

A generate-and-test approach for computing “optimal” plans in SAT-based planning

Enrico Giunchiglia and Marco Maratea

DIST - University of Genova, Viale F. Causa 15, Genova, Italy
{enrico,marco@dist.unige.it}

Abstract

Planning as Satisfiability (SAT) is the best approach for optimally (wrt makespan) solving classical planning problems. SAT-based planners, like SATPLAN, can thus return plans having minimal makespan guaranteed. However, the returned plan does not take into account plan quality issues introduced in the last two International Planning Competitions (IPCs): such issues include plans which minimize action costs and plans with “soft” goals, where a metric has to be optimized over actions/goals. Recently, an approach to address such issues has been presented, in the framework of planning as satisfiability with preferences: by imposing that one prefers “not to perform” actions, or “to satisfy” soft goals, the related system (called SATPLAN(P)) is guaranteed to return plans with minimal number of actions, or with maximal number of soft goals satisfied (thus in a qualitative case), and is extended to deal with the quantitative case. But, besides such feature, and the fact that the first computed plan is guaranteed to be “optimal”, it is well-known that introducing ordering in SAT heuristics can lead, at least theoretically, to significant degradation in performances: in SATPLAN(P), this phenomenon also happened experimentally on large planning problems with many preferences.

In this paper, we present a different, generate-and-test, approach to tackle the problem of dealing with such optimization issues: without imposing any ordering, a (candidate optimal) plan is first generated, and then a constraint is added imposing that the new plan (if any) has to be “better” than the last computed, i.e., the plan quality is increased at each iteration. We implemented this idea in SATPLAN, and compared the resulting systems wrt SATPLAN(P) and SGPLAN on planning problems coming from IPCs. The analysis shows performance benefits for the new approach, in particular on planning problems with many preferences.

Introduction

Planning as Satisfiability (SAT) (Kautz and Selman 1992) is the best approach for optimally (wrt makespan) solving classical planning problems. The SAT-based planner SATPLAN (Kautz and Selman 1999) has been the winner in the deterministic track for optimal planners in the 4th International Planning Competition (IPC-4) (Hoffmann and Edelkamp 2005) and co-winner in the IPC-5 (Gerevini et

al. 2009) (together with another SAT-based planner, MAXPLAN (Xing, Chen, and Zhang 2006)). SAT-based planners inherit from the approach the property that the returned plan has minimal makespan guaranteed. However, the returned plan does not take into account plan quality issues introduced in the last two International Planning Competitions (IPCs): such issues include plans which minimize action costs and plans with “soft” goals, where a metric has to be optimized over actions/goals. The availability of planning systems able to deal with such optimization issues is relatively recent, and, even if search-based and IP-based planners have shown positive results, it is not yet clear what is the best solving approach. A first SAT-based approach to tackle this problem has been recently presented, in the framework of planning as satisfiability with preferences (Giunchiglia and Maratea 2007a; 2007b): by imposing that the heuristic of the underlying SAT solver first select literals which correspond to “not to perform” actions, or “to satisfy” soft goals, the related system, called SATPLAN(P), is guaranteed to return plans with minimal number of actions (or, minimal-actions), or with the maximal number of soft goals satisfied, i.e., no subset of the plan’s actions or soft goals achieve the goals, given the optimal makespan. But, besides such feature, and the fact that the first computed plan is guaranteed to be “optimal”, it is well-known that introducing ordering in SAT heuristics can lead, at least theoretically, to significant degradation in performances (Järvisalo, Juntila, and Niemelä 2005): in SATPLAN(P), this phenomenon also happened experimentally on large planning problems with many preferences.

In this paper, we present a different, generate-and-test, approach to tackle the problem of dealing with such optimization issues, at fixed, optimal makespan: without imposing any ordering, a (candidate optimal) plan is first generated, and then a constraint is added imposing that the new plan (if any) has to be “better” than the last computed, extending what has been done in different contexts in both CSP and SAT, e.g., (Castell et al. 1996; Gavanelli 2002; DiRosa, Giunchiglia, and Maratea 2008). Thus, the plan quality is increased at each iteration of the algorithm. We implemented this idea in SATPLAN, also considering the “quantitative” approach to the problem, i.e., considering also plans with the minimum number of actions, or with maximum number of soft goals satisfied, by means of a known re-

duction from quantitative to qualitative preferences. We then compared the resulting system wrt SATPLAN(P) on both classical planning problems coming from IPCs, and on planning problems coming from the “SimplePreferences” track of the IPC-5, having all goals as “soft”. The results of our preliminary analysis point out that:

- on planning problems with many preferences, SATPLAN-GNT performs better than SATPLAN(P);
- on planning problems with (relatively) few preferences, SATPLAN-GNT and SATPLAN(P) perform similarly; and
- SATPLAN-GNT and SATPLAN(P) are both overall competitive to SGPLAN on planning problems coming from the “SimplePreferences” track of the IPC-5, where SGPLAN was the clear winner.

The paper is structured as follows. We first present some basic preliminaries about planning (as satisfiability). Then, we define optimal plans in term of preferences. We continue by showing the new algorithm, along with some formal results, whose implementation and experimental evaluation is presented in the next section. The last part of the paper discusses interactions with other, related work and drawn some conclusions and possible topics for future research.

Preliminaries

Let \mathcal{F} and \mathcal{A} be the set of *fluents* and *actions*, respectively. A *state* is an interpretation of the fluent signature. A *complex action* α is an interpretation of the action signature, and models the concurrent execution of the actions satisfied by α , i.e., it is a set of actions that can be executed in parallel. A *planning problem* is a triple $\langle I, tr, G \rangle$ where

- I is a Boolean formula over \mathcal{F} and represents the set of *initial states*;
- tr is a Boolean 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* of the automaton describing how (complex) actions affect states (we assume $\mathcal{F} \cap \mathcal{F}' = \emptyset$);
- G is a Boolean formula over \mathcal{F} and represents the set of *goal states*.

The above definition of planning problem differs from the traditional ones in which the description of actions’ effects on a state is described in an high-level action language like STRIPS or PDDL. We used this formulation because the techniques we are going to describe are largely independent from the action language used, at least from a theoretical point of view. The only assumption that we make is that the description is deterministic: there is only one state satisfying I and the execution of a (complex) action α in a state s can lead to at most one state s' . More formally, for each state s and complex action α there is at most one interpretation extending $s \cup \alpha$ and satisfying tr . Consider a planning problem $\Pi = \langle I, tr, G \rangle$. In the following, for any integer i

- if F is a formula in the fluent signature, F_i is obtained from F by substituting each $f \in \mathcal{F}$ with f_i ,
- 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 .

If n is an integer, the *planning problem* Π with *makespan* n is the Boolean formula Π_n defined as

$$I_0 \wedge \bigwedge_{i=1}^n tr_i \wedge G_n, n \geq 0 \quad (1)$$

and a *plan* is an interpretation (or, equivalently, a set of literals) satisfying (1).¹ For example, consider the planning problem of going to work from home for an husband and a wife. Assume that they can use the car or the bus or the bike, but one has to stay home with the child, this scenario can be formalized using two fluent variables $AtWorkH$ and $AtWorkW$, and three action variables Car , Bus and $Bike$. The problem with makespan 1 can be expressed by the conjunction (here indicated with “;”) of the formulas:

$$\begin{aligned} & \neg AtWorkH_0, \neg AtWorkW_0, \\ AtWorkH_1 & \equiv \neg AtWorkH_0 \equiv (Car_0 \vee Bus_0 \vee Bike_0), \\ AtWorkW_1 & \equiv \neg AtWorkW_0 \equiv (Car_0 \vee Bus_0 \vee Bike_0), \\ & \neg AtWorkH_1 \vee \neg AtWorkW_1, \end{aligned} \quad (2)$$

in which the first two formulas correspond to the initial state, the third and the fourth to the transition relation, and the last indicates that at most one goal can be reached, mimicking, together with the other formulas, that goals are soft. The planning problem has many solutions, each corresponding to a non-empty subset of $\{Car_0, Bus_0, Bike_0\}$ for each fluent. Among those plans, in a minimal-actions plan a single action is performed. About “soft” goals, the simple characterization in (2) is extended to include “preferences” among the goals, by means of weights associated to the violation of the goals, or an ordering on them.

