SAT Solving und Anwendungen

MaxSAT und Pseudo Boolesche Optimierung

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

Universität Tübingen

19. Juni 2012

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Motivierendes Beispiel: Minimum Vertex Cover

Minimum Vertex Cover

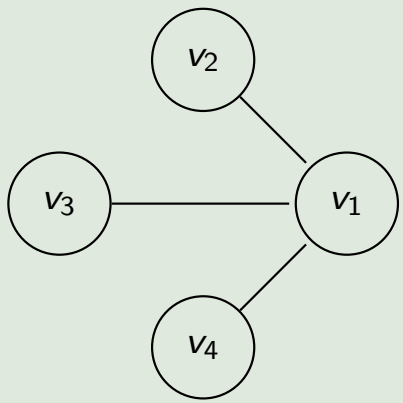
Gegeben:

- Graph $G = (V, E)$

Gesucht:

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

Beispiel



Motivierendes Beispiel: Minimum Vertex Cover

Minimum Vertex Cover

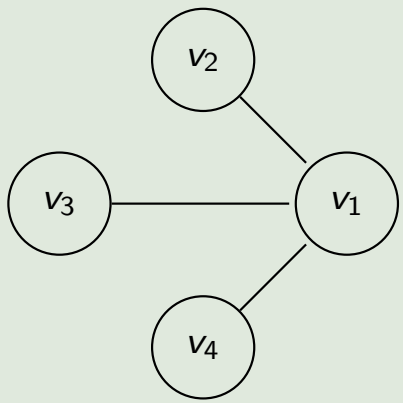
Gegeben:

- Graph $G = (V, E)$

Gesucht:

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

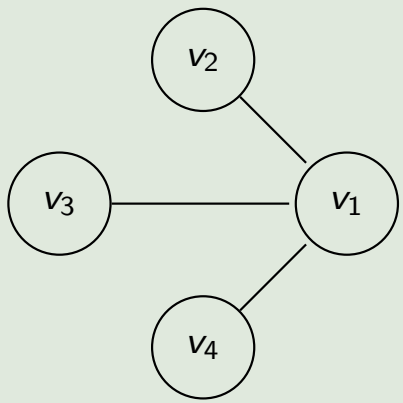
Gegeben:

- Graph $G = (V, E)$

Gesucht:

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

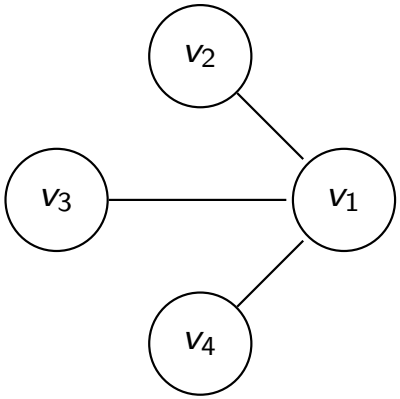
- c ist eine Menge an Literalen (Klausel)
- w ist ein nicht-negativer Integerwert oder ∞ (oder $\top$)
  - Kosten (Strafe), wenn man c nicht erfüllt

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 $(v_i, v_j) \in E$ ist entweder $v_1 \in U$ oder $v_2 \in U$
- 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: $x_1, \ldots, x_n$
- 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

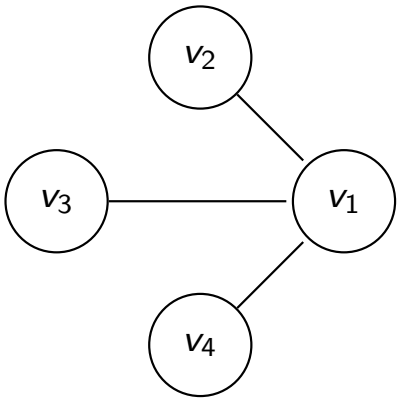
Minimiere

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

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 $(v_i, v_j) \in E$ 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 $x_1 = 1$ 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

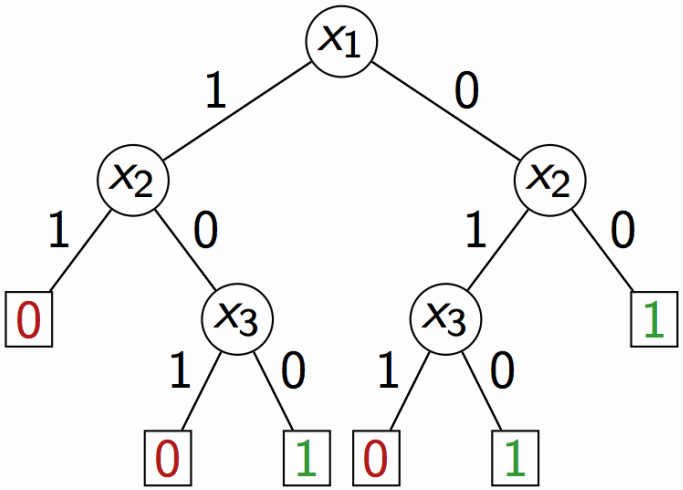
Allgemeine Form: $\sum_{j=1}^{n} a_j x_j \leq b$

