Knowledge Representation and Learning

5. Maximum Satisfiability

Luciano Serafini

Fondazione Bruno Kessler

June 15, 2023

Not all interpretation are the same

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

$$\langle \mathrm{models}(\boldsymbol{C}), \prec \rangle$$

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:

Person	gender	M	K	V	H
Alice	f	1	1	1	1
Bea	f	3	0	2	0
Celine	f	1	3	0	0
Dania	f	1	0	0	3
Enrico	m	1	0	3	0
Felix	m	2	1	0	0

Example (Team building (cont'd))

Formalization in SAT:

$$
\begin{array}{rl}
M \wedge K \wedge V \wedge H & \text{You want all 4 competences} \\
A + B + C + D + E + F = 4 & \text{You have to select 4 people}
\end{array}
$$

$$
\begin{aligned}
& M \rightarrow A \vee B \vee C \vee D \vee E \vee F \\
& K \rightarrow A \vee C \vee F \\
& V \rightarrow A \vee B \vee E \\
& H \rightarrow A \vee D
\end{aligned}
$$

Example (Team building (cont'd))

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

$$
\begin{aligned}
Team_1 &= \{A, B, C, D, M, K, V, H\} \\
Team_2 &= \{A, B, C, E, M, K, V, H\} \\
Team_3 &= \{A, B, C, F, M, K, V, H\} \\
Team_4 &= \{A, B, D, E, M, K, V, H\} \\
Team_5 &= \{A, B, D, F, M, K, V, H\} \\
Team_6 &= \{A, B, E, F, M, K, V, H\} \\
Team_7 &= \{A, C, D, E, M, K, V, H\} \\
Team_9 &= \{A, C, D, F, M, K, V, H\} \\
Team_9 &= \{A, C, E, F, M, K, V, H\} \\
Team_{10} &= \{A, D, E, F, M, K, V, H\} \\
Team_{11} &= \{B, C, D, E, M, K, V, H\} \\
Team_{12} &= \{B, C, D, F, M, K, V, H\} \\
Team_{13} &= \{B, D, E, F, M, K, V, H\} \\
Team_{14} &= \{C, D, E, F, M, K, V, H\}
\end{aligned}
$$

Team building (cont'd)

Example (Team building (cont'd))

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

- C1 $gb = 1 - \frac{|\#\text{male} - \#\text{female}|}{4}$
- C2 $cpt(Y) = \sum_{X \in team} level\_of\_cpt(X, Y)$

Team building (cont'd)

Team	gb	$cmp(M)$	$cmp(K)$	$cmp(V)$	$cmp(H)$
$Team_1 = \{A, B, C, D\}$	0.0	6	4	3	4
$Team_2 = \{A, B, C, E\}$	0.5	6	4	6	1
$Team_3 = \{A, B, C, F\}$	0.5	7	5	3	1
$Team_4 = \{A, B, D, E\}$	0.5	6	1	6	4
$Team_5 = \{A, B, D, F\}$	0.5	7	2	3	4
$Team_6 = \{A, B, E, F\}$	1.0	7	2	6	1
$Team_7 = \{A, C, D, E\}$	0.5	4	4	4	4
$Team_9 = \{A, C, D, F\}$	0.5	5	5	1	4
$Team_9 = \{A, C, E, F\}$	1.0	5	5	4	1
$Team_{10} = \{A, D, E, F\}$	1.0	5	2	4	4
$Team_{11} = \{B, C, D, E\}$	0.5	6	3	5	3
$Team_{12} = \{B, C, D, F\}$	0.5	7	4	2	3
$Team_{13} = \{B, D, E, F\}$	1.0	7	1	5	3
$Team_{14} = \{C, D, E, F\}$	1.0	5	4	3	3

Weighted formulas

- A general method to express preference relation between iterpretations is via weighted formulas. I.e.,

$$w : \phi \tag{1}$$

  $w \in \mathbb{R}$ is the weight of the propositional formula ϕ

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

$$\mathcal{I} \preceq \mathcal{J} \text{ if and only if } w_F(\mathcal{I}) \leq w_F(J) \tag{2}$$

  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:

$1 : (x \wedge y)$ for every $x, y \in \{A, B, C, D, E, F\}$ such that x is a male and y 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 exercize).

