Lecture 9: Algorithms for MaxSAT.
February 25, 2016

Matti Järvisalo
On This Lecture

Algorithms for exact MaxSAT solving

- Branch-and-bound
- Integer Programming (IP)
- SAT-Based Algorithms
  - Iterative / “model-based”
  - Core-based
- Implicit hitting set algorithms (IP/SAT hybrid).
Algorithms for \textsc{MaxSat} Solving
Examples of Recent MaxSAT Solvers by Category

**Branch-and-bound:**
- MaxSatz  
- ahmaxsat  
  [http://www.lsis.org/habetd/Djamal_Habet/MaxSAT.html](http://www.lsis.org/habetd/Djamal_Habet/MaxSAT.html)

**Iterative, model-based:**
- QMaxSAT  
  [https://sites.google.com/site/qmaxsat/](https://sites.google.com/site/qmaxsat/)

**Core-based:**
- Eva  
  [http://www.maxsat.udl.cat/14/solvers/eva500a__](http://www.maxsat.udl.cat/14/solvers/eva500a__)
- MSCG  
- OpenWBO  
- WPM  
  [http://web.udl.es/usuaris/q4374304/#software](http://web.udl.es/usuaris/q4374304/#software)
- maxino  
  [http://alviano.net/software/maxino/](http://alviano.net/software/maxino/)

**IP-SAT Hybrids:**
- MaxHS  
  [http://maxhs.org](http://maxhs.org)
- LMHS  
Branch and Bound
Branch and Bound

- $UB = \text{cost of the best solution so far.}$
- $\text{mincost}(n) = \text{minimum cost achievable under node } n$
- Backtrack when $\text{mincost}(n) \geq UB$
  (no solution under $n$ can improve $UB$).
- Goal: compute a lower bound $LB$ s.t.
  $\text{mincost}(n) \geq LB$.
- When $LB \geq UB$:
  $\text{mincost}(n) \geq LB \geq UB$
  $\Rightarrow$ backtrack.
Lower Bounds

Common LB technique in MaxSAT solvers:
Look for inconsistencies that force some soft clause to be falsified.
Lower Bounds

Common LB technique in MAXSAT solvers:

Look for inconsistencies that force some soft clause to be falsified.

\[ F = \ldots \land (x, 2) \ldots \land (\neg x, 3) \ldots \]

Ignoring clause costs, \( \kappa = \{(x) \land (\neg x)\} \) is inconsistent.
Lower Bounds

**Common LB technique in MaxSAT solvers:**

Look for inconsistencies that force some soft clause to be falsified.

\[ F = \ldots \land (x, 2) \ldots \land (\neg x, 3) \ldots \]

Ignoring clause costs, \( \kappa = \{(x) \land (\neg x)\} \) is inconsistent.

Let \( \kappa' = \{(\emptyset, 2) \land (\neg x, 1)\} \).

- Then \( \kappa' \) is MaxSAT-equivalent to \( \kappa \):
  - the cost of each truth assignment is preserved.
Lower Bounds

Common LB technique in MaxSAT solvers:
Look for inconsistencies that force some soft clause to be falsified.

\[ F = \ldots \land (x, 2) \ldots \land (\neg x, 3) \ldots \]

Ignoring clause costs, \( \kappa = \{(x) \land (\neg x)\} \) is inconsistent.

Let \( \kappa' = \{((\emptyset, 2) \land (\neg x, 1))\} \).

- Then \( \kappa' \) is MaxSAT-equivalent to \( \kappa \):
  the cost of each truth assignment is preserved.

Let \( F' = F \setminus (\kappa \cup \kappa') \).

Then \( F' \) is MaxSAT-equivalent to \( F \), and the cost of \( \emptyset \) has been incremented by 2.

Cost of \( (\emptyset, 2) \) must be incurred: 2 is a LB
Lower Bounds

1. Detect an inconsistent subset $\kappa$ (aka core) of the current formula
   - e.g. $\kappa = \{(x, 2) \land (\neg x, 3)\}$

2. Apply sound transformation to the clauses in $\kappa$ that result in an increment to the cost of the empty clause $\emptyset$
   - e.g. $\kappa$ replaced by $\kappa' = \{ (\emptyset, 2) \land (\neg x, 1) \}$
   - This replacement increases cost of $\emptyset$ by 2.

3. Repeat 1 and 2 until no LB cannot be incremented (or $LB \geq UB$)
Fast detection of some cores

Treat the soft clauses as if they were hard and then:

- Run **Unit Propagation** (UP).
  - If UP falsifies a clause we can find a core.
  - On \(\{(x, 2), (\neg x, 3)\}\) UP falsified a clause.
- The falsified clause and the clauses that generated it form a core.
- This can find inconsistent sub-formulas quickly.
  - But only limited set of inconsistent sub-formulas.
Transforming the Formula

- Various sound transformations of cores into increments of the empty clause have been identified.
- **MaxRes** generalizes this to provide a sound and complete inference rule for **MaxSAT**

  [Larrosa and Heras, 2005]
  [Bonet, Levy, and Manyà, 2007]

Other Lower Bounding Techniques

- Falsified soft learnt clauses and hitting sets over their proofs
  [Davies, Cho, and Bacchus, 2010]

- Minibuckets, width-restricted BDDs
  [Dechter and Rish, 2003]
  [Bergman, Ciré, van Hoeve, and Yunes, 2014]
Branch-and-Bound: Summary

- Can be effective on small combinatorially hard problems, e.g., maxclique in a graph.
- Once the number of variables gets to 1,000 or more it is less effective: LB techniques become weak or too expensive.
SAT-Based MAXSAT Solving
SAT-Based MaxSAT Solving

- Solve a sequence or SAT instances where each instance encodes a decision problem of the form

  "Is there a truth assignment of falsifying at most weight $k$ soft clauses?"

  for different values of $k$.

- SAT-based MaxSAT algorithms mainly do two things:
  1. Develop better ways to encode this decision problem.
  2. Find ways to exploit information obtained from the SAT solver at each stage in the next stage.

Assume unit weight soft clauses for now
SAT-Based MaxSAT Solving

- Iterative search methods
- Improving by using cores
- Improving by using cores and new variables
Iterative Search

Basic approach:

- To check whether \( F \) has a solution of cost \( \leq k \):
  
  - SAT solve \((C_1 \lor r_1) \land (C_2 \lor r_2) \land \cdots \land (C_n \lor r_n) \land (\sum_{i=1}^{n} r_i \leq k)\)

- Iterate over \( k \in \{1, \ldots, n\} \) to find the optimal \( k \)
  
  - …and an optimal solution.
  
  - …proving that no solutions of cost \(< k\) exist.
Iterative Search

Basic approach:

- **To check whether $F$ has a solution of cost $\leq k$:**
  - SAT solve $(C_1 \lor r_1) \land (C_2 \lor r_2) \land \cdots \land (C_n \lor r_n) \land (\sum_{i=1}^{n} r_i \leq k)$

- **Iterate** over $k \in \{1, \ldots, n\}$ to find the optimal $k$
  - ...and an optimal solution.
  - ...proving that no solutions of cost $< k$ exist.
Iterating over $k$

- Different ways of iterating over values of $k$.
- Three “standard” approaches:

1. Linear search (not effective)
   - Start from $k = 1$.
   - Increment $k$ by 1 until a solution is found.

2. Binary search (effective with core-based reasoning)
   - $UB = \# \text{ of soft clauses}; LB = 0$.
   - Solve with $k = (UB + LB)/2$.
   - If SAT: $UB = k$; if UNSAT: $LB = k$.
   - When $UB = LB + 1$, $UB$ is solution.
Iterating over $k$

3. **SAT to UNSAT**
   1. Find a satisfying assignment $\pi$ of the hard clauses.
   2. Solve with $k = (\# \text{ of clauses falsified by } \pi) - 1$
   3. If SAT found better assignment. Reset $k$ and repeat 2.
   4. If UNSAT last assignment $\pi$ found is optimal.