Optimal plans and qualitative preferences

In this section we formalize the definition of optimal plans in the framework of planning as satisfiability with preferences, first presented in (Giunchiglia and Maratea 2007a). Let Π_n be a planning problem Π with makespan n . A *qualitative preference* (for Π_n) is a pair $\langle P, \prec \rangle$ where P is a set of literals (the preferences, in our case they will be built on action variables or soft goals) and \prec is a partial order on P . Thus, the focus of our presentation is on the *qualitative* approach to the problem, and to preferences on literals. This is not a limitation, given that in (Giunchiglia and Maratea 2007a) it is showed how to deal with quantitative preferences (where a preference is a pair $\langle P, c \rangle$ where P has the same meaning as before, and c is a function that maps literals to positive integer) and with (qualitative and quantitative) preferences on formulas, by means of a reduction to our framework of qualitative preferences on literals. Qualitative and quantitative approaches both received attention in planning, each having its pros and cons, as commented in, e.g., Sec. 2.3 of (Gerevini et al. 2009).

Consider a qualitative preference $\langle P, \prec \rangle$. The partial order can be extended to plans for Π_n in the following way. Let π_1 and π_2 be two plans for Π_n . π_1 is *preferred* to π_2 (wrt $\langle P, \prec \rangle$) iff

¹In the following, we continuously switch between plans and satisfying interpretations.

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 $p_1 \prec p_2$.

The second condition says that if π_1 does not satisfy a preference p_2 which is satisfied by π_2 , then π_1 is preferred to π_2 only if there is a good reason for $\pi_1 \not\models p_2$, i.e., π_1 satisfies a “more preferred” preference (not satisfied by π_2). We write $\pi_1 \prec \pi_2$ to mean that π_1 is preferred to π_2 . It is easy to see that \prec defines a partial order on plans for Π_n wrt $\langle P, \prec \rangle$. A plan π is *optimal* for Π_n (wrt $\langle P, \prec \rangle$) if it is a minimal element of the partial order on plans for Π_n , i.e., if there is no plan π' for Π_n with $\pi' \prec \pi$ (wrt $\langle P, \prec \rangle$). As an example, consider the planning problem in (2); if we have the qualitative preference (in the following, we show only the action variables assigned to true in the optimal plan):

- a. $\langle \{-Bike_0, -Bus_0, -Car_0\}, \emptyset \rangle$, i.e., the situation in which we prefer not to perform actions, and the preferences are equally important ($\prec = \emptyset$), there are three optimal, i.e., minimal-actions, plans, corresponding to $\{Bike_0\}$, $\{Bus_0\}$, $\{Car_0\}$, for each fluent.
- b. $\langle \{-Bike_0, -Bus_0, -Car_0\}, \{-Bike_0 \prec -Car_0\} \rangle$, i.e., the situation in which again we prefer not to perform actions, and not to take the bike is preferred over not to take the car, then there are two optimal plans, i.e., $\{Bus_0\}$, $\{Car_0\}$, for each fluent.
- c. $\langle \{AtWorkH, AtWorkW\}, \emptyset \rangle$, i.e., the situation in which we prefer to satisfy the (soft) goals, and the goals are equally important ($\prec = \emptyset$), there are 14 optimal plans, each corresponding to a non-empty subset of $\{Car_0, Bus_0, Bike_0\}$ for each soft goal.
- d. $\langle \{AtWorkH, AtWorkW\}, \{AtWorkH \prec AtWorkW\} \rangle$, i.e., the situation in which again we prefer to satisfy the soft goals, and satisfying the first is preferred to the second, there are 7 optimal plans, each corresponding to a non-empty subset of $\{Car_0, Bus_0, Bike_0\}$ and with the fluent $AtWorkH$ true.

Computing optimal plans via generate-and-test

We have already discussed in the introduction that the algorithm in (Giunchiglia and Maratea 2007a; 2007b) has drawbacks related to imposing an ordering on preferences to be followed while branching: from (Järvisalo, Juntila, and Niemelä 2005), it is well-known that such ordering can significantly degrade the performances. Here we present a different, generate-and-test, approach for solving the problem of generating plans with minimal number of actions, or with maximal number of soft goal satisfied, thus in the qualitative case (in the next section we briefly mention how to also deal with the quantitative case). It first generates a (candidate optimal) plan, and then a constraint is added imposing that the new plan (if any) has to be “better” than the last computed, extending what has been done in both CSP and SAT, e.g., in (Castell et al. 1996; Gavanelli 2002; DiRosa, Giunchiglia, and Maratea 2008).

