Current trends in Max-SAT solving

Javier Larrosa
Universitat Politècnica de Catalunya
Barcelona, Spain

Why Max-SAT?

- **Optimization version of SAT**
  - Most famous NP-complete problem
  - Modeling language for optimization problems
  - SAT (Max-SAT) Strongly related to CSP (weighted CSP)
- **Many practical applications**
  - **Physics**: ising models (statistical mech.)
  - **Telecommunications**: frequency assignment
  - **Scheduling**: satellite scheduling (SPOT5)
  - **Electronic markets**: combinatorial auctions
  - **Bioinformatics**: protein alignment
  - **Circuits design**: debugging, logic synthesis
  - ...

Why now?

- Before 2005
  - Significant theoretical work
  - Very little algorithmic work
- After 2005
  - Significant algorithmic work
  - New applications identified
  - 3 competitions/evaluations
  - more than 10 solvers

Summary

- Preliminaries: SAT and Max-SAT
- Solving by reducing to SAT
  - Unsatisfiable cores
- Solving by inference
  - Max-sat resolution
- Solving by search (lower bounds)
  - SAT Unit Propagation
- Adding weights and hard clauses

Notation

x, y, ...	Boolean variables
$\neg x$, $\neg y$	Negation
l_1, l_2,...	Literals
$C=l_1 \vee l_2 \vee \ldots \vee l_k$	Clause
l	Unit clause
$\square$	Empty clause

Notation

$\Phi = \{C_1, C_2, \ldots\}$	CNF formula
$A=\{l_1, l_2, \ldots\}$	Assignment
➢ $x \in A$	"x is set to *true*"
➢ $\neg x \in A$	"x is set to *false*"

Example: $\Phi = \{x, y, x \vee \neg y, \neg x \vee \neg p\}$

$A=\{x, \neg y, \neg p\}$ does not satisfy Φ

$A=\{x, y, \neg p\}$ satisfes Φ

Boolean SATisfiability

- SAT(Φ)= is it possible to simultaneously satisfy every clause in Φ (i.e. does it have a model)?
- Ex: Φ ={x,y,¬x∨¬y}
  - SAT(Φ)=false
- If $\square \in \Phi$ then SAT(Φ)= false

Unsatisfiable Cores (UC)

- Let Φ be a unsatisfiable CNF formula.
- **Definition**: $\delta \subseteq \Phi$ is an *unsatisfiable core* (UC) if SAT(δ)=false
- **Example**: Φ ={x,y,¬x∨¬y,p,q}
  δ ={x,y,¬x∨¬y}
- **Observation**:
  UCs needs not to be minimal
  There may be several UCs
  Current SAT solvers can provide near-minimal UCs

Unit Propagation (UP)

- Simplification process
- Apply the following rule until fixpoint:
  If $\{l, \neg l \vee C\} \subseteq \Phi$ then replace $\neg l \vee C$ by C
- **Example**: $\Phi = \{x, q, \neg x \vee \neg y, y \vee p, \neg x \vee \neg p\} =$
  $= \{x, q, \neg y, y \vee p, \neg p\} =$
  $= \{x, q, \neg y, p, \neg p\} =$
  $= \{x, q, \neg y, p, \square\}$

Unit propagation (UP)

- Polytime process
- Sometimes, they can identify a special type of unsatisfiable core
- **Example**: $\Phi = \{x, \neg x \vee q, \neg x \vee \neg y, y \vee p, \neg x \vee \neg p\} =$
  $= \{x, q, \neg y, y \vee p, \neg p\} =$
  $= \{x, q, \neg y, p, \neg p\} =$
  $= \{x, q, \neg y, p, \square\}$

Solving SAT

- **Inference** (e.g. DP [Davis & Putnam, 60])
  - Resolution
- **Backtracking search** (e.g. DPLL [Davis et al, 62]
  - Splitting + Unit propagation

Maximum Satisfiability (Max-SAT)