- Finds a sequence of improved solutions
- Used in QMaxSAT, can be effective on certain problems
SAT-based $\text{MaxSAT}$ Solving using Cores
Core-Based MaxSAT Solving

Motivation

- In the linear approach:
  add $CNF(\sum r_i \leq k)$ to the SAT solver.
  - One $r_i$ per each soft clause.
  - The cardinality constraint could be over 100,000s of variables
    ... and is very loose:
    No information about which particular relaxation variables to make true.

- This makes SAT solving inefficient:
  could have to explore many choices of subsets of $k$ soft clauses to remove.

However:

*Obtaining a core gives a more powerful constraints over which particular soft clauses to relax.*
Constraints from Cores

- If $\kappa$ is a MaxSAT core, then at least one of the soft clauses in it must be removed
  - No truth assignment satisfies every clause in $\kappa$ along with all of the hard clauses.

- Typically cores are much smaller than the set of all soft clauses.
Core-Guided MaxSAT Algorithms: Fu-Malik

The first core-guided MaxSAT algorithm

**Fu-Malik Algorithm**

Iteratively:
- Find an UNSAT core using a SAT solver
- Add relaxation variables to clauses in the core
- Add an AtMost-1 constraint over the new relaxation variables
  - Soft clauses remain soft after relaxing them

...until the SAT solver reports *satisfiable*.

**Key observation**

Each iteration *lowers the cost of solutions by 1* (on an unweighted formula)
Core-Guided MaxSAT Algorithms: Fu-Malik

The first core-guided MaxSAT algorithm

Fu-Malik Algorithm

Iteratively:

- Find an UNSAT core using a SAT solver
- Add relaxation variables to clauses in the core
- Add an AtMost-1 constraint over the new relaxation variables
  - Soft clauses remain soft after relaxing them

...until the SAT solver reports *satisfiable*.

Key observation

Each iteration *lowers the cost of solutions by 1* (on an unweighted formula)
Fu-Malik: Example

(On an unweighted formula)

\[ C_1 = x_6 \lor x_2 \]
\[ C_4 = \neg x_1 \]
\[ C_7 = x_2 \lor x_4 \]
\[ C_{10} = \neg x_7 \lor x_5 \]
\[ C_2 = \neg x_6 \lor x_2 \]
\[ C_5 = \neg x_6 \lor x_8 \]
\[ C_{11} = \neg x_5 \lor x_3 \]
\[ C_3 = \neg x_2 \lor x_1 \]
\[ C_8 = \neg x_4 \lor x_5 \]
\[ C_9 = x_7 \lor x_5 \]
\[ C_{12} = \neg x_3 \]

1. UNSAT core: \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\}
2. Relax the clauses in the core with variables \(r_1, \ldots, r_6\)
3. Add \(\sum_{i=1}^{6} r_i \leq 1\)
4. UNSAT core: \{C_1, C_2, C_3, C_4, C_9, C_{10}, C_{11}, C_{12}\}
5. Relax the clauses in the core with variables \(r_7, \ldots, r_{14}\)
6. Add \(\sum_{i=7}^{14} r_i \leq 1\)
7. Satisfiable, terminate.
Optimal cost: 2 (the number of iterations)
Fu-Malik: Example

(On an unweighted formula)
\[
\begin{align*}
C_1 &= x_6 \lor x_2 \\
C_4 &= \neg x_1 \\
C_7 &= x_2 \lor x_4 \\
C_{10} &= \neg x_7 \lor x_5 \\
C_2 &= \neg x_6 \lor x_2 \\
C_5 &= \neg x_6 \lor x_8 \\
C_8 &= \neg x_4 \lor x_5 \\
C_{11} &= \neg x_5 \lor x_3 \\
C_3 &= \neg x_2 \lor x_1 \\
C_6 &= x_6 \lor \neg x_8 \\
C_9 &= x_7 \lor x_5 \\
C_{12} &= \neg x_3
\end{align*}
\]

1. UNSAT core: \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\}
2. Relax the clauses in the core with variables \(r_1, \ldots, r_6\)
3. Add \(\sum_{i=1}^{6} r_i \leq 1\)
4. UNSAT core: \{C_1, C_2, C_3, C_4, C_9, C_{10}, C_{11}, C_{12}\}
5. Relax the clauses in the core with variables \(r_7, \ldots, r_{14}\)
6. Add \(\sum_{i=7}^{14} r_i \leq 1\)
7. Satisfiable, terminate.

Optimal cost: 2 (the number of iterations)
Fu-Malik: Example

(On an unweighted formula)

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \lor r_1 \]
\[ C_4 = \neg x_1 \lor r_2 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \lor r_3 \quad C_8 = \neg x_4 \lor x_5 \lor r_4 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \lor r_5 \quad C_{12} = \neg x_3 \lor r_6 \]

1. **UNSAT core:** \{ \( C_3, C_4, C_7, C_8, C_{11}, C_{12} \) \}
2. **Relax the clauses in the core with variables** \( r_1, \ldots, r_6 \)
3. **Add** \( \sum_{i=1}^{6} r_i \leq 1 \)
4. **UNSAT core:** \{ \( C_1, C_2, C_3, C_4, C_9, C_{10}, C_{11}, C_{12} \) \}
5. **Relax the clauses in the core with variables** \( r_7, \ldots, r_{14} \)
6. **Add** \( \sum_{i=7}^{14} r_i \leq 1 \)
7. **Satisfiable, terminate.**
   **Optimal cost:** 2 (the number of iterations)
Fu-Malik: Example

(On an unweighted formula)

\[
\begin{align*}
C_1 &= x_6 \lor x_2 \\
C_4 &= \neg x_1 \lor r_2 \\
C_7 &= x_2 \lor x_4 \lor r_3 \\
C_{10} &= \neg x_7 \lor x_5 \\
C_2 &= \neg x_6 \lor x_2 \\
C_5 &= \neg x_6 \lor x_8 \\
C_8 &= \neg x_4 \lor x_5 \lor r_4 \\
C_{11} &= \neg x_5 \lor x_3 \lor r_5 \\
C_3 &= \neg x_2 \lor x_1 \lor r_1 \\
C_6 &= x_6 \lor \neg x_8 \\
C_9 &= x_7 \lor x_5 \\
C_{12} &= \neg x_3 \lor r_6
\end{align*}
\]

1. UNSAT core: \(\{ C_3, C_4, C_7, C_8, C_{11}, C_{12} \}\)

2. Relax the clauses in the core with variables \(r_1, \ldots, r_6\)

3. Add \(\sum_{i=1}^{6} r_i \leq 1\)

4. UNSAT core: \(\{ C_1, C_2, C_3, C_4, C_9, C_{10}, C_{11}, C_{12} \}\)

5. Relax the clauses in the core with variables \(r_7, \ldots, r_{14}\)

6. Add \(\sum_{i=7}^{14} r_i \leq 1\)

7. Satisfiable, terminate.
   Optimal cost: 2 (the number of iterations)
Fu-Malik: Example

(On an unweighted formula)

\[ \begin{align*}
C_1 &= x_6 \lor x_2 \\
C_2 &= \neg x_6 \lor x_2 \\
C_3 &= \neg x_2 \lor x_1 \lor r_1 \\
C_4 &= \neg x_1 \lor r_2 \\
C_5 &= \neg x_6 \lor x_8 \\
C_6 &= x_6 \lor \neg x_8 \\
C_7 &= x_2 \lor x_4 \lor r_3 \\
C_8 &= \neg x_4 \lor x_5 \lor r_4 \\
C_9 &= x_7 \lor x_5 \\
C_{10} &= \neg x_7 \lor x_5 \\
C_{11} &= \neg x_5 \lor x_3 \lor r_5 \\
C_{12} &= \neg x_3 \lor r_6
\end{align*} \]

1. UNSAT core: \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\}
2. Relax the clauses in the core with variables \( r_1, \ldots, r_6 \)
3. Add \( \sum_{i=1}^{6} r_i \leq 1 \)
4. UNSAT core: \{C_1, C_2, C_3, C_4, C_9, C_{10}, C_{11}, C_{12}\}
5. Relax the clauses in the core with variables \( r_7, \ldots, r_{14} \)
6. Add \( \sum_{i=7}^{14} r_i \leq 1 \)
7. Satisfiable, terminate.

