Logic for Knowledge Representation, Learning, and Inference

Luciano Serafini

serafini@fbk.eu

Version August 8, 2023

Contents

Chapter 1. Maximum Satisfiability 5
1. Ordering interpretations 7
2. The MaxSAT problem 10
3. MaxSAT exact algorithms 12
4. Solving problems with MaxSAT 21
5. MaxSAT for machine learning 23
6. Exercises 24

Bibliography 33

CHAPTER 1

Maximum Satisfiability

So far we have considered the interpretations of a propositinal language as a plain set. However, in many situations it is important to represent relations between propositional interpretations, i.e, to impose some structure on the set of interpretations of a propositional language. A tipical, and very important example of structure definable on the set of propositional interpretation is the one that states the fact that some interpretations are "better" then others.

EXAMPLE 1.1. *Suppose that you want to build a team of four people to develop a project that requires competences in machine learning (M), knowledge representation (K) vision (V), and human computer interaction (H). You can select the team between 6 people who have following degree of experteese for each of the competence.*

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*

The hard constraints on the formation of the team is that you have to select four people, and each competence should be in the team. These constraints can be expressed in terms of propositional logic formulas. For instance, the fact that you want all the four competences in the team (independently from the competence level) can be formalized by requiring that $M \wedge K \wedge V \wedge H$ is true and by stating which person can provide the various competences (independently from the expertese level) using the following implications:

$$\begin{array}{ll} M & \rightarrow A \vee B \vee C \vee D \vee E \vee F \\ K & \rightarrow A \vee C \vee F \\ V & \rightarrow A \vee B \vee E \\ H & \rightarrow A \vee D \end{array} \tag{1}$$

The constraint about the team size can be expressed by the cardinality constraints, exactly four among $A, \ldots, F$. I.e.,

$$A + B + C + D + E + F = 4 \tag{2}$$

There are many choices that satisfy this constraints, they correspond to the assignments to $A, \ldots, F$ that satisfy the above formulas. All the interpretations that satisfy formulas (1) and (2) are shown in Figure 1 For every interpretation $\mathcal{I}_i$ that satisfy the hard constraint you can extract the corresponding team. However, you would also like to be able to express some preference on the teams as, for instance, you prefer teams with gender and competence balance, but also the higher the total

$$\mathcal{I}_1 = \{A, B, C, D, M, K, V, H\} \qquad \mathcal{I}_8 = \{A, C, D, F, M, K, V, H\}$$
$$\mathcal{I}_2 = \{A, B, C, E, M, K, V, H\} \qquad \mathcal{I}_9 = \{A, C, E, F, M, K, V, H\}$$
$$\mathcal{I}_3 = \{A, B, C, F, M, K, V, H\} \qquad \mathcal{I}_{10} = \{A, D, E, F, M, K, V, H\}$$
$$\mathcal{I}_4 = \{A, B, D, E, M, K, V, H\} \qquad \mathcal{I}_{11} = \{B, C, D, E, M, K, V, H\}$$
$$\mathcal{I}_5 = \{A, B, D, F, M, K, V, H\} \qquad \mathcal{I}_{12} = \{B, C, D, F, M, K, V, H\}$$
$$\mathcal{I}_6 = \{A, B, E, F, M, K, V, H\} \qquad \mathcal{I}_{13} = \{B, D, E, F, M, K, V, H\}$$
$$\mathcal{I}_7 = \{A, C, D, E, M, K, V, H\} \qquad \mathcal{I}_{14} = \{C, D, E, F, M, K, V, H\}$$

FIGURE 1. The models of the formulas (1) and (2), In red the people of the team.

		Competence level			
Team	*gb*	*M*	*K*	*V*	*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

FIGURE 2. Ranking of the teams w.r.t, the different criteria.

amount of each competence the better. In summary you can rank all the different teams that satisfy the hard criteria according to some preference criteria. We can for instance consider the gender balance criteria $gb = 1 - \frac{|\#male - \#female|}{4}$ *and the crieria of the sum of the competence level for each competence. The evaluation of the four teams (interpretations) according to tese 5 criteria are shown in Figure* **??**.

There is no team that maximizes all the criteria, hoever one could decide to give priority to the gender balance, preferring $team_2$ *and* $team_4$ *and then to the uniform distribution among competence. which implies that* $team_2$ *is seleccted.*

EXAMPLE 1.2. *Consider the situation in which you have to build a team of* n *people with a good gender balance. You will prefer teams with gender balance degree, defined as* $b = 1 - \frac{|\#male - \#female|}{n}$ *is close to* 1. *The best option would be a team with an even number of male and female, with* $b = 1$. *Formalizing it in propositional logic, if* p_i *represents the proposition that the* i*-th member of the team is a female,*

you prefer the interpretations in which the fraction of p_i set to true and those set to false are closer. This amount in introducing an order in the set of interpretations as shown in Figure3 following picture that shows how interpretations of $p_1, \ldots, p_4$ (denoted by a sequence of four 0/1) can be orderred according to their preference.

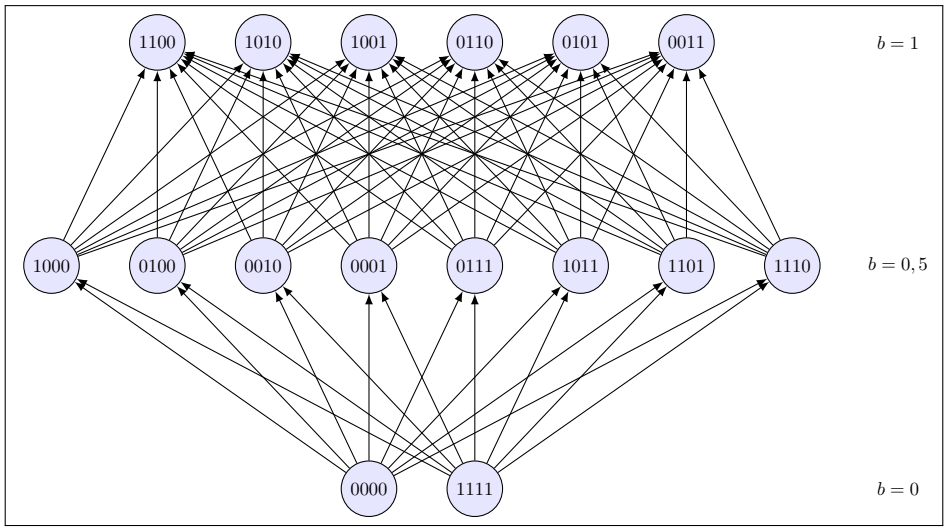


FIGURE 3.

Maximum satisfiability problem focus on this type of relation, which formally correspond to partial orders. In particular maximum satisfiabiliy focus on the problem of finding the "best" interpretation that satisfy a certain set of formulas. However, Before proceeding with the definition of the problem of maximum satisfiability and the relative solution methods let us discuss how it is possible to impose an ordering structure on the set of interpretations of a propositional language.

1. Ordering interpretations

To state that an interpretation is "better" than another, of that we "prefer" an interpretation w.r.t., another we should be able to order the set of interpretations from the less preferred to the most preferred. The mathematical notion that can be used for this aim is the notion of *preorder*

1.1. Partial and total orders. A *preorder* is a structure $(S, \precsim)$ where S is a set and $\precsim$ is a binary relation on S i.e., $\precsim \subseteq S \times S$ that satisfies the following properties: v

Reflexivity:: $s \precsim s$, for all $s \in S$
Transitivity:: $s \precsim t$ and $t \precsim u$ implies that $s \precsim u$.

The preorder is said to be *total* if

Totality:: $s \precsim s$ or $t \precsim s$ for all $s, t \in S$

We say that s and t are *equi-preferrable*, in symbol $s \sim t$ if $a \precsim t$ and $t \precsim s$. Finally we say that t is *strictly preferable* to s, in symbol $s \prec t$ if $s \precsim t$ but not $s \sim t$. One of the simplest way to specify a total preorder on the set $\mathcal{S}$ is by defining a function

$w : \mathcal{S} \to \mathbb{R}$, often called *weight function*, that associates to each element $s \in S$ a real number $w(s)$. The order on S is defined as $s \precsim t$ if and only if $w(s) \leq w(t)$. In the following we will use only total preorders specified by some weight function.

Let $\mathcal{P}$ be a set of propositional variables, and $\mathbb{I}$ be the set of interpretations of $\mathcal{P}$, a *weight function* for $\mathcal{P}$ is a function $w : \mathbb{I} \to \mathbb{R}$, that associates to each interpretation $\mathcal{I}$ of $\mathcal{P}$ a real number $w(\mathcal{I})$. Such a function defines the following total preorder on the models of ϕ

$$\mathcal{I} \precsim \mathcal{J} \text{ if and only if } w(\mathcal{I}) \leq w(\mathcal{J})$$

It is easy to show that this definition satisfies the property that defines a total pre-order (by exercise).

EXAMPLE 1.3. *Consider the example of forming a gender balanced team introduced above. Let $\mathcal{P} = \{p_1, \ldots, p_n\}$. If we define the weight function as:*

$$w(\mathcal{I}) = -\left|\textstyle\sum_i \mathcal{I}(p_i) - \frac{n}{2}\right| \tag{3}$$

we have that $w(\mathcal{I})$ reaches it's maximum equal to 0, when there is an even number of variables set to true and false (if n is even), or $-\frac{1}{2}$, which is reached when the number of variables assigned to true and false differs of one unit, in case n is odd.

Notice that, the nominal value of the weight of an interpretation is not really important, what matters is the order between the weights that determines the order of the interpretations. In the above exaple, for instance, if we define the weight function as $w(\mathcal{I}) = -(\sum_i \mathcal{I}(p_i) - \frac{n}{2})^2$ we obtain exactly the same order betwen the interpretations.

1.2. Specifying weight function with weighted formulas. In the most general case, the specification of $w : \mathbb{I} \to \mathbb{R}$ could involce the specification of $2^n - 1$ parameters. The "-1" is due to the fact that, without loss of generality we can suppose that the "worse" interpretation is weightd $-\infty$. However there are more compact and easy to interpret ways to specify a weight functionf on interpretations, one of this is by associating weights to formulas.

