Not all interpretation are the same

- Satisfiability searches for any interpretation that satisfies a set of clauses \( \mathcal{C} = \{ C_1, \ldots, C_n \} \);
- In many situation some interpretations are better than others’ preference relation between interpretations can be represented with a partial ordered set (poset)

\[
\langle \text{models}(\mathcal{C}), \prec \rangle
\]
Example (Team building)

Build a team with competences in machine learning ($M$), knowledge representation ($K$) vision ($V$), and human computer interaction ($H$), selecting four people from:

<table>
<thead>
<tr>
<th>Person</th>
<th>gender</th>
<th>$M$</th>
<th>$K$</th>
<th>$V$</th>
<th>$H$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>f</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Bea</td>
<td>f</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Celine</td>
<td>f</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dania</td>
<td>f</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Enrico</td>
<td>m</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Felix</td>
<td>m</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example (Team building (cont’d))

Formalization in SAT:

\[ M \land K \land V \land H \]
\[ A + B + C + D + E + F = 4 \]

You want all 4 competences

You have to select 4 people

\[ M \rightarrow A \lor B \lor C \lor D \lor E \lor F \]
\[ K \rightarrow A \lor C \lor F \]
\[ V \rightarrow A \lor B \lor E \]
\[ H \rightarrow A \lor D \]
Example (Team building (cont’d))

models($\mathbf{C}$) contains the following assignments

Team$_1$ = \{A, B, C, D, M, K, V, H\}
Team$_2$ = \{A, B, C, E, M, K, V, H\}
Team$_3$ = \{A, B, C, F, M, K, V, H\}
Team$_4$ = \{A, B, D, E, M, K, V, H\}
Team$_5$ = \{A, B, D, F, M, K, V, H\}
Team$_6$ = \{A, B, E, F, M, K, V, H\}
Team$_7$ = \{A, C, D, E, M, K, V, H\}
Team$_9$ = \{A, C, D, F, M, K, V, H\}
Team$_9$ = \{A, C, E, F, M, K, V, H\}
Team$_{10}$ = \{A, D, E, F, M, K, V, H\}
Team$_{11}$ = \{B, C, D, E, M, K, V, H\}
Team$_{12}$ = \{B, C, D, F, M, K, V, H\}
Team$_{13}$ = \{B, D, E, F, M, K, V, H\}
Team$_{14}$ = \{C, D, E, F, M, K, V, H\}
you would like to express also some preference on the teams:

- C1 you prefer teams with gender balance;
- C2 you prefer teams with higher competence level;

Rank the potential teams according to the respective criteria

\[ gb = 1 - \frac{|\#\text{male} - \#\text{female}|}{4} \]

\[ \text{cpt}(Y) = \sum_{X \in \text{team}} \text{level\_of\_cpt}(X, Y) \]
<table>
<thead>
<tr>
<th>Team</th>
<th>gb</th>
<th>$\text{cmp}(M)$</th>
<th>$\text{cmp}(K)$</th>
<th>$\text{cmp}(V)$</th>
<th>$\text{cmp}(H)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Team}_1$</td>
<td>0.0</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$\text{Team}_2$</td>
<td>0.5</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$\text{Team}_3$</td>
<td>0.5</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>$\text{Team}_4$</td>
<td>0.5</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$\text{Team}_5$</td>
<td>0.5</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$\text{Team}_6$</td>
<td>1.0</td>
<td>7</td>
<td>2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$\text{Team}_7$</td>
<td>0.5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$\text{Team}_9$</td>
<td>0.5</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>$\text{Team}_9$</td>
<td>1.0</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>$\text{Team}_{10}$</td>
<td>1.0</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$\text{Team}_{11}$</td>
<td>0.5</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>$\text{Team}_{12}$</td>
<td>0.5</td>
<td>7</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>$\text{Team}_{13}$</td>
<td>1.0</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>$\text{Team}_{14}$</td>
<td>1.0</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
A general method to express preference relation between interpretations is via **weighted formulas**. I.e.,

$$w : \phi$$  \hspace{1cm} (1)

\(w \in \mathbb{R}\) is the **weight** of the propositional formula \(\phi\)

A set of weighted formulas \(F = \{w_1 : \phi_1, w_2 : \phi_2, \ldots, w_k : \phi_k\}\) defines a total order between the interpretations such as

\[\mathcal{I} \preceq \mathcal{J} \text{ if and only if } w_F(\mathcal{I}) \leq w_F(\mathcal{J})\]  \hspace{1cm} (2)

where

\[w_F(\mathcal{I}) = \sum_{i=1}^{k} w_i \cdot \mathcal{I}(C_i) = \sum_{\mathcal{I} \models C_i} w_i\]
Example of weighted formulas

Example (Team building (cont’d))

To rank the models according to the gender balance criteria we can use the following weighted formulas:

\[ 1 : (x \land y) \text{ for every } x, y \in \{A, B, C, D, E, F\} \text{ such that } x \text{ is a male and } y \text{ a female} \]