Example of weighted formulas

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

$ExpertLevel(M, x) : x$	for every $x \in \{A, B, C, D, E, F\}$ where $ExpertLevel(M, x)$ is the expert level of x 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 $\mathcal{I}, \mathcal{J}$

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

Two sets of weighted formulas F_1 and F_2 are opposite if

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

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. $F \cup \{w : \phi \wedge \psi\}$ *is not equivalent to* $F \cup \{w : \phi,\ w : \psi\}$

$F = \{3 : p \wedge q,\ 1 : \neg p,\ 1 : \neg q\}$

$$
\begin{aligned}
w(p, q) &= 3 \\
w(p, \neg q) &= 1 \\
w(\neg p, q) &= 1 \\
w(\neg p, \neg q) &= 2
\end{aligned}
$$

$2 + F = \{5 : \neg p \vee q,\ 3 : \neg p,\ 3 : \neg q\}$

$$
\begin{aligned}
w(p, q) &= 5 \\
w(p, \neg q) &= 3 \\
w(\neg p, q) &= 3 \\
w(\neg p, \neg q) &= 6
\end{aligned}
$$

$F = \{1.5 : p \wedge q,\ 1 : \neg p,\ 1 : \neg q\}$

$$
\begin{aligned}
w(p, q) &= 1.5 \\
w(p, \neg q) &= 1 \\
w(\neg p, q) &= 1 \\
w(\neg p, \neg q) &= 2
\end{aligned}
$$

$F' = \{1.5 : p,\ 1.5 : q,\ 1 : \neg p,\ 1 : \neg q\}$

$$
\begin{aligned}
w(p, q) &= 3 \\
w(p, \neg q) &= 2.5 \\
w(\neg p, q) &= 2.5 \\
w(\neg p, \neg q) &= 2
\end{aligned}
$$

MaxSAT - Maximum satisfiability

Definition (MaxSAT)

Given a set of weighted clauses $w_1 : C_1, \dots, w_n : C_n$ and a set of standard clauses $D_1, \dots, D_m$, 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 satisied 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.

Remark

- First of all let us show that maximizing $\sum_{i=1}^{n} w_i \cdot \mathcal{I}(C_i)$ is the same as minimizing $\sum_{i=1}^{n} w_i \cdot \mathcal{I}(\neg C_i)$. Since $\mathcal{I}(\neg C) = 0$ iff $\mathcal{I}(C) = 1$ we have that:

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

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

MaxSAT - Maximum satisfiability

- 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 (∞) then at least one hard constraint is not satisfied, and therefore we say the the entire problem ϕ is unsatisfiable.
- in practice ∞ is replaced with $\sum_{i=1} w_i + 1$

MaxSAT application to optimization problems

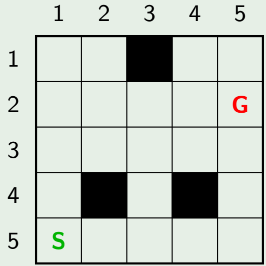
Most real-world problems involve an optimization component
Examples:

- Find a shortest path/plan/execution/ . . . to a goal state
- Find a smallest explanation of a certain phenomena in terms of causes 007060336298007060336298
- 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 enter in the black squares

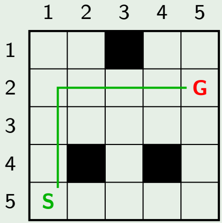


Encoding in propositional logic

- 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

Example



Hard Constraints

- S and G are visited $c_{51} \wedge c_{25}$;
- S and G have a single visited neighbour