Let $F = \{w_i : \phi_i\}_{i=1}^n$ be a multiset[1] of of n propositional formulas the set of propositional variables $\mathcal{P}$ each of which is assigned a real number, called *weight*. We can use F to define a weight function on the set of truth assignments of $\mathcal{P}$ as follows:

$$w_F(\mathcal{I}) = \sum_{w:\phi \in F} w \cdot \mathcal{I}(\phi) \tag{4}$$

which implies that the ordering $\prec_F$ is defined as

$$\mathcal{I} \prec_F \mathcal{J} \Leftrightarrow \sum_{w:\phi \in F} w \cdot \mathcal{I}(\phi) \leq \sum_{w:\phi \in F} w \cdot \mathcal{J}(\phi)$$

One important intuition to bear in mind is the following: For every weighted formula $w : \phi \in F$

- if $w_i > 0$ we prefer interpretations that satisfy ϕ;
- if $w_i < 0$ we prefer interpretations that do not satisfy ϕ;
- if $w_i = 0$ we are indifferent about the truth value of γ.

[1]A multiset is a set that can contain multiple copies of the same elements. For instance $\{1, 2, 3, 5, 2, 1\}$ is a multiset, which is equal to $\{1, 1, 2, 2, 3, 5\}$.

1.3. Properties of weight function. In the MaxSAT problem the nominal weight of a model in not important, what matters is the ordering that a certain weight function induces on a set of iunterpretations. As a consequence the same ordering can be obtained by different weight functions. In the following we define the notion of *equivalence* between weight functions, that intuitively means that they define the same partial order on a set of interpretations. We also introduce the notion of a weight formula being the *opposite* of another weight formula with the intuitive meaning that the order defined by the two weight funciton are one the inverse of the other.

DEFINITION 1.1. *Two sets of weighted formulas on the propositional variables $\mathcal{P}$, F_1 and F_2 are* equivalent *if they define the same order. I.e., if for all interpretations $\mathcal{I}, \mathcal{J}$ of $\mathcal{P}$*

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

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

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

PROPOSITION 1.1. *(1) F is equivalent to $a \cdot F = \{a \cdot w : \phi \mid w : \phi \in F\}$ for $a > 0$;*
(2) F is opposite of $a \cdot F = \{a \cdot w : \phi \mid w : \phi \in F\}$ 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 $F \cup \{w_1 + w_2 : \phi\}$

PROOF. Let use w_F to denote the weight function defined by the set of weighted formulas F. Notice that $w_{a\cdot F}(\mathcal{I}) = a \cdot w_F(\mathcal{I})$. Indeed

$$\begin{aligned} w_{a\cdot F}(\mathcal{I}) &= \sum_{w:\phi\in F} a \cdot w \cdots \mathcal{I}(\phi) \\ &= a \cdot \sum_{w:\phi\in F} w \cdots \mathcal{I}(\phi) = a \cdot w_F(\mathcal{I}) \end{aligned}$$

Property (1) and (2) directly derives from this fact. For property (3) we have that

$$\begin{aligned} w_{F\cup\{-w:\neg\phi\}} &= w_F(\mathcal{I}) - w \cdot (1 - \mathcal{I}(\phi)) \\ &= w_F(\mathcal{I}) - w + w \cdot \mathcal{I}(\phi) \\ &= w_{F\cup\{w:\phi\}}(\mathcal{I}) - w \end{aligned}$$

This implies that

$$\begin{aligned} w_{F\cup\{w:\phi\}}(\mathcal{I}) < w_{F\cup\{w:\phi\}}(\mathcal{J}) &\Leftrightarrow w_{F\cup\{w:\phi\}}(\mathcal{I}) - w < w_{F\cup\{w:\phi\}}(\mathcal{J}) - w \\ &\Leftrightarrow w_{F\cup\{-w:\neg\phi\}}(\mathcal{I}) < w_{F\cup\{-w:\neg\phi\}}(\mathcal{J}) \end{aligned}$$

The proof of properties (1) and (2) are left by exercize. □

Property (1) of Proposition 1.1 states that, if we re-scale the weights of a positive factor, the order on the interpretations does not change. Instead property (2) says that if we rescale with a negative factor then we obtaing the opposite ordering. Property refitem:F-phi-equiv-F–not-phi states that the weight function obtained by inverting the weight associated to the formula and negating the formulas differs from a constant from the original weight. This implies that optimizing the two weight function will lead to the same result. This property guarantees that,

without loss of generality we can assume that all the weights are positive. Indeed every negatively weighted formula $w : \phi$ can be replaced by the positively weighted formula $-w : \neg\phi$, without changing the order between the interpretations. Property (1) implies that, if two formulas are logically equivalent, then adding one with a weight or the other with the same weight has the same effect. In other words the specification of the weight function using weighted formulas is independent from the syntactic spefication of the formula but depends only from the semantics of the formula.

2. The MaxSAT problem

There are various versions of MaxSAT problems

- *Basic MaxSAT:* There are no hard clauses and all the soft clauses have the same weight (equal to 1). The solution of this problem is the assignment that satisfies the maximize number of soft clauses, or equivalently minimize the number of unsatisfied soft clauses.
- *Partial MaxSAT:* the set of hard clauses could be not empty and the soft clauses have the same weight (equal to 1). The solution need to satisfy them and to minimize the number of unsatisfied soft clauses
- *Weighted MaxSAT:* No hard clauses and different weights associated with soft clauses The solution has to minimize the sum of weights of unsatisfied soft clauses.
- *Weighted Partial MaxSAT:* The set of hard clauses could be non empty and they need to be satisfied by the solution. The soft clauses can be associated with different weight, and the solution has to minimize the sum of the weeight of the soft clauses that are not satisfieed.

The last version os the most general version and it. If no specification is given with the terms MaxSAT, we refer to this general formulation. In the following we provide formal definitions of the different versions.

A general definition of the maximum satiisfiability problem is the following:

Definition 1.2 (General maximum satisfiability problem). *Given a partial order $(\mathbb{I}, \prec)$ defined on the interpretations of a set of propositional variables $\mathcal{P}$, and a formula ϕ, the* maximum satisfiability problem *is the problem of finding a model $\mathcal{I}^*$ of ϕ such that:*

$$\mathcal{I}^* \in \sup_{\prec}(\{I \in \mathbb{I} \mid \mathcal{I} \models \phi\}) \tag{5}$$

When $\prec$ is a total order, then it can be specified by a weight function $w : \mathbb{I} \to \mathbb{R}$, then the problem of maximum satisfiability can be rewritten as the problem of finding the maximum of the weighted formula, i.e.

$$\mathcal{I}^* = \operatorname*{argmax}_{\mathcal{I} \models \phi} w(\mathcal{I}) \tag{6}$$

The literature contains multiple definition and variants of the MaxSat problem, that are specific cases, or can be reformulated in terms of (6). All the approaches to MaxSAT assumes that weighted and hard formulas are specified in CNF. In the following we report the various definitons. of the different MaxSAT problems

Definition 1.3 (Unweighted MaxSat). *Given a set of clauses $C_1, \ldots, C_n$, the* unweighted maximum satisfiability problem *is the problem of finding an assignment*

that maximizes the total number of satisfied clauses:, i.e.,

$$\mathcal{I}^* = \operatorname*{argmax}_{\mathcal{I}} \sum_i \mathcal{I}(C_i) \tag{7}$$

Much studied in Theoretical Computer Science are dedicated to the MaxSat formulation (7). This formulation can is a special case of the general definition (5) where ϕ is $\top$ and the weight function speficied by the weighted formulas $1 : C_1$, $\dots 1 : C_n$. It has been proved that unweighted MaxSat is NP-complete. Even Max2Sat, the restriction to instances in which each clause has at most two literals in it, is NP-complete.

DEFINITION 1.4 (Weighted MaxSat). *Given a set of weighted clauses* $w_1 : C_1, \dots, w_n : C_n$*, the* weighted maximum satisfiability problem *is the problem of finding an assignment that maximizes the sum of the weights of the clauses satisfied by the assignment.*

$$\mathcal{I}^* = \operatorname*{argmax}_{\mathcal{I}} \sum_i w_i \cdot \mathcal{I}(C_i) \tag{8}$$

DEFINITION 1.5 (Partial MaxSat). *Given a set of clauses* $C_1, \dots, C_n$*, called* soft clauses*, and a second set of clauses* $D_1, \dots, D_n$*, called* hard clauses*, the* partial maximum satisfiability problem *is the problem of finding an assignment that satisfies the hard clauses and that maximizes the number of satisfied soft clauses:*

$$\mathcal{I}^* = \operatorname*{argmax}_{\mathcal{I} \models D_1 \dots, D_n} \sum_i \mathcal{I}(C_i) \tag{9}$$

DEFINITION 1.6 (Partial weighte MaxSat). *Given a set of weighted clauses* $C_1, \dots, C_n$*, called* soft clauses*, and a second set of clauses* $D_1, \dots, D_n$*, called* hard clauses*, the* partial maximum satisfiability problem *is the problem of finding an assignment that satisfies the hard clauses and that maximizes the sum of the weight of the satisfied soft clauses:*

$$\mathcal{I}^* = \operatorname*{argmax}_{\mathcal{I} \models D_1 \dots, D_n} \sum_i w_i \cdot \mathcal{I}(C_i) \tag{10}$$

Inspite of the different definition each of the formulation can be rewritten in terms of the others and in particular a general MaxSat problem can be rewritten (in polinomial time) in an equivalent unweighted MaxSat problem.