Optimal cost: 2 (the number of iterations)
Fu-Malik: Example

(On an unweighted formula)

\[ C_1 = x_6 \lor x_2 \lor r_7 \]
\[ C_4 = \neg x_1 \lor r_2 \lor r_{10} \]
\[ C_7 = x_2 \lor x_4 \lor r_3 \]
\[ C_{10} = \neg x_7 \lor x_5 \lor r_{12} \]
\[ C_2 = \neg x_6 \lor x_2 \lor r_8 \]
\[ C_5 = \neg x_6 \lor x_8 \]
\[ C_8 = \neg x_4 \lor x_5 \lor r_4 \]
\[ C_{11} = \neg x_5 \lor x_3 \lor r_5 \lor r_{13} \]
\[ C_3 = \neg x_2 \lor x_1 \lor r_1 \lor r_9 \]
\[ C_6 = x_6 \lor \neg x_8 \]
\[ C_9 = x_7 \lor x_5 \lor r_{11} \]
\[ C_{12} = \neg x_3 \lor r_6 \lor r_{14} \]

1. UNSAT core: \( \{ C_3, C_4, C_7, C_8, C_{11}, C_{12} \} \)
2. Relax the clauses in the core with variables \( r_1, \ldots, r_6 \)
3. Add \( \sum_{i=1}^{6} r_i \leq 1 \)
4. UNSAT core: \( \{ C_1, C_2, C_3, C_4, C_9, C_{10}, C_{11}, C_{12} \} \)
5. Relax the clauses in the core with variables \( r_7, \ldots, r_{14} \)
6. Add \( \sum_{i=7}^{14} r_i \leq 1 \)
7. Satisfiable, terminate.

Optimal cost: 2 (the number of iterations)
Fu-Malik: Example

(On an unweighted formula)

\[
\begin{align*}
    C_1 &= x_6 \lor x_2 \lor r_7 \\
    C_2 &= \neg x_6 \lor x_2 \lor r_8 \\
    C_3 &= \neg x_2 \lor x_1 \lor r_1 \lor r_9 \\
    C_4 &= \neg x_1 \lor r_2 \lor r_{10} \\
    C_5 &= \neg x_6 \lor x_8 \\
    C_6 &= x_6 \lor \neg x_8 \\
    C_7 &= x_2 \lor x_4 \lor r_3 \\
    C_8 &= \neg x_4 \lor x_5 \lor r_4 \\
    C_9 &= x_7 \lor x_5 \lor r_{11} \\
    C_{10} &= \neg x_7 \lor x_5 \lor r_{12} \\
    C_{11} &= \neg x_5 \lor x_3 \lor r_5 \lor r_{13} \\
    C_{12} &= \neg x_3 \lor r_6 \lor r_{14}
\end{align*}
\]

1. UNSAT core: \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\}

2. Relax the clauses in the core with variables \(r_1, \ldots, r_6\)

3. Add \(\sum_{i=1}^{6} r_i \leq 1\)

4. UNSAT core: \{C_1, C_2, C_3, C_4, C_9, C_{10}, C_{11}, C_{12}\}

5. Relax the clauses in the core with variables \(r_7, \ldots, r_{14}\)

6. Add \(\sum_{i=7}^{14} r_i \leq 1\)

7. Satisfiable, terminate.

Optimal cost: 2 (the number of iterations)
Fu-Malik: Example

(On an unweighted formula)

\[ C_1 = x_6 \lor x_2 \lor r_7 \]
\[ C_4 = \neg x_1 \lor r_2 \lor r_{10} \]
\[ C_7 = x_2 \lor x_4 \lor r_3 \]
\[ C_{10} = \neg x_7 \lor x_5 \lor r_{12} \]
\[ C_2 = \neg x_6 \lor x_2 \lor r_8 \]
\[ C_5 = \neg x_6 \lor x_8 \]
\[ C_8 = \neg x_4 \lor x_5 \lor r_4 \]
\[ C_{11} = \neg x_5 \lor x_3 \lor r_5 \lor r_{13} \]
\[ C_3 = \neg x_2 \lor x_1 \lor r_1 \lor r_9 \]
\[ C_6 = x_6 \lor \neg x_8 \]
\[ C_9 = x_7 \lor x_5 \lor r_{11} \]
\[ C_{12} = \neg x_3 \lor r_6 \lor r_{14} \]

1. UNSAT core: \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\}
2. Relax the clauses in the core with variables \( r_1, \ldots, r_6 \)
3. Add \( \sum_{i=1}^{6} r_i \leq 1 \)
4. UNSAT core: \{C_1, C_2, C_3, C_4, C_9, C_{10}, C_{11}, C_{12}\}
5. Relax the clauses in the core with variables \( r_7, \ldots, r_{14} \)
6. Add \( \sum_{i=7}^{14} r_i \leq 1 \)
7. Satisfiable, terminate.

Optimal cost: 2 (the number of iterations)
Input: MaxSAT instance $F$

1. $k = 0$
2. $(\kappa, SAT?) = SAT(F)$
3. If $SAT?$ return $k$.
4. $k = k + 1$
5. Update $F$:
   a. $R = \emptyset$
   b. For $c \in \kappa$
      Let $r_c$ be a new relaxation variable
      $c = c \cup \{r_c\}$
      $R = R \cup \{r_c\}$
   c. Add $CNF(\sum_{r \in R} r \leq 1)$
6. GOTO 3

- Don’t care if $c$ already has relaxation variable, add new one.
- The cardinality constraint is over new core only and is always $\leq 1$.
- A soft clause can accumulate multiple relaxation variables (one for every core it appears in).
Core-Guided MaxSAT Algorithms: MSU3

Similarities with Fu-Malik:
- Relax clauses on-demand
- Add AtMost constraints over the relaxation variables of clauses in found cores

Differences:
- Re-use already introduced relaxation variables
  ▶ Introduce only at most one relaxation variable to each soft clause
- Add a single AtMost-\(k\) constraint
  ▶ Instead of adding one AtMost-1/Exactly-1 constraint per iteration
  ▶ Update the AtMost-\(k\), \(k\) noting the \(k\)th iteration
- Relaxed soft clauses become hard
Core-Guided MaxSAT Algorithms: MSU3

(On an unweighted formula)

\[ C_1 = x_6 \lor x_2 \]
\[ C_4 = \neg x_1 \]
\[ C_7 = x_2 \lor x_4 \]
\[ C_{10} = \neg x_7 \lor x_5 \]
\[ C_2 = \neg x_6 \lor x_2 \]
\[ C_5 = \neg x_6 \lor x_8 \]
\[ C_{11} = \neg x_5 \lor x_3 \]
\[ C_3 = \neg x_2 \lor x_1 \]
\[ C_6 = x_6 \lor \neg x_8 \]
\[ C_{12} = \neg x_3 \]

1. UNSAT core: \( \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\} \)
2. Relax the clauses in the core with variables \( r_1, \ldots, r_6 \)
3. Add \( \sum_{i=1}^{6} r_i \leq 1 \) AtMost-\( k \) where \( k = 1 \)
4. UNSAT core: \( \{C_1, C_2, , C_9, C_{10}\} \)
5. Relax the clauses in the core with variables \( r_7, \ldots, r_{10} \)
6. Update the AtMost-1 to: \( \sum_{i=1}^{10} r_i \leq 2 \) AtMost-\( k \) where \( k = 2 \)
7. Satisfiable, terminate.

