MaxSAT Solving with CDCL SAT solvers

Ruben Martins
rmartins@cs.utexas.edu

Automated Logical Reasoning, February 16, 2016
Software Package Upgradeability Problem

<table>
<thead>
<tr>
<th>Package</th>
<th>Dependencies</th>
<th>Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>${p_2 \lor p_3}$</td>
<td>${p_4}$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>${p_3}$</td>
<td>${}$</td>
</tr>
<tr>
<td>$p_3$</td>
<td>${p_2}$</td>
<td>${p_4}$</td>
</tr>
<tr>
<td>$p_4$</td>
<td>${p_2 \land p_3}$</td>
<td>${}$</td>
</tr>
</tbody>
</table>

- Set of packages we want to install: $\{p_1, p_2, p_3, p_4\}$
- Each package $p_i$ has a set of **dependencies**: Packages that must be installed for $p_i$ to be installed
- Each package $p_i$ has a set of **conflicts**: Packages that cannot be installed for $p_i$ to be installed
### Solving Software Package Upgradeability Problem

<table>
<thead>
<tr>
<th>Package</th>
<th>Dependencies</th>
<th>Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>${p_2 \lor p_3}$</td>
<td>${p_4}$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>${p_3}$</td>
<td>${}$</td>
</tr>
<tr>
<td>$p_3$</td>
<td>${p_2}$</td>
<td>${p_4}$</td>
</tr>
<tr>
<td>$p_4$</td>
<td>${p_2 \land p_3}$</td>
<td>${}$</td>
</tr>
</tbody>
</table>

How can we encode this problem to Boolean Satisfiability?
### Solving Software Package Upgradeability Problem

<table>
<thead>
<tr>
<th>Package</th>
<th>Dependencies</th>
<th>Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>${ p_2 \lor p_3 }$</td>
<td>${ p_4 }$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>${ p_3 }$</td>
<td>${ }$</td>
</tr>
<tr>
<td>$p_3$</td>
<td>${ p_2 }$</td>
<td>${ p_4 }$</td>
</tr>
<tr>
<td>$p_4$</td>
<td>${ p_2 \land p_3 }$</td>
<td>${ }$</td>
</tr>
</tbody>
</table>

How can we encode this problem to Boolean Satisfiability?

**Hint** Encode dependencies, conflicts, and installing all packages
Solving Software Package Upgradeability Problem

<table>
<thead>
<tr>
<th>Package</th>
<th>Dependencies</th>
<th>Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>${p_2 \lor p_3}$</td>
<td>${p_4}$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>${p_3}$</td>
<td>${}$</td>
</tr>
<tr>
<td>$p_3$</td>
<td>${p_2}$</td>
<td>${p_4}$</td>
</tr>
<tr>
<td>$p_4$</td>
<td>${p_2 \land p_3}$</td>
<td>${}$</td>
</tr>
</tbody>
</table>

How can we encode this problem to Boolean Satisfiability?

- **Encoding dependencies:**
  - $p_1 \Rightarrow (p_2 \lor p_3) \equiv (\neg p_1 \lor p_2 \lor p_3)$
  - $p_2 \Rightarrow p_3 \equiv (\neg p_2 \lor p_3)$
  - $p_3 \Rightarrow p_2 \equiv (\neg p_3 \lor p_2)$
  - $p_4 \Rightarrow (p_2 \land p_3) \equiv (\neg p_4 \lor p_2) \land (\neg p_4 \lor p_3)$
Solving Software Package Upgradeability Problem

<table>
<thead>
<tr>
<th>Package</th>
<th>Dependencies</th>
<th>Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>${p_2 \lor p_3}$</td>
<td>${p_4}$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>${p_3}$</td>
<td>${}$</td>
</tr>
<tr>
<td>$p_3$</td>
<td>${p_2}$</td>
<td>${p_4}$</td>
</tr>
<tr>
<td>$p_4$</td>
<td>${p_2 \land p_3}$</td>
<td>${}$</td>
</tr>
</tbody>
</table>

How can we encode this problem to Boolean Satisfiability?

- **Encoding conflicts:**
  - $p_1 \Rightarrow \neg p_4 \equiv (\neg p_1 \lor \neg p_4)$
  - $p_3 \Rightarrow \neg p_4 \equiv (\neg p_3 \lor \neg p_4)$
Solving Software Package Upgradeability Problem

<table>
<thead>
<tr>
<th>Package</th>
<th>Dependencies</th>
<th>Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>${p_2 \lor p_3}$</td>
<td>${p_4}$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>${p_3}$</td>
<td>${}$</td>
</tr>
<tr>
<td>$p_3$</td>
<td>${p_2}$</td>
<td>${p_4}$</td>
</tr>
<tr>
<td>$p_4$</td>
<td>${p_2 \land p_3}$</td>
<td>${}$</td>
</tr>
</tbody>
</table>

How can we encode this problem to Boolean Satisfiability?

