SAT-based **planning** with minimal-#actions plans and “soft” goals

Enrico Giunchiglia and **Marco Maratea**

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

Villa Mondragone, Rome, September 13 2007

SAT-based based with minimal-#actions plans and “soft” goals

Enrico Giunchiglia and **Marco Maratea**

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

Villa Mondragone, Rome, September 13 2007

Context

SAT-based planning is the best approach for "optimally" solve planning problems by

1. constructing a SAT formula ϕ_n for a fixed *makespan* n;
2. verifying if ϕ_n is satisfiable; if not, n is increased.

Main advantages:

- simplicity
- effectiveness, can take advantage on the continuous progress in the SAT area
- optimal makespan guaranteed

International Planning Competitions

SATPLAN has been the winner of the IPC-4 and co-winner of the last IPC-5 (held in 2006) in the optimal track.

On the other hand . . .

SATPLAN's deficiency

- it can only handle a very limited part of the PDDL language; and
- it does not take into account other "plan quality" issues, e.g., number of actions in the plan and the possibility to express "soft" goals.

On the other hand . . .

SATPLAN's deficiency

it can only handle a very limited part of the PDDL language; and

√ it does not take into account other "plan quality" issues, e.g., number of actions in the plan and the possibility to express "soft" goals.

Contribution of the work

1. We present SATPLANP, modification of SATPLAN, which returns plans
   - having minimal number of actions;
   - having maximal number of “soft” goals satisfied.

   wrt both
   - subset inclusions (qualitative)
   - cardinality (quantitative)

2. We show that SATPLANP does not sacrifice efficiency wrt SATPLAN, but on a small fraction of instances, while returning “optimal” solutions.

Planning as Satisfiability

Planning problem

Is a triple $\langle I, tr, G \rangle$ where (given the sets of fluents $\mathcal{F}$ and actions $\mathcal{A}$)

- I is a SAT formula over $\mathcal{F}$ and represents the set of *initial states*;
- tr is a SAT formula over $\mathcal{F} \cup \mathcal{A} \cup \mathcal{F}'$ where $\mathcal{F}' = \{f' : f \in \mathcal{F}\}$ is a copy of the fluent signature and represents the *transition relation*
- G is a SAT formula over $\mathcal{F}$ and represents the set of *goal states*.

Plan

The *planning problem* Π *with makespan* n is the SAT formula Π_n

$$I_0 \wedge \wedge_{i=1}^{n} tr_i \wedge G_n \qquad (n \geq 0) \tag{1}$$

- tr_i is the formula obtained from tr by substituting each symbol $p \in \mathcal{F} \cup \mathcal{A}$ with p_{i-1} and each $f \in \mathcal{F}'$ with f_i
- I_0 and G_n are obvious

A *plan for* Π *with makespan* n is an interpretation satisfying (1).

SATPLAN's algorithm

function SATPLAN(Π,n)
1 **return** DLL($cnf(\Pi_n)$,$\emptyset$)

function DLL(φ,S)
2 **if** ($\emptyset \in \varphi$) **return** FALSE;
3 **if** ($\varphi = \emptyset$) **return** S;
4 **if** ($\{l\} \in \varphi$) **return** DLL($\varphi_l, S \cup \{l\}$);
5 l := *ChooseLiteral*(φ);
6 **return** DLL($\varphi_{\bar{l}}, S \cup \{\bar{l}\}$) **or**
7 **return** DLL($\varphi_{\bar{l}}, S \cup \{l\}$).

φ_l

φ_l returns the formula obtained from φ by (i) deleting the clauses containing l, and (ii) deleting $\bar{l}$ from the others.

