Improving Plan Quality in SAT-based Planning

Enrico Giunchiglia and **Marco Maratea**

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

AI*IA 2009 - RCRA session, Reggio Emilia
December 11 19 2009

Context

Planning as SAT and Preferences are a hot topic in Planning and CP:

1. In (ECAI'06) we presented a simple framework and procedure for expressing and reasoning with qualitative and quantitative preferences in SAT
2. In (AAAI'07) we extended the procedure and applied to Planning as Satisfiability
3. In (ECAI'08) we proposed a new algorithm for reasoning about qualitative preferences in SAT

Here....

1. we briefly review the planning as SAT and the (simple) formalism we use for expressing preferences in SAT
2. present and discuss the (ECAI'06) and (ECAI'08) procedures for reasoning with preferences in SAT
3. extend (ECAI'08) to planning as Satisfiability

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

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

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.

Review: Planning as Satisfiability with Preferences

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

$\pi_1 \prec \pi_2$

Consider two plans (i.e., assignments) π_1 and π_2.
$\pi_1 \prec \pi_2$ if and only if

1. they satisfy different sets of preferences, i.e., $\{p : p \in P, \pi_1 \models p\} \neq \{p : p \in P, \pi_2 \models p\}$, and
2. for each preference p_2 satisfied by π_2 and not by π_1 there is another preference p_1 satisfied by π_1 and not by π_2 with $p_1 \prec p_2$.

Review (II): Planning as Satisfiability with Preferences

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

$$\sum_{p \in P : \pi \models p} c(p). \tag{2}$$

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

3. Idea: Encode the value of the objective function (2) as a sequence of bits $b_{n-1}, \ldots, b_0$ and then consider the qualitative preference

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

(e.g., (Warners, IPL 1999))

Qualitative SATPLANP's algorithm — AAAI'07

$\langle P, \prec \rangle$:= a qualitative preference;
function QL-SATPLANP(Π,n)
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',$\prec'$) (ECAI'06)
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 $\prec$ (first solution is optimal, but (Jarvisalo et al., AMAI 2005)).

Qualitative generate-and-test algorithm (I)

Idea

1. generate a candidate optimal solution π; and
2. test if the solution is optimal, pruning less preferred plans if it is not the case.

The idea extends previous works done in both CSP and SAT, e.g. (Castell et al., ECAI 1996; Gavanelli, ECAI 2002; Di Rosa, Giunchiglia and Maratea, ECAI 2008).

Given a qualitative preference $\langle P, \prec \rangle$, the assignments that are preferred to π are those satisfying the formula (where l and l' are literals):

$$(\vee_{l:l \in P, l \notin \pi} l) \wedge (\wedge_{l':l' \in P, l' \in \pi} (\vee_{l:l \in P, l \notin \pi, l \prec l'} l \vee l')). \quad (3)$$

Qualitative generate-and-test algorithm (II)

$\langle P, \prec \rangle$:= a qualitative preference; $\psi := \top$; $\pi_{opt} := \emptyset$

function QL-PLAN-GNT(Π,n)
16 **return** PREF-DLL($cnf(\Pi_n \wedge_{p \in P} (v(p) \equiv p))$,$\emptyset$,$v(P)$,$v(\prec)$)

function PREF-DLL($\varphi \cup \psi$,π,P',$\prec'$) (ECAI' 08)
17 **if** ($\emptyset \in (\varphi \cup \psi)_\pi$) **return** FALSE;
18 **if** (π is total) $\pi_{opt} := \pi$; $\psi := Reason(\pi, P', \prec')$; **return** FALSE;
19 **if** ($\{l\} \in (\varphi \cup \psi)_\pi$) **return** PREF-DLL($\varphi \cup \psi, \pi \cup \{l\}$);
20 l := *ChooseLiteral*($\varphi \cup \psi, \pi$);
21 **return** PREF-DLL($\varphi \cup \psi, \pi \cup \{l\}$) **or**
PREF-DLL($\varphi \cup \psi, \pi \cup \{\bar{l}\}$).

where

- *Reason* returns the set of clauses corresponding to (3);
- *ChooseLiteral* does not have any "restriction".

Experimental analysis: Setting and goals

Setting

- SATPLANP implements QL-SATPLANP;
- SATPLAN-GNT implements QL-PLAN-GNT;
- SGPLAN used as reference system on problems with soft goals (solves a different problem).