- Encoding installing all packages:
  - $(p_1) \land (p_2) \land (p_3) \land (p_4)$
Solving Software Package Upgradeability Problem

CNF Formula:

\[
\neg p_1 \lor p_2 \lor p_3 \quad \neg p_2 \lor p_3 \quad \neg p_3 \lor p_2 \\
\neg p_4 \lor p_2 \quad \neg p_4 \lor p_3 \quad \neg p_1 \lor \neg p_4 \quad \neg p_3 \lor \neg p_4
\]

\[
p_1 \quad p_2 \quad p_3 \quad p_4
\]

Boolean Satisfiability (SAT):

- Decide if the formula is satisfiable or unsatisfiable
Solving Software Package Upgradeability Problem

CNF Formula:

\[ \neg p_1 \lor p_2 \lor p_3 \quad \neg p_2 \lor p_3 \quad \neg p_3 \lor p_2 \]

\[ \neg p_4 \lor p_2 \quad \neg p_4 \lor p_3 \quad \neg p_1 \lor \neg p_4 \quad \neg p_3 \lor \neg p_4 \]

- Formula is unsatisfiable
- Can you find an unsatisfiable subformula?

**Hint** There are several with 3 clauses!
Solving Software Package Upgradeability Problem

CNF Formula:

\[
\neg p_1 \lor p_2 \lor p_3 \quad \neg p_2 \lor p_3 \quad \neg p_3 \lor p_2 \\
\neg p_4 \lor p_2 \quad \neg p_4 \lor p_3 \quad \neg p_1 \lor \neg p_4 \quad \neg p_3 \lor \neg p_4
\]

- Formula is unsatisfiable
- We cannot install all packages
- How many packages can we install?
What is Maximum Satisfiability?

- **Maximum Satisfiability (MaxSAT):**
  - Optimized version of SAT
  - All clauses in the formula are soft
  - Minimize number of unsatisfied soft clauses
What is Maximum Satisfiability?

- **Maximum Satisfiability (MaxSAT):**
  - Optimized version of SAT
  - All clauses in the formula are soft
  - Minimize number of unsatisfied soft clauses

- **Partial MaxSAT:**
  - Clauses in the formula are soft or hard
  - Hard clauses must be satisfied
  - Minimize number of unsatisfied soft clauses

- **Weighted Partial MaxSAT:**
  - Clauses in the formula are soft or hard
  - Weights associated with soft clauses
  - Minimize sum of weights of unsatisfied soft clauses
How to encode Software Package Upgradeability?

Software Package Upgradeability problem as MaxSAT:

- What are the hard constraints?
  - *(Hint)* Dependencies, conflicts or installation packages?

- What are the soft constraints?
  - *(Hint)* Dependencies, conflicts or installation packages?
How to encode Software Package Upgradeability?

Software Package Upgradeability problem as MaxSAT:

- What are the hard constraints?
  - Dependencies and conflicts

- What are the soft constraints?
  - Installation of packages
Software Package Upgradeability Problem as MaxSAT

Partial MaxSAT Formula:

\[ \varphi_h \text{ (Hard): } \neg p_1 \lor p_2 \lor p_3 \quad \neg p_2 \lor p_3 \quad \neg p_3 \lor p_2 \]

\[ \neg p_4 \lor p_2 \quad \neg p_4 \lor p_3 \quad \neg p_1 \lor \neg p_4 \quad \neg p_3 \lor \neg p_4 \]

\[ \varphi_s \text{ (Soft): } p_1 \quad p_2 \quad p_3 \quad p_4 \]

- Dependencies and conflicts are encoded as hard clauses
- Installation of packages are encoded as soft clauses
- **Goal:** maximize the number of installed packages
Software Package Upgradeability Problem as MaxSAT

Partial MaxSAT Formula:

\[ \varphi_h \text{ (Hard):} \quad \neg p_1 \lor p_2 \lor p_3 \quad \neg p_2 \lor p_3 \quad \neg p_3 \lor p_2 \]

\[ \quad \neg p_4 \lor p_2 \quad \neg p_4 \lor p_3 \quad \neg p_1 \lor \neg p_4 \quad \neg p_3 \lor \neg p_4 \]

\[ \varphi_s \text{ (Soft):} \quad p_1 \quad p_2 \quad p_3 \quad p_4 \]

- Dependencies and conflicts are encoded as hard clauses
- Installation of packages are encoded as soft clauses
- **Optimal solution** (3 out 4 packages are installed):
  \[ \mu = \{p_1 = 1, p_2 = 1, p_3 = 1, p_4 = 0\} \]
Why is MaxSAT Important?

- Many real-world applications can be encoded to MaxSAT:
  - Software package upgradeability:
    - Eclipse platform uses MaxSAT for managing the plugins dependencies
  - Error localization in C code
  - Debugging of hardware designs
  - Haplotyping with pedigrees
  - Reasoning over Biological Networks
  - Course timetabling
  - Combinatorial auctions
  - ...

- MaxSAT algorithms are very effective for solving real-world problems
The MaxSAT (r)evolution – plain industrial instances

Number x of instances solved in y seconds

CPU time in seconds

Number of instances

Open-WBO-In
pmifumax-13
WPM1-11
wbo-1.4a-10
wbo1.6-cnf-12

Source: [MaxSAT 2014 organizers]

48.1% more instances solved!
The MaxSAT (r)evolution — plain industrial instances

Source: [MaxSAT 2014 organizers]
The MaxSAT (r)evolution — partial

Source: [MaxSAT 2014 organizers]
The MaxSAT (r)evolution — partial

Number x of instances solved in y seconds

Source: [MaxSAT 2014 organizers]

71.5% more instances solved!
The MaxSAT (r)evolution – weighted partial

Number x of instances solved in y seconds

Source: [MaxSAT 2014 organizers]
The MaxSAT (r)evolution — weighted partial

Number x of instances solved in y seconds

- Eva500a
- WPM1-2013
- WPM1-11
- pwbo2.1-12
- wbo-1.4a-wcnf-10

CPU time in seconds

Number of instances

51.5% more instances solved!

