Algorithms for Computational Logic

Overconstrained Problems

Emmanuel Hebrard (adapted from) João Marques Silva

Outline

1. Maximum Satisfiability
2. Modeling Examples
3. Problems with MaxSAT Solving
4. MaxSAT Algorithms with Iterative Search
5. Core-Guided MaxSAT
6. The MaxHS algorithm for MaxSAT
1 Maximum Satisfiability

2 Modeling Examples

3 Problems with MaxSAT Solving

4 MaxSAT Algorithms with Iterative Search

5 Core-Guided MaxSAT
   ● Fu&Malik’s Algorithm
   ● MSU3 Algorithm

6 The MaxHS algorithm for MaxSAT

Unsatisfiable formula

Find largest subset of clauses that is satisfiable: the complement of a *minimum-size correction set*

For above example, MaxSAT solution is 2:
  ▶ By removing 2 clauses, the remaining are satisfiable
MaxSAT problem(s)

<table>
<thead>
<tr>
<th>Hard Clauses?</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights?</td>
<td>No</td>
<td>Plain</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>Weighted</td>
</tr>
</tbody>
</table>

- **Must** satisfy hard clauses, if any
- Compute set of satisfied soft clauses with maximum cost
  - Without weights, cost of each falsified soft clause is 1
- Or, compute set of falsified soft clauses with minimum cost
  (s.t. hard & remaining soft clauses are satisfied)
- **Note**: goal is to compute set of satisfied (or falsified) clauses; not just the cost!

Outline

1. **Maximum Satisfiability**
2. **Modeling Examples**
3. Problems with MaxSAT Solving
4. MaxSAT Algorithms with Iterative Search
5. Core-Guided MaxSAT
   - Fu&Malik’s Algorithm
   - MSU3 Algorithm
6. The MaxHS algorithm for MaxSAT
The problem:

- Graph $G = (V, E)$
- Vertex cover $U \subseteq V$
  - For each $(v_i, v_j) \in E$, either $v_i \in U$ or $v_j \in U$
- Minimum vertex cover: vertex cover $U$ of minimum size

Minimum vertex cover

```
V2
  \\
V3
  \\
V1
  \\
V4
```

Vertex cover: \{v_2, v_3, v_4\}
Min vertex cover: \{v_1\}

Partial MaxSAT formulation:

- Variables: $x_i$ for each $v_i \in V$, with $x_i = 1$ iff $v_i \in U$
- Hard clauses: $(x_i \lor x_j)$ for each $(v_i, v_j) \in E$
- Soft clauses: $(\neg x_i)$ for each $v_i \in V$
  - i.e. preferable not to include vertices in $U$

```
F_H = \{(x_1 \lor x_2), (x_1 \lor x_3), (x_1 \lor x_4)\}
F_S = \{(\neg x_1), (\neg x_2), (\neg x_3), (\neg x_4)\}
```

- Hard clauses have cost $\infty$
- Soft clauses have cost 1
Independent sets and cliques

Given undirected graph $G = (V, E)$:

- A **clique** is a complete subgraph of $G$, i.e. it is a set $L \subseteq V$ such that $\forall u, v \in L (u \neq v) \rightarrow (u, v) \in E$
- A **vertex cover** $C \subseteq V$ is such that $\forall (u, v) \in E u \in C \lor v \in C$
- An **independent set** $I \subseteq V$ is such that $\forall u, v \in I (v, u) \not\in E$

Properties:

- If $I$ is an independent set of $G = (V, E)$, then
  - $V - I$ is a vertex cover of $G$
  - $I$ is a clique of the complement graph of $G$, $G^C$
- A maximum independent set of $G$ corresponds to a maximum clique of $G^C$

**Modeling Examples**

- $G$:

  ![Diagram of graph G]

  - $\{v_1, v_2, v_3\}$ is clique of $G$ and an independent set of $G^C$
  - $\{v_4\}$ is a vertex cover of $G^C$

- $G^C$:

  ![Diagram of graph G^C]

  - $\{v_1, v_2\}$
  - $\{v_4, v_3\}$
Maximum clique with MaxSAT

\[ \mathcal{F}_H \triangleq (\neg x_1 \lor \neg x_4) \land (\neg x_3 \lor \neg x_4) \]
\[ \mathcal{F}_S \triangleq \{(x_1), (x_2), (x_3), (x_4)\} \]