Thus, crucial for the above procedure is a condition which enables us to say which are the plans that are preferred (wrt $\langle P, \prec \rangle$) to a plan π : such a formula, where l is a literal, is

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

which codifies conditions 1. and 2. in the previous section. Note that if the partial order is empty, the second part of (3) reduces to a conjunctive clause (i.e., a conjunction of literals), thus this part is very efficiently propagated by unit propagation.

A plan π' is preferred to π wrt $\langle P, \prec \rangle$ iff π' satisfies (3), as stated by the following theorem.

Theorem 1 *Let π and π' be two plans. Let $\langle P, \prec \rangle$ be a qualitative preference. π' is preferred to π wrt $\langle P, \prec \rangle$ if and only if π' satisfies the preference formula in (3).*

Theorem 1 follows from (3) by construction, and corresponds to a formalization of a concept already known in literature, e.g., (Giunchiglia and Maratea 2007a).

In Figure 1 we present the new solving procedures. Informally, QL-PLAN-GNT-1 and QL-PLAN-GNT-2 find an optimal plan by translating the planning problem Π at makespan n and the preference into a propositional formula (and a CNF conversion cnf is needed). Then, the CNF formula together with the preference are fed to PREF-DLL, a modified version of DLL that can find “optimal” SAT solution. An optimal solution corresponds to an optimal plan of the original problem.

More in details about Figure 1:

- in QL-PLAN-GNT-2: for each $p \in P$, $v(p)$ is a newly introduced variable; $v(P)$ 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 $p \prec p'$; in QL-PLAN-GNT-1: no changes are made to the preference, thus in the parameters passed to PREF-DLL $P' = P$ and $\prec' = \prec$.
- π is a consistent set of literals (or, equivalently, a (candidate) plan for Π_n), and corresponds to the partial interpretation mapping to true the literals $l \in \pi$.
- $cnf(\varphi)$, where φ is a formula, is a set of clauses (i.e., set of sets of literals, with \top denoting the empty set of clauses) such that:

- for any interpretation π in the signature of $cnf(\varphi)$ such that $\pi \models cnf(\varphi)$ it is true also that $\pi' \models \varphi$, where π' is the interpretation π restricted to the signature of φ ; and
- (ii) interpretation π , $\pi \supseteq \pi'$, such that $\pi \models cnf(\varphi)$.

There are well known methods for computing $cnf(\varphi)$ in linear time by introducing additional variables, e.g., (Tseitin 1970).

- l is a literal and \bar{l} is the complement of l ;
- φ_l returns the set of clauses obtained from φ by (i) deleting the clauses $C \in \varphi$ with $l \in C$, and (ii) deleting \bar{l} from the other clauses in φ ;
- *Reason* returns the set of clauses corresponding to (3);
- *ChooseLiteral* returns an *unassigned* literal l (i.e., such that $\{l, \bar{l}\} \cap \pi = \emptyset$) in φ .

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

function QL-PLAN-GNT-1(Π, n)

1 **return** PREF-DLL($cnf(\Pi_n), \emptyset, P, \prec$)

function QL-PLAN-GNT-2(Π, n)

2 **return** PREF-DLL($cnf(I_0 \wedge \bigwedge_{i=1}^n tr_i \wedge_{p \in P} (v(p) \vee p)), \emptyset, v(P), v(\prec)$)

function PREF-DLL($\varphi \cup \psi, \pi, P', \prec'$)

3 **if** ($\emptyset \in (\varphi \cup \psi)_\pi$) **return** FALSE;

4 **if** (π is total) $\pi_{opt} := \pi$; $\psi := \text{Reason}(\pi, P', \prec')$;

return FALSE;

5 **if** ($\{l\} \in (\varphi \cup \psi)_\pi$) **return** PREF-DLL($\varphi \cup \psi, \pi \cup \{l\}$);

6 $l := \text{ChooseLiteral}(\varphi \cup \psi, \pi)$;

7 **return** PREF-DLL($\varphi \cup \psi, \pi \cup \{l\}$) **or**

PREF-DLL($\varphi \cup \psi, \pi \cup \{\bar{l}\}$).

Figure 1: The algorithms of SATPLAN-GNT.