Source: [MaxSAT 2014 organizers]
Outline

- MaxSAT Algorithms:
  - Linear search algorithms
  - Unsatisfiability-based algorithms

- Partitioning in MaxSAT:
  - Use the structure of the problem to guide the search

- Using MaxSAT solvers
MaxSAT Algorithms

- MaxSAT algorithms build on SAT solver technology
- MaxSAT algorithms use constraints not defined in causal form:
  - AtMost1 constraints, $\sum_{j=1}^{n} x_j \leq 1$
  - General cardinality constraints, $\sum_{j=1}^{n} x_j \leq k$
  - Pseudo-Boolean constraints, $\sum_{j=1}^{n} a_j x_j \leq k$
- Efficient encodings to CNF
CNF Encodings

Naive encoding for AtMost1 Constraints:

\[ x_1 + x_2 + x_3 \leq 1: \]

\[ (\neg x_1 \lor \neg x_2) \land (\neg x_1 \lor \neg x_3) \land (\neg x_2 \lor \neg x_3) \]

For a general AtMost1 constraint \( r_1 + r_2 + \ldots + r_n \leq 1: \)

For each pair \((r_i, r_j)\) add the clause \((\neg r_i \lor \neg r_j)\)

Complexity: \(O(n^2)\) clauses

More efficient encodings can be used! (PBLib’15)
CNF Encodings

Sequential counters

AtMost1 constraints:
- Clauses/Variables: $O(n)$

General cardinality constraints:
- Clauses/Variables: $O(n \kappa)$

Sequential weighted counters

Pseudo-Boolean constraints:
- Clauses/Variables: $O(n \kappa)$

(Sinz (CP’05))

(Hölldobler et al. (KI’12))
MaxSAT algorithms

SAT-UNSAT Linear Search algorithm:

- Optimum solution (OPT):
  - Assignment with **minimum** cost

- Upper Bound (UB) value:
  - Cost **greater than or equal** to OPT

SAT-UNSAT Linear search algorithms:
- Iterative calls to a SAT solver
- Refine UB value until OPT is found
MaxSAT algorithms

SAT-UNSAT Linear Search algorithm:

- **Optimum solution (OPT):** Assignment with *minimum* cost
- **Upper Bound (UB) value:** Cost *greater than or equal* to OPT

**SAT-UNSAT Linear search algorithms:**
- Iterative calls to a SAT solver
- Refine UB value until OPT is found
MaxSAT algorithms

SAT-UNSAT Linear Search algorithm:

- Optimum solution (OPT):
  - Assignment with minimum cost

- Upper Bound (UB) value:
  - Cost greater than or equal to OPT

- SAT-UNSAT Linear search algorithms:
  - Iterative calls to a SAT solver
  - Refine UB value until OPT is found
Partial MaxSAT Formula:

\( \varphi_h \) (Hard):
\[ \neg x_2 \lor \neg x_1 \lor x_2 \lor \neg x_3 \]

\( \varphi_s \) (Soft):
\[ x_1 \lor x_3 \lor x_2 \lor \neg x_1 \lor \neg x_3 \lor x_1 \]
Partial MaxSAT Formula:

\[ \varphi_h : \bar{x}_2 \lor \bar{x}_1 \lor x_2 \lor \bar{x}_3 \]

\[ \varphi_s : x_1 \lor r_1 \lor x_3 \lor r_2 \lor x_2 \lor \bar{x}_1 \lor r_3 \lor \bar{x}_3 \lor x_1 \lor r_4 \]

- Relax all soft clauses
- Relaxation variables:
  - \( V_R = \{r_1, r_2, r_3, r_4\} \)
  - If a soft clause \( \omega_i \) is unsatisfied, then \( r_i = 1 \)
  - If a soft clause \( \omega_i \) is satisfied, then \( r_i = 0 \)
Partial MaxSAT Formula:

\[
\varphi_h : \bar{x}_2 \lor x_1 \quad x_2 \lor \bar{x}_3
\]

\[
\varphi_s : x_1 \lor r_1 \quad x_3 \lor r_2 \quad x_2 \lor \bar{x}_1 \lor r_3 \quad \bar{x}_3 \lor x_1 \lor r_4
\]

\[V_R = \{r_1, r_2, r_3, r_4\}\]

- Formula is satisfiable
  - \[\nu = \{x_1 = 1, x_2 = 0, x_3 = 0, r_1 = 0, r_2 = 1, r_3 = 1, r_4 = 0\}\]

**Goal:** Minimize the number of relaxation variables assigned to 1
Partial MaxSAT Formula:

\[ \varphi_h : \bar{x}_2 \lor \bar{x}_1 \lor x_2 \lor \bar{x}_3 \]
\[ \varphi_s : x_1 \lor r_1 \lor x_3 \lor r_2 \lor x_2 \lor \bar{x}_1 \lor r_3 \lor \bar{x}_3 \lor x_1 \lor r_4 \]

\[ \mu = 2 \quad V_R = \{ r_1, r_2, r_3, r_4 \} \]