Qualitative SATPLANP's algorithm (I)

Going at work from home

$$\begin{gathered} \neg AtWork_0, \\ AtWork_1 \equiv \neg AtWork_0 \equiv (Car_0 \vee Bus_0 \vee Bike_0), \\ AtWork_1, \end{gathered} \tag{2}$$

- There are 7 ways of going from home to work, each corresponing to a non empty subset of $\{Bus, Car, Bike\}$.
- The idea is to rely on Preferences, i.e.,additional constraints that we would like to be satisfied, e.g.:
  - p_1: we do not like moving: $\neg Bus_0 \wedge \neg Bike_0 \wedge \neg Car_0$
  - p_2: we prefer to avoid taking the bus and the bike: $\neg Bus_0 \wedge \neg Bike_0$.
- It is possible that not all preferences can be satisfied. In this case, they can be ordered.

A (qualitative) preference for a planning problem Π_n is a partially ordered set $\langle P, \prec \rangle$ of formulas in the signature of Π_n.

Qualitative SATPLANP's algorithm (II)

function QL-SATPLANP(Π,n,P,$\prec$)

8 **return** OPT-DLL($cnf(\Pi_n \wedge \wedge_{p \in P}(v(p) \equiv p))$,$\emptyset$,$v(P)$,$v(\prec)$)

function OPT-DLL(φ,S,P',$\prec'$)

9 **if** $(\emptyset \in \varphi)$ **return** FALSE;

10 **if** $(\varphi = \emptyset)$ **return** S;

11 **if** $(\{l\} \in \varphi)$ **return** OPT-DLL$(\varphi_l, S \cup \{l\}, P', \prec')$;

12 l := *ChooseLiteral*$(\varphi, S, P', \prec')$;

13 V := OPT-DLL$(\varphi_l, S \cup \{l\}, P', \prec')$;

14 **if** $(V \neq$ FALSE$)$ **return** V;

15 **return** OPT-DLL$(\varphi_{\bar{l}}, S \cup \{\bar{l}\}, P', \prec')$.

where

- $v(P)$ is the set of new variables, i.e., $\{v(p) : p \in P\}$;
- $v(\prec) = \prec'$ is the partial order on $v(P)$ defined by $v(p) \prec' v(p')$ iff $p \prec p'$;
- *ChooseLiteral* initially selects literals according to $\prec$.

Qualitative SATPLANP's algorithm: Example

We have two preferences

1. $p_1 = (\neg Bike_0 \wedge \neg Bus_0 \wedge \neg Car_0)$
2. $p_2 = (\neg Bike_0 \wedge \neg Bus_0)$

with $p_1 \prec p_2$.

OPT-DLL on (2) returns the plan corresponding to $\{Car_0\}$ determined after exploring

1. $v(p_1)$, $v(p_2)$: no plan exists extending this branch. OPT-DLL backtracks, and explores branches with
2. $v(p_1)$, $\neg v(p_2)$: no plan exists extending this branch. OPT-DLL backtracks, and explores branches with
3. $\neg v(p_1)$, $v(p_2)$: OPT-DLL finds the optimal plan.

Quantitative SATPLANP's algorithm (I)

1. A quantitative preference for Π_n is a pair $\langle P, c \rangle$, where $c : P \mapsto \mathcal{N}$. A plan π is *optimal (wrt $\langle P, c \rangle$)* if it maximizes

$$\sum_{p \in P : \pi \models p} c(p). \tag{3}$$

2. Optimal planning with quantitative preferences is reduced to the qualitative case.

3. Idea: Encode the value of the objective function (3) as a sequence of bits $b_{n-1}, \ldots, b_0$ and then consider the qualitative preference

$$\langle \{b_{n-1}, \ldots, b_0\}, \{b_{n-1} \prec b_{n-2}, \ldots, b_1 \prec b_0\} \rangle.$$

Quantitative SATPLANP's algorithm (II)

function QT-SATPLANP(Π,n,P,c)
16 **return** OPT-DLL($cnf(\Pi_n \wedge adder(P,c))$,$\emptyset$,$b(c)$,$p(c)$)

$adder(P,c)$ is a SAT formula, e.g., (Warners, IPL 1999)