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

$N(i,j) = \{(i'j') \mid 1 \leq i', j' \leq 5, \; ||(i,j) - (i,' j')||_1 = 1\}$, 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:

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

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

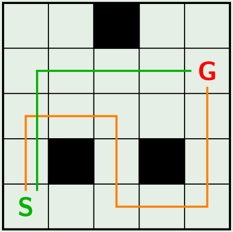
Soft Constraints

- visit the minimum number of cells: $-1 : c_{ij}$, (or equivalently $1 : \neg c_{ij}$)

¿

Example - Shortest Path

Example



- Every interpretastion that satisfies the hard constraints correspond to a (set of paths) from S to G;
- the weigth 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(ϕ : CNF, ψ : weighted CNF, $\mathcal{I}$: Partial assignment)

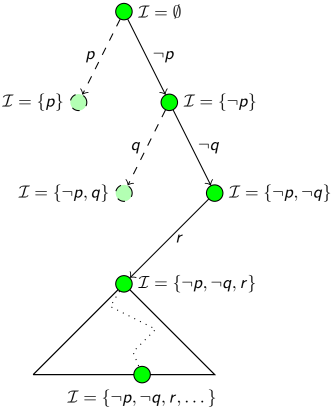
$\mathcal{I}, \phi \leftarrow$ UnitPropagation$(\mathcal{I}, \phi)$
$\psi \leftarrow \psi|_{\mathcal{I}}$
if $\{\} \in \phi$ **then**
  return $\mathcal{I}, \infty$
end if
if $\phi = \{\}$ and ψ contains only empty weighted clauses **then**
  return $\mathcal{I}, \sum_{(w:D)\in\psi} w$
else
  select a l from some clause in ϕ or in ψ
  $\mathcal{I}, c \leftarrow$ MaxSAT-DPLL$(\phi|_l, \psi|_l, \mathcal{I} \cup \{l\})$
  $\mathcal{I}', c' \leftarrow$ MaxSAT-DPLL$(\phi|_{\bar{l}}, \psi|_{\bar{l}}, \mathcal{I} \cup \{\bar{l}\})$
  if $c \leq c'$ **then**
    return $\mathcal{I}, c$
  else
    return $\mathcal{I}', c'$
  end if
end if

Early stop seearch using Lower/Upper Bound

- MAXSAT-DPLL performs an exhaustive search on all the models of ϕ and evaluates their cost on ψ;
- not very efficient
- possible imporvement:
  - remember the best model found so far $\mathcal{I}_{UB}$ and its cost UB (UB stands for upper-bound)
  - we are not insterested in models with cost $> UB$.
  - compute the lower bound (LB) of the cost of the current partial assignment $\mathcal{I}$,
  - if $LB > UB$ stop expanding $\mathcal{I}$, and backtrack

Branch and bound

$p \vee q \vee r \vee s \vee t : \infty \qquad \neg p : 3 \qquad \neg q : 2 \qquad \neg r : 4$



- UB = cost of the best solution so far;
- LB minimum cost achievable under the node;
- Abandone the subtree of a node if $LB > UB$ (no solution under this node);

Branch and Bound

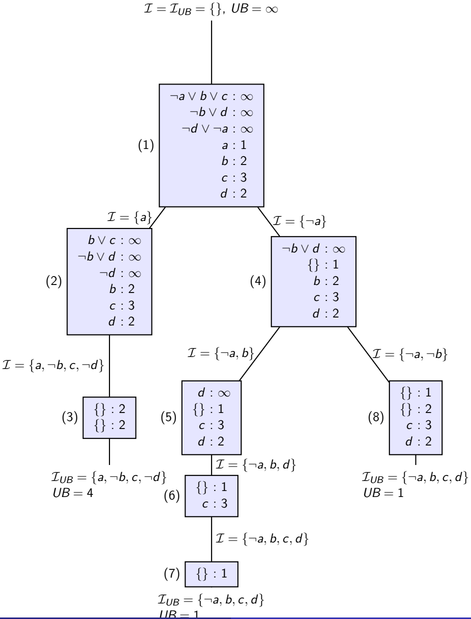
B&B(ϕ: CNF, ψ: Weighted CNF, $\mathcal{I}$: Partial assignment $\mathcal{I}_{UB}$: Best previously found solution, UB: Cost of $\mathcal{I}_{UB}$)