- 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

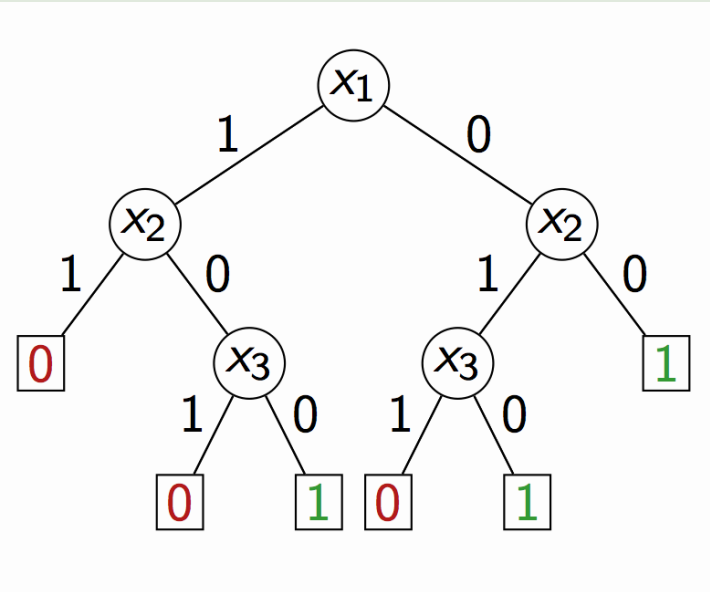
Allgemeine Form: $\sum_{j=1}^{n} a_j x_j \leq b$

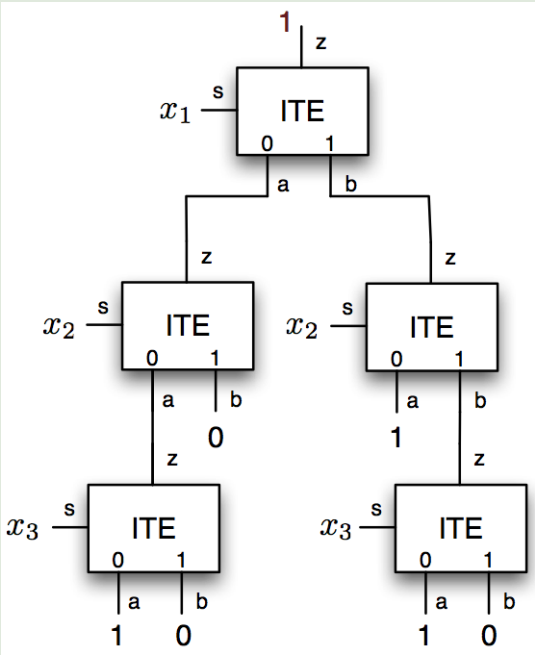
- Kodierung z.B. mit BDD
- Im worst case: Exponentielle Anzahl an Klauseln

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





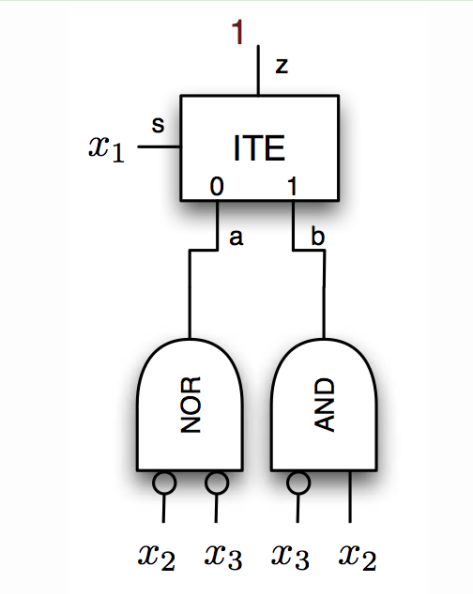
Pseudo-Boolesche Constraints — 2

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 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
  - $LB \leq UB$: 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(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
$x = \text{selectVariable}(\varphi)$
$UB = \min(UB, MaxSAT(\varphi_{\bar{x}}, UB))$
return $UB = \min(UB, MaxSAT(\varphi_x, UB))$

- 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?

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.

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 ein Spezialfall von Resolution. 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.