Goals

- find a "minimal-actions" plan wrt both qualitative (SATPLANP() and SATPLAN-GNT()) and quantitative (SATPLANP(W) and SATPLAN-GNT(W)) preferences on IPC domains; with $\langle \neg act(\Pi_n), \emptyset \rangle$, $\langle \neg act(\Pi_n), 1 \rangle$.
- solve problems from IPC-5 "SimplePreferences" track via compilation into a STRIPS problem (so far, we considered domains where all goals are soft, and we have made goals's violation cost uniform).

Experimental analysis: “Minimal-actions” plans

	SATPLANP(W)	SATPLAN-GNT(W)	SATPLANP()	SATPLAN-GNT()
pipesworld-notankage	85.57(9)	110.37(9)	40.92(11)	100.21(13)
pipesworld-tankage	193.86(6)	217.72(7)	32.59(7)	97.96(8)
satellite	12.6(2)	7.34(2)	3.34(4)	226.04(4)
promela-optical	58.96(11)	108.2(13)	123.38(9)	18.59(13)
psr-small	34.82(47)	32.08(48)	11.85(44)	15(48)
depots	76.02(5)	43.84(5)	194.24(5)	123.08(9)
zenoTravel	10.44(8)	10.79(8)	64.41(9)	40.76(11)
freeCell	10.8(2)	8.91(2)	89.81(4)	15.19(3)
logistics	97.92(10)	5.77(13)	2.4(22)	78.37(25)
mprime	60.17(19)	60.24(19)	27.59(14)	12.58(19)
mystery	24.47(13)	28.29(15)	32.36(13)	11.69(15)
openstacks	–	–	**717.31(4)**	–
pathways	22.89(5)	13.96(5)	63.78(7)	5.79(7)
storage	16.67(9)	7.83(9)	42.1(12)	24.24(11)
tpp	0.08(5)	151.17(7)	0.14(8)	123.59(19)
elevator	18.99(15)	1.75(15)	54.08(30)	0.63(15)
rovers	83.41(6)	22.9(6)	79.31(8)	110.38(16)

Table: Results on domains coming from IPCs. $x(y)$ stands for y instances solved with x secs of mean CPU time.