$\mathcal{I}, \phi \leftarrow \textsc{UnitPropagation}(\mathcal{I}, \phi)$
$\psi \leftarrow \psi|_{\mathcal{I}}$
if $\{\} \in \phi$ or $UB \leq \sum_{w:\{\}\in\psi} w$ **then**
    return $\mathcal{I}_{UB}, UB$
end if
if $\phi = \{\}$ and ψ contains only empty weighted clauses **then**
    return $\mathcal{I}, \sum_{(w:D)\in\psi} w$
else
    select a l from some clause in ϕ or in ψ
    $\mathcal{I}, c \leftarrow \mathrm{B\&B}(\phi|_l, \psi|_l, \mathcal{I} \cup \{l\}, \mathcal{I}_{UB}, UB)$
    $\mathcal{I}', c' \leftarrow \mathrm{B\&B}(\phi|_{\bar{l}}, \psi|_{\bar{l}}, \mathcal{I} \cup \{\bar{l}\}, \mathcal{I}_{UB}, UB)$
    if $c \leq c'$ **then**
        return $\mathcal{I}, c$
    else
        return $\mathcal{I}', c'$
    end if
end if

Example

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

$$(a \to b \vee c : \infty)$$
$$(b \to d : \infty)$$
$$(d \to \neg a : \infty)$$
$$(a : 1)$$
$$(b : 2)$$
$$(c : 3)$$
$$(d : 2)$$



Branch and Bound Summary

- 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.

SAT-based MaxSAT

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

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

is satisfiable.

Remark

Notice that k may range from 0 to $\sum_{i=1}^{n} w_i$. Therefore the naïve 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

$$\bigwedge_{\substack{B \subseteq \{b_1 \dots, b_n\} \\ |B| = k+1}} \bigvee_{b_i \in B} \neg b_i$$

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

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

[1] More efficient encodings are also possible

Fu and Malik algorithm for MaxSAT

- The Fu and Malik algorithm for MaxSAT uses SAT as an oracle (i.e., it calls SAT as a subrutine)
- $SAT(C_1, \dots, C_N)$ can return one of the following outputs:
  - Satisfiable: if $C_1, \dots, C_n$ are satisfiable
  - (Unsatisfiable,$\mathbb{C}$) for some set of clauses $\mathbb{C}$ that is a minimal subset of $\{C_1, \dots, C_n\}$ which are not satisfiable.

Fu and Malik algorithm for MaxSAT - intuition

1. $MaxSAT((A:1),(B:1),(C:1),(D:\infty),(E:\infty))$
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 $(A \vee a:1)$,
   change $(B:1)$ into $(B \vee b:1)$.
   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. $MaxSAT\left(\begin{array}{c}(A \vee a:1),(B \vee b:1) \\ (C:1)\end{array}, \begin{array}{c}(D:\infty),(E:\infty) \\ (a+b=1:\infty)\end{array}\right)$
9. go to 1

Example

$$MaxSAT\left(\begin{array}{l}(p:1),\ (q,1),\ (r:1),\\ (\neg p\vee\neg q,\infty),\ (\neg p\vee\neg r,\infty),\ (\neg q\vee\neg r:\infty)\end{array}\right)$$

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

$$MaxSat\left(\begin{array}{l}(p\vee a_1:1),\ (q\vee a_2:1),\ (r:1),\\ (\neg p\vee\neg q:\infty),\ (\neg p\vee\neg r:\infty),\ (\neg q\vee\neg r:\infty),\\ (a_1+a_2=1:\infty)\end{array}\right)$$

