

# SAT Solving und Anwendungen

## MaxSAT und Pseudo Boolesche Optimierung

Prof. Dr. Wolfgang Küchlin  
Dipl. Inf. Christoph Zengler

Universität Tübingen

28. Juni 2011



# Motivierendes Beispiel: Minimum Vertex Cover

## Minimum Vertex Cover

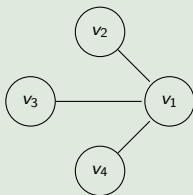
### Gegeben:

- Graph  $G = (V, E)$

### Gesucht:

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

## Beispiel



# Motivierendes Beispiel: Minimum Vertex Cover

## Minimum Vertex Cover

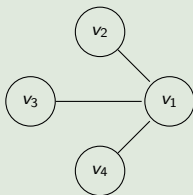
### Gegeben:

- Graph  $G = (V, E)$

### Gesucht:

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

## Beispiel



- Mögliche Vertex Cover:  
 $\{v_2, v_3, v_4\}$
- Minimale Vertex Cover ???

# Motivierendes Beispiel: Minimum Vertex Cover

## Minimum Vertex Cover

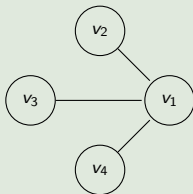
### Gegeben:

- Graph  $G = (V, E)$

### Gesucht:

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

## Beispiel



- Mögliche Vertex Cover:  $\{v_2, v_3, v_4\}$
- Minimale 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

$$x_6 \vee x_2$$

$$\neg x_6 \vee x_2$$

$$\neg x_2 \vee x_1$$

$$\neg x_1$$

$$\neg x_6 \vee x_8$$

$$x_6 \vee \neg x_8$$

$$x_2 \vee x_4$$

$$\neg x_4 \vee x_5$$

$$x_7 \vee x_5$$

$$\neg x_7 \vee x_5$$

$$\neg x_5 \vee x_3$$

$$\neg x_3$$

Ist diese Klauselmenge erfüllbar?

# Was ist MaxSAT?

Nein!

$$x_6 \vee x_2$$

$$\neg x_6 \vee x_2$$

$$\neg x_2 \vee x_1$$

$$\neg x_1$$

$$\neg x_6 \vee x_8$$

$$x_6 \vee \neg x_8$$

$$x_2 \vee x_4$$

$$\neg x_4 \vee x_5$$

$$x_7 \vee x_5$$

$$\neg x_7 \vee x_5$$

$$\neg x_5 \vee x_3$$

$$\neg x_3$$

**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)
- 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: Klausel

$(c, w)$ : gewichtete Klausel

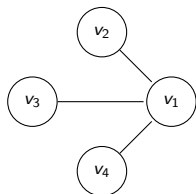
- $c$  ist eine Menga an Literalen (Klausel)
- $w$  ist ein nicht-negativer Integerwert oder  $\infty$  (oder  $\top$ )
  - Kosten (Strafe), wenn man  $c$  nicht erfüllt

## Notation: Klauselmenge

$\varphi$ : Menge gewichteter Klauseln

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

# Modellierungsbeispiel



## Minimum Vertex Cover

- Vertex Cover  $U \subseteq V$ : Für jede Kante  $(v_i, v_j) \in E$  ist entweder  $v_i \in U$  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:

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

# Pseudo-Boolesche Constraints & Optimierung

## Pseudo-Boolesche Constraints

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

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

## Pseudo-Boolesche Optimierung

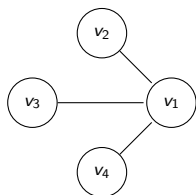
Minimiere

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

bezüglich  $n$  PB Constraints

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

# Modellierungsbeispiel



## Minimum Vertex Cover

- Vertex Cover  $U \subseteq V$ : Für jede Kante  $(v_i, v_j) \in E$  ist entweder  $v_1 \in U$  oder  $v_2 \in U$
- 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

**Problem Instanz:**  $x_1 + x_2 \geq 1, x_1 + x_3 \geq 1, x_1 + x_4 \geq 1$

**Zielfunktion:**  $x_1 + x_2 + x_3 + x_4$

# Von (reinem) MaxSAT zu PBO

Ausgehend von einer unerfüllbaren CNF Formel  $\varphi$ :

- ① Generiere  $\varphi'$  aus  $\varphi$ :
  - 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  $\varphi$ :

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

- Modifizierte Formel  $\varphi'$ :

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

- 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  $\varphi_H$  und  $\varphi_S$

**Generiere PBO Instanz:**

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

- $\varphi_T = \varphi'_H \cup \varphi'_S$
- Jede harte Klausel  $(c, \infty)$  wird auf eine Klausel  $c$  in  $\varphi'_H$  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:  $P = \{p_1, \dots, p_n\}$
- 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

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



# 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  $(x_1 \vee x_2 \vee \dots \vee x_n)$
- $\sum_{j=1}^n x_j \leq 1$ : 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  $\sum_{j=1}^n x_j \geq k$ )