The weight of an interpretation is \(#male \cdot \#female\). This weight function is equivalent to the the weight function produced by the gender balance criteria. (prove by exercise).
Example (Team building (cont’d))

To rank the models with respect to one of the competence (say \( M \)) We can use the expertese level of each member. i.e., the weighted formula

\[
\text{ExpertLevel}(M, x) : x \quad \text{for every } x \in \{A, B, C, D, E, F\} \quad \text{where} \\
\text{ExpertLevel}(M, x) \quad \text{is the expert level of } x \quad \text{in machine learning}
\]

The weight of the models are reported in the column \( M \) of the table with all the possible teams.
Properties of the weight function

**Definition**

Two sets of weighted formulas $F_1$ and $F_2$ are **equivalent** if they define the same order. I.e., if for all interpretations $\mathcal{I}, \mathcal{J}$

$$w_1(\mathcal{I}) < w_1(\mathcal{J}) \quad \text{if and only if} \quad w_2(\mathcal{I}) < w_2(\mathcal{J})$$

Two sets of weighted formulas $F_1$ and $F_2$ are **opposite** if

$$w_1(\mathcal{I}) < w_1(\mathcal{J}) \quad \text{if and only if} \quad w_2(\mathcal{J}) < w_2(\mathcal{I})$$
Properties of the weight function

Proposition

1. $F$ is equivalent to $a \cdot F = \{a \cdot w : \phi \mid w : \phi \in F\}$ for $a > 0$;
2. $F$ is opposite to $a \cdot F = \{a \cdot w : \phi \mid w : \phi \in F\}$ for $a < 0$;
3. $F \cup \{w : \phi\}$ is equivalent to $F \cup \{-w : \neg \phi\}$
4. If $\models \phi \leftrightarrow \psi$, then $F \cup \{w : \phi\}$ is equivalent to $F \cup \{w : \psi\}$
5. $F \cup \{w_1 : \phi, w_2 : \phi\}$ is equivalent to $F \cup \{w_1 + w_2 : \phi\}$

Proposition (Non Properties)

1. $F$ is not equivalent to $a + F = \{a + w : \phi \mid w : \phi \in F\}$;
2. $F \cup \{w : \phi \land \psi\}$ is not equivalent to $F \cup \{w : \phi, w : \psi\}$
\[ F = \{3 : p \land q, 1 : \neg p, 1 : \neg q\} 2 + F = \{5 : \neg p \lor q, 3 : \neg p, 3 : \neg q\} \]

<table>
<thead>
<tr>
<th>( w(p, q) )</th>
<th>= 3</th>
<th>( w(p, q) )</th>
<th>= 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w(p, \neg q) )</td>
<td>= 1</td>
<td>( w(p, \neg q) )</td>
<td>= 3</td>
</tr>
<tr>
<td>( w(\neg p, q) )</td>
<td>= 1</td>
<td>( w(\neg p, q) )</td>
<td>= 3</td>
</tr>
<tr>
<td>( w(\neg p, \neg q) )</td>
<td>= 2</td>
<td>( w(\neg p, \neg q) )</td>
<td>= 6</td>
</tr>
</tbody>
</table>
\[
F = \{1.5 : p \land q, 1 : \neg p, 1 : \neg q\} \quad F' = \{1.5 : p, 1.5 : q, 1 : \neg p, 1 : \neg q\}
\]

\[
w(p, q) = 1.5 \quad w(p, q) = 3
\]

\[
w(p, \neg q) = 1 \quad w(p, \neg q) = 2.5
\]

\[
w(\neg p, q) = 1 \quad w(\neg p, q) = 2.5
\]

\[
w(\neg p, \neg q) = 2
\]
MaxSAT - Maximum satisfiability

Definition (MaxSAT)

Given a set of weighted clauses $w_1 : C_1, \ldots, w_n : C_n$ and a set of standard clauses $D_1, \ldots, D_m$, the MaxSAT problem is the problem of finding the interpretation $\mathcal{I}$ that

1. $\mathcal{I} \models D_1 \wedge \cdots \wedge D_m$

2. $\mathcal{I}$ maximizes the function $\sum_{i=1}^{n} w_i \cdot \mathcal{I}(C_i)$

3. or equivalently minimizes the function $\sum_{i=1}^{n} w_i \cdot \mathcal{I}(\neg C_i)$

- each $C_i$ is called soft constraint and it can be satisfied or not;
- the cost of not satisfying $C_i$ is $w_i$ and in MaxSAT you want to minimize the total cost of not satisfying the soft constraints.
- each $D_i$ is called hard constraint and it must be satisfied.
First of all let us show that maximizing $\sum_{i=1}^{n} w_i \cdot I(C_i)$ is the same as minimizing $\sum_{i=1}^{n} w_i \cdot I(\neg C_i)$. Since $I(\neg C) = 0$ iff $I(C) = 1$ we have that:

$$\sum_{i=1}^{n} w_i \cdot I(\neg C_i) = \sum_{i=1}^{n} w_i - \sum_{i=1}^{n} w_i \cdot I(C_i)$$

Since the term $\sum_i w_i$ is constant and $\sum_{i=1}^{n} w_i \cdot I(C_i)$ occurs in on the right of the “=” symbol with negative sign; maximizing $\sum_{i=1}^{n} w_i \cdot I(C_i)$ is the same as minimizing $\sum_{i=1}^{n} w_i \cdot I(\neg C_i)$.