- $MUS=\left\{\begin{array}{l}p\vee a_1,\ q\vee a_2,\ r,\\ \neg p\vee\neg q,\ \neg p\vee\neg r,\ \neg q\vee\neg r,\\ a_1+a_2=1\end{array}\right\}$
- cost $=2$
- extend with three new variables b_1, b_2, and b_3

$$MaxSAT\left(\begin{array}{l}(p\vee a_1\vee b_1:1),\ (q\vee a_2\vee b_2:1),\ (r\vee b_3:1),\\ (\neg p\vee\neg q):\infty),\ (\neg p\vee\neg r:\infty),\ (\neg q\vee\neg r:\infty)\\ (a_1+a_2=1:\infty),\ (b_1+b_2+b_3=1:\infty)\end{array}\right)$$

- soft and hard clauses are all satisfiable $\rightarrow$ the algorithm terrminate with cost $=$.

Fu and Malik algorithm for MaxSAT

Algorithm 1 Fu and Malik MaxSAT algorithm

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

if $\mathrm{SAT}(D_1, \ldots, D_m) = \mathrm{Unsat}$ **then return** $(\infty, \emptyset))$
end if
$cost \leftarrow 0$
$s \leftarrow 0$
while True **do**
  $(st, \phi_c) \leftarrow SAT(C_1, \ldots, C_n, D_1, \ldots, D_m)$
  if $st = Satisfiable$ **then return** $(cost, \phi)$
  end if
  $s \leftarrow s + 1$
  $A_s = \emptyset$
  for $C_i \in \phi_c$ with $w_i = 1$ **do**
    $b_i^s \leftarrow$ new propositional variable
    $\phi \leftarrow \phi \setminus \{(C_i, 1)\} \cup \{(C_i \vee b_i^s, 1)\}$
    $A_s = A_s \cup \{i\}$
  end for
  $\phi \leftarrow \phi \cup \left\{ (C, \infty) \,\middle|\, C \in CNF\left(\sum_{i \in A_s} b_i^s = 1\right) \right\}$
  $cost \leftarrow cost + 1$
end while

Example

Problem

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:

$$(\neg x_1 \vee \neg x_2, \infty), \ldots, (\neg x_4 \vee \neg x_5, \infty)$$

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

$$(x_1, 1), (x_2, 1), (x_3, 1), (x_4, 1), (x_5, 1)$$

□

FM algorithm on the Pigeon and hole pb.

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

2,3 $cost = 0$, $s = 0$,

6 The unweighted version of ϕ, i.e., $\{(x_1), \dots, (x_5), (\neg x_1, \neg x_2), \dots, (\neg x_4, \neg x_5)\}$ 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_2^1 corresponding to the weighted clauses $(x_1, 1)$ and $(x_2, 1)$, obtaining:

$$
\begin{aligned}
\phi = \{&(x_1, {b^1}_1, 1), (x_2, b_2^1, 1), \\
&(x_3, 1), (x_4, 1), (x_5, 1), (\neg x_1, \neg x_2, \infty), \dots, (\neg x_4, \neg x_5, \infty), \\
&(\neg b_1^1, \neg b_2^1, \infty), (b_1^1, b_2^1, \infty)\}
\end{aligned}
$$

17 $cost = 1$

6 the unweighted version of ϕ 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_1^2 and b_2^2 corresponding to the weighted clauses $(x_3, 1)$ and $(x_4, 1)$ obtaining:

$$
\begin{aligned}
\phi = &\{(x_1, {b^1}_1, 1), (x_2, b_2^1, 1), \\
&\{(x_3, b_1^2, 1), (x_4, b_2^2, 1), \\
&(x_5, 1), (\neg x_1, \neg x_2, \infty), \dots, \neg(x_4, x_5, \infty), \\
&(\neg b_1^1, \neg b_2^1, \infty), (b_1^1, b_2^1, \infty)\} \\
&(\neg b_1^2, \neg b_2^2, \infty), (b_1^2, b_2^2, \infty)\}
\end{aligned}
$$

