OPTSAT: A Tool for Solving SAT-related Optimization Problems

Enrico Giunchiglia[1] **and Marco Maratea**[1,2]

[1] Laboratory of Systems and Technologies for Automated Reasoning (STAR-Lab)
DIST - University of Genova

[2] 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

Optimality by

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

Setting in the case of cardinality

It is convenient to see a problem as

1. a CNF formula φ
2. an optimization function defined on (a subset of) the atoms in φ

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

Optimality

Optimality by

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

Setting in the case of cardinality

It is convenient to see a problem as

1. a CNF formula φ
2. an optimization function defined on (a subset of) the atoms in φ

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

OPT-DLL

function OPT-DLL(φ,μ,S, $\prec$)

if $(\emptyset \in \varphi)$ **return** FALSE;
if $(\varphi = \emptyset)$ **return** μ;
if $(\{l\} \in \varphi)$ **return** OPT-DLL$(assign(l, \varphi), \mu \cup \{l\}, S, \prec)$;
l := a minimal unassigned literal in S, $\prec$ if $\exists$
 and a literal in φ otherwise.
v := OPT-DLL$(assign(l, \varphi), \mu \cup \{l\}, S, \prec)$;
if $(v \neq$ FALSE$)$ **return** v;
return OPT-DLL$(assign(\bar{l}, \varphi), \mu \cup \{\bar{l}\}, S, \prec)$.

Facts: [Boutilier, et al. 2004]

OPT-DLL returns an "optimal" model μ if φ is satisfiable, and FALSE otherwise.

***assign*()**

assign(l, φ) returns the formula obtained from φ by (*i*) deleting the clauses containing l, and (*ii*) deleting $\bar{l}$ 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$_{\subseteq}$ with OPT-DLL

Setting

φ is a formula ; P is the set of variables $\{x_1, \ldots, x_k\}$ in φ ;
$\overline{P} = \{\overline{x}_i : x_i \in P\}$

MIN-ONE$_{\subseteq}$

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

MIN-ONE

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

From MAX-SAT and MAX-SAT$_\subseteq$ to MIN-ONE and MIN-ONE$_\subseteq$

Intuition

1. Given a formula φ, 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$_\subseteq$ correspond to minimize the set of variables v_i assigned to 1, and thus reduce to MIN-ONE and MIN-ONE$_\subseteq$ problems

MIN-ONE and MIN-ONE$_{\subseteq}$: experimental results

	instance	$\#C$	OPBDP	PBS4	MSAT+	OPTSAT	$\#C_{\subseteq}$	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$_\subseteq$: experimental results

	instance	$\#C$	BF	OPBDP	PBS4	MSAT+	OPTSAT	$\#C_\subseteq$	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.

More on OPTSAT . . .

- 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

Optimality of a model

Setting

1. $\prec$ is a partial order on the set of total assignments satisfying the formula φ.
2. Intuitively, $\mu \prec \mu'$ means than μ is preferred to μ'.

Optimality of an assignment

1. A model μ is optimal (for φ) if there exists a total assignment extending μ 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.

$\mu \prec \mu'$

μ and μ' 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

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

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