SAT Solving und Anwendungen

MaxSAT und Pseudo Boolesche Optimierung

Prof. Dr. Wolfgang Küchlin
Dipl. Inform. Christoph Zengler
Rouven Walter, M.Sc. Informatik

Universität Tübingen

25. Juni 2013

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Motivierendes Beispiel: Minimum Vertex Cover

Minimum Vertex Cover

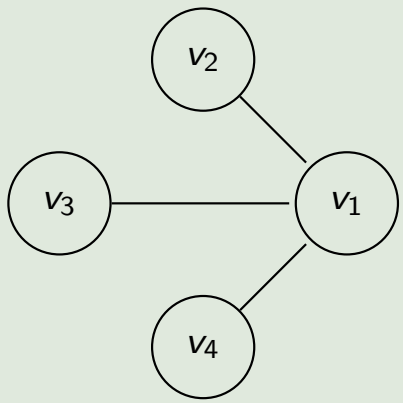
Gegeben:

- Graph $G = (V, E)$

Gesucht:

- Minimum Vertex Cover
  - Vertex Cover $U \subseteq V$: Für jede Kante $(v_i, v_j) \in E$ gilt $v_i \in U$ oder $v_j \in U$
  - Minimum Vertex Cover: Vertex Cover U mit minimaler Größe

Beispiel



Motivierendes Beispiel: Minimum Vertex Cover

Minimum Vertex Cover

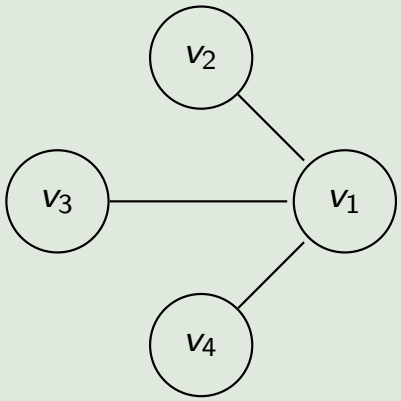
Gegeben:

- Graph $G = (V, E)$

Gesucht:

- Minimum Vertex Cover
  - Vertex Cover $U \subseteq V$: Für jede Kante $(v_i, v_j) \in E$ gilt $v_i \in U$ oder $v_j \in U$
  - Minimum Vertex Cover: Vertex Cover U mit minimaler Größe

Beispiel



- Mögliches Vertex Cover: $\{v_2, v_3, v_4\}$
- Minimales Vertex Cover ???

Motivierendes Beispiel: Minimum Vertex Cover

Minimum Vertex Cover

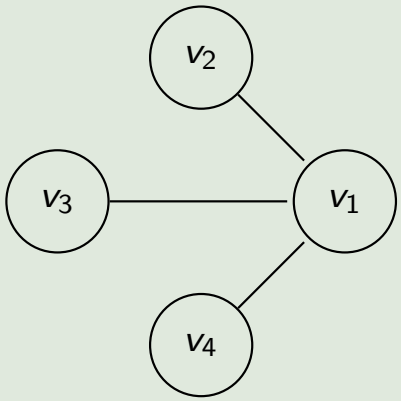
Gegeben:

- Graph $G = (V, E)$

Gesucht:

- Minimum Vertex Cover
  - Vertex Cover $U \subseteq V$: Für jede Kante $(v_i, v_j) \in E$ gilt $v_i \in U$ oder $v_j \in U$
  - Minimum Vertex Cover: Vertex Cover U mit minimaler Größe

Beispiel



- Mögliches Vertex Cover: $\{v_2, v_3, v_4\}$
- Minimales Vertex Cover: $\{v_1\}$

Fahrplan für heute

1. **Boolesche Optimierung**
   - MaxSAT
   - Pseudo-Boolesche Optimierung (PBO)
   - Konversion zwischen beiden Problemen
2. **Beispielanwendungen**
   - Software Package Konfiguration
3. **Techniken**
   - Cardinality Constraints
   - Pseudo-Boolesche Constraints
4. **Praktische Algorithmen**
   - Branch & Bound
   - Iteratives SAT Solving
   - Core-Guided Algorithmen

Was ist MaxSAT?

Beispiel

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

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

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

Ist diese Klauselmenge erfüllbar?

Was ist MaxSAT?

Nein!

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

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

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

Formel ist unerfüllbar!

MaxSAT:

- Finde eine Belegung, welche die Anzahl erfüllter Klauseln maximiert
- In obiger Formel können maximal **10 Klauseln** gleichzeitig erfüllt sein
- Viele Varianten von MaxSAT

Varianten von MaxSAT — 1

MaxSAT

- Alle Klauseln sind weich (soft), d.h. müssen nicht unbedingt erfüllt werden.
- Maximiere Anzahl der erfüllten weichen Klauseln
- Minimiere Anzahl der unerfüllten weichen Klauseln

Partial MaxSAT

- Es gibt harte Klauseln, die erfüllt sein müssen
- Minimiere Anzahl der unerfüllten weichen Klauseln

Anwendungsbeispiel: Softwarekonfiguration

- Harte Klauseln: Bestimmte Pakete/Plugins müssen installiert werden/bleiben
- Weiche Klauseln: Zusätzliche Pakete/Plugins mit Abhängigkeiten
- Frage: Wie viele und welche Pakete können maximal zusätzlich installiert werden?