- \( r_2 \) and \( r_3 \) were assigned truth value 1:
  - Current solution unsatisfies 2 soft clauses
- Can less than 2 soft clauses be unsatisfied?
Partial MaxSAT Formula:

\[ \varphi_h : \bar{x}_2 \lor x_1 \lor x_2 \lor \bar{x}_3 \quad \text{CNF}(\sum_{r_i \in V_R} r_i \leq 1) \]

\[ \varphi_s : x_1 \lor r_1 \lor x_3 \lor r_2 \lor x_2 \lor \bar{x}_1 \lor r_3 \lor \bar{x}_3 \lor x_1 \lor r_4 \]

\[ \mu = 2 \quad V_R = \{ r_1, r_2, r_3, r_4 \} \]

- Add cardinality constraint that excludes solutions that unsatisfies 2 or more soft clauses:
  - CNF\( (r_1 + r_2 + r_3 + r_4 \leq 1) \)
Partial MaxSAT Formula:

\( \varphi_h : \bar{x}_2 \lor x_1 \lor x_2 \lor \bar{x}_3 \lor \text{CNF}(\sum_{n \in V_R} r_i \leq 1) \)

\( \varphi_s : x_1 \lor r_1 \lor x_3 \lor r_2 \lor x_2 \lor \bar{x}_1 \lor r_3 \lor \bar{x}_3 \lor x_1 \lor r_4 \)

\( \mu = 2 \quad V_R = \{r_1, r_2, r_3, r_4\} \)

- Formula is unsatisfiable:
  - There are no solutions that unsatisfy 1 or less soft clauses
Partial MaxSAT Formula:

$$\varphi_h: \bar{x}_2 \lor \bar{x}_1 \lor x_2 \lor \bar{x}_3$$

$$\varphi_s: x_1 \lor x_3 \lor x_2 \lor \bar{x}_1 \lor \bar{x}_3 \lor x_1$$

$$\mu = 2 \quad V_R = \{r_1, r_2, r_3, r_4\}$$

- **Optimal solution**: given by the last model and corresponds to unsatisfying 2 soft clauses:
  - $$\nu = \{x_1 = 1, x_2 = 0, x_3 = 0\}$$

- The same procedure can be generalized to **weighted**
MaxSAT algorithms

- We have just seen a search on the upper bound.

- What other kind of search can we do to find an optimal solution?
MaxSAT algorithms

- We have just seen a search on the **upper bound**

- What other kind of search can we do to find an optimal solution?

- What if we start searching from the **lower bound**?
MaxSAT algorithms

Unsatisfiability-based algorithms:

- **Lower Bound (LB) value:**
  - Cost *smaller than or equal* to OPT

- **Unsatisfiability-based algorithms:**
  - Iteratively increase the LB until a satisfiable call is performed
  - Use unsatisfiable subformulas to refine LB value until OPT is found
MaxSAT algorithms

Unsatisfiability-based algorithms:

- **Lower Bound (LB) value:**
  - Cost $\text{smaller than or equal}$ to OPT

- **Unsatisfiability-based algorithms:**
  - Iteratively increase the LB until a satisfiable call is performed
  - Use unsatisfiable subformulas to refine LB value until OPT is found
MaxSAT algorithms

Unsatisfiability-based algorithms:

- Lower Bound (LB) value:
  - Cost smaller than or equal to OPT

- Unsatisfiability-based algorithms:
  - Iteratively increase the LB until a satisfiable call is performed
  - Use unsatisfiable subformulas to refine LB value until OPT is found
Partial MaxSAT Formula:

\[ \varphi_h : \quad \bar{x}_2 \lor \bar{x}_1 \quad \bar{x}_2 \lor \bar{x}_3 \]

\[ \varphi_s : \quad x_1 \lor r_1 \quad x_3 \lor r_2 \quad x_2 \lor \bar{x}_1 \lor r_3 \quad \bar{x}_3 \lor x_1 \lor r_4 \]

- Relax all soft clauses
- Relaxation variables:
  - \( V_R = \{ r_1, r_2, r_3, r_4 \} \)
  - If a soft clause \( \omega_i \) is **unsatisfied**, then \( r_i = 1 \)
  - If a soft clause \( \omega_i \) is **satisfied**, then \( r_i = 0 \)
Partial MaxSAT Formula:

\[ \varphi_h : \quad \overline{x_2} \lor \overline{x_1} \quad x_2 \lor \overline{x_3} \]

\[ \varphi_s : \quad x_1 \lor r_1 \quad x_3 \lor r_2 \quad x_2 \lor \overline{x_1} \lor r_3 \quad \overline{x_3} \lor x_1 \lor r_4 \]

\[ \mu = 2 \quad V_R = \{ r_1, r_2, r_3, r_4 \} \]

- Add cardinality constraint that excludes solutions that unsatisfies 1 or more soft clauses:
  - \( \text{CNF}(r_1 + r_2 + r_3 + r_4 \leq 0) \)
Linear Search Algorithms UNSAT-SAT

Partial MaxSAT Formula:

\[ \varphi_h : \overline{x}_2 \lor \overline{x}_1 \lor x_2 \lor x_3 \lor \text{CNF}(\sum_{i \in V_R} r_i \leq 0) \]

\[ \varphi_s : x_1 \lor r_1 \lor x_3 \lor r_2 \lor x_2 \lor \overline{x}_1 \lor r_3 \lor \overline{x}_3 \lor x_1 \lor r_4 \]

- Formula is unsatisfiable:
  - There are no solutions that unsatisfy 0 or less soft clauses