This can be donw in the following way:

PROPOSITION 1.2. *Let* $\{D_i\}$ *and* $\{w_i : C_i\}$ *be a set of hard and soft clauses respectively.*

1. *Let* δ *be the the smallest value* $|w_i - w_j|$ *different from* 0*, otherwise let* $\delta = w_i$*;*
2. *Let* $v_i = \lceil \frac{w_i}{\delta} \rceil$
3. *let* $v^* = \sum_i v_i + 1$
4. *let* $\mathcal{C}$ *be the multiset of clauses that contains* v^* *copies of each hard clause and* v_i *copies of each clause* v_i*.*

I is the solution of the partial weighted maximum satisfiability problem (10) *if and only if* $\mathcal{I}$ *is the solution of the unweighted maximum satisfiability problem* (7) *on* $\mathcal{C}$ *and* $\mathcal{I} \models \boldsymbol{D}$ *where* $\boldsymbol{D} = \bigwedge_i D_i$*.*

PROOF. By exercize. Hint. First show that if $\mathcal{I} \not\models \boldsymbol{D}$ and $j \models \boldsymbol{D}$ then $w(\mathcal{I}) < w(\mathcal{J})$. Then show that Then show that if $\mathcal{I} \models \boldsymbol{D}$ and $\mathcal{J} \models \boldsymbol{D}$, then $w(\mathcal{I}) < w(J)$ if and only if the number of clauses in $\mathcal{C}$ satisfied by $\mathcal{I}$ are less then the number of clauses satisfied by $\mathcal{J}$. □

3. MaxSAT exact algorithms

There are different approaches to solve the MaxSAT problem. They can be classified in exact algorithms, which provide an exact solution to the problem, and approximated algorithms, which guarantees only sub-optimal solutions. All the algorithms assumes that weighted formulas are clauses. In this section we will present some of the basic algorithms of the first categori. Inside this category we can distinguish three main apporaches:

- Branch and bound algorithns;
- Transformation into Integer Programming;
- Algorithms that use SAT as oracle;
- Algorithms based on implicit hitting sets.

In the following section we introduce the basis of each of the above categories.

3.1. Branch and Bound. Branch and Bound (B&B) algorithms explore the search tree of all partial assignments for the soft and hard clauses, in a depth-first manner, in order to find the interpretation thas satisfy all the herd clauses and maximizes the weight of the satisfied soft clauses. For every interpretation, $\mathcal{I}$ let *loss*($\mathcal{I}$) is the sum of the weights of the clauses that are not satisfied by $\mathcal{I}$. Solving the MaxSAT problems coincides to find the interpretation $\mathcal{I}$ that minimizes *loss*($\mathcal{I}$).

Let us start by introducing a base algorithm that performs an exhaustive search of all the assignments that satisfies the hard clauses and select the one with minimal loss (or equivalently maximal weigh). The search lgorithm for MasSAT can be obtained by modifying the DPLL decision procedure, which searches for *any* assignment that satisfies a formula ϕ, so that it does not stop when one model of ϕ is found, but it continues to search other models of ϕ. The pseudocode is shown in Algorithm 1.

EXAMPLE 1.4. *Let us see with a simple example how the algorithm works. Consider the following sets of hard and soft clauses:*

$$\phi = \left\{ \begin{array}{c} \{A, B, C\} \\ \{\neg A, \neg B, \neg C\} \end{array} \right\} \qquad \psi \left\{ \begin{array}{c} 2{:}\{A, \neg B\} \\ 3{:}\{\neg A, C\} \\ 4{:}\{B, \neg C\} \end{array} \right\}$$

Aa possible expansion of the search tree of the max-DPLL procedure is showin in Figure 4. Notice that the best assignment, is the one with minimal loss which is equal to 2. In the tree we choose the literals $\neg A$, $\neg B$, and $\neg C$ in this order.

Algorithm 1 MAX-DPLL(ϕ : CNF, ψ : weighted CNF, $\mathcal{I}$: Partial assignment)

1: $\mathcal{I}, \phi, \psi \leftarrow$ UNITPROPAGATION$(\mathcal{I}, \phi, \psi)$
2: **if** $\{\} \in \phi$ **then**
3: **return** $\mathcal{I}, \infty$ c
4: **end if**
5: **if** $\phi = \{\}$ and ψ contains only empty weighted clauses **then**
6: **return** $\mathcal{I}, \sum_{(w:C)\in\psi} w$
7: **else**
8: select a l from a clause in ϕ or in ψ
9: $\mathcal{I}, loss \leftarrow$ MAX-DPLL$(\phi|_l, \psi|_l, \mathcal{I} \cup \{l\})$
10: $\mathcal{I}', loss' \leftarrow$ MAX-DPLL$(\phi|_{\bar{l}}, \psi|_{\bar{l}}, \mathcal{I} \cup \{\bar{l}\})$
11: **if** $loss \leq loss'$ **then**
12: **return** $\mathcal{I}, loss$
13: **else**
14: **return** $\mathcal{I}', loss'$
15: **end if**
16: **end if**

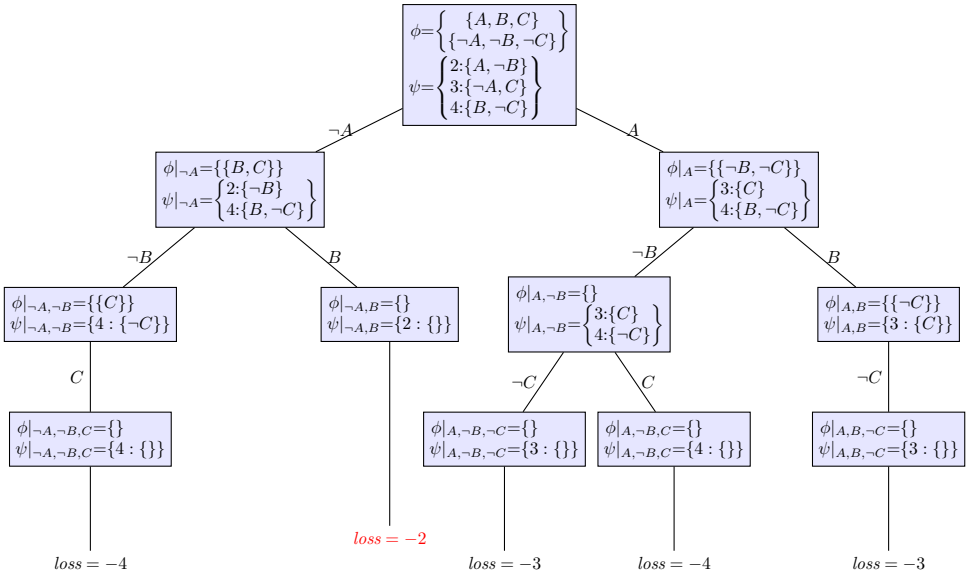


FIGURE 4. The search tree of the max-DPLL procedure

The MAX-DPLL algorithm recursively performs a depth-first visit of the tree of the partial assignments to the propositional variables of the input, searching for the assignment that minimizes the loss. MAX-DPLL takes in input a partial assignment, which is empty in the first call, a set of hard clauses ϕ and a set of soft clauses with the associated weights. MAX-DPLL starts by applying unit propagation (line 1) only on hard clauses, This means that, for all $\{l\} \in \phi$ the current partial assignment $\mathcal{I}$ is extended with $\{l\}$ and ϕ is set to $\phi|_l$ and ψ to $\psi|_l$. UNITPROPAGATION repeatedly applies this reduction until ϕ does not contain

any unit clause. Then, if the set of hard clauses contains an empty clause (i.e., the current assignment does not satisfy the hard clauses), then MAX-DPLL returns the current partial assignment $\mathcal{I}$ with infinite loss (line 3), In this situation the infinite loss has the effect that this solution will be the worse possible from the minimisation point of vies, and any other solution with finite cost would be preferred to it. If instead (line 5) the set of hard clauses are empty (i.e., all of them are satisfied by the current partial interpretation $\mathcal{I}$) and the soft clauses contains only empty (i.e.,, unsatisfiable) clauses, then the MAXSAT-DPLL returns the current interpretation with its cost, which is the sum of the weight of the soft clauses, which are all empty, and therefore they are not satisfied by $\mathcal{I}$. Otherwise, i.e., if there are hard clauses that need to be satisfied, and soft clauses that could be satisfied, (i,e, if $C \in \phi$ for some non empty C and $w : D \in \psi$ for some non empty D), the algorithm computes the solutions that contains the current assignment extended with l, (line 9) and with $\bar{l}$ (line 10) for some literal l, and choose the one of minimal cost (lines 11–14)

The MAX-DPLL algoritm is not very useful in practice as it potentially visits the entire tree of all the partial interpretations, which is exponentially large, w.r.t., the size of the input clauses. One should find some criteria for early stopping the search when there is some evidence that this will not lead to any better solution. Suppose that MAX-DPLL has already found a solution $\mathcal{I}$ with loss equal to x. Then every assignment with loss greater than x are not solutions of the maxSat problem. This means that x acts as an upper bound on the cost of the solutions, let denote x with UB. If MAX-DPLL is expanding anothe partial interpretation $\mathcal{I}'$. Let LB be the minimal loss of all the assignments which are expansions of $\mathcal{I}'$. A naïve wey to find a value for LB is by summing the weights of the soft-clauses not satisfied by $\mathcal{I}'$. If $LB \geq x$, then the loss of any extension of $\mathcal{I}'$ will have cost larger or equal to x, and they will not be better than the solution $\mathcal{I}$ already found which means that we can stop expanding $\mathcal{I}'$ and look to other alternative partial interpretations. This idea is implemented in the *Branch and Bound* algorithm shown in Algorithm 2. Algorithm B&B takes in input two additional parameters than MAX-DPLL which

Algorithm 2 B&B(ϕ: CNF, ψ: Weighted CNF, $\mathcal{I}$: Partial assignment, $\mathcal{I}_{UB}$: Best previously found solution, UB: Cost of $\mathcal{I}_{UB}$)

1: $\mathcal{I}, \phi, \psi \leftarrow$ UNITPROPAGATION$(\mathcal{I}, \phi, \psi)$
2: $LB \leftarrow$ LOWERBOUND(ϕ, ψ)
3: **if** $\{\} \in \phi$ or $UB \leq LB$ **then**
4: **return** $\mathcal{I}_{UB}, UB$
5: **end if**
6: **if** $\phi = \{\}$ and ψ contains only empty weighted clauses **then**
7: **return** $\mathcal{I}, \sum_{(w:D)\in\psi} w$
8: **else**
9: select a l from some clause in ϕ or in ψ
10: $\mathcal{I}, UB \leftarrow$ B&B$(\phi|_l, \psi|_l, \mathcal{I} \cup \{l\}, \mathcal{I}_{UB}, UB)$
11: $\mathcal{I}', UB' \leftarrow$ B&B$(\phi|_{\bar{l}}, \psi|_{\bar{l}}, \mathcal{I} \cup \{\bar{l}\}, \mathcal{I}_{UB}, UB)$
12: **return** $\mathcal{I}', UB'$
13: **end if**

are the best solution found sofar $\mathcal{I}_{UB}$ and its loss UB. B&B proceeds as MAX-DPLL with the only exception that it first computes a lower bound of the current

partial assignment (line **??**) and checks it it is larger than than the loss of the best solution found until now (line **??**). In this case the previous solution is returned. The first call of B&B is done with the empty interpretation $\mathcal{I}$, and $\mathcal{I}_{UB}$ and the infinite cost $UB = \infty$.