Varianten von MaxSAT — 2

Weighted MaxSAT

- Alle Klauseln sind weich
- Alle Klauseln werden mit Gewichten versehen
- Minimiere die Summe der Gewichte der unerfüllten Klauseln

Weighted Partial MaxSAT

- Es gibt harte Klauseln, die erfüllt sein müssen
- Weiche Klauseln werden mit Gewichten versehen
- Minimiere die Summe der Gewichte der unerfüllten weichen Klauseln

Anwendungsbeispiel: PC Konfiguration

- Harte Klauseln: Ein PC kann nur ein Motherboard haben, nur bestimmte Prozessoren funktionieren mit bestimmten Motherboards, ...
- Weiche Klauseln: Spezielle Kundenwünsche mit Priorisierung (Am liebsten 8GB Ram, wenn am Ende noch möglich ein Diskettenlaufwerk, ...)
- Frage: Optimal mögliche PC Konfiguration?

Notation

Notation: gewichtete Klausel

(c, w): gewichtete Klausel

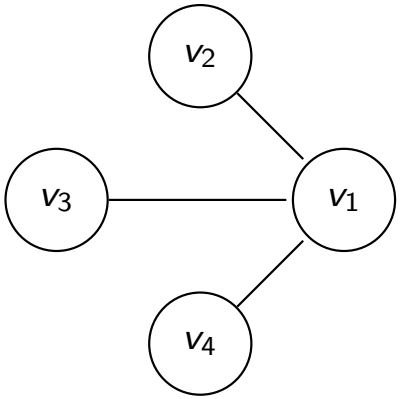
- c ist eine Menge an Literalen (Klausel)
- w ist ein nicht-negativer Integerwert oder ∞ (oder $\top$)
  - Kosten (Strafe), wenn man c nicht erfüllt

Notation: Klauselmenge

φ: Menge gewichteter Klauseln

- Weiche Klauseln: (c, w) mit $w < \infty$
  - Kosten, wenn man c nicht erfüllt
- Harte Klauseln: (c, ∞)
  - Klausel c muss erfüllt werden

Modellierungsbeispiel



Minimum Vertex Cover

- Vertex Cover $U \subseteq V$: Für jede Kante $(v_i, v_j) \in E$ gilt $v_i \in U$ oder $v_j \in U$
- Minimum Vertex Cover: Vertex Cover U mit minimaler Größe

Beispiel

Partial MaxSAT Codierung

- Variablen: x_i für jeden Knoten $v_i \in C$, mit $x_i = 1$, genau dann wenn $v_i \in U$
- Harte Klauseln: $(x_i \vee x_j)$ für jede Kante $(v_i, v_j) \in E$ *(... ist Vertex Cover)*
- Weiche Klauseln: $(\neg x_i)$ für jeden Knoten $v_i \in V$ *(... ist minimal)*
  - Je weniger Variablen auf 1 gesetzt werden, desto weniger die Strafe

Codierung:

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

MaxSAT vs MinUNSAT

MinUNSAT

- Finde Belegung, welche die Anzahl U der unerfüllten Klauseln minimiert
  - diese Belegung maximiert dann auch die Anzahl erfüllter Klauseln
- $\mathrm{MinUNSAT}(\varphi) := U$.
- Varianten Partial, Weighted und Partial Weighted möglich (wie bei MaxSAT)
  - Weighted MinUNSAT: minimiere das Gewicht der unerfüllten Klauseln

Zusammenhang

Sei φ Klauselmenge, dann gilt:

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

Hinweis: Zusammenhang gilt auch für alle Varianten

- Oft werden Algorithmen für MinUNSAT angegeben, welche aufgrund des Zusammenhangs auch für MaxSAT benutzt werden können

MaxSAT vs MinUNSAT — Beispiel

Beispiel

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

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

- $\mathrm{PartialWeightedMaxSAT}(\varphi) = 6$
  Belegung: $x \mapsto 1$, $y \mapsto 1$
- $\mathrm{PartialWeightedMinUNSAT}(\varphi) = 7$
  Belegung: $x \mapsto 1$, $y \mapsto 1$
- Es gilt:

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

Pseudo-Boolesche Constraints & Optimierung

Pseudo-Boolesche Constraints

- Boolesche Variablen: $x_1, \ldots, x_n$
- Lineare Ungleichungen

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

Pseudo-Boolesche Optimierung

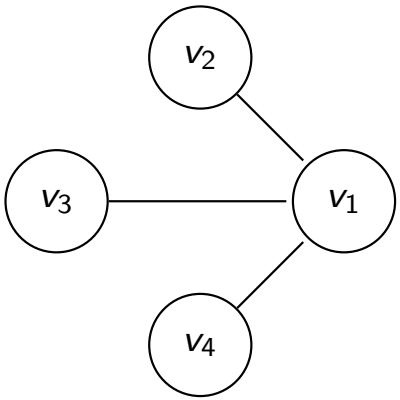
Minimiere

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

bezüglich n PB Constraints

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

Modellierungsbeispiel



Minimum Vertex Cover

- Vertex Cover $U \subseteq V$: Für jede Kante $(v_i, v_j) \in E$ gilt $v_i \in U$ oder $v_j \in U$
- Minimum Vertex Cover: Vertex Cover U mit minimaler Größe

Codierung als PBO

- Variablen: x_i für jeden Knoten $v_i \in V$, mit $x_i = 1$ wenn $v_i \in U$
- PB Constraints: $x_i + x_j \geq 1$ für jede Kante $(v_i, v_j) \in E$
- Zielfunktion: Minimiere die Anzahl an Variablen, die auf 1 gesetzt sind
  $\Rightarrow$ D.h. minimiere Anzahl an Knoten in der Vertex Cover

Probleminstanz: $x_1 + x_2 \geq 1, x_1 + x_3 \geq 1, x_1 + x_4 \geq 1$
Zielfunktion: $x_1 + x_2 + x_3 + x_4$

Von (reinem) MaxSAT zu PBO

Ausgehend von einer unerfüllbaren CNF Formel φ:

1. Generiere φ' aus φ:
   - Ersetze jede Klausel c_i mit $c_i' = c_i \cup \{r_i\}$
   - r_i ist eine neue Variable (Selektorvariable)
   - Problem ist jetzt trivial zu lösen, indem alle r_i auf 1 gesetzt werden
2. Minimiere Zielfunktion $\sum r_i$

Beispiel

- CNF Formel φ:

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

- Modifizierte Formel φ':

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

- PB Constraints: $x_1 + \bar{x_2} + r_1 \geq 1, x_1 + x_2 + r_2 \geq 1, \bar{x_1} + r_3 \geq 1$
- Zielfunktion zu minimieren: $r_1 + r_2 + r_3$

Von Partial (Weighted) MaxSAT zu PBO

Ausgehend von einer Partial (Weighted) MaxSAT Instanz mit φ_H und φ_S

Generiere PBO Instanz:

$$\text{minimiere } \sum w_i r_i, \text{ so dass } \varphi_T \text{ gilt, mit}$$

- $\varphi_T = \varphi'_H \cup \varphi'_S$
- Jede harte Klausel (c, ∞) wird auf eine Klausel c in φ'_H abgebildet
- Jede weiche Klausel (c, w) wird auf eine Klausel $(c_i \vee r_i)$ abgebildet und der Term $w_i r_i$ wird zur Zielfunktion addiert

Beispiel

- Originalproblem: $(\{x, y, \neg z\}, \infty), (\{x, \neg y\}, 4), (\{\neg x\}, 8), (\{x, z\}, 2)$
- $\varphi_T = \underbrace{(x, y, \neg z)}_{\varphi'_H}, \underbrace{(x, \neg y, r_1), (\neg x, r_2), (x, z, r_3)}_{\varphi'_S}$
- PB Constraints: $x + y + \bar{z} \geq 1, x + \bar{y} + r_1 \geq 1, \bar{x} + r_2 \geq 1, x + z + r_3 \geq 1$
- Zielfunktion zu minimieren: $4r_1 + 8r_2 + 2r_3$

Anwendung: Software Package Upgrades

- Mögliche Software Pakete: $P = \{p_1, \ldots, p_n\}$
- Variable x_i für jedes Paket $p_i \in P$. $x_i = 1$, gdw. p_i installiert ist
- Constraints für jedes Paket p_i: (p_i, D_i, C_i)
  - D_i: Abhängigkeiten (required packages) beim Installieren von p_i
  - C_i: Konflikte (disallowed packages) beim Installieren von p_i
- Beispielproblem: Maximum Installability
  - Maximale Anzahl an Paketen, die installiert werden können
  - Paket Constraints sind harte Klauseln
  - Jedes einzelne Paket ist eine weiche Klausel

Beispiel

Paket 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 Codierung

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

Cardinality Constraints

Fragestellung: Wie behandelt man Cardinality Constraints

- Allgemeine Form: $\sum_{j=1}^{n} x_j \bowtie k$ mit $\bowtie \in \{<, \leq, =, \geq, >\}$
- Im Speziellen AtMost1 Constraints: $\sum_{j=1}^{n} x_j \leq 1$

Lösung 1

Benutze speziellen PB Solver

- Schwer, mit Fortschritten im SAT Bereich mitzuhalten
- Für SAT/UNSAT codieren die besten Solver bereits in CNF
  - Z.B. Minisat+, QMaxSat, MSUnCore, (W)PM2

Lösung 2

- Codiere Cardinality Constraints zu CNF
- Benutze SAT solver

Equals, AtLeast1 & AtMost1 Constraints

Spezielle Behandlung von Constraints mit 1 auf der rechten Seite — Kommen sehr häufig vor:

- Fahrzeug muss genau einen Motor haben ...
- Genau ein Grafikkartentreiber muss ausgewählt sein ...
- Höchstens ein SAT Solver kann in Eclipse installiert sein ...