- Add cardinality constraint that excludes solutions that unsatisfies 2 or more soft clauses:
  - \( \text{CNF}(r_1 + r_2 + r_3 + r_4 \leq 1) \)
Partial MaxSAT Formula:

\[ \varphi_h : \bar{x}_2 \lor \bar{x}_1 \lor x_2 \lor \bar{x}_3 \quad \text{CNF}(\sum_{r_i \in V_R} r_i \leq 1) \]

\[ \varphi_s : x_1 \lor r_1 \lor \bar{x}_3 \lor r_2 \lor x_2 \lor \bar{x}_1 \lor r_3 \lor \bar{x}_3 \lor x_1 \lor r_4 \]

- Formula is unsatisfiable:
  - There are no solutions that unsatisfy 1 or less soft clauses

- Add cardinality constraint that excludes solutions that unsatisfies 3 or more soft clauses:
  - \( \text{CNF}(r_1 + r_2 + r_3 + r_4 \leq 2) \)
Partial MaxSAT Formula:

\[ \varphi_h : \quad \bar{x}_2 \lor \bar{x}_1 \quad x_2 \lor \bar{x}_3 \quad \text{CNF}(\sum_{r_i \in V_R} r_i \leq 2) \]

\[ \varphi_s : \quad x_1 \lor r_1 \quad x_3 \lor r_2 \quad x_2 \lor \bar{x}_1 \lor r_3 \quad \bar{x}_3 \lor x_1 \lor r_4 \]

- Formula is satisfiable:
  - \( \mu = \{x_1 = 1, x_2 = 0, x_3 = 0, r_1 = 0, r_2 = 1, r_3 = 1, r_4 = 0\} \)

- Optimal solution unsatisfies 2 soft clauses

- The same procedure can be generalized to weighted
Unsatisfiability-based Algorithms

What are the problems of this algorithm?

(Hint) Number of relaxation variables? Size of the cardinality constraint? Other?
Unsatisfiability-based Algorithms

What are the problems of this algorithm?

(Hint) Number of relaxation variables? Size of the cardinality constraint? Other?

- We relax all soft clauses!
- The cardinality constraint contain as many literals as we have soft clauses!
- Can we do better?
Partial MaxSAT Formula:

\[ \varphi_h \text{ (Hard)}: \quad \bar{x}_2 \lor \bar{x}_1 \quad x_2 \lor x_3 \]

\[ \varphi_s \text{ (Soft)}: \quad x_1 \quad x_3 \quad x_2 \lor x_1 \quad \bar{x}_3 \lor x_1 \]
Partial MaxSAT Formula:

\[ \varphi_h: \overline{x}_2 \lor \overline{x}_1 \lor x_2 \lor \overline{x}_3 \]

\[ \varphi_s: x_1 \lor x_3 \lor x_2 \lor \overline{x}_1 \lor \overline{x}_3 \lor x_1 \]

Formula is unsatisfiable
Partial MaxSAT Formula:

\( \varphi_h: \bar{x}_2 \lor \bar{x}_1 \lor x_2 \lor \bar{x}_3 \)

\( \varphi_s: x_1 \lor x_3 \lor x_2 \lor \bar{x}_1 \lor \bar{x}_3 \lor x_1 \)

- Formula is unsatisfiable
- Identify an unsatisfiable core
Partial MaxSAT Formula:

\[ \varphi_h: \quad \bar{x}_2 \lor \bar{x}_1 \quad x_2 \lor \bar{x}_3 \quad \text{CNF}(r_1 + r_2 \leq 1) \]

\[ \varphi_s: \quad x_1 \lor r_1 \quad x_3 \lor r_2 \quad x_2 \lor \bar{x}_1 \quad \bar{x}_3 \lor x_1 \]

- Relax non-relaxed soft clauses in unsatisfiable core:
  - Add cardinality constraint that excludes solutions that unsatisfies 2 or more soft clauses:
    - \( \text{CNF}(r_1 + r_2 \leq 1) \)
  - Relaxation on demand instead of relaxing all soft clauses eagerly
Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

$\varphi_h: \bar{x}_2 \lor \bar{x}_1 \quad x_2 \lor \bar{x}_3 \quad \text{CNF}(r_1 + r_2 \leq 1)$

$\varphi_s: \quad x_1 \lor r_1 \quad x_3 \lor r_2 \quad x_2 \lor \bar{x}_1 \quad \bar{x}_3 \lor x_1$

Formula is unsatisfiable
Partial MaxSAT Formula:

$$\varphi_h: \overline{x}_2 \lor \overline{x}_1 \lor \overline{x}_3 \lor \text{CNF}(r_1 + r_2 \leq 1)$$

$$\varphi_s: x_1 \lor r_1 \lor x_3 \lor r_2 \lor x_2 \lor \overline{x}_1 \lor \overline{x}_3 \lor x_1$$

- Formula is unsatisfiable
- Identify an unsatisfiable core
Partial MaxSAT Formula:

\[ \varphi_h: \overline{x}_2 \lor \overline{x}_1 \lor x_2 \lor \overline{x}_3 \quad \text{CNF}(r_1 + r_2 + r_3 + r_4 \leq 2) \]

\[ \varphi_s: x_1 \lor r_1 \lor x_3 \lor r_2 \lor x_2 \lor \overline{x}_1 \lor r_3 \lor \overline{x}_3 \lor x_1 \lor r_4 \]