- if $n = \lceil log_2((\sum_{p \in P} c(p)) + 1) \rceil$, $adder(P,c)$ contains n new variables $\{b_{n-1}, \ldots, b_0\} = b(c)$; and
- for any plan π satisfying Π_n, there exists a unique interpretation μ to the variables in $\Pi_n \wedge adder(P,c)$ s.t.
  1. μ extends π and satisfies $\Pi_n \wedge adder(P,c)$;
  2. $\sum_{p \in P:\pi \models p} c(p) = \sum_{i=0}^{n-1} \mu(b_i) \times 2^i$, where $\mu(b_i)$ is 1 if μ assigns b_i to true, and is 0 otherwise.
- $p(c)$ is the partial order $b_{n-1} \prec b_{n-2} \prec \cdots \prec b_0$.

Quantitative SATPLANP's algorithm: Example

We have two preferences

1. $p_1 = (\neg Bike_0 \wedge \neg Bus_0 \wedge \neg Car_0)$, $c(p_1) = 2$;
2. $p_2 = (\neg Bike_0 \wedge \neg Bus_0)$, $c(p_2) = 1$.

The value of the objective function to be maximized can be encoded with two bits b_1 and b_0, with $b_1 \prec b_0$.

OPT-DLL returns the plan corresponding to $\{Car_0\}$ determined after exploring the branches

1. b_1, b_0: since no plan with cost 3 is found, then OPT-DLL backtracks, and explores a branch starting with
2. $b_1, \neg b_0$: since no plan is found with cost 2, then OPT-DLL backtracks, and explores a branch starting with
3. $\neg b_1, b_0$: finding the optimal plan.

Experimental analysis: Goals

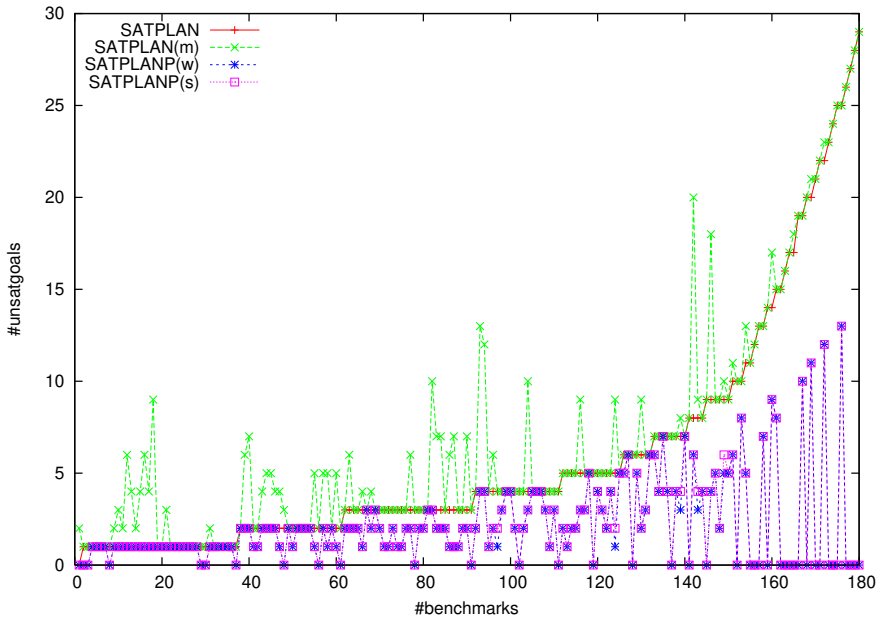
1. Evaluate SATPLANP wrt the state-of-the-art on problems with “soft” goals
   - SATPLANP vs. SGPLAN on classical planning problems whose goals have been turned to be “soft”
2. Evaluate the advantages that can be obtained with SATPLANP over SATPLAN
3. Evaluate the costs of such advantages in terms of performance degradation
   - SATPLANP vs. SATPLAN on classical planning problems
4. In the case of quantitative preferences, evaluate what kind of *adder*() works better
   - (Warners, IPL 1999) vs. (Bailleaux & Boufkhad, CP 2003)

Experimental analysis: 1. & 2. & 4.