- Max-SAT(Φ)=maximum number of clauses that can be simultaneously satisfied

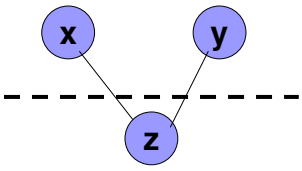
or, equivalently,

- Max-SAT(Φ)=minimum number of clauses that can be simultaneously falsified

Max-SAT

- Ex: $\Phi = \{x, y, \neg x \vee \neg y\}$
  - Solutions: $A_1=\{x,y\}$, $A_2=\{x,\neg y\}$, $A_3=\{\neg x,y\}$,
  - As maximization: Max-SAT(Φ)=2
  - As minimization: Max-SAT(Φ)=1

Example: Max-cut



$\Phi = \{x \vee z, \neg x \vee \neg z, y \vee z, \neg y \vee \neg z\}$

$A = \{x, y, \neg z\}$

Solving Max-SAT with SAT solvers

Why is it a good idea?

- Modern SAT solvers are very efficient
  - Conflict-based Learning
  - Backjumping
  - Two-watched literals
  - Heuristics (VSIDS)

Blocking variables

- **relax(Φ):** adds a fresh literal to each clause (blocking variables)
  relax(Φ)={$C_i \vee b_i$ | $C_i \in \Phi$}

➢ **Example**: Φ ={x,y,¬x∨¬y}
relax(Φ)={$x \vee b_1$,$y \vee b_2$,$\neg x \vee \neg y \vee b_3$}

- Setting blocking variables to true means relaxing their clause

Max-SAT to SAT

Auxiliary clauses

- Max-SAT(Φ)= min {k | SAT(Φ'∪α^k)}
  Φ'=Relax(Φ)
  α^k = CNF(Σb_i=k)

Sorting networks
BDDs
...

➢ **Example**: Φ ={x,y,¬x∨¬y}
Max-SAT(Φ)= 1 because
SAT{$x \vee b_1$,$y \vee b_2$,$\neg x \vee \neg y \vee b_3$}∪CNF($b_1+b_2+b_3=1$)

1st approach (Minisat+)

[Een & Sorensson,06]

Función Max-SAT(Φ) : int

Φ':=relax(Φ);

k:=0

while(**not** SAT($\Phi' \cup \alpha^k$)) **do** *k:=k+1*

return k;

Example

- Φ ={x,y,¬x∨¬y,p,q}
  α={}

1. Φ={x∨b_1,y∨b_2,¬x∨¬y∨b_3, p∨b_4, q∨b_5}
   α={b_1+b_2+b_3+b_4+b_5=0}
2. Φ={x∨b_1,y∨b_2,¬x∨¬y∨b_3, p∨b_4, q∨b_5}
   α={b_1+b_2+b_3+b_4+b_5=1}
3. Φ={x∨b_1,y∨b_2,¬x∨¬y∨b_3, p∨b_4, q∨b_5}
   α={b_1+b_2+b_3+b_4+b_5=2}
4. …

Main drawback: relax(Φ) adds too many new variables
the search space becomes prohibitive

2nd approach [Fu & Malik,06]

- **Motivation**: reduce the number of blocking variables
- **Observation**: Given a Unsatisf. Core, one of its clauses needs to be falsified
- **Idea**: Add blocking variables only to UCs

2nd approach (naive version)

Función Max-SAT(Φ) : int
  k:=0; α:=$\varnothing$
  while(**not** SAT($\Phi \cup \alpha$)) **do**
    (Φ,α)**:=**T(Φ,α)
    k:=k+1
  return *k*;

T(Φ, α) :
  Φ:=relax(Φ)
  α:=$\alpha \cup$CNF($\Sigma b_i = 1$)
  return (Φ,α);

Example

1. $\Phi = \{x, y, \neg x \vee \neg y, p, q\}$
   $\alpha = \{\}$
2. $\Phi = \{x \vee b_1, y \vee b_2, \neg x \vee \neg y \vee b_3, p \vee b_4, q \vee b_5\}$
   $\alpha = \{b_1 + b_2 + b_3 + b_4 + b_5 = 1\}$
3. $\Phi = \{x \vee b_1 \vee b_6, y \vee b_2 \vee b_7, \neg x \vee \neg y \vee b_3 \vee b_8, p \vee b_4 \vee b_9, q \vee b_5 \vee b_{10}\}$
   $\alpha = \{b_1 + b_2 + b_3 + b_4 + b_5 = 1,\ b_6 + b_7 + b_8 + b_9 + b_{10} = 1\}$
4. ...