Optimal cost: 2 (the number of iterations)
Core-Guided MaxSAT Algorithms: MSU3

(On an unweighted formula)

\[
\begin{align*}
C_1 &= x_6 \lor x_2 \\
C_4 &= \neg x_1 \\
C_7 &= x_2 \lor x_4 \\
C_{10} &= \neg x_7 \lor x_5 \\
C_2 &= \neg x_6 \lor x_2 \\
C_5 &= \neg x_6 \lor x_8 \\
C_8 &= \neg x_4 \lor x_5 \\
C_{11} &= \neg x_5 \lor x_3 \\
C_3 &= \neg x_2 \lor x_1 \\
C_6 &= x_6 \lor \neg x_8 \\
C_9 &= x_7 \lor x_5 \\
C_{12} &= \neg x_3
\end{align*}
\]

1. **UNSAT core:** \{\(C_3, C_4, C_7, C_8, C_{11}, C_{12}\}\)
2. Relax the clauses in the core with variables \(r_1, \ldots, r_6\)
3. Add \(\sum_{i=1}^{6} r_i \leq 1\) \(\text{AtMost-}k\) where \(k = 1\)
4. **UNSAT core:** \{\(C_1, C_2, C_9, C_{10}\}\)
5. Relax the clauses in the core with variables \(r_7, \ldots, r_{10}\)
6. Update the AtMost-1 to: \(\sum_{i=1}^{10} r_i \leq 2\) \(\text{AtMost-}k\) where \(k = 2\)
7. Satisfiable, terminate.
   Optimal cost: 2 (the number of iterations)
Core-Guided MaxSAT Algorithms: MSU3

(On an unweighted formula)

\[
\begin{align*}
C_1 &= x_6 \lor x_2 \\
C_4 &= \neg x_1 \lor r_2 \\
C_7 &= x_2 \lor x_4 \lor r_3 \\
C_{10} &= \neg x_7 \lor x_5 \\
C_2 &= \neg x_6 \lor x_2 \\
C_5 &= \neg x_6 \lor x_8 \\
C_8 &= \neg x_4 \lor x_5 \lor r_4 \\
C_{11} &= \neg x_5 \lor x_3 \lor r_5 \\
C_3 &= \neg x_2 \lor x_1 \lor r_1 \\
C_6 &= x_6 \lor \neg x_8 \\
C_9 &= x_7 \lor x_5 \\
C_{12} &= \neg x_3 \lor r_6
\end{align*}
\]

1. UNSAT core: \( \{ C_3, C_4, C_7, C_8, C_{11}, C_{12} \} \)
2. Relax the clauses in the core with variables \( r_1, \ldots, r_6 \)
3. Add \( \sum_{i=1}^{6} r_i \leq 1 \) \quad AtMost-\( k \) where \( k = 1 \)
4. UNSAT core: \( \{ C_1, C_2, C_9, C_{10} \} \)
5. Relax the clauses in the core with variables \( r_7, \ldots, r_{10} \)
6. Update the AtMost-1 to: \( \sum_{i=1}^{10} r_i \leq 2 \) \quad AtMost-\( k \) where \( k = 2 \)
7. Satisfiable, terminate.
Optimal cost: 2 (the number of iterations)
Core-Guided MaxSAT Algorithms: MSU3

(On an unweighted formula)

\[ C_1 = x_6 \lor x_2 \]
\[ C_4 = \neg x_1 \lor r_2 \]
\[ C_7 = x_2 \lor x_4 \lor r_3 \]
\[ C_{10} = \neg x_7 \lor x_5 \]
\[ C_2 = \neg x_6 \lor x_2 \]
\[ C_5 = \neg x_6 \lor x_8 \]
\[ C_8 = \neg x_4 \lor x_5 \lor r_4 \]
\[ C_{11} = \neg x_5 \lor x_3 \lor r_5 \]
\[ C_3 = \neg x_2 \lor x_1 \lor r_1 \]
\[ C_6 = x_6 \lor \neg x_8 \]
\[ C_9 = x_7 \lor x_5 \]
\[ C_{12} = \neg x_3 \lor r_6 \]

1. UNSAT core: \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\}
2. Relax the clauses in the core with variables \(r_1, \ldots, r_6\)
3. Add \(\sum_{i=1}^{6} r_i \leq 1\) \quad \text{AtMost-}k \text{ where } k = 1
4. UNSAT core: \{C_1, C_2, , C_9, C_{10}\}
5. Relax the clauses in the core with variables \(r_7, \ldots, r_{10}\)
6. Update the AtMost-1 to: \(\sum_{i=1}^{10} r_i \leq 2\) \quad \text{AtMost-}k \text{ where } k = 2
7. Satisfiable, terminate.
   Optimal cost: 2 (the number of iterations)
Core-Guided MaxSAT Algorithms: MSU3

(On an unweighted formula)

\[ \begin{align*}
C_1 &= x_6 \lor x_2 \\
C_4 &= \neg x_1 \lor r_2 \\
C_7 &= x_2 \lor x_4 \lor r_3 \\
C_{10} &= \neg x_7 \lor x_5 \\
C_2 &= \neg x_6 \lor x_2 \\
C_5 &= \neg x_6 \lor x_8 \\
C_8 &= \neg x_4 \lor x_5 \lor r_4 \\
C_{11} &= \neg x_5 \lor x_3 \lor r_5 \\
C_3 &= \neg x_2 \lor x_1 \lor r_1 \\
C_6 &= x_6 \lor \neg x_8 \\
C_9 &= x_7 \lor x_5 \\
C_{12} &= \neg x_3 \lor r_6 
\end{align*} \]

1. UNSAT core: \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\}
2. Relax the clauses in the core with variables \( r_1, \ldots, r_6 \)
3. Add \( \sum_{i=1}^{6} r_i \leq 1 \) \hspace{1cm} \text{AtMost-}k \text{ where } k = 1
4. UNSAT core: \{C_1, C_2, C_9, C_{10}\}
5. Relax the clauses in the core with variables \( r_7, \ldots, r_{10} \)
6. Update the AtMost-1 to: \( \sum_{i=1}^{10} r_i \leq 2 \) \hspace{1cm} \text{AtMost-}k \text{ where } k = 2
7. Satisfiable, terminate.
   Optimal cost: 2 (the number of iterations)
Core-Guided MaxSAT Algorithms: MSU3

(On an unweighted formula)

\[
\begin{align*}
C_1 &= x_6 \lor x_2 \lor r_7 \\
C_4 &= \neg x_1 \lor r_2 \\
C_7 &= x_2 \lor x_4 \lor r_3 \\
C_{10} &= \neg x_7 \lor x_5 \lor r_{12} \\
C_2 &= \neg x_6 \lor x_2 \lor r_8 \\
C_5 &= \neg x_6 \lor x_8 \\
C_8 &= \neg x_4 \lor x_5 \lor r_4 \\
C_{11} &= \neg x_5 \lor x_3 \lor r_5 \\
C_3 &= \neg x_2 \lor x_1 \lor r_1 \\
C_6 &= x_6 \lor \neg x_8 \\
C_9 &= x_7 \lor x_5 \lor r_{11} \\
C_{12} &= \neg x_3 \lor r_6
\end{align*}
\]

1. UNSAT core: \( \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\} \)
2. Relax the clauses in the core with variables \( r_1, \ldots, r_6 \)
3. Add \( \sum_{i=1}^{6} r_i \leq 1 \)  
   AtMost-\( k \) where \( k = 1 \)
4. UNSAT core: \( \{C_1, C_2, \ldots, C_9, C_{10}\} \)
5. Relax the clauses in the core with variables \( r_7, \ldots, r_{10} \)
6. Update the AtMost-1 to: \( \sum_{i=1}^{10} r_i \leq 2 \)  
   AtMost-\( k \) where \( k = 2 \)
7. Satisfiable, terminate.
   Optimal cost: 2 (the number of iterations)
Core-Guided MaxSAT Algorithms: MSU3

