$ // Blocking variables

$\varphi_w \leftarrow \mathit{Hard} \cup \{C_1 \vee b_1, \ldots, C_m \vee b_m\}$ // Protect soft clauses

$\mathit{cost} \leftarrow 0$; $L \leftarrow \emptyset$ // L = set of unsat cores

while *true* **do**

- $(st, \varphi_c) \leftarrow \mathrm{SAT}(\varphi_w \cup \mathrm{CNF}(\sum_{b \in BV} b \leq \mathit{cost}))$
- **if** $st = \mathit{SAT}$ **then return** *cost*
- Remove hard clauses from φ_c
- **if** $\varphi_c = \emptyset$ **then return** None
- $B \leftarrow \emptyset$ // Blocking variables of the unsat core
- **foreach** $C = C_i \vee b_i$ **do**
  - $B \leftarrow B \cup \{b_i\}$
- $L \leftarrow L \cup \{\varphi_c\}$
- $k \leftarrow |\{\psi \in L \mid \psi \subseteq \varphi_c\}|$ // Number of unsat cores contained in φ_c
- $\varphi_w \leftarrow \varphi_w \cup \mathrm{CNF}(\sum_{b \in B} b \geq k)$
- $\mathit{cost} \leftarrow \mathit{cost} + 1$

WPM1 für Partial Weighted MaxSAT — Idee

Idee des WPM1 Algorithmus

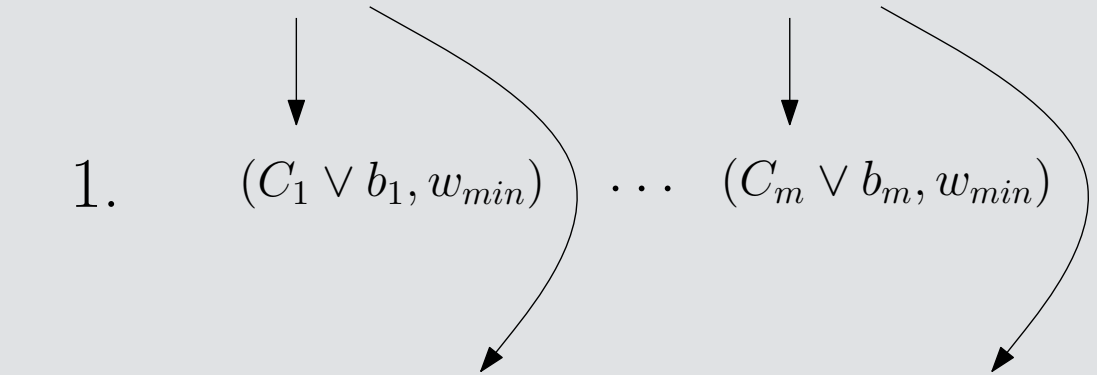
- Erweiterung des Fu & Malik Algorithmus zu Partial **Weighted** MaxSAT
- Prüfe iterativ mit Hilfe eines SAT Solvers (mit unsat core Unterstützung), ob Instanz erfüllbar ist
- Falls ja, so ist Optimum gefunden (alle Klauseln erfüllbar)
- Falls nein, so kann **mindestens** das Minimum $w_{\min}$ der Gewichte der soft Klauseln aus dem gelieferten unsat core **nicht** erfüllt werden!
  - Dupliziere jede soft Klausel (C, w):
    - Klausel $(C \vee b, w_{\min})$ mit blocking Variable b
    - Klausel $(C, w - w_{\min})$
  - Füge cardinality constraint über die eingeführten blocking Variablen hinzu, so dass genau eine Variable erfüllt sein muss. D.h. eine soft Klausel mit Gewicht $w_{\min}$ im unsat core wird bei der nächsten Iteration ausgeschlossen.
  - Erhöhe Kosten um $w_{\min}$

WPM1 für Partial Weighted MaxSAT — Idee

Idee des WPM1 Algorithmus (continued)

- Gefundener unsat core wird aufgeteilt:
  - Klauseln mit blocking Variable und Gewicht $w_{\min}$ (Unsat core mit Klauseln gleichen Gewichtes)
  - Klauseln ohne blocking Variable und Restgewicht (Rest des unsat cores)

$$\varphi_c = \{ \; (C_1, w_1) \quad \cdots \quad (C_m, w_m) \; \}$$



$$1. \quad (C_1 \vee b_1, w_{min}) \quad \cdots \quad (C_m \vee b_m, w_{min})$$

Unsatisfiable Core:
Klauseln gleichen Gewichtes

$$2. \quad (C_1, w_1 - w_{min}) \quad \cdots \quad (C_m, w_m - w_{min})$$

Klauseln mit Differenzgewicht

- Literatur: Ansótegui, Bonet, Levy. On Solving MaxSAT Through SAT. POS-10. Pragmatics of SAT.