As we will see in the next slide the *minimizing* formulation is preferrable because it allows to associate infinite weight to hard constraints.
MaxSAT Variations

- **Basic MaxSAT**: There are no hard clauses and all the soft clauses have the same weight (equal to 1).
- **Partial MaxSAT**: The set of hard clauses could be not empty and the soft clauses have the same weight (equal to 1).
- **Weighted MaxSAT**: No hard clauses and different weights associated with soft clauses
- **Weighted Partial MaxSAT**: The set of hard clauses could be non empty, the soft clauses can be associated with different weight,
for a more uniform representation it is convenient to consider hard constraints $D_1, \ldots, D_m$ as soft constraints with infinite costs
therefore the (weighted) MasSAT problem is defined on a set

$$\phi = \{ w_1 : C_1, \ldots, w_n : C_n, \infty : D_1, \ldots, \infty : D_m \}$$

If the solution of the MaxSAT problem is infinite ($\infty$) then at least one hard constraint is not satisfied, and therefore we say the the entire problem $\phi$ is unsatisfiable.

in practice $\infty$ is replaced with $\sum_{i=1}^n w_i + 1$
MaxSAT application to optimization problems

Most real-world problems involve an optimization component
Examples:

- Find a **shortest** path/plan/execution/ ... to a goal state
- Find a **smallest** explanation of a certain phenomena in terms of causes
- Find a **least** resource-consuming schedule
- Find a **most probable** state (Maximum a Posteriori - MAP)
Example - Shortest Path

Example

Find shortest path in a grid with horizontal/vertical moves. Travel from S to G without entering in the black squares.

Encoding in propositional logic

- one propositional variable $c_{ij}$ for every cell $(i, j)$ with $i, j \in \{1, \ldots, 5\}$.
- $c_{ij}$ is true if the cell $i, j$ is visited in going from S to G.
Example - Shortest Path

**Hard Constraints**
- $S$ and $G$ are visited $c_{51} \land c_{25}$;
- $S$ and $G$ have a single visited neighbour

$$c_{ij} \rightarrow \bigvee_{a \in N(i,j)} c_a \quad (i,j) = (5,1), (2,5)$$

$N(i,j) = \{(i',j') \mid 1 \leq i', j' \leq 5, \| (i,j) - (i',j') \|_1 = 1\}$, i.e., the cells on the left/right/up/down (if any) of cell $i,j$.

- other visited cells must have exactly two visited neighbours:

$$c_{ij} \rightarrow \bigvee_{a \neq b \in N(i,j)} c_a \land c_b$$

- black cells cannot be visited $\neg c_{11} \land \neg c_{31} \land \neg c_{25}$;

**Soft Constraints**
- visit the minimum number of cells: $-1 : c_{ij}$, (or equivalently $1 : \neg c_{ij}$)
Every interpretation that satisfies the hard constraints correspond to a (set of paths) from $S$ to $G$;

the weight of each model $\mathcal{I}$ of the hard constraints, i.e., $w(\mathcal{I})$ is equal to $-k$, where $k$ is the number of $c_{ij}$ which are assigned to true by $\mathcal{I}$;
Algorithms for MaxSAT

- DPLL (naïve)
- Branch-and-bound
- Integer Programming (IP)
- SAT-Based Algorithms
- Implicit hitting set algorithms (IP/SAT hybrid).
Modification of DPLL for MaxSAT

MaxSAT-DPLL($\phi : \text{CNF}, \psi : \text{weighted CNF}, I : \text{Partial assignment}$)

1: $I, \phi \leftarrow \text{UnitPropagation}(I, \phi)$
2: $\psi \leftarrow \psi|_I$
3: if $\{\} \in \phi$ then
4: return $I, \infty$
5: end if
6: if $\phi = \{\}$ and $\psi$ contains only empty weighted clauses then
7: return $I, \sum_{(w,D) \in \psi} w$
8: else
9: select a $l$ from some clause in $\phi$ or in $\psi$
10: $I, c \leftarrow \text{MaxSAT-DPLL}(\phi|_l, \psi|_l, I \cup \{l\})$
11: $I', c' \leftarrow \text{MaxSAT-DPLL}(\phi|_{\neg l}, \psi|_{\neg l}, I \cup \{\neg l\})$
12: if $c \leq c'$ then
13: return $I, c$
14: else
15: return $I', c'$
16: end if
17: end if
Early stop search using Lower/Upper Bound