Experiments with soft goals

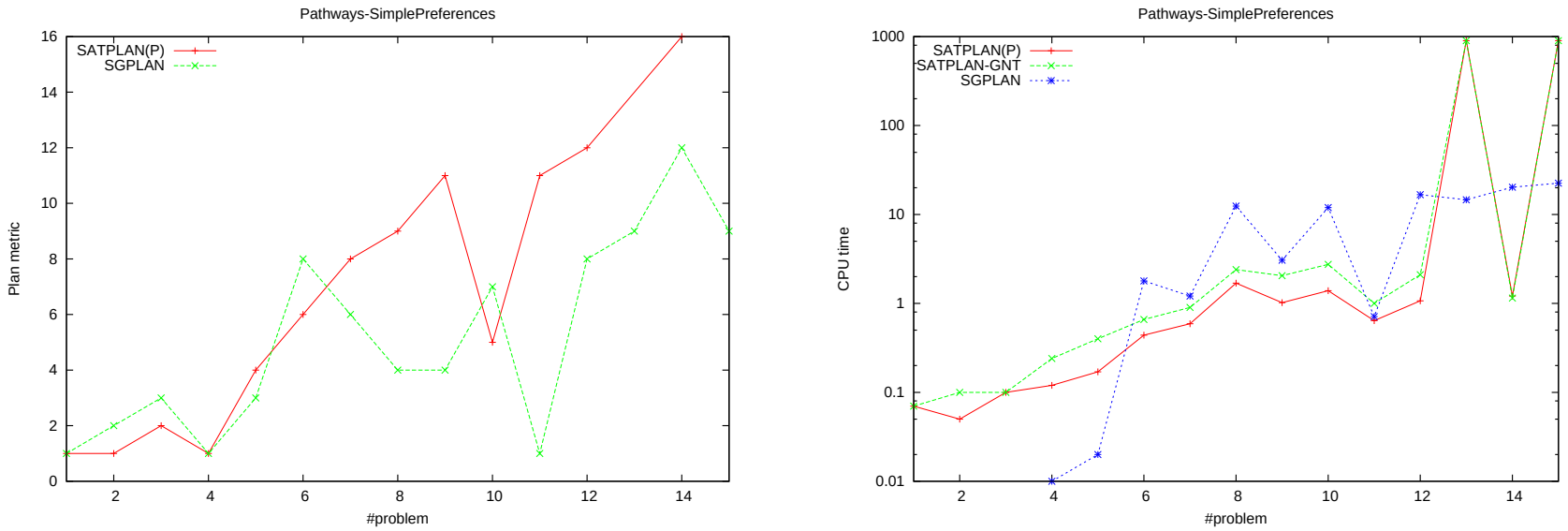

Figure: Pathways domain, "SimplePreferences" track of IPC-5. Left: Plan metric, i.e., number of unsatisfied soft goals, for SATPLAN-GNT(W) (and thus SATPLANP(W)) and SGPLAN. Right: CPU time for SATPLAN-GNT(W), SATPLANP(W) and SGPLAN (in log scale).

Experiments with soft goals

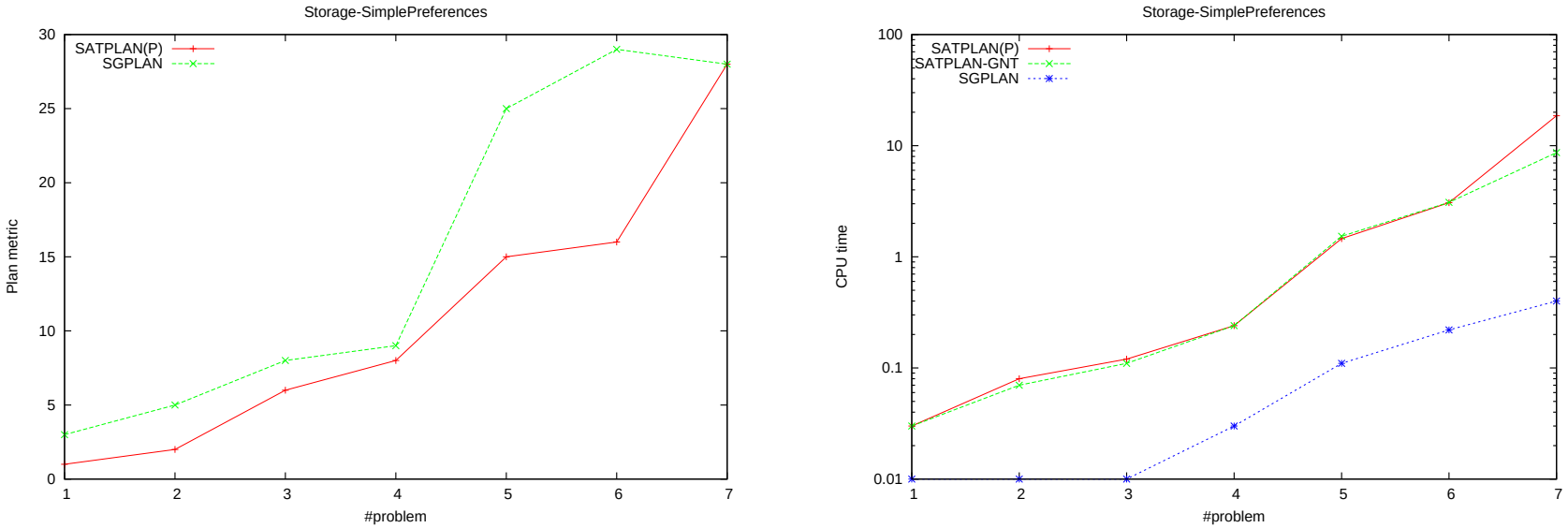

Figure: Storage domain, “SimplePreferences” track of IPC-5. Left: Plan metric, i.e., number of unsatisfied soft goals, for SATPLAN-GNT(W) (and thus SATPLANP(W)) and SGPLAN. Right: CPU time for SATPLAN-GNT(W), SATPLANP(W) and SGPLAN (in log scale).

Summing up

The results of our preliminary analysis point out that

- on planning problems with many preferences, SATPLAN-GNT performs better than SATPLANP;
- on planning problems with (relatively) few preferences, SATPLAN-GNT and SATPLANP perform similarly; and
- SATPLAN-GNT and SATPLANP compare well wrt SGPLAN on planning problems (considering the restriction we have imposed) coming from the “SimplePreferences” track of the IPC-5.

Current and future work

We plan to

- analyze domains with mixed hard/soft goals, and with non-uniform costs;
- compare also to IP-based planners (can handle first problem);
- evaluate the “Planning with soft goals as SAT-related Optimization problem” approach.