(On an unweighted formula)

\[ \begin{align*}
C_1 &= x_6 \lor x_2 \lor r_7 \\
C_4 &= \neg x_1 \lor r_2 \\
C_7 &= x_2 \lor x_4 \lor r_3 \\
C_{10} &= \neg x_7 \lor x_5 \lor r_{12}
\end{align*} \]

\[ \begin{align*}
C_2 &= \neg x_6 \lor x_2 \lor r_8 \\
C_5 &= \neg x_6 \lor x_8 \\
C_8 &= \neg x_4 \lor x_5 \lor r_4 \\
C_{11} &= \neg x_5 \lor x_3 \lor r_5
\end{align*} \]

\[ \begin{align*}
C_3 &= \neg x_2 \lor x_1 \lor r_1 \\
C_6 &= x_6 \lor \neg x_8 \\
C_9 &= x_7 \lor x_5 \lor r_{11} \\
C_{12} &= \neg x_3 \lor r_6
\end{align*} \]

\begin{enumerate}
\item UNSAT core: \{ \( C_3, C_4, C_7, C_8, C_{11}, C_{12} \) \}
\item Relax the clauses in the core with variables \( r_1, \ldots, r_6 \)
\item Add \( \sum_{i=1}^{6} r_i \leq 1 \) AtMost-\( k \) where \( k = 1 \)
\item UNSAT core: \{ \( C_1, C_2, C_9, C_{10} \) \}
\item Relax the clauses in the core with variables \( r_7, \ldots, r_{10} \)
\item Update the AtMost-1 to: \( \sum_{i=1}^{10} r_i \leq 2 \) AtMost-\( k \) where \( k = 2 \)
\item Satisfiable, terminate.
\end{enumerate}

Optimal cost: 2 (the number of iterations)
Core-Guided MaxSAT Algorithms: MSU3

(On an unweighted formula)

\[ C_1 = x_6 \lor x_2 \lor r_7 \]
\[ C_4 = \neg x_1 \lor r_2 \]
\[ C_7 = x_2 \lor x_4 \lor r_3 \]
\[ C_{10} = \neg x_7 \lor x_5 \lor r_{12} \]
\[ C_2 = \neg x_6 \lor x_2 \lor r_8 \]
\[ C_5 = \neg x_6 \lor x_8 \]
\[ C_8 = \neg x_4 \lor x_5 \lor r_4 \]
\[ C_{11} = \neg x_5 \lor x_3 \lor r_5 \]
\[ C_3 = \neg x_2 \lor x_1 \lor r_1 \]
\[ C_6 = x_6 \lor \neg x_8 \]
\[ C_9 = x_7 \lor x_5 \lor r_{11} \]
\[ C_{12} = \neg x_3 \lor r_6 \]

1. UNSAT core: \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\}
2. Relax the clauses in the core with variables \( r_1, \ldots, r_6 \)
3. Add \( \sum_{i=1}^{6} r_i \leq 1 \) \hspace{1cm} AtMost-\( k \) where \( k = 1 \)
4. UNSAT core: \{C_1, C_2, , C_9, C_{10}\}
5. Relax the clauses in the core with variables \( r_7, \ldots, r_{10} \)
6. Update the AtMost-1 to: \( \sum_{i=1}^{10} r_i \leq 2 \) \hspace{1cm} AtMost-\( k \) where \( k = 2 \)
7. Satisfiable, terminate.
   Optimal cost: 2 (the number of iterations)
Input: \textbf{MaxSAT} instance $F$

1. $k = 0; \ R = \emptyset$.
2. $(\kappa, \text{SAT}? \,) = \text{SAT}(F)$
3. If \text{SAT}? \, return \, k.
4. $k = k + 1$
5. Update $F$:
   a. For $C \in \kappa$ if $C$ has no relaxation variable
      \[ C = C \cup \{ r_C \} \]  
         (new relaxation variable)
      \[ R = R \cup \{ r_C \} \]
   b. Remove previous cardinality constraint.
   c. Add $\text{CNF}(\sum_{r \in R} r \leq k)$
6. GOTO 3

- Initially NO relaxation variables!
- The cardinality constraint is always only over soft clauses that have participated in some core.
- The relaxation variables in the cardinality constraint grows as more cores are discovered.
- On many problems however the cardinality constraint remains over a proper subset of the soft clauses.
Core-Based Algorithms that Modify the Instance
State-of-the-Art Core-Based MaxSAT

- Recent advances in SAT-Based MaxSAT solving comes from approaches that **add new variables to the formula**.
- New variables always been used encoding the cardinality constraint
  - but no attention was paid to the structure of these variables.
- Current best SAT-Based approaches EVA, MSCG-OLL, OpenWBO, WPM3, MAXINO use cores and add new variables.
  - EVA, MSCG-OLL and WPM3 explicitly add new variables.
  - OpenWBO and MAXINO more carefully structure the new variables in the cardinality constraints.

Central Research Question

Achieve a better understand of the impact of these new variables on the SAT solving process
Dealing with Weighted Soft Clauses

How to deal with soft clauses with different weights?
Clause Cloning

Methor used to deal with varying weights. [Ansótegui, Bonet, and Levy, 2009; Manquinho, Silva, and Planes, 2009]

1. \( K \) is new core.
2. \( w_{\text{min}} \) is minimum weight in \( K \).
3. Split each clause \((c, w) \in K\) into two clauses:
   (1) \((c, w_{\text{min}})\) and (2) \((c, w - w_{\text{min}})\).
4. Keep all clauses (2) \((c, w - w_{\text{min}})\) as soft clauses (discard zero weight clauses)
5. Let \( K \) be all clauses (1) \((c, w_{\text{min}})\)
6. Process \( K \) as a new core (all clauses in \( K \) have the same weight)
SAT-Based MaxSAT: Summary

- Effective on large MaxSAT instance
  - Especially when there are many hard clauses

- Central innovations:
  efficient ways to encode and solve the individual SAT decision problems that have to be solved.
  - Some work done on understand the core structure and its impact on SAT solving efficiency but more needed.

[Bacchus and Narodytska, 2014]
MaxSAT by Integer Programming (IP)
Solving MaxSAT with an IP Solver

- Optimization problems studied for decades operations research (OR).
- IP solvers are most optimization tool in OR.
  - IBM CPLEX, Gurobi, SCIP, …
- IP solvers solve problems with linear constraints and objective function where some variables are integers.
- Branch-and-cut solver algorithms, essentially:
  - Compute a series of linear relaxations and cuts (new linear constraints that cut off non-integral solutions).
  - Sometimes branch on a bound for an integer variable.
- State-of-the-art IP solvers very powerful and effective: at times also for solving MaxSAT instances!
Relaxation Variables

MaxSAT solving uses technique of blocking variables to relax (block) soft clauses (selector variables).

- To a soft clause \((x_1 \lor x_2 \lor \cdots \lor x_k)\) we add a new variable \(b\):

\[
(b \lor x_1 \lor x_2 \lor \cdots \lor x_k)
\]

\(b\) does not appear anywhere else in the formula.

- If we make \(b\) true the soft clause is automatically satisfied (is relaxed/is blocked).
- If we make \(b\) false the clause becomes hard and must be satisfied.
MaxSAT encoding into IP

1. For each soft clause $C_i$, *relax* $C_i$ by augmenting it with a new relaxation variable $r_i$.

   $$(x \lor \neg y \lor z \lor \neg w) \leadsto (r_i \lor x \lor \neg y \lor z \lor \neg w)$$

2. Convert every augmented clause into a linear constraint:

   $$r_i + x + (1 - y) + z + (1 - w) \geq 1$$

3. Boolean variables: bound integer domains to $\{0, 1\}$

4. Objective function:

   $$\text{minimize } \sum_{C_i \in F_s} r_i \cdot w_i,$$

   where $w_i$ is the weight of the soft clause $C_i \in F_s$
Integer Programming Summary

- IP solvers use Branch and Cut to solve.
  - Compute a series of linear relaxations and cuts (new linear constraints that cut off non-integral solutions).
  - Sometimes branch on a bound for an integer variable.
  - Also use many other techniques.
- Effective on many standard optimization problems, e.g., vertex cover.
- But for problems where there are many boolean constraints IP is not as effective.
Implicit Hitting Set Algorithms for \textsc{MaxSAT}

[Davies and Bacchus, 2011, 2013b,a]
Hitting Sets and UNSAT Cores

Hitting Sets

Given a collection $S$ of sets of elements, 
A set $H$ is a hitting set of $S$ if $H \cap S \neq \emptyset$ for all $S \in S$.