17 $cost = 2$

FM algorithm on the Pigeon and hole pb.(cont'd)

6 the unweighted version of ϕ is not satisfiable and a minimal unsat subset is $(x_1, b_1^1), (x_2, b_2^1), (\neg b_1^1, \neg b_2^1)(x_5), (\neg x_1, \neg x_5), (\neg x_2, \neg x_5)$

11,15 We introduce tree 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:

$$
\begin{aligned}
\phi = \{&(x_1, b^1{}_1 b_1^3, 1), (x_2, b_2^1, b_2^3, 1),\\
&\{(x_3, b_1^2, 1), (x_4, b_2^2, 1),\\
&(x_5, b_3^3, 1),\\
&(\neg x_1, \neg x_2, \infty), \ldots, \neg(x_4, x_5, \infty),\\
&(\neg b_1^1, \neg b_2^1, \infty), (b_1^1, b_2^1, \infty)\}\\
&(\neg b_1^2, \neg b_2^2, \infty), (b_1^2, b_2^2, \infty)\}\\
&(\neg b_1^3, \neg b_2^3, \infty), (\neg b_1^3, \neg b_3^3, \infty), (\neg b_2^3, \neg b_3^3, \infty), (b_1^3, b_2^3, b_3^3, \infty)\}
\end{aligned}
$$

17 $cost = 3$

FM algorithm on the Pigeon and hole pb.(cont'd)

6 the unweighted version of ϕ is not satisfiable and a minimal unsat subset contains all the clauses of ϕ.

11,15 We introduce 5 new variables $b_1^4 \dots b_5^4$ corresponding to all the weighted clauses of ϕ. obtaining:

$$
\begin{aligned}
\phi = \{&(x_1, b^1{}_1 b_1^3, b_1^4, 1), (x_2, b_2^1, b_2^3, b_2^4, 1), \\
&\{(x_3, b_1^2, b_3^4, 1), (x_4, b_2^2, b_4^4, 1), \\
&(x_5, b_3^3, b_5^4, 1), \\
&(\neg x_1, \neg x_2, \infty), \dots, \neg(x_4, x_5, \infty), \\
&(\neg b_1^1, \neg b_2^1, \infty), (b_1^1, b_2^1, \infty)\} \\
&(\neg b_1^2, \neg b_2^2, \infty), (b_1^2, b_2^2, \infty)\} \\
&(\neg b_1^3, \neg b_2^3, \infty), (\neg b_1^3, \neg b_3^3, \infty), (\neg b_2^3, \neg b_3^3, \infty), (b_1^3, b_2^3, b_3^3, \infty)\} \\
&(\neg b_1^4, \neg b_2^4, \infty), (\neg b_1^4, \neg b_3^4, \infty), \dots, (\neg b_4^4, \neg b_5^4, \infty), (b_1^4, b_2^4, b_3^4, b_4^4, b_5^4, \infty)\}
\end{aligned}
$$

17 *cost* $= 4$

6 The unweighted version of ϕ is now satisfiable, e.g., with $b_1^1 = \mathit{True}$, $b_2^3 = \mathit{True}$, $b_1^2 = \mathit{True}$, $b_4^4 = \mathit{True}$, $x_5 = \mathit{True}$ and all the other variables set to False.

7 The FM algorithm terminates and returns cost=4 and the ϕ shown above.

Fu and Malik algorithm for MaxSAT

Detailed description of the algorithm is provided in

- Section 5 of **ansotegui2013sat**, https://www.sciencedirect.com/science/article/pii/S000437021300012X
- Original paper: **fu2006solving**, https://link.springer.com/chapter/10.1007/11814948_25

MaxSat in Python

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

- In addiiton 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 `dimacs` representation of weighted cnf, where every clause is preceeded by its weight. and

```
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
```

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 whaights can be transformed in a MaxSAT problem with only positive weights.