- Relax non-relaxed soft clauses in unsatisfiable core:
  - Add cardinality constraint that excludes solutions that unsatisfies 3 or more soft clauses:
    - \( \text{CNF}(r_1 + r_2 + r_3 + r_4 \leq 2) \)
  - Relaxation on demand instead of relaxing all soft clauses eagerly
Partial MaxSAT Formula:

\[ \varphi_h : \bar{x}_2 \lor x_1 \quad x_2 \lor \bar{x}_3 \quad \text{CNF}(r_1 + r_2 + r_3 + r_4 \leq 2) \]

\[ \varphi_s : x_1 \lor r_1 \quad x_3 \lor r_2 \quad x_2 \lor \bar{x}_1 \lor r_3 \quad \bar{x}_3 \lor x_1 \lor r_4 \]

- Formula is satisfiable:
  - \( \mu = \{x_1 = 1, x_2 = 0, x_3 = 0, r_1 = 0, r_2 = 1, r_3 = 1, r_4 = 0\} \)

- Optimal solution unsatisfies 2 soft clauses

- The same procedure can be generalized to weighted
Unsatisfiability-based Algorithms

What are the problems of this algorithm?

(Hint) Number of relaxation variables? Size of the cardinality constraint? Other?
Unsatisfiability-based Algorithms

- What are the problems of this algorithm?
  
  **(Hint)** Number of relaxation variables? Size of the cardinality constraint? Other?

- We must translate cardinality constraints into CNF!

- If the number of literals is large than we may generate a **very large** formula!

- Can we do better?
Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

\[ \varphi_h \text{ (Hard):} \quad \bar{x}_2 \lor \bar{x}_1 \lor x_2 \lor \bar{x}_3 \]

\[ \varphi_s \text{ (Soft):} \quad x_1 \land x_3 \land x_2 \land \bar{x}_1 \land \bar{x}_3 \lor x_1 \]
Partial MaxSAT Formula:

\[ \varphi_h: \overline{x}_2 \lor x_1 \land x_2 \lor \overline{x}_3 \]

\[ \varphi_s: x_1 \land x_3 \land x_2 \lor \overline{x}_1 \land \overline{x}_3 \lor x_1 \]

Formula is unsatisfiable
Unsatisfiability-based Algorithms (Fu&Malik SAT 2006)

Partial MaxSAT Formula:

\[ \varphi_h: \overline{x_2} \lor x_1 \lor \overline{x_2} \lor x_3 \]

\[ \varphi_s: x_1 \lor x_3 \lor x_2 \lor \overline{x_1} \lor \overline{x_3} \lor x_1 \]

- Formula is unsatisfiable
- Identify an unsatisfiable core
Partial MaxSAT Formula:

\[ \varphi_h: \bar{x}_2 \lor \bar{x}_1 \lor x_2 \lor \bar{x}_3 \]

\[ \varphi_s: x_1 \lor r_1 \lor x_3 \lor r_2 \]

\[ x_2 \lor \bar{x}_1 \lor \bar{x}_3 \lor x_1 \]

Relax unsatisfiable core:
- Add relaxation variables
- Add AtMost1 constraint
Partial MaxSAT Formula:

\[ \varphi_h: \bar{x}_2 \lor x_1 \quad x_2 \lor \bar{x}_3 \quad \text{CNF}(r_1 + r_2 \leq 1) \]

\[ \varphi_s: x_1 \lor r_1 \quad x_3 \lor r_2 \quad x_2 \lor \bar{x}_1 \quad \bar{x}_3 \lor x_1 \]

- Formula is unsatisfiable
Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

- $\varphi_h$: $ar{x}_2 \lor x_1 \lor x_2 \lor \bar{x}_3 \lor \text{CNF}(r_1 + r_2 \leq 1)$
- $\varphi_s$: $x_1 \lor r_1 \lor x_3 \lor r_2 \lor x_2 \lor \bar{x}_1 \lor \bar{x}_3 \lor x_1$

- Formula is unsatisfiable
- Identify an unsatisfiable core
Partial MaxSAT Formula:

$\varphi_h: \bar{x}_2 \lor \bar{x}_1 \quad x_2 \lor \bar{x}_3 \quad \text{CNF}(r_1 + r_2 \leq 1)$

$\varphi_s: x_1 \lor r_1 \lor r_3 \quad x_3 \lor r_2 \lor r_4 \quad x_2 \lor \bar{x}_1 \lor r_5 \quad \bar{x}_3 \lor x_1 \lor r_6$

- Relax unsatisfiable core:
  - Add relaxation variables
  - Add AtMost1 constraint

- Soft clauses may be relaxed multiple times
Partial MaxSAT Formula:

\[ \varphi_h: \quad \bar{x}_2 \lor \bar{x}_1 \quad x_2 \lor \bar{x}_3 \quad \text{CNF}(r_1 + r_2 \leq 1) \quad \text{CNF}(r_3 + \ldots + r_6 \leq 1) \]

\[ \varphi_s: \quad x_1 \lor r_1 \lor r_3 \quad x_3 \lor r_2 \lor r_4 \quad x_2 \lor \bar{x}_1 \lor r_5 \quad \bar{x}_3 \lor x_1 \lor r_6 \]

- Formula is satisfiable
- An optimal solution would be:
  \[ \nu = \{ x_1 = 1, x_2 = 0, x_3 = 0 \} \]
**Partial MaxSAT Formula:**

\[ \varphi_h: \overline{x}_2 \lor \overline{x}_1 \lor x_2 \lor \overline{x}_3 \]

\[ \varphi_s: x_1 \lor x_3 \lor x_2 \lor \overline{x}_1 \lor \overline{x}_3 \lor x_1 \]