2nd approach

Función Max-SAT(Φ) : int
  k:=0; $\alpha := \varnothing$
  while(**not** SAT($\Phi \cup \alpha$)) **do**
    (Φ, α)**:=**T(Φ, α)
    k:=k+1
  return *k*;

T(Φ, α) :
  δ:=Unsat core of $\Phi \cup \alpha$
  $\delta := \delta \cap \Phi$
  δ':=relax(δ)
  $\Phi := (\Phi - \delta) \cup \delta'$
  $\alpha := \alpha \cup \text{CNF}(\Sigma b_i = 1)$
  return (Φ, α);

Example

1. $\Phi = \{x, y, \neg x \vee \neg y, p, q\}$
   $\alpha = \{\}$
2. $\Phi = \{x \vee b_1, y \vee b_2, \neg x \vee \neg y \vee b_3, p, q\}$
   $\alpha = \{b_1 + b_2 + b_3 = 1\}$
3. ...

•May add more than one blocking variable per clause
•May avoid to add blocking variables to some clauses
•Works well If the original formula has small UCs

3rd approach [Marques-Silva & Planes 08]

- **Motivation**: reduce the number of blocking variables
  - Only one blocking variables per clause (as in 1st approach)
  - Only blocking variables to UCs (as in 2nd approach)

3rd approach [Marques-Silva & Planes 08]

- **Idea**: SAT($\Phi \cup \{\sum b_i \leq ub\}$)
  - If satisfiable, then decrease upper bound (ub)
  - If unsatisfiable, then add blocking variables to original clauses in the UC
  - Solution: number of unsat. iterations

Inference (Max-SAT resolution)

SAT truth tables

- Ex: $\Phi = \{x, y, \neg x \vee \neg y\}$

  SAT(Φ)=false

x	y	Φ
F	F	F
F	T	F
T	F	F
T	T	F

Max-SAT cost tables

- Ex: $\Phi = \{x, y, \neg x \vee \neg y\}$

  Max-SAT(Φ)=1

x	y	# falsified clauses Φ
F	F	2
F	T	1
T	F	1
T	T	1

Equivalence

- SAT: $\Phi \equiv \Phi'$ iff same truth table
- Ex: $\Phi = \{x, y, x \vee y\}$, $\Phi' = \{x, y\}$,

x	y	Φ	Φ'
0	0	F	F
0	1	F	F
1	0	F	F
1	1	T	T

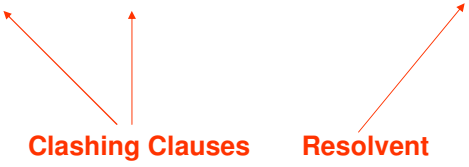
Equivalence

- Max-SAT: $\Phi \equiv \Phi'$ iff same cost table
- Ex: $\Phi = \{x, y, x \vee y\}$, $\Phi' = \{x, y\}$,

x	y	Φ	Φ'
F	F	3	2
F	T	1	1
T	F	1	1
T	T	0	0

SAT resolution

$\{x \vee C, \neg x \vee C'\} \equiv \{x \vee C, \neg x \vee C', C \vee C'\}$



x	C	C'	
F	F	F	F
F	F	T	F
F	T	F	T
F	T	T	T
T	F	F	F
T	F	T	T
T	T	F	F
T	T	T	F

Does it work for Max-SAT?

$\{x \vee C, \neg x \vee C'\} \equiv$
$\{C \vee C', x \vee C, \neg x \vee C'\}$

x	C	C'		
F	F	F	1	1+1
F	F	T	1	1
F	T	F	0	0
F	T	T	0	0
T	F	F	1	1+1
T	F	T	0	0
T	T	F	1	1
T	T	T	0	0

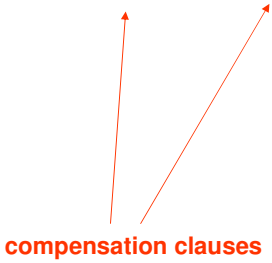
Does it work for Max-SAT?

$\{x \vee C, \neg x \vee C'\} \equiv$
$\{C \vee C'\}$

x	C	C'		
F	F	F	1	1
F	F	T	1	0
F	T	F	0	0
F	T	T	0	0
T	F	F	1	1
T	F	T	0	0
T	T	F	1	0
T	T	T	0	0

Does it work for Max-SAT?