Kodierungen

- $\sum_{j=1}^{n} x_j = 1$: Kodiere mit $(\sum_{j=1}^{n} x_j \leq 1) \wedge (\sum_{j=1}^{n} x_j \geq 1)$
- $\sum_{j=1}^{n} x_j \geq 1$: Kodiere mit $(x_1 \vee x_2 \vee \cdots \vee x_n)$
- $\sum_{j=1}^{n} x_j \leq 1$: Kodiere mit:
  - Paarweise Kodierung:
    - Klauseln: $\mathcal{O}(n^2)$; Keine Hilfsvariablen
  - Sequentieller Counter
    - Klauseln: $\mathcal{O}(n)$; Hilfsvariablen: $\mathcal{O}(n)$
  - Bitweise Kodierung:
    - Klauseln: $\mathcal{O}(n \log n)$; Hilfsvariablen $\mathcal{O}(\log n)$

Allgemeine Cardinality Constraints

Allgemeine Form: $\sum_{j=1}^{n} x_j \leq k$ (oder $\sum_{j=1} x_j \geq k$)

Kodierungen

- Sequentielle Counter
  - Klauseln/Variablen: $\mathcal{O}(nk)$
- BDDs
  - Klauseln/Variablen: $\mathcal{O}(nk)$
- Sortiernetzwerke
  - Klauseln/Variablen: $\mathcal{O}(n \log^2 n)$
- Cardinality Netzwerke
  - Klauseln/Variablen: $\mathcal{O}(n \log^2 k)$

Pseudo-Boolesche Constraints — 1

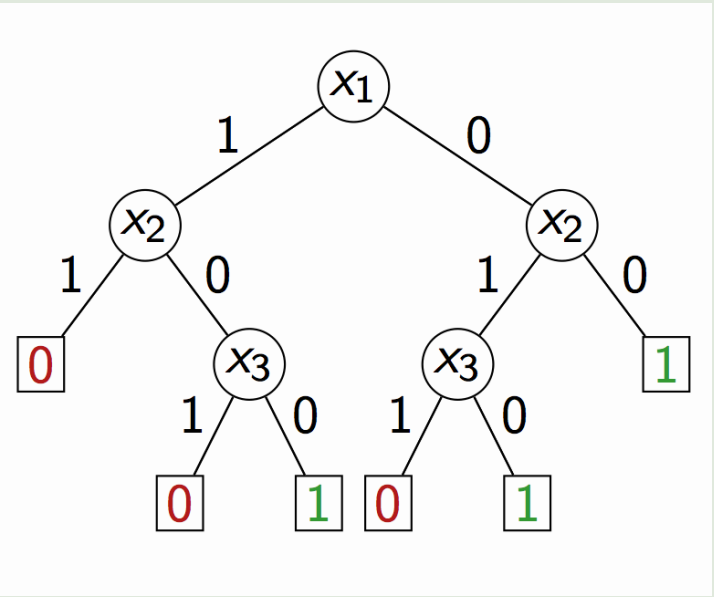
Allgemeine Form: $\sum_{j=1}^{n} a_j x_j \leq b$

- Kodierung z.B. mit BDD
- Im worst case: Exponentielle Anzahl an Klauseln

Beispiel

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

- Kodiere BDD, d.h. analysiere Variablen durch Abziehen der Koeffizienten
- Extrahiere ITE Schaltkreis aus dem BDD



Pseudo-Boolesche Constraints — 1

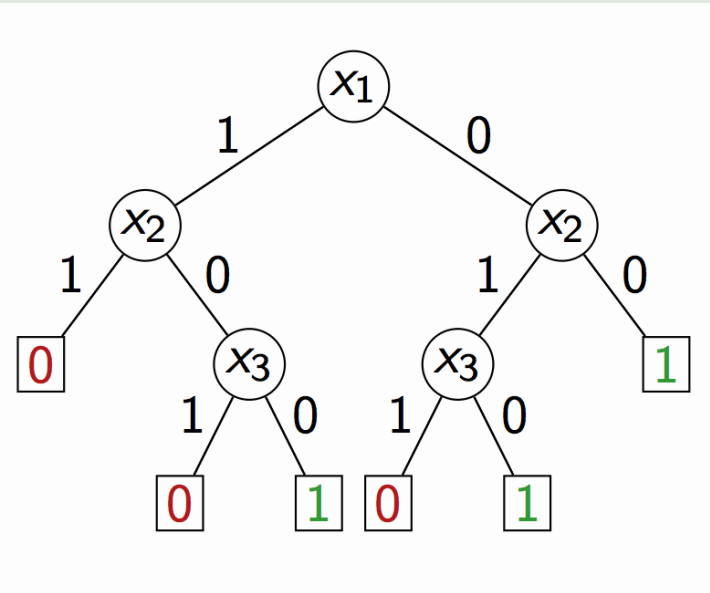
Allgemeine Form: $\sum_{j=1}^{n} a_j x_j \leq b$