The simplest way to compute the lower bound of a partial interpretation is by summing the weights of the empty clauses in ψ. i.e.,

$$\text{LOWWERBOUND}(\phi, \psi) = \sum_{w:\{\} \in \psi} w \tag{11}$$

Later we will see more sophisticated algorithm that compute higher lower bounds and therefore that prevents B&B to explore larger parts of the search space. For the time being, let us see an example of B&B with this simple method of estimating lower bound.

EXAMPLE 1.5. *Consider the following set of hard and soft clauses with relative weight (hard clauses are labelled with infinite weight).*

$$\begin{array}{lll} (\neg a \vee b \vee c : \infty) & (a : 1) & (c : 3) \\ (\neg b \vee d : \infty) & (b : 2) & (d : 2) \\ (\neg d \vee \neg a : \infty) & & \end{array}$$

3.1.1. *Lower bound computation.* One step forward w.r.t., (11) can be done by incorporating an underestimation of the sum of the weights of clauses that will become unsatisfied if the current partial assignment is extended to a complete assignment.

EXAMPLE 1.6. *Suppose that ψ contains the two weighted unary clauses $w : \{x\}$ and $v : \{\neg x\}$. This implies that every assignment for ϕ, ψ will falsify one of the two clauses. Therefore, the loss for ϕ, ψ will not be less then* $\min(w, v)$. *A little more complicated example is when ψ contains multiple pairs of contradicting unary weighted clauses.*

$$\left\{ \begin{array}{c} w_1 : \{x_1\}, v_1 : \{\neg x_1\} \\ w_2 : \{x_2\}, v_2 : \{\neg x_2\} \\ \cdots \\ w_n : \{x_n\}, v_n : \{\neg x_n\} \end{array} \right\} \subseteq \psi$$

then every assignment for ϕ, ψ will falsify one of the two unary clause for every pair. Therefore a lowerbound for the loss is therefore equal to:

$$\sum_{i=1}^{n} \min(w_i, v_i)$$

Finally suppose that ϕ contains the hard clause $\{a, b\}$ and ψ contains the two weighted unary clauses $w : \{\neg a\}$ and $v : \{\neg b\}$ Then every assignment that satisfy ϕ will not satisfy one of the two weighted unary clauses, which implies that the loss will be at least $\min(w, v)$.

The following method to find a lowerbound for the loss of a maxSAT problem ϕ, ψ is a generalization of the previous example and uses a SAT solver as an oracle.

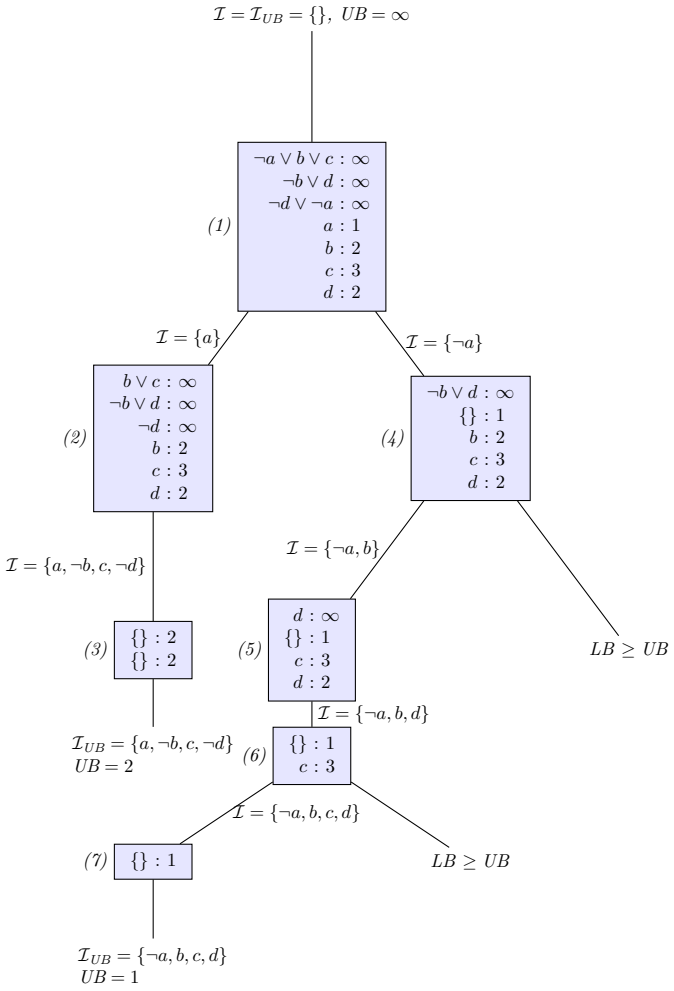


FIGURE 5. Exploration tree of the B&B algorithm for the hard and soft clauses of Example 1.5. Nodes are labelled in order of expansion. B&B find the first solution at node (3), the cost of this solution is 4. Therefore $\mathcal{I}_{UB}$ is set to be this solution and the UB (upper bound cost) is set to be 4. Then the algorithm find a second solution at node (7) whose cost is equal to 1 smaller than UB. Therefore $\mathcal{I}_{UB}$ is set to this new solution and UB is set to 1. Successively the B&B reaches the node (8), which does not correspond to a solution. but the solutions that are derivable from the partial assignment at this node will have a cost higher or equal to 3 (the sum of the costs of the two empty soft clauses) which is higher than the current UB and therefore B&B stops and returns $\mathcal{I}_{UB}$ and UB.

Suppose that ψ contains n disjoint subsets $\psi_1, \ldots, \psi_n$ and each $\{C \mid w : C \in \psi_i\} \cup \phi$

is not satisfiable, then

$$\text{lowerBound}(\phi, \psi) = \sum_{i=1}^{n} \min_{w:C\in\psi_i} w \tag{12}$$

Indeed notice that every assignment that satisfy ϕ will not satisfy at least one clause in each ϕ_i. Since we don't know which of the clauses of ϕ_i will be falsified, we can only infer that the loss will be increased with the minimum weights among those of the clauses in ϕ_i. Notice that (11) is a special case of (12).

Example 1.7. *Consider the set of hard and soft clauses:*

$$\phi = \left\{ \begin{matrix} \{a, b\} \\ \{\neg a, \neg b\} \end{matrix} \right\} \qquad \psi = \begin{Bmatrix} 1{:}\{a, \neg c\} & 2{:}\{\neg a, c\} \\ 3{:}\{b, \neg c\} & 4{:}\{\neg b, c\} \\ 5{:}\{c\} & 6{:}\{\neg c\} \end{Bmatrix}$$

ψ contains the following two disjoint sets of weighted clauses

$$\psi_1 = \begin{Bmatrix} 1{:}\{a, \neg c\} \\ 3{:}\{b, \neg c\} \\ 5{:}\{c\} exercize \end{Bmatrix} \qquad \psi_2 = \begin{Bmatrix} 2{:}\{\neg a, c\} \\ 4{:}\{\neg b, c\} \\ 6{:}\{\neg c\} \end{Bmatrix}$$

such that, if we extend their unweighted version with ϕ, we obtain the following two sets of clauses

$$\left\{ \begin{matrix} \{a, b\} \\ \{\neg a, \neg b\} \\ \{a, \neg c\} \\ \{b, \neg c\} \\ \{c\} \end{matrix} \right\} \qquad \left\{ \begin{matrix} \{a, b\} \\ \{\neg a, \neg b\} \\ \{\neg a, c\} \\ \{\neg b, c\} \\ \{\neg c\} \end{matrix} \right\}$$

which are both unsatisfiable. This means that every assignment does not satisfy at least one clause in ψ_1 and one in ψ_2. This implies that the minimum value of the loss of any assignments for ϕ, ψ is $1 + 2 = 3$ which is the sum of the minimum weights of the clauses in ψ_1 and ψ_2.

3.1.2. *Literal selection.* One of the key aspect of the B&B algorithm is the literal selection performed on line 9. Selecting the "right" literal will avoid the expansion of many parts of the search tree and go straight to the maxSAT solution. For instance in figure 5 if the algorithm would have selected $\neg a$ instead of a in the first branching it would have reached the solution straightaway without expanding the subtree on the left (the one under the selection of a). Most of the exact MaxSAT solvers incorporate variable selection heuristics that take into account the number of literal occurrences in such a way that each occurrence has an associated weight that depends on the length of the clause that contains the literal. MaxSAT heuristics give priority to literals occurring in binary clauses instead of literals occurring in unit clauses as SAT heuristics do. Things are getting to technical and detailed here. So I decided to stop here with this topic. For those interested in the argument you can check works like Mohamedou and Planes 2009,

3.2. Core based algorithm. Core based algorithms for maximum satisfiability uses a SAT solver as subrutine. The main idea behind the core based approaches is the following:

If ϕ, ψ is satisfiable then any interpretation $\mathcal{I}$ that satisfies ϕ and ψ is a solution of the MaxSAT problem with cost equal to 0. If ϕ, ψ the ideas is to weakening the

soft clauses ψ (by adding literals to some of the clauses in ψ) but at the same time impose some additional constraints by extending the hard clauses, requiring that the added literals will be minimally true. Then check the satisfiability of the new MaxSAT problem. Let us see a concrete example on how this works:

EXAMPLE 1.8. *Consider the MaxSAT instance:*

$$\phi = \left\{ \begin{array}{c} \{p, q\} \\ \{\neg p, \neg r\} \\ \{\neg q, \neg r\} \end{array} \right\} \qquad \psi = \left\{ \begin{array}{c} 1{:}\{p\} \\ 1{:}\{q\} \\ 1{:}\{r\} \end{array} \right\}$$

Notice that the union of ϕ with (the unweighted version of) ψ is not satisfiable. Which implies that a solution of this simple MaxSAT should not satisfy at least one of the soft clauses. The idea is to weaken the soft clauses so that they become satisfiable and add a constraint that states that only one of them can be falsified.

