

# OPTSAT: A Tool for Solving SAT-related Optimization Problems

**Enrico Giunchiglia<sup>1</sup> and Marco Maratea<sup>1,2</sup>**

<sup>1</sup>Laboratory of Systems and Technologies for Automated Reasoning (STAR-Lab)  
DIST - University of Genova

<sup>2</sup>Department of Mathematics, University of Calabria

10th European Conference on Logics in Artificial Intelligence  
Liverpool, September 13, 2006

# Goals of the work

- 1 Solve SAT-related optimization problems without a branch-and-bound schema
- 2 Show how optimal models can be computed via a simple modification to DLL
- 3 Show how MIN-ONE, MAX-SAT (but also other) problems can be recasted and solved in the framework
- 4 Show the viability of the approach with experimental analysis

# The approach

## Main idea

Modify the DLL heuristic by imposing a (partial) ordering to the literals to be followed while branching

What do we need? [Giunchiglia and Maratea, ECAI 2006]

A partial order  $\prec$  on a set of literals  $S$

# The approach

## Main idea

Modify the DLL heuristic by imposing a (partial) ordering to the literals to be followed while branching

## What do we need? [Giunchiglia and Maratea, ECAI 2006]

A partial order  $\prec$  on a set of literals  $S$

## Optimality by

- 1 cardinality (e.g., MIN-ONE), or
- 2 sub-set inclusions (e.g.,  $\text{MIN-ONE}_{\subseteq}$ )

## Setting in the case of cardinality

It is convenient to see a problem as

- 1 a CNF formula  $\varphi$
- 2 an optimization function defined on (a subset of) the atoms in  $\varphi$

## Why two different types?

For applications: e.g., in planning as satisfiability, if optimality is computed on the set of actions

- $\text{MIN-ONE}$   $\rightarrow$  as few as possible actions are executed
- $\text{MIN-ONE}_{\subseteq}$   $\rightarrow$  no redundant sequence of (possibly parallel) actions are executed

## Optimality by

- 1 cardinality (e.g., MIN-ONE), or
- 2 sub-set inclusions (e.g.,  $\text{MIN-ONE}_{\subseteq}$ )

## Setting in the case of cardinality

It is convenient to see a problem as

- 1 a CNF formula  $\varphi$
- 2 an optimization function defined on (a subset of) the atoms in  $\varphi$

## Why two different types?

For applications: e.g., in planning as satisfiability, if optimality is computed on the set of actions

- MIN-ONE  $\rightarrow$  as few as possible actions are executed
- $\text{MIN-ONE}_{\subseteq}$   $\rightarrow$  no redundant sequence of (possibly parallel) actions are executed

```

function OPT-DLL( $\varphi, \mu, \mathcal{S}, \prec$ )
1 if ( $\emptyset \in \varphi$ ) return FALSE;
2 if ( $\varphi = \emptyset$ ) return  $\mu$ ;
3 if ( $\{I\} \in \varphi$ ) return OPT-DLL(assign( $I, \varphi$ ),  $\mu \cup \{I\}, \mathcal{S}, \prec$ );
4  $I :=$  a minimal unassigned literal in  $\mathcal{S}, \prec$  if  $\exists$ 
   and a literal in  $\varphi$  otherwise.
5  $v :=$  OPT-DLL(assign( $I, \varphi$ ),  $\mu \cup \{I\}, \mathcal{S}, \prec$ );
6 if ( $v \neq$  FALSE) return  $v$ ;
7 return OPT-DLL(assign( $\bar{I}, \varphi$ ),  $\mu \cup \{\bar{I}\}, \mathcal{S}, \prec$ ).

```

**Facts:** [Boutilier, et al. 2004]

OPT-DLL returns an “optimal” model  $\mu$  if  $\varphi$  is satisfiable, and FALSE otherwise.

*assign()*

*assign*( $I, \varphi$ ) returns the formula obtained from  $\varphi$  by (i) deleting the clauses containing  $I$ , and (ii) deleting  $\bar{I}$  from the others.

# OPTSAT's input format and options

## Example

```
c min ONE
c subset
p cnf k m
```

## Format features

- extension of the well-known DIMACS format for SAT
- can be easily translated in other formats, e.g., the Weighted SAT format for MAX-SAT

## Options

```
optsat -solver <name> -<Minimality> -enc <Encoding>
```

- name  $\in \{minisat, zchaff\}$
- Minimality  $\in \{card, subset\}$
- Encoding  $\in \{wm, w, bb, bbm\}$

# Solving MIN-ONE and MIN-ONE<sub>⊆</sub> with OPT-DLL

## Setting

$\varphi$  is a formula ;  $P$  is the set of variables  $\{x_1, \dots, x_k\}$  in  $\varphi$  ;  
 $\bar{P} = \{\bar{x}_i : x_i \in P\}$

## MIN-ONE<sub>⊆</sub>

OPT-DLL( $\varphi, \emptyset, \bar{P}, \emptyset$ ) returns an optimal model if  $\varphi$  is satisfiable, and FALSE otherwise.