$\{x \vee C, \neg x \vee C'\} \equiv$
$\{C \vee C', x \vee C \vee \neg C', \neg x \vee \neg C \vee C'\}$



x	C	C'		
F	F	F	1	1
F	F	T	1	1
F	T	F	0	0
F	T	T	0	0
T	F	F	1	1
T	F	T	0	0
T	T	F	1	1
T	T	T	0	0

Resolution [L. et al, 06]

$\{x \vee C, \neg x \vee C'\} \equiv \{C \vee C', x \vee C \vee \neg C', \neg x \vee \neg C \vee C'\}$

- **Problem**: compensation clauses are not clauses
- **Example**:

$\{x \vee y \vee z, \neg x \vee p \vee q\} \equiv$
$\{y \vee z \vee p \vee q, x \vee y \vee z \vee \neg(p \vee q), \neg x \vee \neg(y \vee z) \vee p \vee q\}$

Negation

- **Solution**:

$A \vee \neg(C \vee B) \equiv \{A \vee \neg C \vee B , A \vee \neg B\}$

- **Example**:

$x \vee \neg(y \vee z \vee p \vee q) \equiv \{x \vee \neg y \vee z \vee p \vee q, x \vee \neg(z \vee p \vee q)\}$
$\equiv \ldots \equiv$
$\{x \vee \neg y \vee z \vee p \vee q, x \vee \neg z \vee p \vee q, x \vee \neg p \vee q, x \vee \neg q\}$

Resolution

- Resolution is
  - Correct: preserves Max-SAT equivalence
  - Complete: Suffices to solve Max-SAT

Resolution completeness

Theorem [Bonet et al, 07] :

if Max-SAT(Φ)=k then

$$\Phi \equiv \{\underbrace{\square,\ldots,\square}_{\text{k times}}\} \cup \Phi' \quad \text{satisfiable}$$

and we can make the transformation using only resolution

Resolution completeness

Alg.: Directional resolution (Davis and Putnam 60)

Let $(x_1,x_2,...,x_n)$ a variable ordering

1. Resolve over x_1 until saturation
2. Resolve over x_2 until saturation
3. ...
4. Resolve over x_n until saturation

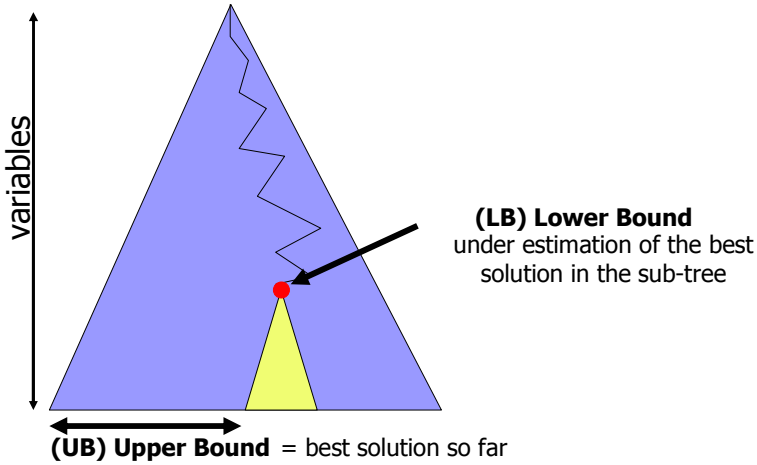
Complexity

- **Theorem** [L et al, 07]: The complexity is time and space exponential on the formula's *induced width* (i.e. cyclicity)

- Same complexity as:
  - SAT case (Davis and Putnam, 60)
  - Decomposition Methods (CSPs)
  - Bucket Elimination (Soft CSPs)

Search (branch and bound)

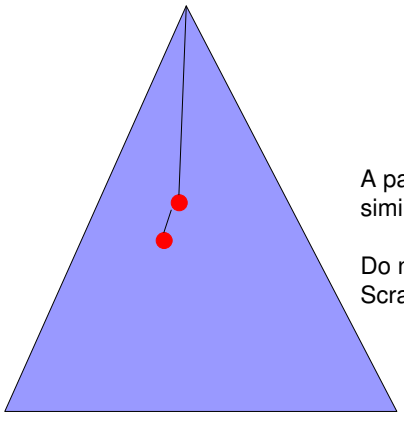
Branch and bound



Lower Bounds

- Pruning condition
  - If LB(current subproblem)≥ UB then backtrack
- Good Lower Bounds are essential
- Desired properties:
  - Cheap/Tight (tradeoff)
  - Incremental