	SGPLAN	SATPLAN	SATPLANP(w)	SATPLANP(b)	SATPLANP(s)
pipe	0/0	0/7	0/18	0/18	0/17
pipet	0/0	0/5	0/11	0/11	0/11
sat	0/10	0/4	0/4	0/4	0/4
air	0/23	0/9	0/11	0/11	0/11
phil	29/0	0/29	0/464	0/464	0/464
opt	12/0	0/12	0/90	0/90	0/90
psr	12/157	0/48	0/231	0/231	0/231
dep	2/3	0/4	0/7	0/7	0/7
driv	0/71	0/10	0/54	0/54	0/50
zeno	0/44	0/9	0/24	0/24	0/24
free	0/12	0/3	0/8	0/8	0/8
log	0/51	0/10	0/33	0/33	0/33
block	0/33	0/9	0/12	0/12	0/12
mpr	0/0	0/4	0/4	0/4	1/3
myst	0/0	0/2	0/2	0/2	0/2
path	7/0	0/7	0/21	0/21	0/21
stor	0/30	0/9	0/10	0/10	0/10
TPP	5/14	0/9	0/27	0/27	0/27
truck	3/0	0/3	0/3	0/3	0/3
Total	70/448	0/193	0/1034	0/1034	1/1028

Table: x/y stands for x time outs or segmentation faults, y soft goals satisfied.

Experimental analysis: 2. & 4.



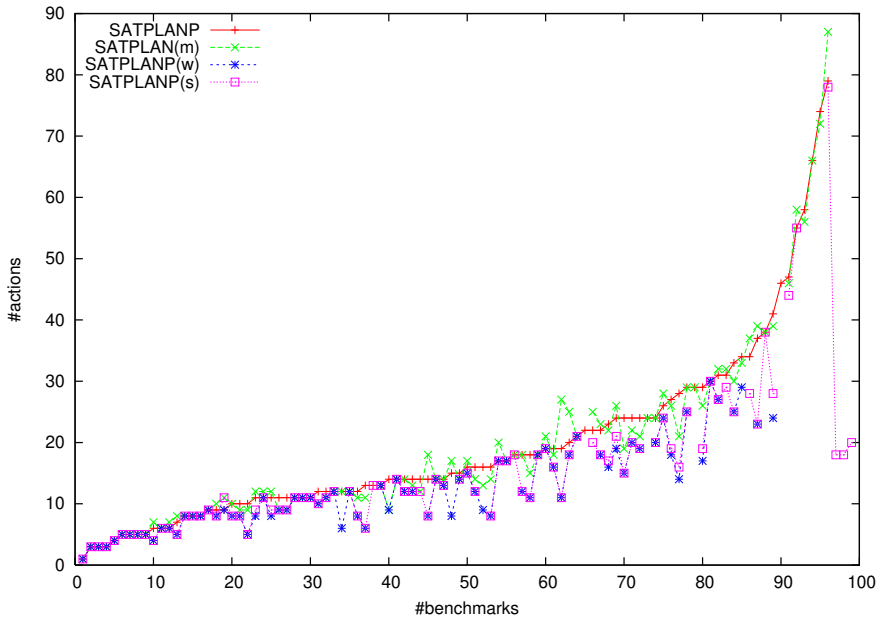
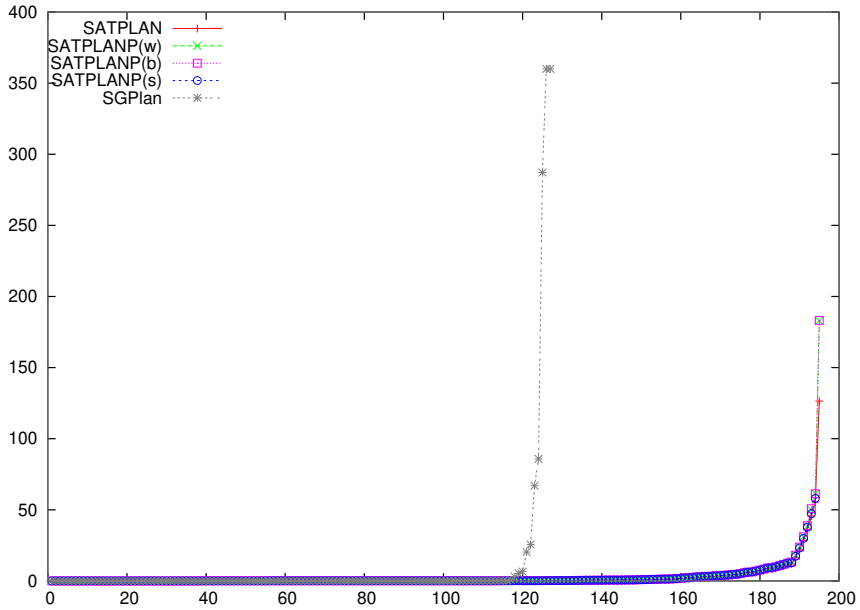