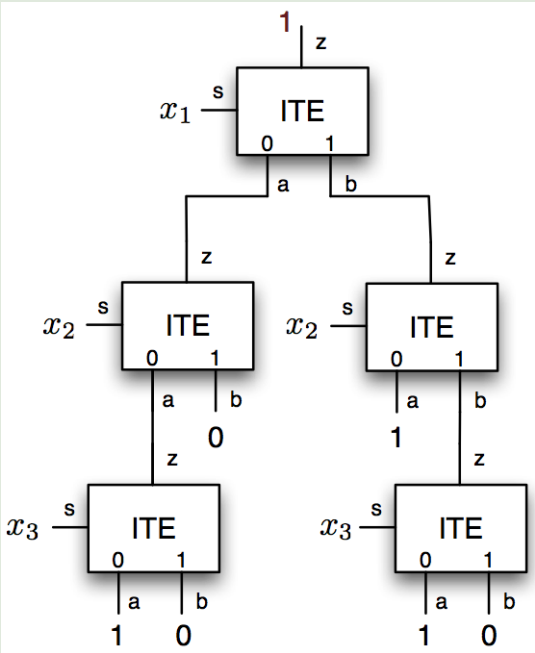
- Kodierung z.B. mit BDD
- Im worst case: Exponentielle Anzahl an Klauseln

Beispiel

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

- Kodiere BDD, d.h. analysiere Variablen durch Abziehen der Koeffizienten
- Extrahiere ITE Schaltkreis aus dem BDD





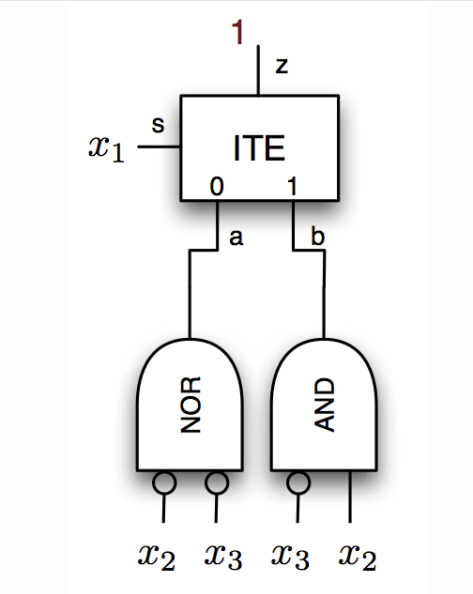
Pseudo-Boolesche Constraints — 2

Beispiel (continued)

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

- Extrahiere ITE Schaltkreis aus dem BDD
- Simplifiziere Schaltkreis

Finaler Schaltkreis:



- Konvertiere Schaltkreis zu CNF

Branch & Bound Algorithmen — 1

- Betrachte MinUNSAT (die Minimierungs-Variante von MaxSAT), die die Anzahl unerfüllter Klauseln *uc* minimiert.
- Exploriere den Suchraum aller Belegungen und betrachte zu jedem Zeitpunkt die Obergrenze $UB \geq uc$ und die Untergrenze $LB \leq uc$ für die Anzahl unerfüllter Klauseln *uc*.
- *UB* ist der beste (globale) Wert von *uc*, der für eine der bisher betrachteten vollständigen Belegungen erzielt wurde – schlechter wird das Ergebnis nicht.
- Wir senken durch weitere Suche die Obergrenze *UB* bis zu einem Minimum.
- Die Untergrenze *LB* ist eine (konservative) Abschätzung der momentanen (lokalen) Situation im Suchraum: besser kann das Ergebnis ausgehend von der momentanen partiellen Belegung π nicht mehr werden.
- *LB* ist die Summe aus den unter π schon leeren Klauseln plus einer minimalen Anzahl von Klauseln, die zusätzlich leer werden, egal wie π weiter vervollständigt wird.
- Vereinfache das Problem durch Inferenzregeln, um Suchraum einzuschränken. Die Vereinfachungen müssen für jede Belegung die Anzahl unerfüllter Klauseln unverändert lassen.