## 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

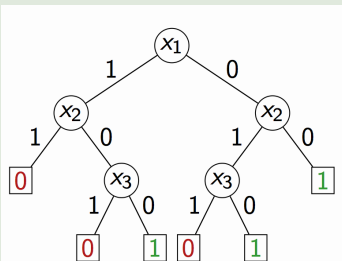
Allgemein Form:  $\sum_{j=1}^n a_j x_j \leq b$

- Kodierung z.B. mit BDD
- Im worstcase: 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

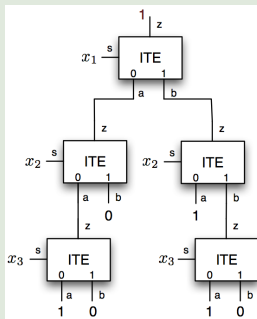
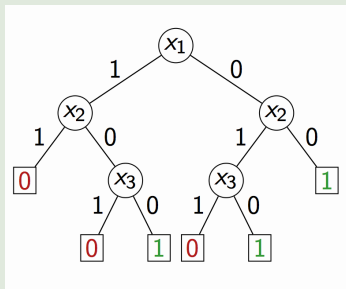
Allgemein Form:  $\sum_{j=1}^n a_j x_j \leq b$

- Kodierung z.B. mit BDD
- Im worstcase: 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



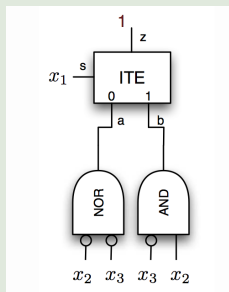
# Pseudo-Boolesche Constraints — 2

## 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

- Betrachtet die Minimierungs-Variante von MaxSAT
- Bei MaxSAT darf keine UP ausgeführt werden:

$$\{\{x_1\}, \{\neg x_1, \neg x_2\}, \{\neg x_1, \neg x_3\}, \{x_2\}, \{x_3\}\}$$

⇒ Standard B&B Suche führt keine UP aus

- Nähere Obergrenze  $UB$  für Anzahl unerfüllter Klauseln an ein Optimum an
- Schätze Untergrenze  $LB$  ab
- Inferenzregeln, um Suchraum einzuschränken
- Optional: Benutze Lokale Suche um  $UB$ s zu identifizieren

# Branch & Bound Algorithmen — 2

## Grundlegender B&B Algorithmus

Gegeben MaxSAT Instanz  $\varphi$

- B&B exploriert den Suchbaum, der den Suchraum aller möglichen Belegungen von  $\varphi$  repräsentiert, in Tiefensuche
- An jedem Knoten:
  - *UB* Upper Bound/Obergrenze: die beste Lösung, die bisher für eine komplette Belegung gefunden wurde
  - *LB* Lower Bound/Untergrenze: Summe der Anzahl der Klauseln, die mit der aktuellen partiellen Belegung unerfüllt sind + Unterbewertung der Anzahl an Klauseln, die noch unerfüllt werden, wenn die aktuelle Belegung komplettiert wird
- 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
  - $LB \leq UB$ : 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: MaxSAT( $\varphi$ , $UB$ )

**Input:** MaxSAT Instanz  $\varphi$ , Obergrenze  $UB$

$\varphi = \text{simplifyFormula}(\varphi)$

**if**  $\varphi = \emptyset$  *oder*  $\varphi$  enthält nur leere Klauseln **then**

**return**  $\#emptyClauses(\varphi)$

$LB = \#emptyClauses(\varphi) + \text{underestimation}(\varphi)$

**if**  $LB \geq UB$  **then**

**return**  $UB$

$x = \text{selectVariable}(\varphi)$

$UB = \min(UB, \text{MaxSAT}(\varphi_{\bar{x}}, UB))$

**return**  $UB = \min(UB, \text{MaxSAT}(\varphi_x, UB))$

- $UB$  ist initial Anzahl an Klauseln oder Anzahl an Klauseln, die durch eine beliebige Belegung unerfüllt sind
- **Interessante Frage:** Wie wird  $\text{underestimation}(\varphi)$  implementiert?

# Wie findet man eine gute Unterbewertung?

Berechnung der Lower Bound an einem Knoten mit partieller Belegung:

- Einfach: Zähle alle Klauseln, die durch die aktuelle partielle Belegung unerfüllt sind
- Besser: Schätze ab, wie viele Klauseln noch unerfüllt werden, wenn diese partielle Belegung komplettiert wird (und benutze dann eine Unterbewertung, d.h. Schranke, die sicher gilt)

## Basisvariante nach Wallace und Freuder

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

- $\varphi$ : CNF Belegung mit aktueller partiellen Belegung
- $ic(x)$  bzw.  $ic(\bar{x})$ : Inconsistency Count. Anzahl der Unit Klauseln von  $\varphi$ , die  $x$  bzw.  $\bar{x}$  enthalten.
- Nur ein Beispiel (sehr basic!)
- Viele weitere Verbesserungen denkbar, z.B. weitere Inferenzregeln