- MaxSAT formulation:
  - \( x_i \): assigned 1 if \( v_i \in V \) included in clique
  - If \( \{x_i, x_j\} \notin E \), add hard clause \((\neg x_i \lor \neg x_j)\)
  - Soft clauses \((x_i)\) for \( v_i \in V \)
  - Why? Add as many vertices as possible to the clique such that non-adjacent vertices are not both selected

### Design debugging

#### Correct circuit

\[ \langle r, s \rangle = \langle 0, 1 \rangle \]
\[ \text{Valid output: } \langle y, z \rangle = \langle 0, 0 \rangle \]

- The model:
  - Hard clauses: Input and output values
  - Soft clauses: CNF representation of circuit, each gate aggregated in group of clauses

#### Faulty circuit

\[ \langle r, s \rangle = \langle 0, 1 \rangle \]
\[ \text{Invalid output: } \langle y, z \rangle = \langle 0, 0 \rangle \]

- The problem:
  - Maximize number of satisfied clauses (i.e. circuit gates)
Software package upgrades with MaxSAT

- Universe of software packages: \( \{ p_1, \ldots, p_n \} \)
- Difference with respect to original installation: \( \{ p_1^\Delta, \ldots, p_n^\Delta \} \)
- Incompatibilities, dependencies and non-regression
  - Hard clauses
  - Objective: minimize \( \sum_{i=1}^n p_i^\Delta \)
  - Soft clauses \( (p_1^\Delta) \land (p_2^\Delta) \land \ldots \land (p_i^\Delta) \)

Many other applications

- Error localization in C code [JM’11]
- Haplotyping with pedigrees [GLMSO’10]
- Course timetabling [AN’10]
- Combinatorial auctions [HLGS’08]
- Minimizing Disclosure of Private Information in Credential-Based Interactions [AVFPS’10]
- Reasoning over Biological Networks [GL’12]
- Binate/unate covering
  - Haplotype inference [GMSLO’11]
  - Digital filter design [ACFM’08]
  - FSM synthesis [e.g. HS’96]
  - Logic minimization [e.g. HS’96]
  - ...
- ...
Problems with MaxSAT Solving

1. Example formula:
   \[ \mathcal{F} \triangleq (x_1) \land (x_2) \land (x_3) \land (\neg x_1 \lor \neg x_2) \land (\neg x_1 \lor \neg x_3) \]

2. Unit propagation falsifies two clauses: \((\neg x_1 \lor \neg x_2)\) and \((\neg x_1 \lor \neg x_3)\)

3. But, the MaxSAT solution is 1; \(S \subseteq \mathcal{F}\) is satisfiable:
   \[ S \triangleq (x_2) \land (x_3) \land (\neg x_1 \lor \neg x_2) \land (\neg x_1 \lor \neg x_3) \]

4. **Cannot** apply unit propagation when solving MaxSAT

5. **Cannot** apply hallmarks of CDCL SAT solving

6. MaxSAT solving requires dedicated algorithms
MaxSAT Algorithms with Iterative Search

Cost of assignment:
- Sum of weights of falsified clauses

Optimum solution (OPT):
- Assignment with minimum cost

Upper Bound (UB):
- Assignment with cost $\geq$ OPT
- E.g. $\sum_{c_j \in F} w_j + 1$; hard clauses may be inconsistent

Lower Bound (LB):
- No assignment with cost $\leq$ LB
- E.g. $-1$; it may be possible to satisfy all soft clauses

Relax each soft clause $c_j$: $(c_j \lor r_j)$ (on-demand in core-guided)
MaxSAT with iterative SAT solving – refine UB

\[ i \leftarrow 0 \\
UB_i \leftarrow \text{ComputeUB} \]

\[ i \leftarrow i + 1 \\
UB_i \leftarrow \text{UpdateUB} \]

\[ G \leftarrow F \cup (\sum w_j r_j < UB_i) \]

\( \text{SAT}(G)? \)

\( \text{return } UB_i \)

- Worst-case \# of iterations \textit{exponential} on instance size (\# bits)