To weaken a soft clause C we can add a fresh atom b obtaining $C \cup \{b\}$. In our example we have

$$\psi_1 = \left\{ \begin{array}{c} 1{:}\{p, b_1\} \\ 1{:}\{q, b_2\} \\ 1{:}\{r, b_3\} \end{array} \right\}$$

and then add the constraint that at most one among b_1, b_2 and b_3 can be true. I.e., $\sum_{i=1}^{3} b_i$. The new MaxSAT problem becomes:

$$\phi^{(1)} = \left\{ \begin{array}{c} \{p, q\} \\ \{\neg p, \neg r\} \\ \{\neg q, \neg r\} \\ \sum_{i=1}^{3} b_i \leq 1 \end{array} \right\} \qquad \psi^{(1)} = \left\{ \begin{array}{c} 1{:}\{p, b_1\} \\ 1{:}\{q, b_2\} \\ 1{:}\{r, b_3\} \end{array} \right\}$$

Notice that every interpretation $\mathcal{I}$ that satisfies $\phi^{(1)}$ and the unweighted version of $\psi^{(1)}$ must satisfy at least one b_i. Otherwise the initial $\phi \cup \psi$ would have been satisfiable. In other words b_i being true indicates that the initial i-th clause can be falsified. Furthermore it $\mathcal{I}$ is a solution of the MaxSAT problem $\phi^{(1)}, \psi^{(1)}$ it will also be a solution of the initial MaxSAT problem with a cost augmented of one unit.

In this simple case we have that $\phi^{(1)} \cup \psi^{(1)}$ is satisfiable, by the assignment:

$$p = 1 \qquad q = 0 \qquad r = 0 \qquad b_1 = 0 \qquad b_2 = 0 \qquad b_3 = 1$$

which contains a solution of the initial MaxSAT problem, with cost/loss equal to 1.

One can minimize the number of soft clauses to weaken by adding new variable, by considering only the minimal set of weak clause ψ' such that $\phi \cup \psi'$ is inconsistent. In the example we have two alternatives such a minimal sets.

$$\psi' = \left\{ \begin{array}{c} 1{:}\{p\} \\ 1{:}\{r\} \end{array} \right\} \qquad \psi'' = \left\{ \begin{array}{c} 1{:}\{q\} \\ 1{:}\{r\} \end{array} \right\}$$

We can consider one of the two subsets and proceed as before by adding two new variables b_1 and b_2 (instead of 3) obtaining the set of hard and soft clasuses:

$$\phi^{(1)} = \left\{ \begin{array}{c} \{p, q\} \\ \{\neg p, \neg r\} \\ \{\neg q, \neg r\} \\ \sum_{i=1}^{2} b_i \leq 1 \end{array} \right\} \qquad \psi^{(1)} = \left\{ \begin{array}{c} 1{:}\{p, b_1\} \\ 1{:}\{r, b_2\} \end{array} \right\}$$

The pair $\phi^{(1)} \cup \psi^{(1)}$ is satisfiable with the same assignment as before, with the exception of the assignment to b_3.

Now let us generalize the above example. We first define the notion of *core*. Given a set ϕ of unsatisfiable clauses a core is a subset ϕ' of ϕ such $\phi' \setminus \{C\}$ for every $C \in \phi'$ is consistent. In other words, a core is a minimally inconsistent set of formulas. such that if we remore one formula from it we are left with a consistent set of formulas.

Core of a set of clauses (Exercize 6).

EXAMPLE 1.9. *The core sets of the set of clauses*

$$\{\{a, b\}, \{\neg a, \neg b\}, \{a\}, \{b\}, \{\neg a\}, \{\neg b\}\}$$

are the following:

$$\{\{a\}, \{\neg a\}\} \qquad \{\{b\}, \{\neg b\}\} \quad \{\{a, b\}, \{\neg a\}, \{\neg b\}\} \qquad \{\{\neg a, \neg b\}, \{a\}, \{b\}\}$$

Notice that all the above subsets of the given clauses are not satisfiable, but removing one clause from each of them results in a satisfiable set of clauses.

SAT solvers are such that, when they are called with a a set of clauses ϕ which are inconsistent, they return UNSAT, and in addition they return a core set, i.e. a minimally inconsistent subset of ϕ. The Fu and Malik MaxSAT algorithm exploit this feature of SAT solver in order to solve the MaxSAT problem.

The Fu and Malik algorithm for MaxSAT, originally proposed by Fu and Malik 2006, uses the SAT subrutine; when it is called with a set of clauses ϕ, it returns a pair x, y where $x = SAT$ and $y = \mathcal{I}$ if ϕ is satisfiable and $\mathcal{I}$ is an interpretation of ϕ. or the pair $x = UNSAT$ and ϕ' is a core set. In the following, we consider the special case in which the weights of the soft clauses are equal to 1. Extensions of the algorithm for more general formulations of MaxSAT can be found in Manquinho, Marques-Silva, and Planes 2009.

Algorithm 3 Fu and Malik MaxSAT algorithm

Require: ϕ set of hard clauses; ψ set of soft clauses with weight = 1
1: $x, y \leftarrow \text{SAT}(\phi)$;
2: **if** $x = \text{UNSAT}$ **then**
3: **return** ∞, None
4: **end if**
5: $cost \leftarrow 0$
6: **while** True **do**
7: $x, y \leftarrow \text{SAT}(\phi \cup \psi)$
8: **if** $x = \text{SAT}$ **then**
9: **return** $cort, y$
10: **else** ▷ $x = \text{UNSAT}$, y is a core for $\phi \cup \psi$
11: $B = \emptyset$
12: **for** $C \in y \cap \psi$ **do** ▷ C is a soft clause that appears in the core y
13: $b \leftarrow$ new propositional variable
14: $C \leftarrow C \cup \{b\}$
15: $B \leftarrow B \cup \{b\}$
16: **end for**
17: **end if**
18: $\phi \leftarrow \phi \cup \{\sum_{b \in B} b \leq 1\}$
19: $cost \leftarrow cost + 1$
20: **end while**

EXAMPLE 1.10. *Consider the set of hard and soft clauses ϕ and ψ.*

$$\phi = \{\neg x_1, \neg x_2\}, \{\neg x_1, \neg x_3\}, \{\neg x_1, \neg x_4\}, \{\neg x_2, \neg x_3\}, \{\neg x_2, \neg x_4\}, \{\neg x_3, \neg x_4\}$$
$$\psi = 1 : \{x_1\},\ 1 : \{x_2\},\ 1 : \{x_3\},\ 1 : \{x_4\}$$

- *Since $\phi \cup \psi$ is not satisfiable, then the call to $SAT(\phi \cup \psi)$ returns $x = UNSAT$ and y a core set of $\phi \cup \psi$;*
- *Suppose that the returned core set is the following:*

  $$y = \{\neg x_1, \neg x_2\}, \{x_1\}, \{x_2\}$$

  there are other core sets, but the result of the algorithm is independend from which core is returned by the sat solver.
- *The two waighted clauses $\{x_1\}$, $\{x_2\}$ are extended with two new propositions b_1 and b_2 respectively and the clause $b_1 + b_2 \leq 1$ (which is equivalent to $\{\neg b_1, \neg b_2\}$ is added.*
- *the cost is set to 1.*
- *Therefore we ontain the following new set of hard and soft clauses*

$$\phi = \{\neg x_1, \neg x_2\}, \{\neg x_1, \neg x_3\}, \{\neg x_1, \neg x_4\}, \{\neg x_2, \neg x_3\}, \{\neg x_2, \neg x_4\}, \{\neg x_3, \neg x_4\}, \{\neg b_1, \neg b_2\}$$
$$\psi = 1 : \{x_1, b_1\},\ 1 : \{x_2, b_2\},\ 1 : \{x_3\},\ 1 : \{x_4\}$$

- *The new set of clauses $\phi \cup \psi$ is also inconsistent, and a possible core is $\{\neg x_2, \neg x_4\}, \{x_3\}, \{x_4\}$.*
- *the cost is set to 2.*
- *We proceed as before obtaining the set of hard and soft clauses*

$$\phi = \{\neg x_1, \neg x_2\}, \{\neg x_1, \neg x_3\}, \{\neg x_1, \neg x_4\}, \{\neg x_2, \neg x_3\}, \{\neg x_2, \neg x_4\}, \{\neg x_3, \neg x_4\}, \{\neg b_1, \neg b_2\}, \{\neg b_3, \neg b_4\}$$
$$\psi = 1 : \{x_1, b_1\},\ 1 : \{x_2, b_2\},\ 1 : \{x_3, b_3\},\ 1 : \{x_4, b_4\}$$

- *the new set of clauses are not satisfiable and the only core is the set itself.*
- *the cost is set to 3;*
- *Every soft clause is therefore extended with a new variable. Since there are four soft clauses in the core, we introduce four new variables $b_5, \ldots, b_8$. And we also add the hard clause $b_5 + \cdots + b_8 \leq 1$ obtaining the following set of clauses:*

$$\phi = \{\neg x_1, \neg x_2\}, \{\neg x_1, \neg x_3\}, \{\neg x_1, \neg x_4\}, \{\neg x_2, \neg x_3\}, \{\neg x_2, \neg x_4\}, \{\neg x_3, \neg x_4\}, \{\neg b_1, \neg b_2\}, \{\neg b_3, \neg b_4\}$$
$$\psi = 1 : \{x_1, b_1, b_5\},\ 1 : \{x_2, b_2, b_6\},\ 1 : \{x_3, b_3, b_7\},\ 1 : \{x_4, b_4, b_8\}$$

  which is satisfiable.
- *Therefore the algorithm terminates.*

We have to prove that the algorithm terminates.

PROPOSITION 1.3. *The algorithm Fu and Malik terminates for every input of* ϕ, ψ

PROOF. to do □

3.3. Pseudo Boolean Optimizatiom. A Pseudo-Boolean function is a function $f0,1^n \to \mathbb{R}$. This type of function have beenn tudied since the '60s in operations research in integer linear programming Boros and Hammer 2002. One can consider also restricted versions, where f is represented in a polynomial of a given degree; f is represented as linear form, polinomial of degree 1;

A pseudo-boolean constraint is a constraint defined on a pseudo-boolean function f.