- Formula is satisfiable
- An optimal solution would be:
  - \( \nu = \{ x_1 = 1, x_2 = 0, x_3 = 0 \} \)
- This assignment unsatisfies 2 soft clauses
Partial MaxSAT Formula:

\[ \varphi_h: \bar{x}_2 \lor \bar{x}_1 \quad x_2 \lor \bar{x}_3 \]

\[ \varphi_s: \begin{array}{cccc}
    & x_1 & x_3 & x_2 \lor \bar{x}_1 & \bar{x}_3 \lor x_1 \\
    x_1 &   &   &   &   \\
    x_3 &   &   &   &   \\
    x_2 \lor \bar{x}_1 &   &   &   &   \\
    \bar{x}_3 \lor x_1 &   &   &   &   \\
\end{array} \]

Formula is satisfiable

An optimal solution would be:

\[ \nu = \{ x_1 = 1, x_2 = 0, x_3 = 0 \} \]

This assignment unsatisfies 2 soft clauses
How to improve MaxSAT algorithms?

- MaxSAT algorithms use the **whole** MaxSAT formula at each iteration.
- Performance is related with the **unsatisfiable cores** given by the SAT solver:
  - Some unsatisfiable cores may be unnecessarily large.
How to improve MaxSAT algorithms?

- MaxSAT algorithms use the whole MaxSAT formula at each iteration

- Performance is related with the unsatisfiable cores given by the SAT solver:
  - Some unsatisfiable cores may be unnecessarily large

- **Solution:**
  - Partition the soft clauses into disjoint sets
  - Iteratively increase the size of the MaxSAT formula at each iteration
How to partition the soft clauses?

- **Graph representation** of the MaxSAT formula:
  - Example: Clause-Variable Incidence Graph

- **Community-based partitioning** to split the graph:
  - Minimization of the *modularity measure*
  - Vertices inside a partition should be densely connected
  - Vertices assigned to different partitions should be loosely connected

- Finding a set of partitions with an optimal modularity is **computationally hard**

- In practice, we use an **approximation algorithm** (Louvain)

  (Bondel et al. Journal of Statistical Mechanics 2008)
Community-based partitioning

SATGraf --- https://bitbucket.org/znewsham/satgraf

(normalized-f20c10b_001_area_delay.wcnf)
Graph representations for MaxSAT

There are many ways to represent MaxSAT as a graph:

- Clause-Variable Incidence Graph (CVIG) (Martins et al. SAT 2013)
- Variable Incidence Graph (VIG) (Martins et al. SAT 2013)
- Hypergraph (Martins et al. SAT 2013)
- Resolution Graph (Neves et al. SAT 2015)
- ...
Graph representations for MaxSAT

There are many ways to represent MaxSAT as a graph:

- **Clause-Variable Incidence Graph (CVIG)** (Martins et al. SAT 2013)
- Variable Incidence Graph (VIG) (Martins et al. SAT 2013)
- Hypergraph (Martins et al. SAT 2013)
- Resolution Graph (Neves et al. SAT 2015)
- . . .

How can we represent a SAT formula as a CVIG graph?
Clause-Variable Incidence Graph (CVIG)

- Vertices: Represent each variable and each clause
- Edges: There is an edge between each variable and each clause where the variable occurs
- Each edge has a corresponding weight:
  - More weight is given to clauses that establish edges between variables that occur frequently in soft clauses
Clause-Variable Incidence Graph (CVIG)

Vertices: Represent each variable and each clause

Edges: There is an edge between each variable and each clause where the variable occurs

Each edge has a corresponding weight:

More weight is given to clauses that establish edges between variables that occur frequently in soft clauses

Hard clauses:
\[ c_1 = x_1 \lor x_2 \]
\[ c_2 = \neg x_2 \lor x_3 \]
\[ c_3 = \neg x_1 \lor \neg x_3 \]

Soft clauses:
\[ c_4 = \neg x_1 \]
\[ c_5 = \neg x_3 \]
Clause-Variable Incidence Graph (CVIG)

- Vertices: Represent each variable and each clause
- Edges: There is an edge between each variable and each clause where the variable occurs
- Each edge has a corresponding weight:
  - More weight is given to clauses that establish edges between variables that occur frequently in soft clauses

Hard clauses:
- \( c_1 = x_1 \lor x_2 \)
- \( c_2 = \neg x_2 \lor x_3 \)
- \( c_3 = \neg x_1 \lor \neg x_3 \)

Soft clauses:
- \( c_4 = \neg x_1 \)
- \( c_5 = \neg x_3 \)
Clause-Variable Incidence Graph (CVIG)

Vertices: Represent each variable and each clause

Edges: There is an edge between each variable and each clause where the variable occurs

Each edge has a corresponding weight:

- More weight is given to clauses that establish edges between variables that occur frequently in soft clauses

Hard clauses:
\[
\begin{align*}
c_1 &= x_1 \lor x_2 \\
c_2 &= \neg x_2 \lor x_3 \\
c_3 &= \neg x_1 \lor \neg x_3
\end{align*}
\]