Branch & Bound Algorithmen — 2

Grundlegender B&B Algorithmus

Gegeben MaxSAT Instanz φ

- B&B exploriert den Suchbaum, der den Suchraum aller möglichen Belegungen von φ repräsentiert, in Tiefensuche
- An jedem Knoten:
  - UB Upper Bound/Obergrenze: die beste Lösung für die Anzahl unerfüllter Klauseln, die bisher für eine komplette Belegung gefunden wurde, also $UB \geq uc$
  - LB Lower Bound/Untergrenze: Summe der Anzahl der Klauseln, die mit der aktuellen partiellen Belegung unerfüllt sind + Minimum min der Anzahl an Klauseln, die noch unerfüllt werden, wenn die aktuelle Belegung komplettiert wird. LB ist also ein Schätzwert (underestimation) mit $uc \geq LB$.
- Vergleiche an jedem Knoten UB mit LB
  - $LB \geq UB$: Es muss nicht mehr weitergesucht werden; Backtracking zu einem höheren Level; Untersuchung eines anderen Sub-Baumes
  - $LB < UB$: Versuche bessere Lösung zu finden; Belege eine weitere Variable
- **Finale Lösung:** Wert von UB wenn der ganze Suchbaum exploriert wurde

Der Branch & Bound Algorithmus

Der Algorithmus

Algorithm 1: BnB(φ, UB)

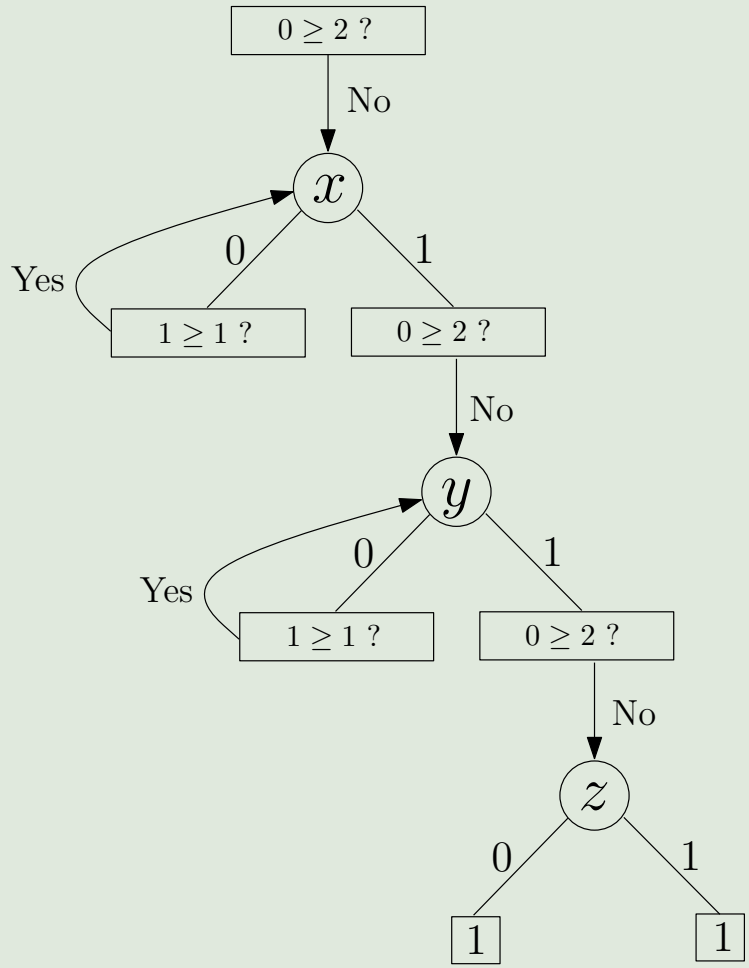
Input: MaxSAT Instanz φ, Obergrenze UB
$\varphi = \text{simplifyFormula}(\varphi)$
if $\varphi = \emptyset$ *oder* φ *enthält nur leere Klauseln* **then**
  return $\#emptyClauses(\varphi)$
$LB = \#\text{emptyClauses}(\varphi) + \text{underestimation}(\varphi)$
if $LB \geq UB$ **then**
  return UB
$x = \text{selectVariable}(\varphi)$
$UB = \min(UB, BnB(\varphi_x, UB))$
return $\min(UB, BnB(\varphi_{\bar{x}}, UB))$

- UB ist initial die Anzahl an Klauseln oder die Anzahl derjenigen Klauseln, die durch eine beliebige Belegung unerfüllt sind
- **Interessante Frage:** Wie wird underestimation(φ) implementiert?

Der Branch & Bound Algorithmus — Beispiel

Beispiel

- $\varphi = \{\{x\}, \{\neg x \vee y\}, \{z \vee \neg y\}, \{x \vee z\}, \{\neg z\}\}$
- Initiale obere Grenze: $UB = 2$
  (alle Variablen auf false gesetzt)
- Variablenordnung: x, y, z
- Berechnung LB: $\#emptyClauses(\varphi)$
  D.h. underestimation$(\varphi) = 0$
- Ergebnis: 1



Wie findet man eine gute Unterbewertung?

Effiziente Berechnung der Lower Bound ausgehend von partieller Belegung π:

- Einfach: Zähle alle Klauseln, die durch π falsifiziert sind
- Besser: Schätze durch untere Schranke ab, wie viele Klauseln in jedem Fall noch falsifiziert werden, wenn π komplettiert wird.

Basisvariante nach Wallace und Freuder

$LB(\varphi) = \#\text{emptyClauses}(\varphi) + \sum_{x \text{ occurs in } \varphi} \min(ic(x), ic(\bar{x}))$

- φ: CNF Belegung mit aktueller partiellen Belegung
- $ic(x)$ bzw. $ic(\bar{x})$: Inconsistency Count. Anzahl der Unit Klauseln von φ, die x bzw. $\bar{x}$ enthalten.