Suggestion: *Notice that*

$$
\begin{aligned}
cost_{\phi}(\mathcal{I}) &= cost_{\phi\setminus\{C,-w\}}(\mathcal{I}) - w \cdot \mathcal{I}(\neg C) \\
&= cost_{\phi\setminus\{C,-w\}}(\mathcal{I}) + w \cdot (1 - \mathcal{I}(\neg\neg C)) \\
&= cost_{\phi\setminus\{C,-w\}}(\mathcal{I}) - w + w \cdot \mathcal{I}(\neg\neg C) \\
&= cost_{\phi\setminus(C,-w)\cup(\neg C,w)}(\mathcal{I}) - w
\end{aligned}
$$

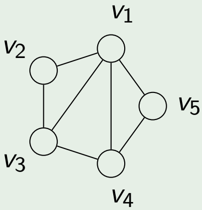
Solving Maximum cut problem with MaxSAT

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



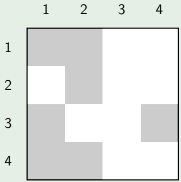
Soft clauses with weight $= 1$

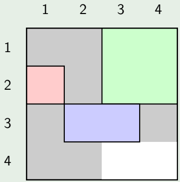
$x_1 \vee x_2$	$\neg x_1 \vee \neg x_2$	$x_1 \vee x_3$	$\neg x_1 \vee \neg x_3$
$x_1 \vee x_4$	$\neg x_1 \vee \neg x_4$	$x_1 \vee x_5$	$\neg x_1 \vee \neg x_5$
$x_2 \vee x_3$	$\neg x_2 \vee \neg x_3$	$x_3 \vee x_4$	$\neg x_3 \vee \neg x_4$
$x_4 \vee x_5$	$\neg x_4 \vee \neg x_5$		

Rectangular bin packing

Example (Rectangular bin packing)

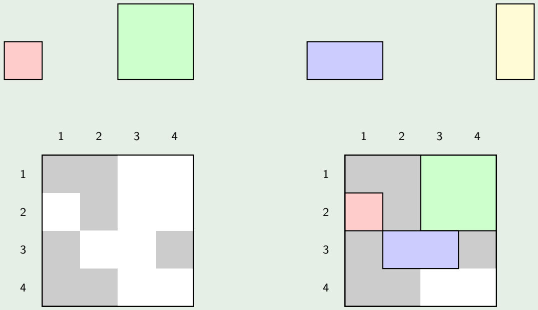
We are given a set of n rectangular pieces of different size which must be placed in a finite rectangular bin of height H and width W. We have to locate as much rectangular pieces as possible in the bin taking into account the size of the pieces. That is, to maximize the sum of the sizes of the located 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_h integers. For every k we have the propositional variables x_k The "k-th rectangle is placed in the bin", r_i^k and c_i^k for "The left-upper corner of the k-th rectangle is in position (i, j);



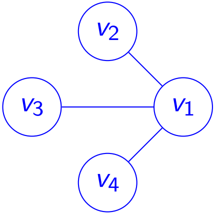
Hard clauses

$H-h_k$ $W-w_k$

Exercizes

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 pe other vertex cover* C', $|C|' > |C|$.
Encode the problem of finding one minimal vertex cover in MaxSAT.

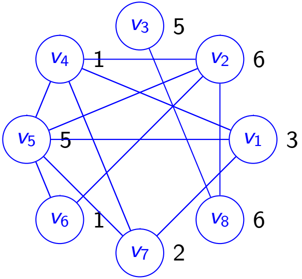


$\{v_2, v_2, v_4\}$	Is a vertex cover
$\{v_1\}$	Is a minimal vertex cover

Exercizes

Problem

Maximum Weighted Clique A weighted undirected graph G si 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.



