# Max-SMT

Albert Rubio

Programación con restricciones,
Facultad de Iniformática, UCM

# Max-SAT

The "maximum satisfiability problem" is defined as:

Given a conjunction of clauses (disjunction) over boolean literals

$$(p \lor q) \land (p \lor \neg q) \land (\neg p \lor q) \land (\neg p \lor \neg q)$$

Find a soluction (model) that

- **maximizes** the number of satisfied clauses

# Partial Max-SAT

Given a conjunction of clauses (disjunction) over boolean literals hard

$$(p \lor q) \land (p \lor \neg q)$$

and a conjunction of clauses (disjunction) over boolean literals soft

$$(\neg p \lor q) \land (\neg p \lor \neg q)$$

Find a solución that

- satisfies all hard clauses
- **maximizes** the number of satified soft clauses

# (Generalized) Partial Max-SAT

Given a conjunction of formulas (constraints) over boolean literals hard

$$(p \lor q) \land (p \lor \neg q)$$

and a conjunction of formulas (constraints) over boolean literals soft

$$(\neg p \lor q) \land (\neg p \lor \neg q)$$

Find a solución that

- satisfies all hard constraints
- **maximizes** the number of satified soft constraints

# Weighted (partial) Max-SAT

Given a conjunction of formulas (constraints) over boolean literals hard

$$(p \lor q) \land (p \lor \neg q)$$

and a conjunction of formulas (constraints) over boolean literals soft with a weight

$$(\neg p \lor q : w(3)) \land (\neg p \lor \neg q : w(2))$$

Find a solución that

- satisfies all hard constraints
- **maximize** the sum of the weights of satified soft constraints

# Weighted (partial) Max-SAT

Given a conjunction of formulas (constraints) over boolean literals hard

$$(p \ \vee \ q) \ \wedge \ (p \ \vee \ \neg q)$$

and a conjunction of formulas (constraints) over boolean literals soft with a weight

$$(\neg p \ \vee \ q \ : w(3)) \ \wedge \ (\neg p \ \vee \ \neg q \ : w(2))$$

Find a solución that

- satisfies all hard constraints
- **maximize** the sum of the weights of satified soft constraints
  **minimize** the sum of the weights of not satified soft constraints

# (Weighted partial) Max-SMT

Given a conjunction of hard constraints over some theory $T$ (e.g. LIA)

$$(x \leq y \ \vee \ x \leq z) \ \wedge \ (x \leq y \ \vee \ x > z)$$

and a conjunction of weighted constraints over some theory $T$

$$(x > y \ \vee \ x \leq z \ : w(3)) \ \wedge \ (x > y \ \vee \ x > z \ : w(2))$$

Find a solución that

- satisfies all hard constraints
- **maximize** the sum of the weights of satified soft constraints
  **minimize** the sum of the weights of not satified soft constraints

# Max-SMT with smtlib2

This is not fully standard, but ...

```
(assert-soft (<= t 8) :weight 2)
```

The weight must be a positive natural number, but is optional.

```
(assert-soft (<= t 8))
```

If omitted, the weight is set to 1.

# Max-SMT with Z3

We can have more than one kind of weights

```
(assert-soft (<= t 8) :weight 2 :id A)
(assert-soft (<= 0 t) :weight 2 :id B)
```

Finds the best solution lexicographically (in order of appearence of tags):

- **maximize** the sum of the weights of satified **A** soft constraints
- If same weight for **A**
  **maximize** the sum of the weights of satified **B** soft constraints
- ...

Choose the names you like for the tags!!

# Max-SMT with z3py

We should use `add_soft` instead of `add`:

```
add_soft (self, arg, weight="1", id=None)
```

Given no optimizer `s` we write:

```
s = Optimize()
...
s.add_soft (t <= 8)
s.add_soft (t <= 8, 2)
s.add_soft (t <= 8, 2, "A")
```

# Optimization Modulo Theories (OMT)

We can add an objective function.

"still not fully competitive", but ...

```
(minimize t)
(maximize t)
```

where $t$ is an expression of type integer or real.

Must be added before `(check-sat)`

With z3py use:

```
s.minimize(t)
s.maximize(t)
```

# OMT extensions and tools

We can have multiple objective functions.

Finds the best solution lexicographically (in order of appearence of objectives).

Other combination options can be considered: pareto, independent, ... (check the solver)

Available tools:

- Z3 `https://github.com/Z3Prover/z3`
- (Opti)MathSat `http://optimathsat.disi.unitn.it`

# Example

(Borrowed from Z3 tutorial)

Queremos crear tres máquinas virtuales que requieren 100, 50 y 15 GB de disco respectivamente.

Disponemos de tres servidores con 100, 75 and 200 GB de disco respectivamente.
Además estos tres servidores tienen un coste diario de 10, 5 y 20 euros respectivamente.

Encontrar una solución (usando Max-SMT) que decida en que servidor se crea cada máquina virtual tal que:

- Minimize el número de servidores usados
- Minimize el coste diario

# Example

Usad variables booleanas

- `x_i_j` para expresar que la MV $i$ se crea en el servidor $j$
- `y_j` para expresar que el servidor $j$ se está usando

```
(declare-fun x_1_1 () Bool)
(declare-fun x_1_2 () Bool)
(declare-fun x_1_3 () Bool)
(declare-fun x_2_1 () Bool)
(declare-fun x_2_2 () Bool)
(declare-fun x_2_3 () Bool)
(declare-fun x_3_1 () Bool)
(declare-fun x_3_2 () Bool)
(declare-fun x_3_3 () Bool)

(declare-fun y_1 () Bool)
(declare-fun y_2 () Bool)
(declare-fun y_3 () Bool)
```

# Example

Otra codificación:

Usad variables

- x_i Int, para expresar que la MV $i$ se crea en el servidor $j$
- y_j Bool, para expresar que el servidor $j$ se está usando

```
(declare-fun x_1 () Int)
(declare-fun x_2 () Int)
(declare-fun x_3 () Int)

(declare-fun y_1 () Bool)
(declare-fun y_2 () Bool)
(declare-fun y_3 () Bool)
```