- **MaxSAT-DPLL** performs an exhaustive search on all the models of $\phi$ and evaluates their cost on $\psi$;
- not very efficient

possible improvement:
- remember the best model found so far $I_{UB}$ and its cost $UB$ (UB stands for upper-bound)
- we are not interested in models with cost $> UB$.
- compute the lower bound (LB) of the cost of the current partial assignment $I$,
- if $LB > UB$ stop expanding $I$, and backtrack
Branch and bound

\[ p \lor q \lor r \lor s \lor t : \infty \]
\[ \neg p : 3 \quad \neg q : 2 \quad \neg r : 4 \]

- \( UB = \text{cost of the best solution so far}; \)
- \( LB = \text{minimum cost achievable under the node}; \)
- Abandone the subtree of a node if \( LB > UB \) (no solution under this node);
Branch and Bound

B&B(ϕ: CNF, ψ: Weighted CNF, ℐ: Partial assignment ℐ_{UB}: Best previously found solution, UB: Cost of ℐ_{UB})

1: ℐ, ϕ ← UnitPropagation(ℐ, ϕ)
2: ψ ← ψ|ℐ
3: if {} ∈ ϕ or UB ≤ ∑w:{}∈ψ w then
4:    return ℐ_{UB}, UB
5: end if
6: if ϕ = {} and ψ contains only empty weighted clauses then
7:    return ℐ, ∑(w:D)∈ψ w
8: else
9:    select a l from some clause in ϕ or in ψ
10:   ℐ, c ← B&B(ϕ|l, ψ|l, ℐ ∪ {l}, ℐ_{UB}, UB)
11:   ℐ′, c′ ← B&B(ϕ|¯l, ψ|¯l, ℐ ∪ {¯l}, ℐ_{UB}, UB)
12: if c ≤ c′ then
13:    return ℐ, c
14: else
15:    return ℐ′, c′
16: end if
17: end if
Example

Resolve the MaxSAT problem for the following weighted formulas by applying B&B.

\[(a \rightarrow b \lor c : \infty)\]
\[(b \rightarrow d : \infty)\]
\[(d \rightarrow \neg a : \infty)\]

(a : 1)
(b : 2)
(c : 3)
(d : 2)
The algorithm presented here is the vanilla (simplest) version of B&B. There are many possible improvements of it’s efficiency:

- estimation of the LB (cores) e.g., replace \(((x) : 2, (\neg x) : 3)\) with \(((\) : 2, (\neg x), 1)\);
- propagation rules for weighted clauses (MaxRes) e.g. replace \(((a, b), w_1), (\neg a, c) : w_2)\) with \(((b, c) : min(w_1, w_2))\) and additional compensating formulas.

B&B 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 optimization techniques become weak or too expensive.
A Weighted Partial MaxSAT problem

\[ \phi = \{(C_1, w_1), \ldots, (C_n, w_n), (D_1, \infty), \ldots, (D_m, \infty)\} \]

can be solved by finding the minimal \( k \in \mathbb{N} \) such that the formula

\[
\phi_k = \{C_1 \lor b_1, \ldots, C_n \lor b_n, D_1, \ldots, D_m\} \cup \text{CNF} \left( \sum_{i=1}^{n} w_i \cdot b_i \leq k \right)
\]

is satisfiable.

**Remark**

Notice that \( k \) may range from 0 to \( \sum_{i=1}^{n} w_i \). Therefore the na"ive algorithm that run SAT to the above formula for \( k = 0, 1, \ldots, \sum w_i \) will solve the MaxSAT problem within a finite time.
Encoding Cardinality constraints:

How can we encode the following constraint?

\[ \sum_{i=1}^{n} w_i b_i \leq k \]

- Suppose that each \( w_i = 1 \): \( b_1 + b_2 + \cdots + b_n \leq k \) can be encoded as
  \[ \bigwedge_{B \subseteq \{b_1, \ldots, b_n\}} \bigvee_{b_i \in B} \neg b_i \]

- when \( w_i \) is an integer > 1 then we introduce \( w_i \) copies of \( b_i^1 \) \((b_1^1 \equiv b_i, \ldots, b_i^{w_i} \equiv b_i)\), with the clause
  \[ \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{w_i} \left( (\neg b_i \lor b_j^i) \land (b_i \lor \neg b_j^i) \right) \]
  and then encode the constraint:

\[ \sum_{i=1}^{n} \sum_{v=1}^{w_i} b_i^v \leq k \]

1More efficient encodings are also possible.
The Fu and Malik algorithm for MaxSAT uses SAT as an oracle (i.e., it calls SAT as a subroutine).