A hitting set $H$ is optimal if no $H' \subset \bigcup S$ with $|H'| < |H|$ is a hitting set of $S$.

• Note: Under weight function $c : S \rightarrow \mathbb{R}^+$,
  $c(H') < c(H)$ where $c(H) = \sum_{h \in H} c(h)$.

What does this have to do with MaxSAT?

For any MaxSAT instance $F$:
for any optimal hitting set $H$ of the set of UNSAT cores of $F$,
there is an optimal solutions $\tau$ to $F$ such that $\tau$ satisfies exactly the clauses $F \setminus H$. 
Hitting Sets and UNSAT Cores

**Hitting Sets**

Given a collection $S$ of sets of elements,
A set $H$ is a *hitting set* of $S$ if $H \cap S \neq \emptyset$ for all $S \in S$.

A hitting set $H$ is *optimal* if no $H' \subseteq \bigcup S$ with $|H'| < |H|$ is a hitting set of $S$.

- **Note:** Under weight function $c : S \rightarrow \mathbb{R}^+$,
  
  $c(H') < c(H)$ where $c(H) = \sum_{h \in H} c(h)$.

What does this have to do with MAXSAT?

For any MAXSAT instance $F$:

- for any optimal hitting set $H$ of the set of UNSAT cores of $F$,
  there is an optimal solutions $\tau$ to $F$ such that $\tau$ satisfies exactly the clauses $F \setminus H$. 

Hitting Sets

Given a collection $S$ of sets of elements, a set $H$ is a hitting set of $S$ if $H \cap S \neq \emptyset$ for all $S \in S$.

A hitting set $H$ is optimal if no $H' \subseteq \bigcup S$ with $|H'| < |H|$ is a hitting set of $S$.

Note: Under weight function $c : S \rightarrow \mathbb{R}^+$,
$c(H') < c(H)$ where $c(H) = \sum_{h \in H} c(h)$.

What does this have to do with $\text{MaxSAT}$?

For any $\text{MaxSAT}$ instance $F$:
for any optimal hitting set $H$ of the set of UNSAT cores of $F$, there is an optimal solutions $\tau$ to $F$ such that $\tau$ satisfies exactly the clauses $F \setminus H$. 
Hitting Sets and UNSAT Cores

Key insight

To find an optimal solution to a MaxSAT instance $F$, it suffices to:

- Find an (implicit) hitting set $F$ of the UNSAT cores of $F$. Implicit refers to not necessarily having all MUSes of $F$.
- Find a solution to $F \setminus H$. 
Implicit Hitting Set Approach to MaxSAT

Iterate over the following steps:

- Accumulate a collection $\mathcal{K}$ of UNSAT cores using a SAT solver.
- Find an optimal hitting set $H$ over $\mathcal{K}$, and rule out the clauses in $H$ for the next SAT solver call using an LP solver.

... until the SAT solver returns satisfying assignment.

Hitting Set Problem as Integer Programming

\[
\begin{align*}
\min & \quad \sum_{C \in \mathcal{K}} c(C) \cdot r_C \\
\text{subject to} & \quad \sum_{C \in \mathcal{K}} r_C \geq 1 \quad \forall K \in \mathcal{K}
\end{align*}
\]

- $r_C = 1$ iff clause $C$ in the hitting set.
- Weight function $c$: works also for weighted MaxSAT.
Implicit Hitting Set Approach to $\text{MaxSAT}$

Iterate over the following steps:

- Accumulate a collection $\mathcal{K}$ of UNSAT cores using a SAT solver

- Find an optimal hitting set $H$ over $\mathcal{K}$, and rule out the clauses in $H$ for the next SAT solver call using an IP solver

... until the SAT solver returns satisfying assignment.

---

**Hitting Set Problem as Integer Programming**

\[
\begin{align*}
\min & \quad \sum_{C \in \cup \mathcal{K}} c(C) \cdot r_C \\
\text{subject to} & \quad \sum_{C \in K} r_C \geq 1 \quad \forall K \in \mathcal{K}
\end{align*}
\]

- $r_C = 1$ iff clause $C$ in the hitting set
- Weight function $c$: works also for weighted $\text{MaxSAT}$
Implicit Hitting Set Approach to \textsc{MaxSAT}

“Best out of both worlds”

Combining the main strengths of SAT and IP solvers:

- SAT solvers are very good at proving unsatisfiability
  - Provide explanations for unsatisfiability in terms of cores
  - Instead of adding clauses to / modifying the input MaxSAT instance: each SAT solver call made on a subset of the clauses in the instance

- IP solvers at optimization
  - Instead of directly solving the input MaxSAT instance: solve a sequence of simpler hitting set problems over the cores

Instantiation of the implicit hitting set approach

[Moreno-Centeno and Karp, 2013]

- Also possible to instantiate beyond \textsc{MaxSAT}

[Saikko, Wallner, and Järvisalo, 2016]
Input:
hard clauses $F_h$, soft clauses $F_s$, weight function $c : F_s \rightarrow \mathbb{R}^+$
**Input:**
hard clauses $F_h$, soft clauses $F_s$, weight function $c : F_s \rightarrow \mathbb{R}^+$

1. Initialize

   $F_h, F_s$
   $hs := \emptyset$
   $\mathcal{K} := \emptyset$

   $\mathcal{K} := \mathcal{K} \cup \{K\}$

**SAT solver**

$F_h \land (F_s \setminus hs)$

**IP solver**

$\min \sum_{C \in \mathcal{K}} c(C) \cdot r_C$
\[\sum_{C \in \mathcal{K}} r_C \geq 1 \ \forall K \in \mathcal{K}\]

**unsat**

**sat**

**Optimal solution found**

**UNSAT core extraction**

**Min-cost Hitting Set**
Solving \textbf{MaxSAT} by SAT and Hitting Set Computations

**Input:**
hard clauses $F_h$, soft clauses $F_s$, weight function $c : F_s \mapsto \mathbb{R}^+$

2. **UNSAT core**

\[ F_h, F_s \]
\[ hs := \emptyset \]
\[ \mathcal{K} := \emptyset \]

\[ \mathcal{K} := \mathcal{K} \cup \{K\} \]

\[ c \]

\[ F_h \land (F_s \setminus hs) \]

\[ SAT \ solver \]

\[ unsat \]

\[ sat \]

**UNSAT core extraction**

**Min-cost Hitting Set**

\[ \text{Optimal solution found} \]

**IP solver**

\[ \min \sum_{C \in \mathcal{K}} c(C) \cdot r_C \]
\[ \sum_{C \in \mathcal{K}} r_C \geq 1 \ \forall K \in \mathcal{K} \]
Solving \textbf{MaxSAT} by SAT and Hitting Set Computations

**Input:**
hard clauses $F_h$, soft clauses $F_s$, weight function $c : F_s \mapsto \mathbb{R}^+$

3. Update core set

\[ F_h, F_s \]
\[ hs := \emptyset \]
\[ \mathcal{K} := \emptyset \]
\[ \mathcal{K} := \mathcal{K} \cup \{ K \} \]
\[ c \]

\[ F_h \land (F_s \setminus hs) \]

\[ \text{unsat} \]

**UNSAT core extraction**

\[ \text{sat} \]

**Min-cost Hitting Set**

\[ \text{optimal solution found} \]