$\{v_1, v_4, v_5, v_7\}$	Is a clique with weight $= 11$
$\{v_2, v_5, v_6\}$	Is a maximal clique with weight $= 12$
$\{v_2, v_8\}$	Is a maximal clique with weight $= 12$

Exercizes

Problem

MaxSAT equivalence Prove that the problem

$$MaxSAT((A : w_1), (p; w_2), (D : \infty)) \tag{3}$$

with $w_1 \leq w_2$ is equivalent to

$$MaxSAT((A|_P, w_1), (p : w_2 - w_1), (D, \infty) \tag{4}$$

Exercizes

Problem

MaxSAT equivalence Prove that the problem

$$MaxSAT((A : w), (D : \infty)) \tag{5}$$

with $A = a_1 \vee \cdots \vee a_n$

$$MaxSAT\left((a : w), \begin{array}{c} (D : \infty), (\neg a \vee A, \infty), \\ (\neg a_1 \vee a, \infty), \ldots, (\neg a_n \vee a : \infty) \end{array}\right) \tag{6}$$

SolutionHint Show that any solution of (5) is a solution of (6) and viceversa. □

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.*

Pseudo-Boolean Function

Problem

Optimisation of Pseudo-Boolean functions A pseudo boolean function is any function $f : \{0,1\}^n \to \mathbb{R}$, 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.

Pseudo-Boolean Function

SolutionHintAn 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 $a \equiv x_1$, $b \equiv x_2 \wedge x_3$ and $c = x_1 \wedge x_3$. Now you can optimize

$$f'(a, b, c) = 3a + 10b - 2c$$

with the constraints: $a = x_1$, $b = x_2 \wedge x_3$ and $c = x_1 \wedge x_3$. in MaxSAT:

$$MaxSat \left(\begin{array}{ccc} (a:3), & & (\neg a \vee x_1 : \infty), (\neg x_1 \vee a : \infty), \\ (b:10), & , & (\neg b \vee x_2 : \infty), (\neg b \vee x_3 : \infty), (\neg x_2 \vee \neg x_3 \vee b : \infty) \\ (c:-2) & & (\neg c \vee x_1 : \infty), (\neg c \vee x_3, \infty), (\neg x_1 \vee \neg x_3 \vee c : \infty) \end{array} \right)$$

Transform negative weights into positive by replacing $c : -2$ with $\neg c : 2$. $\square$

Optimal correlation clustering

Problem

Optimal correlation clustering Given a set of n points $V = \{v_1, \ldots, v_n\}$ *and a symmetric similarity function* $s : V \times V \rightarrow \{0, 1\}$ *(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 cluster* $\mathbb{C} = C_1, \ldots, C_k$ *for some (unknown)* $k \geq 1$ *such that the global correlation* $G(\mathbb{C})$ *is minimized:*

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

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

Optimal correlation clustering

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

hard clauses for every $i < j < k$

$$x_{ij} \wedge x_{jk} \rightarrow x_{ik} \qquad\qquad x_{ij} \wedge x_{ik} \rightarrow x_{jk}$$

Soft clauses for every $i < j$

$$\begin{array}{rl} x_{ij} : 1 & \text{If } s(v_i, v_j) = 1 \\ \neg x_{ij} : 1 & \text{If } s(v_i, v_j) = 0 \end{array}$$

□

bibliography

Chakraborty, Supratik, Kuldeep S Meel, and Moshe Y Vardi (2021). "Approximate model counting". In: *Handbook of Satisfiability*. IOS Press, pp. 1015–1045.

Colnet, Alexis de and Kuldeep S Meel (2019). "Dual hashing-based algorithms for discrete integration". In: *International Conference on Principles and Practice of Constraint Programming*. Springer, pp. 161–176.

Ermon, Stefano et al. (2013). "Taming the curse of dimensionality: Discrete integration by hashing and optimization". In: *International Conference on Machine Learning*. PMLR, pp. 334–342.

Sang, Tian, Paul Beame, and Henry Kautz (2005). "Solving Bayesian networks by weighted model counting". In: *Proc.AAAI-05*. Vol. 1, pp. 475–482.