$SAT(C_1, \ldots, C_N)$ can return one of the following outputs:

- Satisfiable: if $C_1, \ldots, C_n$ are satisfiable
- (Unsatisfiable, $\mathcal{C}$) for some set of clauses $\mathcal{C}$ that is a minimal subset of $\{C_1, \ldots, C_n\}$ which are not satisfiable.
**Fu and Malik algorithm for MaxSAT - intuition**

1. \( \text{MaxSAT}((A : 1), (B : 1), (C : 1), (D : \infty), (E : \infty)) \)
2. \((A, B, C, D, E)\) is unsat
3. \((A, B, E)\) minimal unsat subset of \((A, B, C, D, E)\) is unsat
4. at least one among \(A\) and \(B\) should be false
5. change \((A : 1)\) into \((A \lor a : 1)\),
   change \((B : 1)\) into \((B \lor b : 1)\).
   If one among \(a\) and \(b\) is true, then one among \(A\) and \(B\) can be false
6. add the constraint \(a + b = 1\)
7. cost = cost + 1
8. \( \text{MaxSAT} \left( (A \lor a : 1), (B \lor b : 1), (D : \infty), (E : \infty), (a + b = 1 : \infty) \right) \)
9. go to 1
Example

\[
\text{MaxSAT} \left( (p : 1), (q, 1), (r : 1), \\
(\neg p \lor \neg q, \infty), (\neg p \lor \neg r, \infty), (\neg q \lor \neg r : \infty) \right)
\]

- cost = 0
- MUS = \{ p, q, \neg p \lor \neg q \}
- cost = 1
- extend with two new variables \( a_1 \) and \( a_2 \)

\[
\text{MaxSat} \left( (p \lor a_1 : 1), (q \lor a_2 : 1), (r : 1), \\
(\neg p \lor \neg q : \infty), (\neg p \lor \neg r : \infty), (\neg q \lor \neg r : \infty), \\
(a_1 + a_2 = 1 : \infty) \right)
\]

- MUS = \{ p \lor a_1, q \lor a_2, r, \\
\neg p \lor \neg q, \neg p \lor \neg r, \neg q \lor \neg r, \\
a_1 + a_2 = 1 \}
- cost = 2
- extend with three new variables \( b_1, b_2, \) and \( b_3 \)

\[
\text{MaxSAT} \left( (p \lor a_1 \lor b_1 : 1), (q \lor a_2 \lor b_2 : 1), (r \lor b_3 : 1), \\
(\neg p \lor \neg q : \infty), (\neg p \lor \neg r : \infty), (\neg q \lor \neg r : \infty), \\
(a_1 + a_2 = 1 : \infty), (b_1 + b_2 + b_3 = 1 : \infty) \right)
\]

- soft and hard clauses are all satisfiable \( \rightarrow \) the algorithm terminate with cost = .
Algorithm 1 Fu and Malik MaxSAT algorithm

Input: $\phi = \{(C_1, 1), \ldots, (C_n, 1), (D_1, \infty), \ldots, (D_n, \infty)\}$

1: if $\text{SAT}(D_1, \ldots, D_m) = \text{Unsat}$ then return $(\infty, \emptyset)$
2: end if
3: cost $\leftarrow$ 0
4: $s \leftarrow 0$
5: while True do
6: $(st, \phi_c) \leftarrow \text{SAT}(C_1, \ldots, C_n, D_1, \ldots, D_m)$
7: if $st = \text{Satisfiable}$ then return $(\text{cost}, \phi)$
8: end if
9: $s \leftarrow s + 1$
10: $A_s = \emptyset$
11: for $C_i \in \phi_c$ with $w_i = 1$ do
12: $b_i^s \leftarrow$ new propositional variable
13: $\phi \leftarrow \phi \setminus \{(C_i, 1)\} \cup \{(C_i \lor b_i^s, 1)\}$
14: $A_s = A_s \cup \{i\}$
15: end for
16: $\phi \leftarrow \phi \cup \left\{(C, \infty) \mid C \in \text{CNF} \left(\sum_{i \in A_s} b_i^s = 1\right)\right\}$
17: $\text{cost} \leftarrow \text{cost} + 1$
18: end while
Consider the pigeon-hole problem. There are 5 pigeons and one hole and no two pigeons can go to the same hole. Suppose that we prefer the models in which a pigeon goes indeed to a hole. How can we formulate this simple problem in MaxSat?

### Solution

- $x_i$ is a propositional variable that represents the fact that the $i$-th pigeon goes in a hole
- no two pigeon can occupy the hole is a strong constraint:

$$\left( \neg x_1 \lor \neg x_2, \infty \right), \ldots, \left( \neg x_4 \lor \neg x_5, \infty \right)$$

- we prefer the situation in which at least one pigeon occupy the hole:

$$\left( x_1, 1 \right), \left( x_2, 1 \right), \left( x_3, 1 \right), \left( x_4, 1 \right), \left( x_5, 1 \right)$$
FM algorithm on the Pigeon and hole pb.

input \( \phi = \{(x_1, 1), \ldots, (x_5, 1), (\neg x_1, \neg x_2, \infty), \ldots, (\neg x_4, \neg x_5, \infty)\} \)

2,3 \(\text{cost} = 0, s = 0,\)