**IP solver**

\[ \min \sum_{C \in \mathcal{K}} c(C) \cdot r_c \]

\[ \sum_{C \in \mathcal{K}} r_c \geq 1 \quad \forall K \in \mathcal{K} \]
Solving \textsc{MaxSAT} by SAT and Hitting Set Computations

\textbf{Input:}

hard clauses $F_h$, soft clauses $F_s$, weight function $c : F_s \rightarrow \mathbb{R}^+$

4. Min-cost HS of $\mathcal{K}$

\begin{align*}
F_h, F_s \\
hs := \emptyset \\
\mathcal{K} := \emptyset
\end{align*}

\begin{align*}
\mathcal{K} := \mathcal{K} \cup \{K\} \\

\text{min} \sum_{C \in \mathcal{K}} c(C) \cdot r_C \\
\sum_{C \in \mathcal{K}} r_C \geq 1 \quad \forall K \in \mathcal{K}
\end{align*}

\begin{align*}
\text{Optimal solution found}
\end{align*}
Solving MaxSAT by SAT and Hitting Set Computations

**Input:**
hard clauses $F_h$, soft clauses $F_s$, weight function $c : F_s \mapsto \mathbb{R}^+$

5. UNSAT core

- $F_h, F_s$
- $hs := \emptyset$
- $\mathcal{K} := \emptyset$

**SAT solver**

$F_h \land (F_s \setminus hs)$

**UNSAT core extraction**

**Min-cost Hitting Set**

- $\mathcal{K} := \mathcal{K} \cup \{K\}$
- $c$

**IP solver**

$\min \sum_{C \in \mathcal{K}} c(C) \cdot r_C$

$\sum_{C \in K} r_C \geq 1 \ \forall K \in \mathcal{K}$

Optimal solution found
Solving $\text{MaxSAT}$ by SAT and Hitting Set Computations

Input:
hard clauses $F_h$, soft clauses $F_s$, weight function $c : F_s \mapsto \mathbb{R}^+$

iterate until “sat”

$F_h, F_s$
$hs := \emptyset$
$\mathcal{K} := \emptyset$
$\mathcal{K} := \mathcal{K} \cup \{K\}$

$F_h \land (F_s \setminus hs)$

SAT solver

UNSAT core extraction

IP solver

Min-cost Hitting Set

$\min \sum_{C \in \mathcal{K}} c(C) \cdot r_C$
$\sum_{C \in \mathcal{K}} r_C \geq 1 \ \forall K \in \mathcal{K}$

unsat

sat

hs of $\mathcal{K}$

Optimal solution found
Solving \textbf{MaxSAT} by SAT and Hitting Set Computations

**Input:**
hard clauses $F_h$, soft clauses $F_s$, weight function $c : F_s \rightarrow \mathbb{R}^+$

iterate until “sat”

\[
\begin{align*}
F_h, F_s \\
hs := \emptyset \\
\mathcal{K} := \emptyset
\end{align*}
\]

unsat

\[
\begin{align*}
\mathcal{K} := \mathcal{K} \cup \{K\} \\
c
\end{align*}
\]

unsat

\[
\begin{align*}
\text{SAT solver} \\
F_h \land (F_s \setminus hs)
\end{align*}
\]

unsat

\[
\begin{align*}
\text{Optimal solution found}
\end{align*}
\]

unsat

\[
\begin{align*}
\text{IP solver} \\
\min \sum_{C \in \mathcal{K}} c(C) \cdot r_C \\
\sum_{C \in \mathcal{K}} r_C \geq 1 \ \forall K \in \mathcal{K}
\end{align*}
\]
Solving MaxSAT by SAT and Hitting Set Computations

**Intuition:** After **optimally** hitting all cores of \( F_h \land F_s \) by \( hs \): any solution to \( F_h \land (F_s \setminus hs) \) is **guaranteed to be optimal.**

**Diagram:**
- **SAT solver:** \( F_h, F_s \)
  - \( hs := \emptyset \)
  - \( \mathcal{K} := \emptyset \)
- **IP solver**
  - \( \text{unsat} \)
  - \( c \)
  - \( \text{optimal solution found} \)

**Algorithm:**
- **Iterate until “sat”**
  - \( F_h \land (F_s \setminus hs) \)
  - \( K := \emptyset \)
  - \( K := \mathcal{K} \cup \{K\} \)
  - \( \text{hs of } \mathcal{K} \)
  - \( \text{min } \sum_{C \in \mathcal{K}} c(C) \cdot r_C \)
  - \( \sum_{C \in \mathcal{K}} r_C \geq 1 \ \forall K \in \mathcal{K} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[
\begin{align*}
C_1 &= x_6 \lor x_2 \\
C_4 &= \neg x_1 \\
C_7 &= x_2 \lor x_4 \\
C_{10} &= \neg x_7 \lor x_5 \\
C_2 &= \neg x_6 \lor x_2 \\
C_5 &= \neg x_6 \lor x_8 \\
C_8 &= \neg x_4 \lor x_5 \\
C_{11} &= \neg x_5 \lor x_3 \\
C_3 &= \neg x_2 \lor x_1 \\
C_6 &= x_6 \lor \neg x_8 \\
C_9 &= x_7 \lor x_5 \\
C_{12} &= \neg x_3
\end{align*}
\]
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \]
\[ C_4 = \neg x_1 \]
\[ C_7 = x_2 \lor x_4 \]
\[ C_{10} = \neg x_7 \lor x_5 \]
\[ C_2 = \neg x_6 \lor x_2 \]
\[ C_5 = \neg x_6 \lor x_8 \]
\[ C_{11} = \neg x_5 \lor x_3 \]
\[ C_3 = \neg x_2 \lor x_1 \]
\[ C_6 = x_6 \lor \neg x_8 \]
\[ C_9 = x_7 \lor x_5 \]
\[ C_{12} = \neg x_3 \]

\[ \mathcal{K} := \emptyset \]
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \emptyset \]

SAT solve \( F_h \land (F_s \setminus \emptyset) \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \]
\[ C_4 = \neg x_1 \]
\[ C_7 = x_2 \lor x_4 \]
\[ C_{10} = \neg x_7 \lor x_5 \]
\[ C_2 = \neg x_6 \lor x_2 \]
\[ C_5 = \neg x_6 \lor x_8 \]
\[ C_8 = \neg x_4 \lor x_5 \]
\[ C_{11} = \neg x_5 \lor x_3 \]
\[ C_3 = \neg x_2 \lor x_1 \]
\[ C_6 = x_6 \lor \neg x_8 \]
\[ C_9 = x_7 \lor x_5 \]
\[ C_{12} = \neg x_3 \]

\[ K := \emptyset \]

- SAT solve \( F_h \land (F_s \setminus \emptyset) \leadsto \) UNSAT core \( K = \{ C_1, C_2, C_3, C_4 \} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \]
\[ C_4 = \neg x_1 \]
\[ C_7 = x_2 \lor x_4 \]
\[ C_{10} = \neg x_7 \lor x_5 \]
\[ C_2 = \neg x_6 \lor x_2 \]
\[ C_5 = \neg x_6 \lor x_8 \]
\[ C_8 = \neg x_4 \lor x_5 \]
\[ C_{11} = \neg x_5 \lor x_3 \]
\[ C_3 = \neg x_2 \lor x_1 \]
\[ C_6 = x_6 \lor \neg x_8 \]
\[ C_9 = x_7 \lor x_5 \]
\[ C_{12} = \neg x_3 \]

\[ \mathcal{K} := \left\{ \{C_1, C_2, C_3, C_4\} \right\} \]

- Update \( \mathcal{K} := \mathcal{K} \cup \{K\} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \]
\[ C_4 = \neg x_1 \]
\[ C_7 = x_2 \lor x_4 \]
\[ C_{10} = \neg x_7 \lor x_5 \]
\[ C_2 = \neg x_6 \lor x_2 \]
\[ C_5 = \neg x_6 \lor x_8 \]
\[ C_8 = \neg x_4 \lor x_5 \]
\[ C_{11} = \neg x_5 \lor x_3 \]
\[ C_3 = \neg x_2 \lor x_1 \]
\[ C_6 = x_6 \lor \neg x_8 \]
\[ C_9 = x_7 \lor x_5 \]
\[ C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{ \{ C_1, C_2, C_3, C_4 \} \} \]