- Improvement: use \textit{binary search} instead

- Many example solvers: Minisat+, SAT4J, QMaxSat

\[ \text{Example CNF formula Relax all clauses; Set } UB = 12 + 1 \text{ Formula is SAT; E.g. all } x_i = 0 \text{ and } r_1 = r_7 = r_9 = 1 \text{ (i.e. cost } = 3) \text{ Refine } UB = 3 \text{ Formula is SAT; E.g. } x_1 = x_2 = 1; \]

\[ x_3 = \ldots = x_8 = 0 \text{ (cost } = 2) \text{ Refine } UB = 2 \text{ All (possibly many) soft clauses relaxed} \]

\[ \sum_{i=1}^{12} r_i \leq 12 \sum_{i=1}^{12} r_i \leq 2 \sum_{i=1}^{12} r_i \leq 1 \]
MaxSAT with iterative SAT solving – binary search

\[ m_0 = \lfloor \frac{LB_0 + UB_0}{2} \rfloor \]

- Invariant: \( LB_k \leq UB_k - 1 \)
- Require \( \sum w_i r_i \leq m_0 \)
- Repeat
  - If UNSAT, refine \( LB_1 = m_0, \ldots \)
  - Compute new mid value \( m_1, \ldots \)
  - If SAT, refine \( UB_3 = m_2, \ldots \)
- Until \( LB_k = UB_k - 1 \)
- Worst-case # of iterations linear on instance size

Branch&bound MaxSAT algorithm

**Input:** \( \text{max-sat}(\phi, UB) \): A CNF formula \( \phi \) and an upper bound \( UB \)

1. \( \phi \leftarrow \text{simplifyFormula}(\phi) \);
2. if \( \phi = \emptyset \) or \( \phi \) only contains empty clauses then
3. return \#emptyClauses(\( \phi \));
4. end if
5. \( LB \leftarrow \#emptyClauses(\phi) + \text{underestimation}(\phi, UB) \);
6. if \( LB \geq UB \) then
7. return \( UB \);
8. end if
9. \( x \leftarrow \text{selectVariable}(\phi) \);
10. \( UB \leftarrow \min(UB, \text{max-sat}(\phi_x, UB)) \);
11. return \( \min(UB, \text{max-sat}(\phi_x, UB)) \);

**Output:** The minimal number of unsatisfied clauses of \( \phi \)

- Many techniques for computing lower bounds, i.e. for lower bounding the search
Goal: Do not relax all clauses

Why?
- Some clauses never relevant for computing MaxSAT solution
- Simplify cardinality/PB constraints

How to relax clauses on demand?
- Relax clauses given computed unsatisfiable cores
  - Many alternative ways to instrument code-guided algorithms
Example CNF formula Formula is UNSAT; \( \text{OPT} \leq |\phi| - 1 \); Get unsat core Add relaxation variables and AtMost1 constraint Formula is (again) UNSAT; \( \text{OPT} \leq |\phi| - 2 \); Get unsat core Add new relaxation variables and AtMost1 constraint Instance is now SAT MaxSAT solution is \( |\phi| - I = 12 - 2 = 10 \) Only AtMost1 constraints used Some clauses not relaxed Relaxed soft clauses remain soft

Another example

\[
\mathcal{F}_S \equiv (x_1) \land (\neg x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2) \land (x_3) \land (\neg x_3) \land (x_4 \lor \neg x_5) \land (\neg x_4 \lor x_5)
\]
Example CNF formula. Formula is **UNSAT**; \( \text{OPT} \leq |\varphi| - 1 \); Get unsat core. Add relaxation variables and AtMost1 constraint. Formula is (again) **UNSAT**; \( \text{OPT} \leq |\varphi| - 2 \); Get unsat core. Add new relaxation variables and AtMost1 constraint. Instance is **SAT**. MaxSAT solution is \( |\varphi| - I = 10 \).

Another example

\[ F_S \triangleq (x_1) \land (\neg x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2) \land (x_3) \land (\neg x_3) \land (x_4 \lor \neg x_5) \land (\neg x_4 \lor x_5) \]
1 Maximum Satisfiability

2 Modeling Examples

3 Problems with MaxSAT Solving

4 MaxSAT Algorithms with Iterative Search

5 Core-Guided MaxSAT
   - Fu&Malik’s Algorithm
   - MSU3 Algorithm

6 The MaxHS algorithm for MaxSAT

Remark 1: The MaxSAT solution is a smallest MCS

Remark 2: Any MCS is a hitting set of all MUSes

Approach:

1. Let \( \mathcal{K} \) be a set of unsatisfiable cores (or MUSes)
2. Find a minimum hitting set \( \mathcal{H} \) of the set \( \mathcal{K} \) of already computed cores (or MUSes)
3. Check satisfiability of \( \mathcal{F} \setminus \mathcal{H} \)
4. If satisfiable, then \( \mathcal{H} \) is a smallest MCS; terminate and return \( \mathcal{H} \)
5. Otherwise, compute core (or MUS) and add it to \( \mathcal{K} \)
6. Loop from 2

Issue: worst-case number of iterations worst-case exponential on number of clauses

   ▶ But, quite effective in practice
MHS approach for MaxSAT – example

\[ c_1 = x_6 \lor x_2 \lor A_1 A_1 \]
\[ c_2 = \neg x_6 \lor x_2 \lor A_2 A_2 \]
\[ c_3 = \neg x_2 \lor x_1 \lor A_3 A_3 \]
\[ c_4 = \neg x_1 \lor A_4 A_4 \]
\[ c_5 = \neg x_6 \lor x_8 \lor A_5 A_5 \]
\[ c_6 = x_6 \lor \neg x_8 \lor A_6 A_6 \]
\[ c_7 = x_2 \lor x_4 \lor A_7 A_7 \]
\[ c_8 = \neg x_4 \lor x_5 \lor A_8 A_8 \]
\[ c_9 = x_7 \lor x_5 \lor A_9 A_9 \]
\[ c_{10} = \neg x_7 \lor x_9 \lor A_{10} A_{10} \]
\[ c_{11} = \neg x_5 \lor x_3 \lor A_{11} A_{11} \]
\[ c_{12} = \neg x_3 \lor A_{12} A_{12} \]

\[ K = \emptyset \]

- To every \( c_i \in F \), add a new literal \( A_i \). Set \( A_i \) to true to relax \( c_i \), or to false to activate it.
- Find MHS of \( K \): \( \emptyset \)
  \[ K = \{ \{ c_1, c_2, c_3, c_4 \}, \{ c_9, c_{10}, c_{11}, c_{12} \} \} \]
- Core of \( F \): \( \{ c_1, c_2, c_3, c_4 \} \). Update \( K \)
  \[ K = \{ \{ c_1, c_2, c_3, c_4 \}, \{ c_9, c_{10}, c_{11}, c_{12} \}, \{ c_3, c_4, c_7, c_8, c_{11}, c_{12} \} \} \]
- Find MHS of \( K \): E.g. \( \{ c_1 \} \)
- Core of \( F \): \( \{ c_9, c_{10}, c_{11}, c_{12} \} \). Update \( K \)
- Find MHS of \( K \): E.g. \( \{ c_1, c_9 \} \)
- Core of \( F \): \( \{ c_3, c_4, c_7, c_8, c_{11}, c_{12} \} \). Update \( K \)
- Find MHS of \( K \): E.g. \( \{ c_4, c_9 \} \)
- Terminate & return 2

Core Extraction Using CDCL

- Assign the activation literals at a special decision level (-1)
- CDCL fails when finding a contradiction at level 0
  - The implication graph must involve some activation literals
- Do clause resolution until the cut contains only activation literals
- The resulting clause is a MUS of the original formula

Level Dec. Unit Prop.

\[ \begin{array}{c}
-1 & A_1 & \neg A_3 & \neg A_7 & A_4 \\
0 & \emptyset & b & d & e & \bot \\
\end{array} \]
**Algorithm:** MAXHS

\[ K \emptyset \]  // The MUSs  
\[ \sigma \leftarrow \emptyset \]  // The optimal model  

while satisfiability \( \neq \) SAT do

\[ hs \leftarrow \text{Find-MinCost-HittingSet}(K) \];

\( (sat, \kappa, \sigma) \leftarrow \text{CDCL}(F \setminus hs) \);

add \( \kappa \) to \( K \);

end

return \( \sigma \);

- **CDCL** returns the tuple \( (sat, \kappa, \sigma) \) where:
  - \( sat \) is in \{SAT, UNSAT, UNKNOWN\}
  - \( \kappa \) is a MUS
  - \( \sigma \) is a solution if \( \models (SAT) = \text{true} \)

---

- **Je recrute un postdoc!**
  - Planification des prises de vue et vidages d'une constellation de satellites d'observation (Projet JAPETUS – PROMETHEE, CNES, CNRS, LEANSPACE)