Figure: Left: Number of unsatisfied soft goals by SATPLAN, SATPLAN(m), and SATPLANP(w)/(s). Right: Number of actions in the returned plan for SATPLAN, SATPLAN(m), and SATPLANP(w)/(s).

Experimental analysis: 1. & 3. & 4.



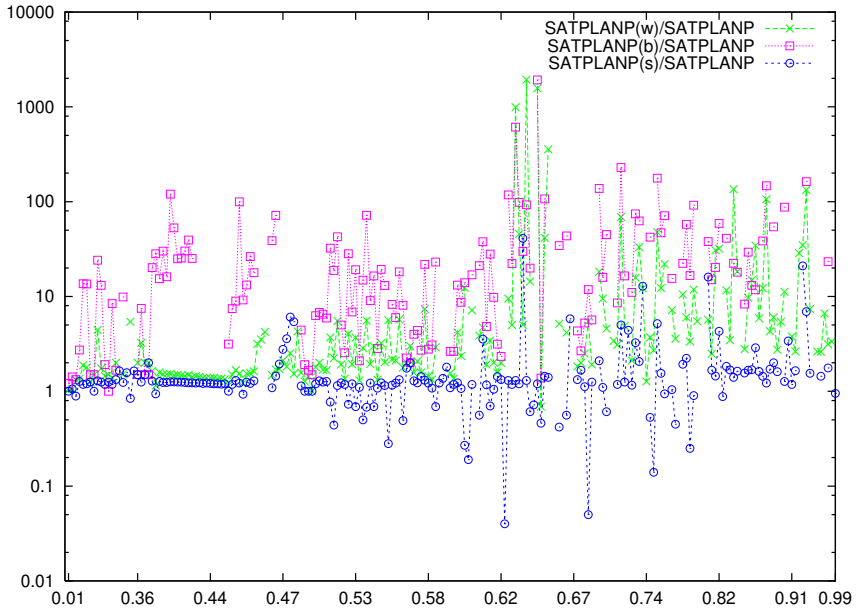


Figure: Left: Performances of SGPLAN, SATPLANP/(w)/(b)/(s). Right: Performances of SATPLANP(w)/(b)/(s) wrt SATPLAN as a function of the ration between the number of preferences and the number of variables.

Conclusions and future work

Done . . . We have

- √ extended SAT-based planning to deal with (other) issues related to plan quality;
- √ showed that our approach is viable and competitive, often without sacrificing efficiency;
- √ evaluated different minimalities and *adder*()s.

To be done . . .

- × allow SATPLAN to increase the makespan even when a solution is found, for further improving the plan quality.

More on this work . . .

- SATPLANP's web page at
  `http://www.star.dist.unige.it/~marco/SATPLANP/`
- Our ECAI 2006 and AAAI 2007 papers:
  - "Solving Optimization Problems with DLL"; and
  - "Planning as Satisfiability with Preferences"
- Enrico's invited talk at ICAPS'06