- Solve minimum-cost hitting set problem over \( \mathcal{K} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{\{C_1, C_2, C_3, C_4\}\} \]

- Solve minimum-cost hitting set problem over \( \mathcal{K} \leadsto hs = \{C_1\} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ \begin{align*}
C_1 &= x_6 \lor x_2 \\
C_4 &= \neg x_1 \\
C_7 &= x_2 \lor x_4 \\
C_{10} &= \neg x_7 \lor x_5 \\
C_2 &= \neg x_6 \lor x_2 \\
C_5 &= \neg x_6 \lor x_8 \\
C_8 &= \neg x_4 \lor x_5 \\
C_{11} &= \neg x_5 \lor x_3 \\
C_3 &= \neg x_2 \lor x_1 \\
C_6 &= x_6 \lor \neg x_8 \\
C_9 &= x_7 \lor x_5 \\
C_{12} &= \neg x_3
\end{align*} \]

\[ \mathcal{K} := \{\{C_1, C_2, C_3, C_4\}\} \]

- SAT solve \( F_h \land (F_s \setminus \{ C_1 \}) \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{\{C_1, C_2, C_3, C_4\}\} \]

- SAT solve \( F_h \land (F_s \setminus \{C_1\}) \) \(\leadsto\) UNSAT core \( K = \{C_9, C_{10}, C_{11}, C_{12}\} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{\{ C_1, C_2, C_3, C_4 \}, \{ C_9, C_{10}, C_{11}, C_{12} \}\} \]

- Update \( \mathcal{K} := \mathcal{K} \cup \{ K \} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{\{C_1, C_2, C_3, C_4\}, \{C_9, C_{10}, C_{11}, C_{12}\}\} \]

- Solve minimum-cost hitting set problem over \( \mathcal{K} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \left\{ \{C_1, C_2, C_3, C_4\}, \{C_9, C_{10}, C_{11}, C_{12}\} \right\} \]

- Solve minimum-cost hitting set problem over \( \mathcal{K} \leadsto hs = \{C_1, C_9\} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{\{C_1, C_2, C_3, C_4\}, \{C_9, C_{10}, C_{11}, C_{12}\}\} \]

- SAT solve \( F_h \land (F_s \setminus \{C_1, C_9\}) \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \]
\[ C_4 = \neg x_1 \]
\[ C_7 = x_2 \lor x_4 \]
\[ C_{10} = \neg x_7 \lor x_5 \]
\[ C_2 = \neg x_6 \lor x_2 \]
\[ C_5 = \neg x_6 \lor x_8 \]
\[ C_8 = \neg x_4 \lor x_5 \]
\[ C_{11} = \neg x_5 \lor x_3 \]
\[ C_3 = \neg x_2 \lor x_1 \]
\[ C_6 = x_6 \lor \neg x_8 \]
\[ C_9 = x_7 \lor x_5 \]
\[ C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{\{C_1, C_2, C_3, C_4\}, \{C_9, C_{10}, C_{11}, C_{12}\}\} \]

- SAT solve \( F_h \land (F_s \setminus \{C_1, C_9\}) \)
  - UNSAT core \( K = \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{ \{ C_1, C_2, C_3, C_4 \}, \{ C_9, C_{10}, C_{11}, C_{12} \}, \{ C_3, C_4, C_7, C_8, C_{11}, C_{12} \} \} \]

- Update \( \mathcal{K} := \mathcal{K} \cup \{ K \} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{ \{ C_1, C_2, C_3, C_4 \}, \{ C_9, C_{10}, C_{11}, C_{12} \}, \{ C_3, C_4, C_7, C_8, C_{11}, C_{12} \} \} \]

- Solve minimum-cost hitting set problem over \( \mathcal{K} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{\{C_1, C_2, C_3, C_4\}, \{C_9, C_{10}, C_{11}, C_{12}\}, \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\}\} \]

- Solve minimum-cost hitting set problem over \( \mathcal{K} \sim hs = \{C_4, C_9\} \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{ \{C_1, C_2, C_3, C_4\}, \{C_9, C_{10}, C_{11}, C_{12}\}, \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\} \} \]

- SAT solve \( F_h \land (F_s \setminus \{C_4, C_9\}) \)
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{\{C_1, C_2, C_3, C_4\}, \{C_9, C_{10}, C_{11}, C_{12}\}, \{C_3, C_4, C_7, C_8, C_{11}, C_{12}\}\} \]

- SAT solve \( F_h \land (F_s \setminus \{C_4, C_9\}) \) \( \Rightarrow \) SATISFIABLE.
MaxSAT by SAT and Hitting Set Computation: Example

\[ C_1 = x_6 \lor x_2 \quad C_2 = \neg x_6 \lor x_2 \quad C_3 = \neg x_2 \lor x_1 \]
\[ C_4 = \neg x_1 \quad C_5 = \neg x_6 \lor x_8 \quad C_6 = x_6 \lor \neg x_8 \]
\[ C_7 = x_2 \lor x_4 \quad C_8 = \neg x_4 \lor x_5 \quad C_9 = x_7 \lor x_5 \]
\[ C_{10} = \neg x_7 \lor x_5 \quad C_{11} = \neg x_5 \lor x_3 \quad C_{12} = \neg x_3 \]

\[ \mathcal{K} := \{ \{ C_1, C_2, C_3, C_4 \}, \{ C_9, C_{10}, C_{11}, C_{12} \}, \{ C_3, C_4, C_7, C_8, C_{11}, C_{12} \} \} \]

- SAT solve \( F_h \land (F_s \setminus \{ C_4, C_9 \}) \) \( \leadsto \) SATISFIABLE.
  Optimal cost: 2 (cost of hs).
Optimizations

Solvers implementing the implicit hitting set approach include several optimizations, such as

- a *disjoint phase* for obtaining several cores before/between hitting set computations
- combinations of greedy and exact hitting sets computations
- ...

Some of these optimizations are *integral* for making the solvers competitive.

For more on some of the details, see [Davies and Bacchus, 2011, 2013b,a]
Implicit Hitting Set Approach to MaxSAT

- Effective on range of MaxSAT problems including large ones.
- Superior to other methods when there are many distinct weights.
- Usually superior to CPLEX.
- On problems with no weights or very few weights can be outperformed by SAT based approaches.
Iterative Use of SAT Solvers for MaxSAT
Implementing **MaxSAT** Algorithms via Assumptions

SAT-based MaxSAT algorithms make use of the assumptions interface in SAT solvers

- **Instrument each soft clause** $C_i$ **with a new “assumption” variable** $a_i$
  - $\sim$ **replace** $C_i$ **with** $(C_i \lor a_i)$ **for each soft clause** $C_i$
- $a_i = 0$ **switches** $C_i$ **“on”**, $a_i = 1$ **switches** $C_i$ **“off”**
- **MaxSAT core:** a subset of the assumptions variables $a_i$s
  - Heavily used in **core-based MaxSAT algorithms**
  - In the **implicit hitting set approach**:
    - hitting sets over sets of assumption variables
  - Cost of including $a_i$ in a core (i.e., assigning $a_i = 1$): weight of the soft clause $C_i$

- **Can state cardinality constraints directly over the assumption variables**
  - Heavily used in **MaxSAT algorithms employing cardinality constraints**
Summary

Take-home message
- Various approaches to MaxSAT solving
- Core-based (and implicit hitting set) approaches currently best performing on most MaxSAT instances encoding real-world problems
- IP also effective

Study goals
- Types of approaches to exact MaxSAT solving
- Details on some key algorithms: B&B, core-based, implicit hitting set

Next time:
- SAT-based Counterexample-guided abstraction refinement
- Quantified Boolean formula satisfiability