Star Rule

Suche Teilmenge der Art $\{\{\ell_1\}, \{\ell_2\}, \ldots, \{\ell_k\}, \{\bar{\ell_1} \vee \bar{\ell_2} \vee \ldots \vee \bar{\ell_k}\}\}$. Jede solche Klauselmenge erzwingt für jede Belegung mindestens *eine* unerfüllte Klausel.

Weitere Regeln existieren.

Iteratives SAT Solving

MaxSAT lösen mit Hilfe von SAT Solving

- Reduziere MaxSAT auf SAT Solving
- Vorteile:
  - Verwendung von effizienten Techniken aus dem SAT Bereich (Unit Propagation, Watched Literals, Non-Chronological Backtraking, Clause Learning, etc), welche nicht oder nur teilweise auf MaxSAT übertragen werden können
  - MaxSAT Solver muss nicht von Grund auf neu geschrieben (und optimiert) werden
- Idee:
  - Rufe iterativ SAT Solver auf bis Optimum gefunden ist
  - Restriktion der Klauseln durch blocking Variablen und Cardinality Constraints
- Zwei Ansätze:
  - Reduktion auf (reines) SAT Solving
  - Reduktion auf SAT Solving mit unsatisfiable core Erweiterung

Binäre Suche für MaxSAT

Der Algorithmus

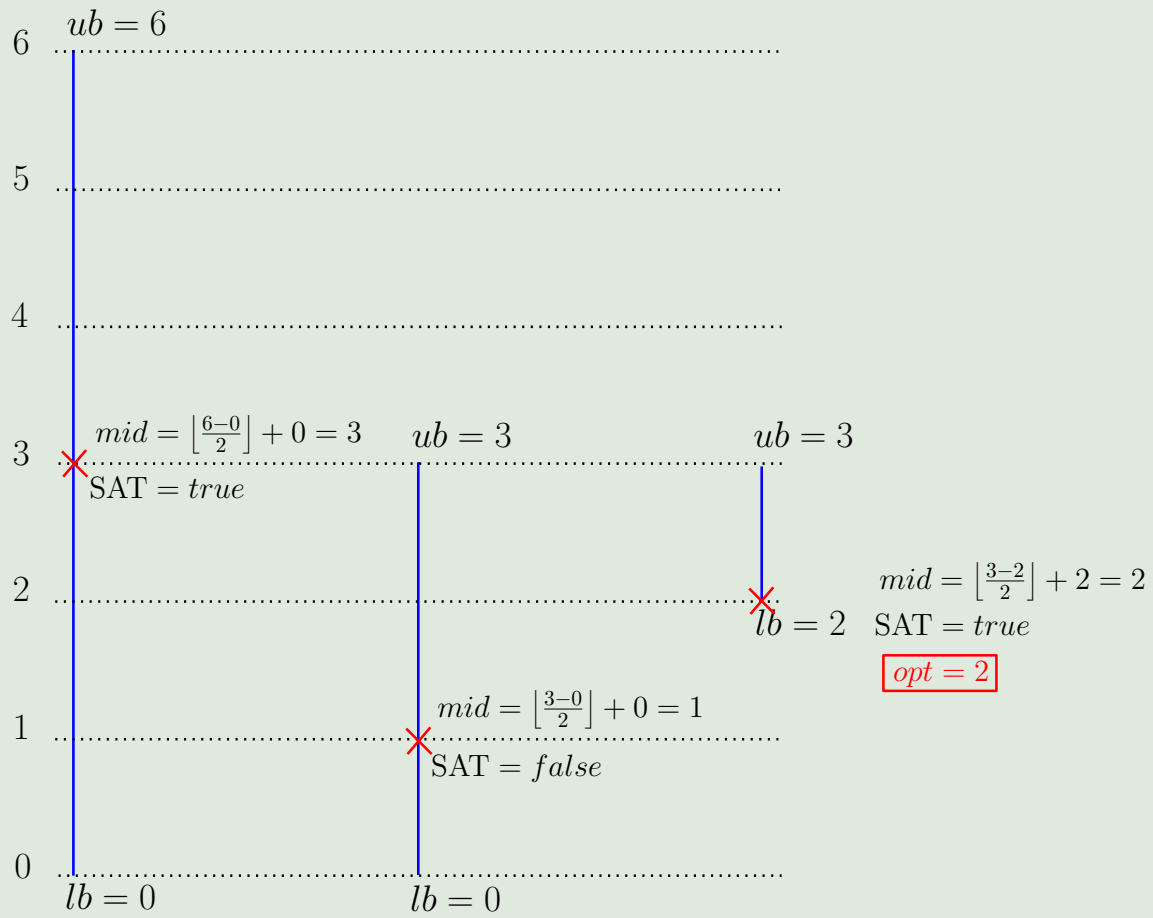
Algorithm 2: BinarySearch(φ)