## MIN-ONE

- 1  $adder(P)$  is an *adder* of  $P$  (SAT-encoding of  $\sum_{i=1}^k x_i$ ), with output  $b_{n-1}, \dots, b_0$  ( $n = \lceil \log_2(|P| + 1) \rceil$ ),  
 $\sum_{i=1}^k x_i = \sum_{i=0}^{n-1} 2^i b_i$  for  $w^*$ ; and  $n = k$ ,  $\sum_{i=1}^k x_i = \sum_{i=1}^n b_{i-1}$  for  $bb^*$  encoding)
- 2 OPT-DLL( $\varphi \cup adder(P), \emptyset, P, \bar{b}_n \prec \bar{b}_{n-1} \prec \dots \prec \bar{b}_0$ ) returns an optimal model if  $\varphi$  is satisfiable, and FALSE otherwise.

# From MAX-SAT and MAX-SAT<sub>⊆</sub> to MIN-ONE and MIN-ONE<sub>⊆</sub>

## Intuition

- 1 Given a formula  $\varphi$ , a new variable  $v_i$  is added to the clause  $C_i \in \varphi$ , e.g.,  $\{\{x_1\}, \{x_2, x_3\}\}$  becomes  $\{\{v_1, x_1\}, \{v_2, x_2, x_3\}\}$ .
- 2 MAX-SAT and MAX-SAT<sub>⊆</sub> correspond to minimize the set of variables  $v_i$  assigned to 1, and thus reduce to MIN-ONE and MIN-ONE<sub>⊆</sub> problems

# MIN-ONE and MIN-ONE<sub>C</sub>: experimental results

	instance	#C	OPBDP	PBS4	MSAT+	OPTSAT	#C <sub>C</sub>	OPTSAT
1	bcomp5	39	0.95	2.34	0.4	7.08	85	0
2	bmax6	61	120.05	TIME	8.42	337.18	131	0.01
3	ibm2	940	TIME	TIME	19.73	1513.48	1054	0.03
4	ibm3	6356	TIME	–	TIME	52.16	6422	0.41
5	gal8		MEM	–	SF	TIME	14207	1.05
6	4blocks	108	TIME	115.96	50.94	TIME	116	0.06
7	log.b	138	TIME	TIME	8.99	4.69	138	0.02
8	qg1-8	64	TIME	81.46	31.06	117.71	64	0.88
9	qg2-8	64	MEM	54.26	21.83	38.77	64	12.7

- 1 OPTSAT implements OPT-DLL on top of MINISAT ver. 1.14.
- 2 *adder(P)* is implemented using [Warners 98] (“wum”).
- 3 OPBDP [Barth 95], PBS4 [Aloul, Ramani, Markov, Sakallah 02], MSAT+ [Eén, Sörensson 06] are PB solvers

# MAX-SAT and MAX-SAT<sub>C</sub>: experimental results

	instance	#C	BF	OPBDP	PBS4	MSAT+	OPTSAT	#C <sub>C</sub>	OPTSAT
1	barrel7	13764	SF	MEM	–	285.55	634.16	13764	85.01
2	lmult8	11876	SF	MEM	–	297.32	379.59	11876	57.45
3	qvar16	6495	SF	MEM	–	51.33	50.03	6495	0.66
4	c3540	9325	TIME	MEM	–	242.02	368.89	9325	57.38
5	u-bw.b		TIME	MEM	–	TIME	TIME	11487	0.98
6	u-log.c	9506	TIME	MEM	–	383.65	217.33	9506	1.3
7	u-rock.a	1691	13.29	TIME	41.29	206.56	1.74	1690	0.11

- 1 [Borchers, Furman 98] is a dedicated MAX-SAT solver.

- Demonstration in few minutes
- OPTSAT's web page at  
`http://www.star.dist.unige.it/~marco/OPTSAT/`
- Our ECAI'06 paper “Solving Optimization Problems with DLL”
- Enrico's invited talk at ICAPS'06

## Setting

- 1  $\prec$  is a partial order on the set of total assignments satisfying the formula  $\varphi$ .
- 2 Intuitively,  $\mu \prec \mu'$  means that  $\mu$  is **preferred** to  $\mu'$ .

## Optimality of an assignment

- 1 A model  $\mu$  is **optimal** (for  $\varphi$ ) if there exists a total assignment extending  $\mu$  which is a minimal element of the partial order.

# From $\prec$ on literals to $\prec$ on models

## Setting

$\prec$  is a partial order on a set  $S$  of literals representing preferences.

## $\mu \prec \mu'$

$\mu$  and  $\mu'$  are total assignments.  $\mu \prec \mu'$  if and only if

- 1 there exists a literal  $l \in S$  with  $l \in \mu$  and  $\bar{l} \in \mu'$ ; and
- 2  $\forall l' \in S \cap (\mu' \setminus \mu), \exists l \in S \cap (\mu \setminus \mu')$  such that  $l \prec l'$ .

## Example

$S = \{\bar{x}_0, \bar{x}_1\}$  with  $\bar{x}_1 \prec \bar{x}_0$ . Then,

- 1  $\{\bar{x}_1, \bar{x}_0\} \prec \{\bar{x}_1, x_0\} \prec \{x_1, \bar{x}_0\} \prec \{x_1, x_0\}$ ;
- 2 if  $\varphi$  is  $(x_0 \vee x_1)$ , the optimal model is  $\{\bar{x}_1, x_0\}$ .