WPM1 für Partial Weighted MaxSAT

Der Algorithmus

Algorithm 4: $\mathrm{WPM1}(\varphi)$

Input: P. W. MaxSAT Instanz $\varphi = \mathit{Hard} \dot{\cup} \mathit{Soft}$ mit $\mathit{Soft} = \{(C_1, w_1), \ldots, (C_m, w_m)\}$

$\mathit{cost} \leftarrow 0$

while *true* **do**

- $(st, \varphi_c) \leftarrow \mathrm{SAT}(\mathit{Hard} \cup \{C_i \mid (C_i, w_i) \in \mathit{Soft}\})$ `// SAT solver call`
- **if** $st = \mathit{SAT}$ **then return** *cost*
- $BV \leftarrow \emptyset$
- $w_{min} \leftarrow \min\{w_i \mid C_i \in \varphi_c \setminus \mathit{Hard}\}$
- **foreach** $C_i \in \varphi_c$ **do**
  - **if** C_i *is soft* **then**
    - $b \leftarrow$ new blocking variable
    - `// Duplicate soft clause`
    - $\varphi \leftarrow \varphi \setminus \{(C_i, w_i)\} \cup \{(C_i, w_i - w_{min})\} \cup \{(C_i \vee b_i, w_{min})\}$
    - $BV \leftarrow BV \cup \{b_i\}$
- **if** $BV = \emptyset$ **then return** None
- $\varphi \leftarrow \varphi \cup \mathrm{CNF}(\sum_{b \in BV} b = 1)$ `// Cardinality Constraint is hard`
- $\mathit{cost} \leftarrow \mathit{cost} + w_{min}$

WPM1 für Partial Weighted MaxSAT — Beispiel

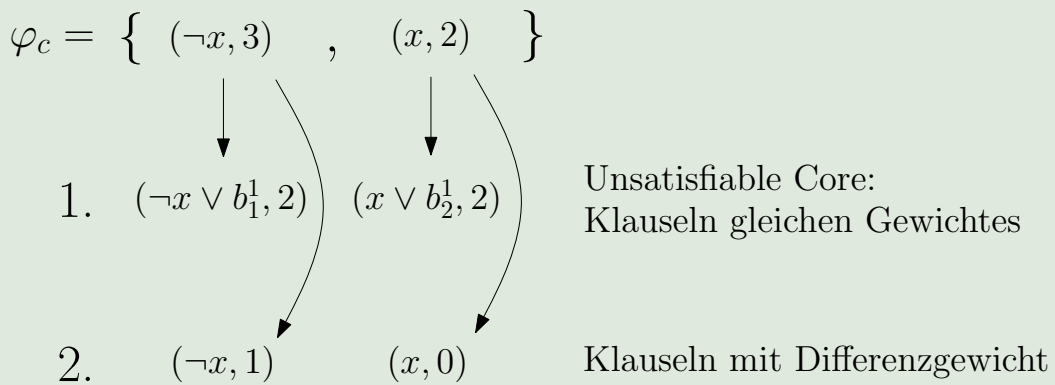
Beispiel

- $\varphi = \mathit{Hard} \dot{\cup} \mathit{Soft}$ mit

$$\begin{aligned} \mathit{Hard} &= \{\{x, y\}\} \\ \mathit{Soft} &= \{(\{\neg x\}, 3), (\{x\}, 2), (\{\neg y\}, 5)\} \end{aligned}$$

- 1. Iteration: $\mathrm{SAT}(\varphi) = \mathit{false}$
  $\varphi_c \leftarrow \{(\neg x, 3), (x, 2)\}$
  $w_{\min} \leftarrow 2$
  $\varphi \leftarrow$
  $\{\{x, y\}\} \cup \{(\{\neg x, b_1^1\}, 2), (\{x, b_2^1\}, 2), (\{\neg x\}, 1), (\{\neg y\}, 5)\} \cup \mathrm{CNF}\left(\sum_{i=1}^{2} b_i^1 = 1\right)$
  $\mathit{cost} \leftarrow 0 + w_{\min} = 2$

- Veranschaulichung der Aufteilung:



WPM1 für Partial Weighted MaxSAT — Beispiel

Beispiel (continued)