EXAMPLE 1.11. *Let* $f(x_1, \ldots, x_n)$ *be the linear pseudo-boolean function* $\sum_{i=1}^{n} x_i$. *An example of constraint on* f *is*

$$\sum_{i=1}^{n} x_i \leq k0$$

for some integer k*, which corresponds to the cardinality constraint "at most* k*" introduced in the previous chapters.*

DEFINITION 1.7 (Pseudo-Boolean optimization). *A pseudo-Boolean optimization is the problem of finding a boolean assignment to the variables* $x_1, \ldots, x_n$*, that satisfy the following:*

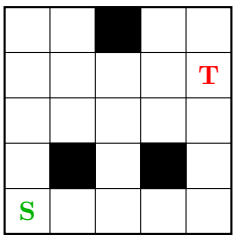
$$\begin{aligned} &\textit{Minimize} && \sum_{i=1}^{n} c_i x_i && \\ &\textit{Subject to} && \sum_{j=1}^{n} a_{ij} x_j \leq b_i && \forall i = 1, \ldots, m \end{aligned}$$

where $a_i, b_i, c_i \in \mathbb{Z}$.

[Section to be completed]

4. Solving problems with MaxSAT

4.1. Minimal path. Find shortest path in a grid with horizontal/vertical moves. Travel from **S** to **T** without enter in the black squares



$$p_{0,5,5} : \infty \tag{13}$$

$$\bigvee_{i=1}^{22} p_{i,2,5} : \infty \tag{14}$$

$$\bigwedge_{\substack{r,c=1 \\ (r,c)\neq(2,5)}}^{5} \left(p_{i,r,c} \rightarrow \bigvee_{\substack{r'c'=1 \\ |(r,c)-(r',c')|=1}}^{5} p_{i+1,r',c'} \right) : \infty \tag{15}$$

$$\bigwedge_{i=1}^{22} (\neg p_{i,1,3} \wedge \neg p_{i,4,1} \wedge \neg p_{i,4,4}) : \infty \tag{16}$$

$$\neg p_{i,r,c} : 1 \tag{17}$$

4.2. Optimal correlation clustering. The problem of optimal correlation clustering can be formulatecd as follows: 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 $v \in C_i$.

A MaxSAT formulation of the optimal correlation clustering problem is based on a set of indicator variables x_{ij} , where $i < j$, with the interpretation that x_{ij} is true if points v_i and v_j are put in the same cluster. Using this variables we can formulate the following hard and soft clauses

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

$$\neg x_{ij} \vee \neg x_{jk} \vee x_{ik} \qquad\qquad \neg x_{ij} \vee \neg x_{ik} \vee x_{jk}$$

Soft clauses:: for every $i < j$

$$x_{ij} : 1 \qquad \text{If } s(v_i, v_j) = 1$$

$$\neg x_{ij} : 1 \qquad \text{If } s(v_i, v_j) = 0$$

The first hard clause states that, if v_i and v_j are put in the same cluster and v_j and v_k are also put in the same cluster then v_i and v_k must be put in the same cluster. This clause is the CNF form of transidivity: $x_{ij} \wedge x_{jk} \rightarrow x_{ik}$. The second hard clause has similar meaning; it corresponds to the euclidean property of a relation $x_{ij} \wedge x_{ik} \rightarrow x_{jk}$. Notice that clustering is a special type of equivalence relation and therefore it is both transitive and euclidean. We don't need symmetry and reflexivity because for $i < j$ x_{ji} and x_{ii} are not propositional variables. Soft clauses mimic what happens in the cost function $G(\mathbb{V})$. Notice that if v_i and v_j are symilar (i.e., $s(v_i, v_j) = 1$) and they are assigned with different clusters, i.e., x_{ij} is false, then the cost of the interpretation is increased by 1. This corresponds to the term $s(v_i, v_j)$ of the second summation of $G(\mathbb{V})$. Vice-versa, if two points v_i and v_j which are dissimilar (i.e., $v(v_i, v_j) = 0$) are put in the same cluster, i.e., x_{ij} is true then the cost of the interpretation will increase by 1, These corresponds to the term $1 - s(v_i, v_j)$ (which is equal to 1) of the first summations of $G(\mathbb{V})$

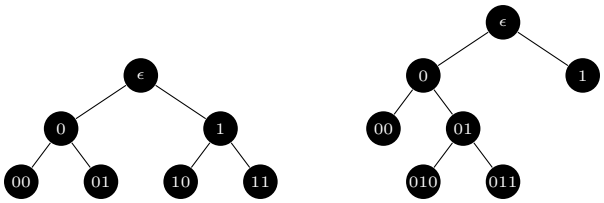


FIGURE 6. Two examples of binary trees with 7 nodes. Notice that the maximum depth of a binary tree with 7 nodes, is $\frac{7-1}{2} = 3$. Therefore we define B all the 0/1 strings of length less then or equal to 3. B indicates all the potential nodes of a binary tree.

Notice that the solution $\mathcal{I}$ of the MaxSAT problem does not provide directly the mapping of the v_i to the clusters. One has to derive such a clustering from the truth assignments of x_{ij}. This can be easily extracted by the following procedure; Let $\mathbb{C} = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$ be the initial clustering, and $x_{1,2}, \ldots, x_{1,n}, x_{2,3}, \ldots, x_{n-1,n}$ be an enumeration of the propositional variables, Then iterate on the elements of such a list, and at any step revise $\mathbb{C}$ as follows: If you are analyzing $x_{i,j}$ and $\mathcal{I}(x_{i,j}) = 1$, and $v_i \in C_q$ and $v_j \in C_r$ in the current clustering and $r \neq q$, then rmove C_q and C_r from $\mathbb{V}$ and add $C_q \cup C_r$.

5. MaxSAT for machine learning

5.1. Learning optimal decision tree. Let $\left\{\boldsymbol{x}^{(i)}, y^{(i)}\right\}_{i=1}^{d}$ be a dataset that we use to train a classifier for a class C. Suppose that $\boldsymbol{x}^{(i)}$ is a vector $(x_1, \ldots, x_k)$ of boolean features and that $y^{(i)}$ is equal to 0 if the i-th individual doesn't belong to C and 1 otherwise. Our goal is to construct a binary decision tree for the class C with at most n nodes. To cast this task in a MaxSat problem, we proceed in two steps, First, we formalize binary trees with at most n nodes in propositional logic. Successively, we associate to the internal node of the tree the corresponding features.

Let us start by formalizing binary trees in propositional logic. Before doing this let us see how a binary tree with n nodes looks like. Figure **??** shows two examples of binary trees with 7 nodes. To identify all possible nodes of a binary tree we use 0/1 strings, as follows. Suppose we want to consider a binary tree with at most n nodes. The maximum height of a binary tree with with n nodes. is $H = \frac{n-1}{2}$. Let B be the set $B = \bigcup_{h=0}^{H} \{0,1\}^h$. Intuitively B is all the potential nodes of a binary tree with n nodes. However we have to select only n elements of B. A binary tree can be seen as a non enpty subset of $T \subseteq B$ closed under substring. More formally $T \subseteq B$ is a tree iff

(1) $\epsilon \in T$ where ϵ is the empty string
(2) For all $\boldsymbol{b}i \in B$, $\boldsymbol{b}i \in T \Rightarrow \boldsymbol{b} \in T$

The above conditions can be easily formalized in propositional logic. Let $v_{\boldsymbol{b}i}$ for $\boldsymbol{b} \in B$ and $i \in \{0,1\}$ be a propositional variable indicating that the node $\boldsymbol{b}$ belongs to a binary tree T. For unifromity of formalization we add an extra depth step $H+1$ but we require that nodes with labels of length $H+1$ never belong to T.

The above conditions can be formalized as follows:

$$v_\epsilon \tag{18}$$

$$v_{\boldsymbol{b}i} \rightarrow v_{\boldsymbol{b}} \qquad \text{for } \boldsymbol{b} \in B \text{ and } i \in \{0,1\} \tag{19}$$

$$\neg v_{\boldsymbol{b}i} \qquad \text{for } \boldsymbol{b} \in B \text{ and } i \in \{0,1\} \text{ with } |\boldsymbol{b}| = H \tag{20}$$

Since we want a tree to be perfectly binary we have to guarantee that every node has either no children or two children. This can be formalized as:

$$v_{\boldsymbol{b}0} \leftrightarrow v_{\boldsymbol{b}1} \tag{21}$$

We can now pass to the second step in which we associate to each node $\boldsymbol{b} \in T$ one of the k features. To represent which feature is associated to which node we introduce the set of propositional variables $v_{\boldsymbol{b}}^f$ that codifies the fact that at node $\boldsymbol{b}$ we take the decision looking at feature f. We also want that only one feature is associated to internal nodes. This is formalizable by the axiom:

$$v_{\boldsymbol{b}} \wedge v_{\boldsymbol{b}i} \rightarrow \sum_f v_{\mathbf{b}}^f = 1 \tag{22}$$

Every propositional assignment that satisfies the above axioms identifies a unique binary decision tree with n nodes. We are only remained with the problem of formalizing in propositional logic how an item $\boldsymbol{x}^i$ is classified by such a binary tree. To this aim we introduce the propositional variabels $x_{\boldsymbol{b}}^{(i)}$ that indicates that the i-th item is classified in a subtree of the node $\boldsymbol{b}$. We can formalize the decision taken at each node of the tree, by the following set of formulas:

$x_\epsilon^{(i)}$	Every element is classified under the root node
$(x_{\boldsymbol{b}}^{(i)} \wedge v_{\boldsymbol{b}}^f) \leftrightarrow x_{\boldsymbol{b}x_f}^{(i)}$	If an is classified in $\boldsymbol{b}$ and the value of the feature associated to is i, then such item is classified under $\boldsymbol{b}i$

At this point you have to maximize the following measure

$$(x_{\boldsymbol{b}}^{(i)} \wedge \neg v_{\boldsymbol{b}i}) \wedge (C_{\boldsymbol{b}} \leftrightarrow y^{(i)})$$

5.2. Training Binary Neural Networks. [to be done]

6. Exercises

Exercise 1:

Given a list of numbers $a_1, \ldots, a_n$, formalize in propositional logic all the possible ways to split them into two sets. Then define a weight function that is maximal when the sums of the numbers in each set are as close as possible.