Incrementality



A parent and its child are similar subproblems

Do not compute the child LB from Scratch (reuse its parent's)

Lower bounds

- **Property**: Let Φ be a unsatisfiable CNF formula and $\{U_1, U_2, ..., U_k\}$ a set of disjoint unsatisfiable cores. Then,

$$k \leq \text{Max-SAT}(\Phi)$$

- **Idea**: Look for special kinds of Ucs
  - E.g. use SAT unit propagation

1st approach [Li et al, 05]

- **function** UnitProp(Φ) **returns** (*bool*, δ)

  bool tells if UP has produced the empty clause

  $\delta \subseteq \Phi$ is the set of clauses involved in the conflict

- **Example**:

  $\Phi = \{\neg x, x \vee q, x \vee y, x \vee p, \neg y \vee \neg p\}$

  UnitProp(Φ)=(true,$\{\neg x, x \vee y, x \vee p, \neg y \vee \neg p\}$)

1st approach

```
function LB(Φ) ret int
  k:=0
  repeat
    (b,δ):=UnitProp(Φ)
    if (b) then k:=k+1; Φ:=Φ–δ
  until not b
ret k
```

Example

Φ={¬x, x∨q, x∨y, x∨p, ¬y∨¬p, r, s, ¬r∨¬s}

1. Φ={x∨q, r, s, ¬r∨¬s} LB=1
2. Φ={x∨q} LB=2

Return 2

Drawback: non-incremental

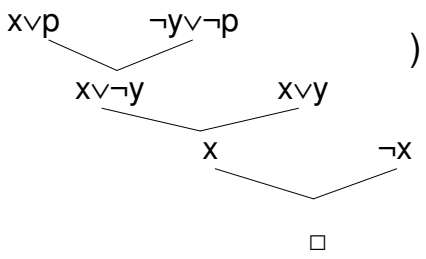
2nd approach (minimaxsat)
[Heras et al, 08]

- **Goal:** incrementality
- **Idea**: make the LB explicit via Max-SAT resolution
- **function** UnitProp(Φ) **returns** (*bool*, T)
  - *bool* tells if UP has produced the empty clause
  - T is a SAT refutation tree associated to the conflict

2nd approach

- **Example**:

  Φ={¬x, x∨q, x∨y, x∨p, ¬y∨¬p}

  UnitProp(Φ)=(true, 

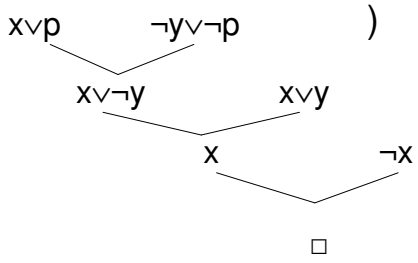
)

2nd approach

function LB(Φ) **ret** CNF
  k:=0
  repeat
    (b,T):=UnitProp(Φ)
    if (b) **then** apply T to Φ /*using Max-sat resolution */
  until not b
ret Φ

Example

Φ={¬x, x∨q, x∨y, x∨p, ¬y∨¬p, r, s, ¬r∨¬s}

UnitProp(Φ)=(true, 

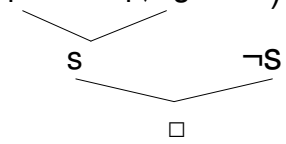
)

Φ={□, x∨p∨y, ¬x∨¬p∨¬y, x∨q, r, s, ¬r∨¬s}

Example (cont.)