6 The unweighted version of \(\phi\), i.e., \(\{(x_1), \ldots, (x_5), (\neg x_1, \neg x_2), \ldots, (\neg x_4, \neg x_5)\}\) is not satisfiable and a minimal unsatisfiable subset is \(\{(x_1), (x_2), (\neg x_1, \neg x_2)\}\)

11,15 We introduce two new variables \(b^1_1\) and \(b^1_2\) corresponding to the weighted clauses \((x_1, 1)\) and \((x_2, 1)\), obtaining:

\[
\phi = \{(x_1, b^1_1, 1), (x_2, b^1_2, 1),
(x_3, 1), (x_4, 1), (x_5, 1), (\neg x_1, \neg x_2, \infty), \ldots, (\neg x_4, \neg x_5, \infty),
(\neg b^1_1, \neg b^1_2, \infty), (b^1_1, b^1_2, \infty)\}
\]

17 \(\text{cost} = 1\)

6 the unweighted version of \(\phi\) is not satisfiable and a minimal unsat subset is \(\{(x_3), (x_4), (\neg x_3, \neg x_4)\}\)

11,15 We introduce two new variables \(b^2_1\) and \(b^2_2\) corresponding to the weighted clauses \((x_3, 1)\) and \((x_4, 1)\) obtaining:

\[
\phi = \{(x_1, b^1_1, 1), (x_2, b^1_2, 1),
\{(x_3, b^2_1, 1), (x_4, b^2_2, 1),
(x_5, 1), (\neg x_1, \neg x_2, \infty), \ldots, (\neg x_4, x_5, \infty),
(\neg b^1_1, \neg b^1_2, \infty), (b^1_1, b^1_2, \infty)\}
(\neg b^2_1, \neg b^2_2, \infty), (b^2_1, b^2_2, \infty)\}
\]

17 \(\text{cost} = 2\)
the unweighted version of $\phi$ is not satisfiable and a minimal unsat subset is
$$(x_1, b_1^1), (x_2, b_2^1), (\neg b_1^1, \neg b_2^1)(x_5), (\neg x_1, \neg x_5), (\neg x_2, \neg x_5)$$

We introduce three new variables $b_1^3, b_2^3$ and $b_3^3$ corresponding to the weighted clauses
$$(x_1, b_1^1, 1), (x_2, b_2^1, 1), (x_5, 1)$$

obtaining:

$$\phi = \{(x_1, b_1^1 b_1^3, 1), (x_2, b_2^1 b_2^3, 1),$$
$$(x_3, b_1^2, 1), (x_4, b_2^2, 1),$$
$$(x_5, b_3^3, 1),$$
$$(\neg x_1, \neg x_2, \infty), \ldots, (\neg x_4, x_5, \infty),$$
$$(\neg b_1^1, \neg b_2^1, \infty), (b_1^1, b_2^1, \infty)\}$$
$$(\neg b_1^2, \neg b_2^2, \infty), (b_1^2, b_2^2, \infty)\}$$
$$(\neg b_3^3, \infty), (\neg b_1^3, \neg b_2^3, \infty), (\neg b_2^3, \neg b_3^3, \infty), (b_1^3, b_2^3, b_3^3, \infty)\}$$

$$cost = 3$$
The unweighted version of $\phi$ is not satisfiable and a minimal unsat subset contains all the clauses of $\phi$.

We introduce 5 new variables $b^4_1 \ldots b^4_5$ corresponding to all the weighted clauses of $\phi$, obtaining:

$$\phi = \{ (x_1, b^1_1 b^3_1, b^4_1, 1), (x_2, b^1_2, b^3_2, b^4_2, 1),
\{ (x_3, b^2_1, b^4_3, 1), (x_4, b^2_2, b^4_4, 1),
(x_5, b^3_3, b^4_5, 1),
(\neg x_1, \neg x_2, \infty), \ldots, (\neg x_4, x_5, \infty),
(\neg b^1_1, \neg b^1_2, \infty), (b^1_1, b^1_2, \infty) \}
(\neg b^2_1, \neg b^2_2, \infty), (b^2_1, b^2_2, \infty) \}
(\neg b^3_3, \neg b^3_3, \infty), (\neg b^1_2, \neg b^1_3, \infty), (\neg b^3_2, \neg b^3_3, \infty), (b^3_1, b^2_2, b^3_3, \infty) \}
(\neg b^4_1, \neg b^4_2, \infty), (\neg b^4_1, \neg b^4_3, \infty), \ldots, (\neg b^4_4, \neg b^4_5, \infty), (b^4_1, b^4_2, b^4_3, b^4_4, b^4_5, \infty) \}$$

$\text{cost} = 4$

The unweighted version of $\phi$ is now satisfiable, e.g., with $b^1_1 = True, b^3_2 = True, b^2_1 = True, b^4_4 = True, x_5 = True$ and all the other variables set to False.