Soft clauses:
\[
\begin{align*}
c_4 &= \neg x_1 \\
c_5 &= \neg x_3
\end{align*}
\]
(1) Partition the soft clauses

\[ \gamma_1, \gamma_2, \gamma_3 \]
Partitioning in MaxSAT

1. Partition the soft clauses
2. Add a new partition to the formula
Partitioning in MaxSAT

(1) Partition the soft clauses
(2) Add a new partition to the formula
(3) While the formula is unsatisfiable:
   - Relax unsatisfiable core
Partitioning in MaxSAT

1. Partition the soft clauses
2. Add a new partition to the formula
3. While the formula is unsatisfiable:
   - Relax unsatisfiable core
4. The formula is satisfiable:
   - If there are no more partitions:
     ▶ Optimum found
   - Otherwise, go back to 2
Partitioning in MaxSAT

(1) Partition the soft clauses

(2) Add a new partition to the formula

(3) While the formula is unsatisfiable:
   - Relax unsatisfiable core

(4) The formula is satisfiable:
   - If there are no more partitions:
     - Optimum found
   - Otherwise, go back to 2
Partitioning in MaxSAT

(1) Partition the soft clauses
(2) Add a new partition to the formula
(3) While the formula is unsatisfiable:
   - Relax unsatisfiable core
(4) The formula is satisfiable:
   - If there are no more partitions:
     ▶ Optimum found
   - Otherwise, go back to 2
Partitioning in MaxSAT

(1) Partition the soft clauses

(2) Add a new partition to the formula

(3) While the formula is unsatisfiable:
   - Relax unsatisfiable core

(4) The formula is satisfiable:
   - If there are no more partitions:
     - Optimum found
   - Otherwise, go back to 2
Partitioning in MaxSAT

1. Partition the soft clauses
2. Add a new partition to the formula
3. While the formula is unsatisfiable:
   - Relax unsatisfiable core
4. The formula is satisfiable:
   - If there are no more partitions:
     ▶ Optimum found
   - Otherwise, go back to 2
Partitioning in MaxSAT

1. Partition the soft clauses
2. Add a new partition to the formula
3. While the formula is unsatisfiable:
   - Relax unsatisfiable core
4. The formula is satisfiable:
   - If there are no more partitions:
     - Optimum found
   - Otherwise, go back to 2
Partitioning in MaxSAT

(1) Partition the soft clauses

(2) Add a new partition to the formula

(3) While the formula is unsatisfiable:
   - Relax unsatisfiable core

(4) The formula is satisfiable:
   - If there are no more partitions:
     ▶ **Optimum found**
   - Otherwise, go back to 2
Experimental Results (Partial MaxSAT)

- **Benchmarks:**
  - 504 industrial partial MaxSAT instances

- **Solvers:**
  - WBO
  - rdm (Random partitioning – 16 partitions)
  - hyp (Hypergraph partitioning – 16 partitions)
  - VIG (Community partitioning – Variable Incidence Graph)
  - CVIG (Community partitioning – Clause-Variable Incidence Graph)
  - VBS (Virtual Best Solver)
Experimental Results (Partial MaxSAT)

- Running times of solvers for industrial partial MaxSAT instances

![Graph showing running times of solvers for industrial partial MaxSAT instances]
Partitioning in MaxSAT

- Partitioning approaches outperform WBO on most instances:
  - Finds smaller unsatisfiable cores

- All algorithms contribute to the VBS:
  - Different graph-based partition methods solve different instances
  - Using the structure of the formula improves the partitioning

- **Partitioning** may be applied to **other algorithms** and **fields**!
Want to try MaxSAT solving?

- **Java:**
  - SAT4J
    - [http://www.sat4j.org/](http://www.sat4j.org/)

- **C++:**
  - Open-WBO
    - Winner of multiples tracks in the MaxSAT Competition 2014, 2015!
    - [http://sat.inesc-id.pt/open-wbo/](http://sat.inesc-id.pt/open-wbo/)

- **Annual competition:**
  - [http://www.maxsat.udl.cat/](http://www.maxsat.udl.cat/)
  - Modify a solver today and enter this year competition!
MaxSAT is everywhere!

Do you know any optimization problems?

- It is very likely that you can use MaxSAT to solve your problem!
MaxSAT is everywhere!

Do you know any optimization problems?

- It is very likely that you can use MaxSAT to solve your problem!

Example — Development Assurance Level (DAL) Allocation Process:

- ARP4754: "The Development Assurance Level is the measure of rigor applied to the development process to limit, to a level acceptable for safety, the likelihood of Errors occurring during the development process of Functions (at aircraft level or system level) and Items that have an adverse safety effect if they are exposed in service."

(Bieber et al. SAFECOMP 2011)
MaxSAT is everywhere!

Do you know any optimization problems?

- It is very likely that you can use MaxSAT to solve your problem!
- Example – Development Assurance Level (DAL) Allocation Process:

(Bieber et al. SAFECOMP 2011)
MaxSAT algorithms:


Community Structure:

C. Ansótegui, J. Giráldez-Cru, Jordi Levy. The Community Structure of SAT Formulas. SAT 2012: 410-423
References

Cardinality and Pseudo-Boolean Encodings:


Web pages of interest:

MaxSAT Evaluation: http://www.maxsat.udl.cat/
Open-WBO: http://sat.inesc-id.pt/open-wbo/
SAT4J: http://www.sat4j.org/
SATGraf: https://bitbucket.org/znewsham/satgraf

Aerospace application:

P. Bieber, R. Delmas, C. Seguin:
DALculus - Theory and Tool for Development Assurance Level Allocation. SAFECOMP 2011: 43-56