Input: MaxSAT Instanz $\varphi = \{C_1, \ldots, C_m\}$
$\varphi \leftarrow \{C_1 \vee b_1, \ldots, C_m \vee b_m\}$
$lb \leftarrow 0, ub \leftarrow m$
$mid \leftarrow \left\lfloor \frac{ub-lb}{2} \right\rfloor$
while $lb < ub$ **do**
  if $\mathrm{SAT}(\varphi \cup \mathrm{CNF}(\sum_{i=1}^{m} b_i \leq mid))$ **then**
    $ub \leftarrow mid$
  else
    $lb \leftarrow mid + 1$
  $mid \leftarrow \left\lfloor \frac{ub-lb}{2} \right\rfloor + lb$
return ub

Binäre Suche für MaxSAT — Beispiel

Beispiel

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



SAT-basierter Ansatz von Le Berre für MaxSAT

Der Algorithmus

Algorithm 3: LeBerre(φ)

Input: MaxSAT Instanz $\varphi = \{C_1, \ldots, C_m\}$
$BV = \{b_1, \ldots, b_m\}$
$\varphi \leftarrow \{C_1 \vee b_1, \ldots, C_m \vee b_m\}$
$ub \leftarrow m$
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*

Literatur: Biere et al. Handbook of Satisfiablity, 19.6, S.635. 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.

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)

- Bei einem minimal unsat core handelt es sich nicht zwangsläufig um den kleinsten unsat core den φ enthält, aber um ein lokales Minimum
- 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! 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

Fu & Malik Algorithmus für Partial MaxSAT

Der Algorithmus

Algorithm 4: 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 5: $\mathrm{PM2}(\varphi)$

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

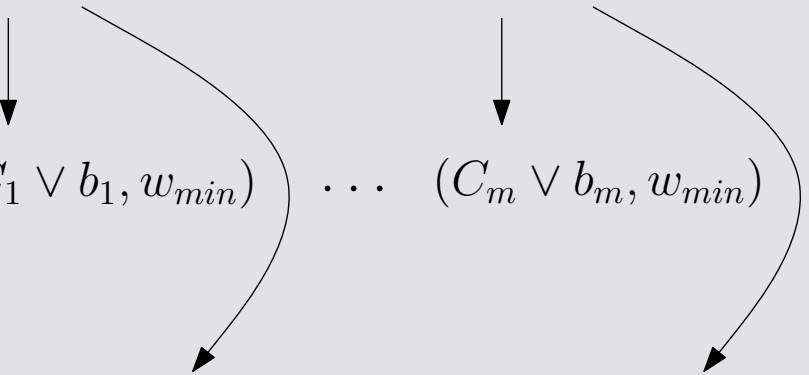
- Erweiterung des Fu & Malik Algorithmus zu Partial **Weighted** MaxSAT
- Prüfe iterativ mit Hilfe eines SAT Solvers (mit unsat core Unterstützung), ob Instanz erfüllbar ist
- Falls ja, so ist Optimum gefunden (alle Klauseln erfüllbar)
- Falls nein, so kann **mindestens** das Minimum $w_{\min}$ der Gewichte 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 6: $\mathrm{WPM1}(\varphi)$

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

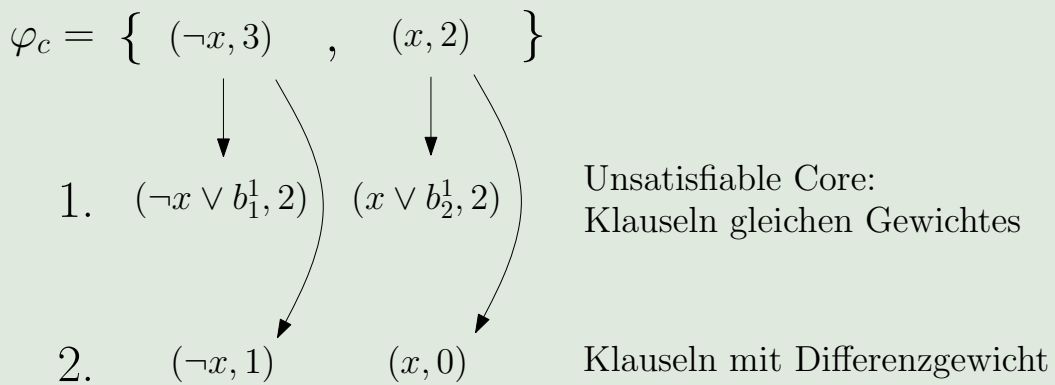
Beispiel

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

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

- Man arbeitet mit Transformationsregeln, die eine Teilmenge der Klauseln durch eine andere Teilmenge ersetzen. Das Hinzufügen von Klauseln verändert i.A. das Ergebnis: $\mathsf{MaxSAT}(\{\{x\}\}) \neq \mathsf{MaxSAT}(\{\{x\},\{x\}\})$.
- 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.