The FM algorithm terminates and returns $\text{cost}=4$ and the $\phi$ shown above.
Fu and Malik algorithm for MaxSAT

Detailed description of the algorithm is provided in

MaxSat in Python

- an implementation of sat based MaxSAT algorithms are available in the python library called PySAT library.

In addition to the FM algorithm the library proposes a more recent implementation of a Sat Based MaxSAT algorithm, called RC2.\(^2\)

RC2 (as MF) uses an extended version of \texttt{dimacs} representation of weighted cnf, where every clause is preceded by its weight. and

\begin{verbatim}
  p wcnf 3 6 4
  1 1 0
  1 2 0
  1 3 0
  4 -1 -2 0
  4 -1 -3 0
  4 -2 -3 0
\end{verbatim}

Infinite weights are represented with a weight equal to \(\sum w_i + 1\), in the above example the infinite weight is represented by a 4.

\(^2\) [ignatiev2019rc2]
Problem

Show that MaxSAT with negative weights can be transformed in a MaxSAT problem with only positive weights.

Suggestion: Notice that

\[
\text{cost}_\phi(I) = \text{cost}_\phi\{c,-w\}(I) - w \cdot I(\neg C)
\]

\[
= \text{cost}_\phi\{c,-w\}(I) + w \cdot (1 - I(\neg\neg C))
\]

\[
= \text{cost}_\phi\{c,-w\}(I) - w + w \cdot I(\neg\neg C)
\]

\[
= \text{cost}_\phi(c,-w) \cup (\neg c,w)(I) - w
\]
MaxCut problem

Let $G = (V, E)$ be an undirected graph. A cut is a partition of the vertices in $V$ into two disjoint subsets $S$ and $T$. Any edge $(u, v) \in E$ with $u \in S$ and $v \in T$ is said to be crossing the cut, and is a cutting edge. The size of the cut is the number of cutting edges. A maximum cut (MaxCut) is then defined as a cut of $G$ of maximum size.
Solving Maximum cut problem with MaxSAT

Example

Soft clauses with weight $= 1$

\[
\begin{align*}
    x_1 \lor x_2 & \quad \neg x_1 \lor \neg x_2 & \quad x_1 \lor x_3 & \quad \neg x_1 \lor \neg x_3 \\
    x_1 \lor x_4 & \quad \neg x_1 \lor \neg x_4 & \quad x_1 \lor x_5 & \quad \neg x_1 \lor \neg x_5 \\
    x_2 \lor x_3 & \quad \neg x_2 \lor \neg x_3 & \quad x_3 \lor x_4 & \quad \neg x_3 \lor \neg x_4 \\
    x_4 \lor x_5 & \quad \neg x_4 \lor \neg x_5
\end{align*}
\]
Rectangular bin packing

Example (Rectangular bin packing)

We are given a set of $n$ rectangular pieces of different size which must be placed in a finite rectangular bin of height $H$ and width $W$. We have to locate as much rectangular pieces as possible in the bin taking into account the size of the pieces. That is, to maximize the sum of the sizes of the located pieces in the bin.
Example (Rectangular bin packing in MaxSat)

**Propositional variables:** Let \(1, \ldots, K\) be the set of rectangles. the \(k\)-th rectangle has dimension \(w_k \times h_k\) for \(w_k\) and \(h_k\) integers. For every \(k\) we have the propositional variables \(x_k\) The “\(k\)-th rectangle is placed in the bin”, \(r^k_i\) and \(c^k_i\) for “The left-upper corner of the \(k\)-th rectangle is in position \((i, j)\);
**Problem**

Minimum Vertex Cover: A vertex cover $C$ of an undirected graph $G = (V, E)$ is a subset of $V$ such that for all $(u, v) \in E$, either $u \in C$ or $v \in C$ or both $u, v \in C$. $C$ is minimal if for every other vertex cover $C'$, $|C'| > |C|$.  

Encode the problem of finding one minimal vertex cover in MaxSAT.

\[
\{v_2, v_2, v_4\} \quad \text{is a vertex cover} \\
\{v_1\} \quad \text{is a minimal vertex cover}
\]
Exercises

Problem

Maximum Weighted Clique A weighted undirected graph \( G \) is a triple \((V, E, w)\) where \((V, E)\) is an undirected graph and \(w: V \to \mathbb{R}^+\). A clique on \( C \) is a set of vertexes such that for every pair \( u, v \in C \), \((u, v)\) is an edge i.e., \((u, v) \in E\). The weighted maximum clique problem is the problem of finding a clique \( C \) with maximum total weight, i.e., that maximizes \( \sum_{v \in C} w(c) \).