Exercise 2:

Define the weight functions that realizes the following total order on the interpretations of the propositinal variables $\mathcal{P} = \{A, B\}$

(1) $\{\} \prec \{A\} \prec \{B\} \prec \{A, B\}$
(2) $\{\} \prec \{B\} \prec \{A\} \prec \{A, B\}$
(3) $\{\} \prec \{A, B\} \prec \{A\} \prec \{B\}$
(4) $\{A, B\} \prec \{A\} \prec \{\} \prec \{B\}$
(5) $\{A\} \prec \{B\} \prec \{A, B\} \prec \{\}$

Exercise 3:
Prove the following facts:

(1) If $\models \phi \leftrightarrow \psi$, then $F \cup \{w : \phi\}$ is equivalent to $F \cup \{w : \psi\}$;
(2) $F \cup \{w_1 : \phi,\ w_2 : \phi\}$ is equivalent to $F \cup \{w_1 + w_2 : \phi\}$

Exercise 4:
Prove that $F \cup \{w : \phi \vee \psi\}$ is equivalent to $F \cup \{w : \phi \wedge \psi, w : \neg\phi \wedge \psi, w : \phi \wedge \neg\psi\}$

Exercise 5:
Suppose that $w < v$, Prove that $F \cup \{w : \phi, v : \neg\phi\}$ is equivalent to $F \cup \{v - w : \neg\psi\}$

Exercise 6:
Let $\mathcal{P}$ be a set of n propositinal variables, and let $w : 2^{\mathcal{P}} \to \mathbb{R}$ be a generic weight function. Define an algorithm that extract a set of weighted clauses F such that w_F is equivalent to w

Exercise 7:
Consider the set of strings that you can build with the letter A, B, C, and D. Define a propositional language such that everty interpretation correspond to a string and define a weight function that orders the strings (or equivalently the corresponding interpretations) lexicographically.

Solution Let us first define a set of propositional variables that we can use toi describe the finite strings composed of the letters A, B, C, and D. For every natural number $n \in \mathbb{N}$, we introduce the propositional variables A_n, B_n, C_n and D_n, where x_n for $x \in \{A, B, C, D\}$ means that the letter the n-th letter of the string is an x.

Then we have to add axioms that restricts the set of interpretations to those corresponding to strings.

(1) There must be at most one character at position n. This can be encoded either by a cardinality constraint AtMost$(1, \{A_n, B_n, C_n, D_n\})$ or by explicit representation:

$$\neg(A_n \wedge B_n) \quad \neg(A_n \wedge C_n) \quad \neg(A_n \wedge D_n) \quad \neg(B_n \wedge C_n) \quad \neg(B_n \wedge D_n) \tag{23}$$

(2) The second constraint states that, if in position n there is a character then there should be a character also in position $n-1$ i.e., we doin't have strings with "blanks" in the middle. This cen be formulated wiith:

$$(A_{n+1} \vee B_{n+1} \vee C_{n+1} \vee D_{n+1}) \to (A_n \vee B_n \vee C_n \vee D_n) \tag{24}$$

Notiche that every string $x_1x_2, \ldots, x_n$, with $x_i \in \{A, B, C, D\}$ can be mapped to an interpretation that makes x_i true and y_i false for eveyr $y \neq x$, and y_j false for every $y \in \{A, B, C, D\}$ and $j \geq n$. Therefore for instance, the string $ABAAC$ corresponds to the interpretation $\mathcal{I}_{ABAAC} = \{A_1, B_2, A_3, A_4, C_5\}$. Notice that the interpretation $\mathcal{I}_{x_1x_2\ldots x_n}$ corresponding to the string $x_1x_2\ldots x_n$ satisfies the axioms (23) and (24). Furtermore, every interpretation that satisfies these axioms corresponds to a (possibly infinite) string.

Let us now define a weight function that allow to order the interpretations that satisfies (23) and (24) lexicographically. I.e., $w(\mathcal{I}_{x_1x_2\ldots x_n\ldots}) < w(\mathcal{I}_{y_1y_2\ldots y_m\ldots})$ if and only if there is a k such that $x_k < y_k$ and $x_h = y_h$ for all $h < k$.

We define our weight function by associating a weight to each propositional variables and defining $w(\mathcal{I}) = \sum_{p\in\mathcal{I}} w(p)$

$$w(A_n) = 1 \cdot 10^{-n} \quad w(B_n) = 2 \cdot 10^{-n} \quad w(C_n) = 3 \cdot 10^{-n} \quad w(D_n) = 4 \cdot 10^{-n}$$

For instance we have the following weights:

$$\begin{aligned} w(\mathcal{I}_A) &= 0.1 \\ w(\mathcal{I}_{AB}) &= 0.1 + 0.02 = 0.12 \\ w(\mathcal{I}_B) &= 0.2 \\ w(\mathcal{I}_{BA}) &= 0.2 + 0.01 = 0.21 \\ w(\mathcal{I}_C) &= 0.3 \\ w(\mathcal{I}_D) &= 0.4 \\ w(\mathcal{I}_{DDDDDDD\ldots}) &= 0.4 + 0.04 + 0.004 + \ldots = 0.\bar{4} \end{aligned}$$

□

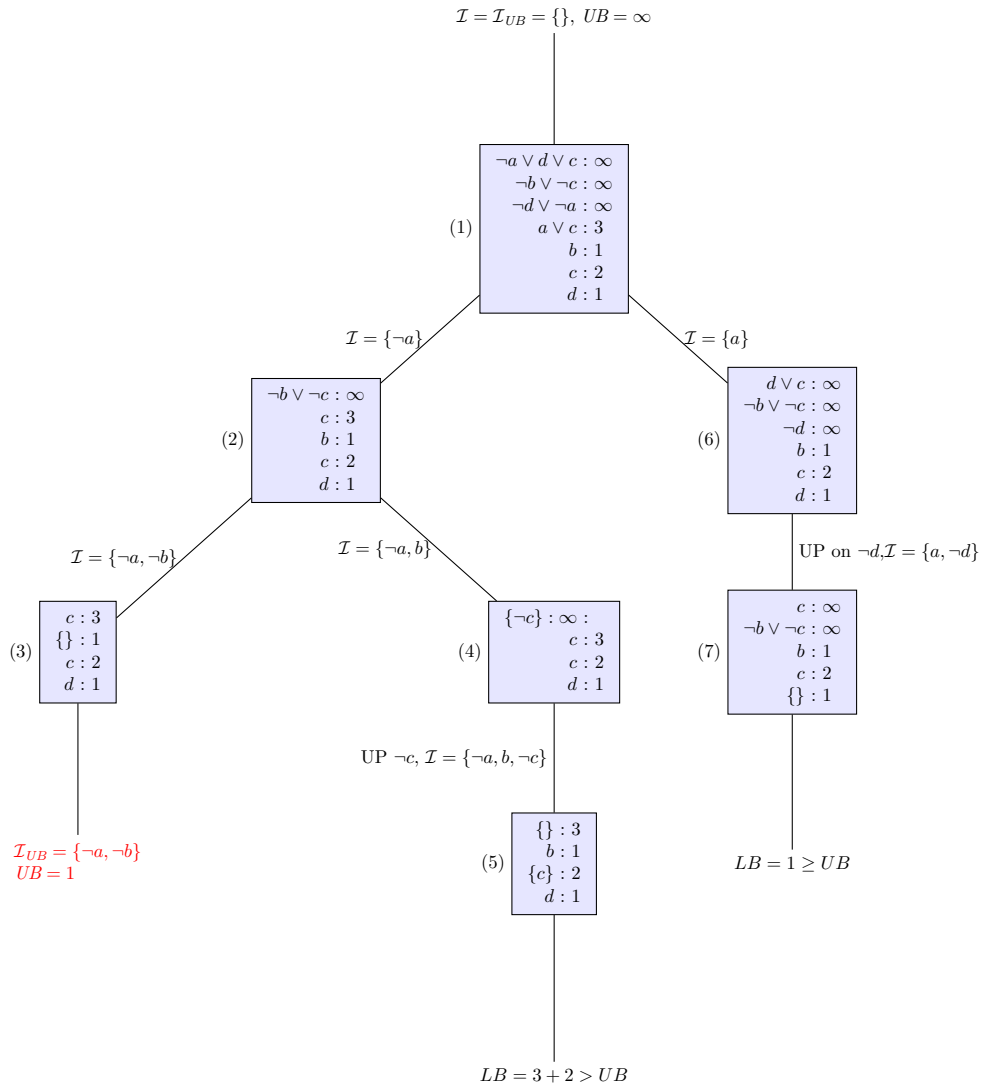
Exercise 8:
Find all the cores of the following set of clauses.

(1) $\{a, b, c\}, \{a, b, d\}, \{\neg a\}, \{\neg b\}, \{\neg c\}, \{\neg d\}$;
(2) $\{\neg a, b\}, \{\neg b, c\}, \{\neg c, d\}, \{\neg d, e\}, \{a, \neg b, \neg d\}$;
(3) $\{a\}, \{\neg a\}, \{b\}, \{\neg b\}, \{\neg a, b\}, \{a, \neg b\}$;
(4) $\{\neg a, b, c\}, \{\neg b, \neg d\}, \{\neg c, \neg d\}, \{a, d\}$;
(5) $\{x_i \mid i \in I\}$, $\{x_j \mid j \in J\}$, for some $I, J \subseteq \{1, \ldots, n\}$, with $I \cap J = \emptyset$ and the set of clauses $\{\neg x_i, \neg x_j\}$ for every $i < j \in \{1, \ldots, n\}$.

Exercise 9:
Use B&B algorithm to solve the following maxSat problem

$$\begin{array}{lll} (\neg a \vee d \vee c : \infty) & (a \vee c : 3) & (c : 2) \\ (\neg b \vee \neg c : \infty) & (b : 1) & (d : 1) \\ (\neg d \vee \neg a : \infty) & & \end{array}$$

Solution



□

Exercise 10:

In the Fu and Malik algorithm for MaxSat, explain what is a minimally inconsistent set of clauses. And provide an example of four clauses, which has a minimally inconsistent set of three clauses.

Solution Given a set of clauses $S = \{C_1, C_2, \ldots, C_n\}$ a minimally inconsistent subset of clauses of S is any subset S' of S such that S' is inconsistent, and for all $C \in S'$, $S' \setminus \{C\}$ is consistent.