- Aus der vorherigen Iteration:
  $\varphi \leftarrow$
  $\{\{x, y\}\} \cup \{(\{\neg x, b_1^1\}, 2), (\{x, b_2^1\}, 2), (\{\neg x\}, 1), (\{\neg y\}, 5)\} \cup \mathrm{CNF}\left(\sum_{i=1}^{2} b_i^1 = 1\right)$
- 2. Iteration: $\mathrm{SAT}(\varphi) = \mathit{false}$
  $\varphi_c \leftarrow \{\{x, y\}, (\{\neg x, b_1^1\}, 2), (\{x, b_2^1\}, 2), (\{\neg x\}, 1), (\{\neg y\}, 5)\} \cup \mathrm{CNF}\left(\sum_{i=1}^{2} b_i^1 = 1\right)$
  $w_{\min} \leftarrow 1$

$$\begin{aligned}\varphi \quad \leftarrow \quad & \{\{x, y\}\} \cup \{(\{\neg x, b_1^1, b_1^2\}, 1), (\{x, b_2^1, b_2^2\}, 1), (\{\neg x, b_3^2\}, 1), (\{\neg y, b_4^2\}, 1), \\ & (\{\neg x, b_1^1\}, 1), (\{x, b_2^1\}, 1), (\{\neg y\}, 4)\} \\ & \cup \mathrm{CNF}\left(\sum_{i=1}^{2} b_i^1 = 1\right) \cup \mathrm{CNF}\left(\sum_{i=1}^{4} b_i^2 = 1\right)\end{aligned}$$

  $\mathit{cost} \leftarrow 2 + w_{\min} = 3$
- 3. Iteration: $\mathrm{SAT}(\varphi) = \mathit{true}$
  Belegung: $x \mapsto 1$, $y \mapsto 0$, $b_1^1 \mapsto 1$, $b_3^2 \mapsto 1$, restliche b_i^j auf 0 belegt
  Ergebnis: $\mathit{cost} = 3$

Vereinfachung durch Inferenzregeln — 1

- Transformiere ein MaxSAT Problem φ in ein äquivalentes Problem φ', das dieselbe Anzahl unerfüllter Klauseln für jede Belegung hat. Eine solche Transformation ist *sound*.
- Boolesche Äquivalenz oder Erfüllbarkeitsäquivalenz genügen nicht.

Beispiel

- $\text{MaxSAT}(\{\{x\}\}) \neq \text{MaxSAT}(\{\{x\},\{x\}\})$.
- $\text{MinUNSAT}(\{\{x\},\{\neg x\}\}) \neq \text{MinUNSAT}(\{\{x\},\{x\},\{\neg x\},\{\neg x\}\})$.
- $\text{MinUNSAT}(\{\{\}\}) \neq \text{MinUNSAT}(\{\{\},\{\}\})$

- Man arbeitet mit Transformationsregeln, die eine Teilmenge der Klauseln durch eine andere Teilmenge ersetzen. Das Hinzufügen von Klauseln verändert i.A. das Ergebnis: $\text{MaxSAT}(\{\{x\}\}) \neq \text{MaxSAT}(\{\{x\},\{x\}\})$.
- Es sind nur weniger und schwächere Inferenzen möglich als bei DPLL.

Vereinfachung durch Inferenzregeln — 2

Ungültige Inferenzen

- **Unit Propagation**: $\{\{x_1\}, \{\neg x_1, \neg x_2\}, \{\neg x_1, \neg x_3\}, \{x_2\}, \{x_3\}\}$ UP generiert 2 leere Klauseln, aber das Minimum ist $uc = 1$ ($\{x_1\}$).
- **Resolution**: Nicht erlaubt, denn durch Resolution kann sich die Anzahl erfüllter Klauseln erhöhen.
- **Lernen**: Nicht erlaubt, denn Resolution ist nicht sound.

Gültige Transformationen

- **One-Literal Rule**: nach der Entscheidung $\ell = \top$, lösche alle Klauseln, die ℓ enthalten und lösche $\neg\ell$ von allen anderen Klauseln.
- **Pure-Literal Rule**: Falls ℓ nur in einer Polarität auftritt, dann lösche alle Regeln, die ℓ enthalten.
- **Complementary Unit Clause Rule**: Gibt es genau zwei Unit Klauseln $\{\ell\}$ und $\{\neg\ell\}$, so ersetze diese durch eine leere Klausel.

Zähle für obige Transformationen, wieviele Klauseln erfüllt wurden und addiere Wert auf das Ergebnis der transformierten MaxSAT Instanz.