Encode the problem of finding one minimal vertex cover in MaxSAT.

\[
\begin{align*}
\{v_1, v_4, v_5, v_7\} & \quad \text{Is a clique with weight = 11} \\
\{v_2, v_5, v_6\} & \quad \text{Is a maximal clique with weight = 12} \\
\{v_2, v_8\} & \quad \text{Is a maximal clique with weight = 12}
\end{align*}
\]
Problem

MaxSAT equivalence Prove that the problem

\[
\text{MaxSAT}(((A : w_1), (p; w_2), (D : \infty)))
\]  \hspace{1cm} (3)

with \( w_1 \leq w_2 \) is equivalent to

\[
\text{MaxSAT}((A|_P, w_1), (p : w_2 - w_1), (D, \infty))
\]  \hspace{1cm} (4)
Exercises

Problem

MaxSAT equivalence Prove that the problem

\[ \text{MaxSAT}((A : w), (D : \infty)) \]  

with \( A = a_1 \lor \cdots \lor a_n \)

\[ \text{MaxSAT} \left( (a : w), (D : \infty), (\neg a \lor A, \infty), (\neg a_1 \lor a, \infty), \ldots, (\neg a_n \lor a : \infty) \right) \]

Solution

Hint Show that any solution of (5) is a solution of (6) and vice versa. \( \square \)
Problem

Scheduling to minimize lateness A single resource is available to process jobs (for instance a printer in an office, a big crane in a building site, etc.). \( n \) jobs \( J_1, \ldots, J_n \) are to be processed by the resource. Once a job starts, it cannot be interrupted. Processing jobs starts at time 0. Each job \( J_i \) has a deadline \( D_i \) and processing time \( p_i \). We need to schedule the jobs so that the lateness \( \max(0, F_i - D_i) \) the difference between the finishing time and deadline will be minimized.
A pseudo boolean function is any function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, which maps 0/1 $n$-vectors into real numbers. A pseudo boolean function can be uniquely represented in the form

$$f(x_1, \ldots, x_n) = \sum_{S \subseteq \{1, \ldots, n\}} a_S \prod_{i \in S} x_i$$

where $a_S \in \mathbb{R}$ is a real number.

Encode the problem of optimising a generic pseudo boolean function $f(x_1, \ldots, x_n)$ in MaxSAT.
**Solution**

An example of pseudo boolean function is the following:

\[ f(x_1, x_2, x_3) = 3x_1 + 10x_2x_3 - 2x_1x_3 + 5 \]

First notice that for the optimization we can ignore the constant term 5. For every non zero coefficient introduce a new propositional variable. In this case \( a, b, \) and \( c \) and define \( a \equiv x_1, b \equiv x_2 \land x_3 \) and \( c = x_1 \land x_3 \). Now you can optimize

\[ f'(a, b, c) = 3a + 10b - 2c \]

with the constraints: \( a = x_1, b = x_2 \land x_3 \) and \( c = x_1 \land x_3 \). In MaxSAT:

\[
\text{MaxSat}
\begin{pmatrix}
(a : 3), & (\neg a \lor x_1 : \infty), (\neg x_1 \lor a : \infty), \\
(b : 10), & (\neg b \lor x_2 : \infty), (\neg b \lor x_3 : \infty), (\neg x_2 \lor \neg x_3 \lor b : \infty), \\
(c : -2) & (\neg c \lor x_1 : \infty), (\neg c \lor x_3 : \infty), (\neg x_1 \lor \neg x_3 \lor c : \infty)
\end{pmatrix}
\]

Transform negative weights into positive by replacing \( c : -2 \) with \( \neg c : 2 \). \( \Box \)
Optimal correlation clustering

Problem

Optimal correlation clustering Given a set of $n$ points $V = \{v_1, \ldots, v_n\}$ and a symmetric similarity function $s : V \times V \to \{0, 1\}$ (such that $s(v_i, v_j) = 1$ (resp. $0$) means that $v_i$ is similar (resp. dissimilar) to $v_j$), the problem of optimal correlation clustering is the problem of partitioning $V$ in a set of clusters $C = C_1, \ldots, C_k$ for some (unknown) $k \geq 1$ such that the global correlation $G(C)$ is minimized:

$$G(C) = \sum_{v_i \neq v_j \in V \atop cl(v_i) = cl(v_j)} (1 - s(v_i, v_j)) + \sum_{v_i \neq v_j \in V \atop cl(v_i) \neq cl(v_j)} s(v_i, v_j)$$

where $cl(v) = i$ means that $v \in C_i$. 

Luciano Serafini (Fondazione Bruno Kessler)  Knowledge Representation and Learning  June 15, 2023  54 / 56
Optimal correlation clustering

**Solution** For every $i < j \in \{1, \ldots, n\}$ $x_{ij}$ means that $v_i$ and $v_j$ belong to the same cluster.

**Hard clauses** for every $i < j < k$

\[ x_{ij} \land x_{jk} \rightarrow x_{ik} \quad \quad \quad x_{ij} \land x_{ik} \rightarrow x_{jk} \]

**Soft clauses** for every $i < j$

\[ x_{ij} : 1 \quad \quad \quad \text{If } s(v_i, v_j) = 1 \]
\[ \neg x_{ij} : 1 \quad \quad \quad \text{If } s(v_i, v_j) = 0 \]