For instance consider the set of clauses

$$S = \{\{A, B\}, \{\neg A, \neg B\}, \{A, C\}, \{B, C\}, \{\neg A, \neg C\}, \{\neg B, \neg C\}, \{C\}\}$$

Then the subset

$$S' = \{\{A, B\}, \{\neg A, \neg C\}, \{\neg B, \neg C\}, \{C\}\}$$

is a minimally inconsistent subset of S. Indeed if we remove from S' any clause we can find always an interpretation. Indeed notice that

$$\begin{aligned}
S' \setminus \{\{A, B\}\} &= \{\{\neg A, \neg C\}, \{\neg B, \neg C\}, \{C\}\} && \text{Is satisfied by } A = F, B = F, C = T \\
S' \setminus \{\{\neg A, \neg C\}\} &= \{\{A, B\}, \{\neg B, \neg C\}, \{C\}\} && \text{Is satisfied by } A = T, B = F, C = T \\
S' \setminus \{\{\neg B, \neg C\}\} &= \{\{A, B\}, \{\neg A, \neg C\}, \{C\}\} && \text{Is satisfied by } A = F, B = T, C = T \\
S' \setminus \{\{C\}\} &= \{\{A, B\}, \{\neg A, \neg C\}, \{\neg B, \neg C\}\} && \text{is satisfied by } A = T, B = T, C = F
\end{aligned}$$

□

Exercise 11:

Consider the following set of formulas

weight : formula
$\infty : \phi_1 = A \leftrightarrow (X \wedge Y \wedge Z)$
$\infty : \phi_2 = B \leftrightarrow (Y \wedge T \wedge W)$
$\infty : \phi_3 = C \leftrightarrow (T \wedge V)$
$\infty : \phi_4 = \neg A \wedge \neg B \wedge \neg C$
$1 : X$
$2 : Y$
$3 : Z$
$4 : W$
$5 : V$
$6 : T$

Find an assignment that minimizes the cost. Remember that the cost of an assignment is the sum of the weights of the clauses that are not satisfied by the assignment

Solution

var	A	B	C	X	Y	Z	W	V	T	ϕ_1	ϕ_2	ϕ_3	ϕ_4
$\mathcal{I}(var)$	0	0	0	0	1	1	1	1	0	1	1	1	1
$cost$	0	0	0	1	0	0	0	0	6	0	0	0	0

$cost(\mathcal{I}) = 1 + 6 = 7$ A different model with the same cost is:

var	A	B	C	X	Y	Z	W	V	T	ϕ_1	ϕ_2	ϕ_3	ϕ_4
$\mathcal{I}'(var)$	0	0	0	1	0	1	1	0	1	1	1	1	1
$cost$	0	0	0	0	2	0	0	5	0	0	0	0	0

$cost(\mathcal{I}') = 2 + 3 = 7$ □ **Exercise 12:**

Suppose that you have to place m queens in an $n \times n$ chess board, with $m \leq n^2$. Every queen i can be placed in $p(i) = (x_i, y_i)$ with $1 \leq x_i, y_i \leq n$, so that it cannot be "eaten" by any other queen. Encode the problem of finding the configuration that maximizes the total distance between the items, i.e.,

$$\sum_{1 \leq i,j \leq m} (x_i - x_j)^2 + (y_i - y_j)^2$$

.

Exercise 13:

Thransform the following MaxSat problem in and Integer Programming problem

$$\begin{aligned} \infty &: a \leftrightarrow \neg b \\ 2 &: a \wedge \neg b \rightarrow r \\ 3 &: b \wedge \neg a \rightarrow r \end{aligned}$$

Solution Since we have two weighted formulas we introduce two new propositional variables b_1 and b_2 and transform the clauses in the following integer constraints:

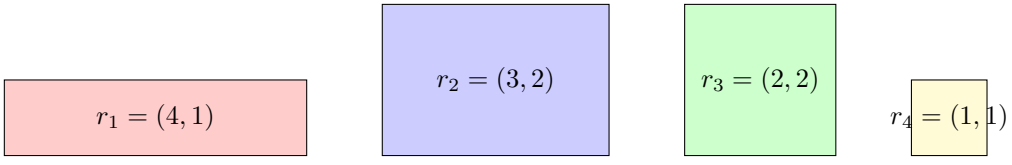
$$\begin{aligned} 1 \leq a \leq 1 \\ 1 \leq b \leq 1 \\ 1 \leq r \leq 1 \\ a + b = 1 \\ 0 \leq b_1 \leq 1 \\ 0 \leq b_2 \leq 1 \\ b_1 + (1 - a) + b + r \geq 1 \\ b2 + (1 - b) + a + r \geq 1 \end{aligned}$$

with the cost function: $2b_1 + 3b_2$. □

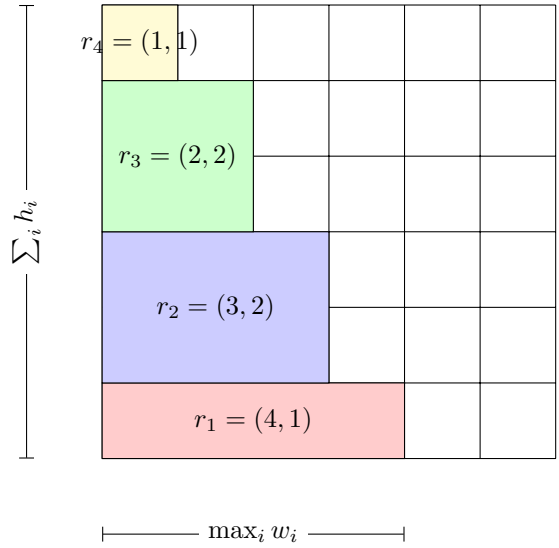
Exercise 14:

Suppose that you hae R rectangles $r_1, \ldots, r_R$, where each rectangle r_i has dimensions (w_i, h_i), with w_i and h_i natural numbers. Suppose that you have to arrange them inside an $n \times n$ square so that they don't overlap. Provide a MaxSat formulation of the problem of funding the smallest n for which theis is possible.

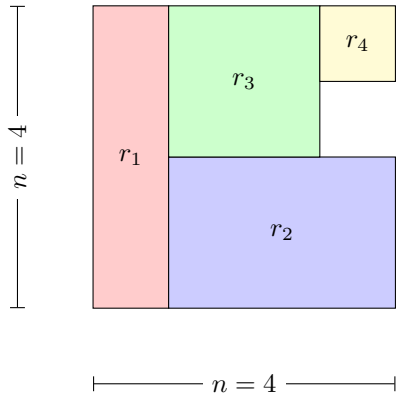
Solution To explain the general solution, let us consider an example with the following four rectangles:



Without loss of generality we can assume that $w_i \leq h_i$. Let $N = \max(\sum_i h_i, \max_j w_j)$. Notice that it is always possible to arrange the rectangles in an $N \times N$ square by stacking them, as shown in the following picture.



However this is not the optimal placement. An optimal placement would be, for instance, the following:



Let us formulate the problem to find such an optimal placement as a MaxSat problem. n We first provide the set of hard constraints that must be satisfied by every solution of the problem (not only the optimal ones). We use the propositions $H(i, j, k)$ and $V(i, j, k)$ to state that the i-th rectangle has been positioned horizontally or vertically respectivelly, in the position j, k where the position refers to the bottom left corner of the rectangle.

For instance the positioning of the first picture is the following:

$$H(1, 0, 0),\ H(2, 0, 1),\ H(3, 0, 3),\ H(4, 0, 5)$$

while that of the second picture is the following:

$$V(1, 0, 0),\ H(2, 1, 0),\ H(3, 1, 2),\ H(4, 3, 3)$$

Let us formulate the hard constraints:

(1) Every rectangle should be positioned either vertically or horizontally inside the $N \times N$ square. For every $i \in \{1, \ldots, R\}$ we add

$$\bigvee_{j=0}^{N-w_i} \bigvee_{k=0}^{N-h_i} H(i,j,k) \veebar V(i,k,j)$$

where $\veebar$ is the disjunctive or.

(2) Rectangles cannot overlap. If a rectangle is positioned in j, k then the other rectangles cannot be positioned in the slots occupied by the rectangle.

$$H(i,j,k) \rightarrow \bigwedge_{j'=0}^{w_i-1} \bigwedge_{k'=0}^{h_i-1} \bigwedge_{i' \neq i} \neg H(i', j+j', k+k') \wedge \neg V(i', j+j', k+k')$$

$$V(i,j,k) \rightarrow \bigwedge_{j'=0}^{h_i-1} \bigwedge_{k'=0}^{w_i-1} \bigwedge_{i' \neq i} \neg H(i', j+j', k+k') \wedge \neg V(i', j+j', k+k')$$

We then have to add some minimization criteria. The main idea is that if the top right corner of a rectangle covers the position j, k you will pay a cost of $(R+1)^{\max(j,k)}$. This can be obtained by adding the following weighted formula, one for every rectangle

$$H(i,j,k) : (R+1)^{\max(j+w_i, k+h_i)}$$

$$V(i,j,k) : (R+1)^{\max(j+h_i, k+w_i)}$$

Why we choose such a cost function. Because the cost of placing one single rectangle outside the $n \times n$ square will be larger than the cost of putting all the rectangles inside the $n \times n$ square. Indeed, if all the rectangles are inside the $n \times n$ square, the maximal cost that you will pay is

$$R(R+1)^n$$

If instead you put one single rectangle which goes out the $n \times n$ square you will pay a cost larger than

$$(R+1)^{n+1}$$

which it is larger than $R(R+1)^n$. This will force the system to search the minimal n. □

Bibliography

Boros, Endre and Peter L Hammer (2002). "Pseudo-boolean optimization". In: *Discrete applied mathematics* 123.1-3, pp. 155–225.

Fu, Zhaohui and Sharad Malik (2006). "On solving the partial MAX-SAT problem". In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer, pp. 252–265.

Manquinho, Vasco, Joao Marques-Silva, and Jordi Planes (2009). "Algorithms for weighted boolean optimization". In: *International conference on theory and applications of satisfiability testing*. Springer, pp. 495–508.

Mohamedou, Nouredine Ould and Jordi Planes (2009). "Solver MaxSatz in Max-SAT Evaluation 2009". In: *SAT 2009 competitive events booklet: preliminary version*, p. 155.