Φ={□, x∨p∨y, ¬x∨¬p∨¬y, x∨q, r, s, ¬r∨¬s}

UnitProp(Φ)=(true, r ¬r∨¬s)



Φ={□, x∨p∨y, ¬x∨¬p∨¬y, x∨q, □, ¬r∨¬s}

Adding weights and hard clauses

(a.k.a. making it practical)

Weights in Max-SAT

- In many applications:
  - different clauses may have different importance

    (C,w) weighted clause
  - Some clauses may be mandatory

    (C,T) mandatory clause

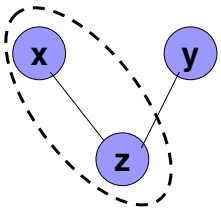
    T not strictly necesary

    (C,10000000) is essentially mandatory

Weights in Max-SAT

- **Observation**:
  - $(\square, w) \in \Phi$ then $w \leq$ Max-SAT(Φ)
  - $(\square, T) \in \Phi$ then the mandatory part of Φ are unsatisfiable

Example: Max-clique



Φ= {($\neg x \vee \neg y$,T), (x,1), (y,1), (z,1)}

A={x,$\neg y$,z}

Equivalence

- $\Phi \equiv_k \Phi'$ iff
  - Max-SAT(Φ)≥k and Max-SAT(Φ')≥k or
  - Max-SAT(Φ)<k and Max-SAT(Φ')<k and Max-SAT(Φ)=Max-SAT(Φ')
- Useful inside branch and bound
  - If Φ is the current node's subproblem and
  - k is the current upper bound
- If $\Phi \equiv_k \Phi'$ and Φ' is simpler, replace Φ by Φ'

Simplification Rules

- Hardening
- Subsumption
- Unit clause reduction
- Pure Literal Rule

Hardening

- Identification of hard clauses
- if $w \geq k$ then $(C,w) \equiv_k (C,T)$
- if $C' \rightarrow C$ and $w'+w \geq k$ then
  $\{(C',w'), (C,w)\} \equiv_K \{(C',w'), (C,T)\}$

Example

$\{(x \vee y,10), (\neg x,2), (\neg y,2), (\square,2)\} \equiv_4$
$\{(x \vee y,T), (\neg x,2), (\neg y,2), (\square,2)\} \equiv_4$
$\{(x \vee y,T), (\neg x,T), (\neg y,T), (\square,2)\}$

Subsumption

- Elimination of redundant clauses
- if $C' \rightarrow C$ then $\{(C',T),(C,w)\} \equiv_k \{(C',T)\}$

- Ex: $\{(x,T), (x \vee y,2), (x \vee z,1)\} \equiv_k \{(x,T)\}$

Unit Clause Reduction

- Propagation of facts
- $\{(l,T), (\neg l \vee C,w)\} \equiv_K \{(l,T), (C,w)\}$

- Ex: $\{(\neg x,T), (x \vee y,2), (x \vee z,1)\} \equiv_k$
  $\{(\neg x,T), (y,2), (z,1)\}$

Example ($\equiv_5$)

$(\square,1), \underline{(\neg x,3)}, \underline{(\neg x,1)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

Example ($\equiv_5$)

$(\square,1), (\underline{\neg x,3}), (\underline{\neg x,1}), (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$(\underline{\square,1}), (\underline{\neg x,4}), (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

Example ($\equiv_5$)

$(\square,1), (\underline{\neg x,3}), (\underline{\neg x,1}), (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$(\underline{\square,1}), (\underline{\neg x,4}), (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$(\square,1), (\underline{\neg x, \top}), (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

Example ($\equiv_5$)

$(\square,1), \underline{(\neg x,3)}, \underline{(\neg x,1)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$\underline{(\square,1)}, \underline{(\neg x,4)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$(\square,1), \underline{(\neg x, \top)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$\underline{(\square,2)}, \underline{(y,3)}, (\neg y,2), (\neg y \vee z,1)$

Example ($\equiv_5$)

$(\square,1), \underline{(\neg x,3)}, \underline{(\neg x,1)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$\underline{(\square,1)}, \underline{(\neg x,4)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$(\square,1), \underline{(\neg x, \top)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$\underline{(\square,2)}, \underline{(y,3)}, (\neg y,2), (\neg y \vee z,1)$

$\downarrow$

$(\square,2), \underline{(y, \top)}, (\neg y,2), (\neg y \vee z,1)$

Example ($\equiv_5$)

$(\square,1), \underline{(\neg x,3)}, \underline{(\neg x,1)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$\underline{(\square,1)}, \underline{(\neg x,4)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$(\square,1), \underline{(\neg x, \top)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$\underline{(\square,2)}, \underline{(y,3)}, (\neg y,2), (\neg y \vee z,1)$

$\downarrow$

$(\square,2), \underline{(y, \top)}, (\neg y,2), (\neg y \vee z,1)$

$\downarrow$

$(\square,4), \underline{(z,1)}$

Example ($\equiv_5$)

$(\square,1), \underline{(\neg x,3)}, \underline{(\neg x,1)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$\underline{(\square,1)}, \underline{(\neg x,4)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$(\square,1), \underline{(\neg x, \top)}, (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$\underline{(\square,2)}, \underline{(y,3)}, (\neg y,2), (\neg y \vee z,1)$

$\downarrow$

$(\square,2), \underline{(y, \top)}, (\neg y,2), (\neg y \vee z,1)$

$\downarrow$

$(\square,4), \underline{(z,1)} \longrightarrow (\square,4), \underline{(z, \top)}$

Example ($\equiv_5$)

$(\square,1), (\underline{\neg x,3}), (\underline{\neg x,1}), (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$(\underline{\square,1}), (\underline{\neg x,4}), (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$(\square,1), (\underline{\neg x, \top}), (x,1), (y,3), (\neg y,2), (x \vee \neg y \vee z,1)$

$\downarrow$

$(\underline{\square,2}), (\underline{y,3}), (\neg y,2), (\neg y \vee z,1)$

$\downarrow$

$(\square,2), (\underline{y, \top}), (\neg y,2), (\neg y \vee z,1)$

$\downarrow$

$(\square,4), (\underline{z,1}) \rightarrow (\square,4), (\underline{z, \top}) \rightarrow (\square,4)$

Extension of Resolution

$$(x \vee C, w), \quad (\neg x \vee C', u) \quad \equiv_k \quad \begin{array}{l} (C \vee C', m), \\ (x \vee C, w\text{-}m), \\ (\neg x \vee C', u\text{-}m), \\ (x \vee C \vee \neg C', m), \\ (\neg x \vee \neg C \vee C', m) \end{array}$$

$m=\min\{w,u\}$

$\forall a$, T-a=T (T is absorptive)

Special cases (I)

$$(x \vee C, T),\ (\neg x \vee C', T) \equiv_k (C \vee C', m),\ (x \vee C, T\text{-}m),\ (\neg x \vee C', T\text{-}m),\ (x \vee C \vee \neg C', m),\ (\neg x \vee \neg C \vee C', m)$$

$\top = \min\{\top, \top\}$
$\forall a, T\text{-}a = T$ (T is absorptive)

Special cases (I)

$$(x \vee C, T),\ (\neg x \vee C', T) \equiv_k (C \vee C', T),\ (x \vee C, T),\ (\neg x \vee C', T),\ (x \vee C \vee \neg C', T),\ (\neg x \vee \neg C \vee C', T)$$

$\top = \min\{\top, \top\}$
$\forall a, T\text{-}a = T$ (T is absorptive)

Special cases (I)

$$(x \vee C, T),\ (\neg x \vee C', T) \equiv_k (C \vee C', T),\ (x \vee C, T),\ (\neg x \vee C', T),$$

$\top = \min\{\top, \top\}$
$\forall a, T\text{-}a = T$ (T is absorptive)

Special cases (II)

$$(x \vee C, w),\ (\neg x \vee C, u) \equiv_k (C \vee C, m),\ (x \vee C, w\text{-}m),\ (\neg x \vee C, u\text{-}m),\ (x \vee C \vee \neg C, m),\ (\neg x \vee \neg C \vee C, m)$$

$m = \min\{w, u\}$
$\forall a, T\text{-}a = T$ (T is absorptive)

Special cases (II)

$$(x \vee C, w),\ (\neg x \vee C, u) \equiv_k (C \vee C, m),\ (x \vee C, w-m),$$

$m=\min\{w,u\}$
$\forall a,\ T\text{-}a=T$ (T is absorptive)

Conclusion

- Max-SAT (with weights and hard clauses) is a handy modeling language with many applications
- In the last years we have witness an algorithmic breakthrough
  - Max-SAT resolution
  - Max-SAT inference rules
  - SAT unsatisfiable cores
- Still space for improvement

Bibliography

- Check the *AI Journal* and the *Journal of AI Research* in the 2006-2008 period