Note that in the algorithm we have not specified $\langle P, \prec \rangle$ to be a preference on literals: in fact, goals are usually formulas. Thus, when dealing with soft goals (QL-PLAN-GNT-2 algorithm), we add what are called “goal selectors”, i.e., variables which are place-holders for the goals: preferences are then expressed over such set of variables, and in this way we are back to our setting of preferences on literals. Goal selectors have the same meaning of clause selectors in Max-SAT. When preferences are expressed over action variables (QL-PLAN-GNT-1 algorithm), adding such variables and modifying the ordering is not needed.

Thus, given a planning problem Π , a makespan n , and a qualitative preference $\langle P, \prec \rangle$, an optimal plan is computed by invoking PREF-DLL (DiRosa, Giunchiglia, and Maratea 2008), a modified version of standard DLL for computing “optimal” solutions, on $cnf()$, i.e., the set of clauses corresponding to the planning problem Π with makespan n , possibly with the modified preferences. The formula ψ corresponds to the clauses corresponding to (3) added because of a (candidate) optimal solution is found: in the first invocation, ψ is the empty set of clauses.

More in details, PREF-DLL is standard DLL except that when a new plan π is found (at line 4), it is set to be the (actual) optimal plan π_{opt} (initially set to \emptyset), a set of clauses corresponding to (3) are assigned to ψ , and FALSE is returned to continue the search looking for “better” plans. When PREF-DLL terminates, the last plan found is optimal (if one exists), i.e., no plan exists which is preferred to the actual π_{opt} , as stated by the following theorem, which extends to planning as satisfiability previous results.

Theorem 2 *Let Π be a planning problem, n the makespan, and $\langle P, \prec \rangle$ a qualitative preference. QL-PLAN-GNT-1 and QL-PLAN-GNT-2 terminate, and then π_{opt} is empty if Π does not have a plan of makespan n , and an optimal plan wrt $\langle P, \prec \rangle$ for makespan n , otherwise.*

Theorem 2 follows from the correctness of PREF-DLL (Theorem 2 in (DiRosa, Giunchiglia, and Maratea 2008))

and the assumptions on cnf .

Notice that each time a new π' is found at line 4 of Figure 1, which is better than the actual π_{opt} , ψ may be overwritten because we can discard the clauses added because of π since they are entailed by the new clauses added because of π' . as stated by the following theorem.

Theorem 3 *Let $\langle P, \prec \rangle$ be a qualitative preference. Let $\pi_1, \pi_2, \dots, \pi_k$ be the sequence of plans computed by QL-PLAN-GNT-1 or QL-PLAN-GNT-2, and $\psi_1, \psi_2, \dots, \psi_k$ be the corresponding formulas computed as in (3). For each i , $0 < i < n$, ψ_{i+1} entails ψ_i .*

As a consequence, in PREF-DLL, the formula ψ is overwritten as soon as a new model π is found (line 4). QL-PLAN-GNT-1 and QL-PLAN-GNT-2 are thus guaranteed to work in polynomial space in the size of the input planning problem, makespan and preference.

Going back to our original problems, if we want to compute plans with minimal number of actions, assuming $act(\Pi_n)$ is the set of variables in Π_n corresponding to action variables, it is enough to set

1. $P := \{\bar{a} \mid a \in act(\Pi_n)\}$; and
2. $\prec := \emptyset$,

and then calling the QL-PLAN-GNT-1 algorithm to obtain the expected result (in the qualitative case). If we are interested in computing plans with the maximal number of soft goals satisfied, given SG to be the set of soft goals of the problem, it is enough to set

1. $P := \{v(p) \mid p \in SG\}$; and
2. $\prec := \emptyset$,

and then calling the QL-PLAN-GNT-2 algorithm to obtain the expected result (again in the qualitative case).

As an example of the behavior of the QL-PLAN-GNT-1 algorithm in Figure 1, consider the planning problem (2) (which thus corresponds to Π_n) and the preference a . at the end of the previous section, the search for a minimal-actions plan may proceed (depending on *ChooseLiteral*) as follows:

1. $\pi_1 = \{Car_0, Bike_0, Bus_0\}$, then $\psi_1 : \overline{Car_0} \vee \overline{Bike_0} \vee \overline{Bus_0}$, i.e., at least one action has to be assigned as in P . Assume the next plan is
2. $\pi_2 = \{Bike_0, Bus_0\}$, then $\psi_2 : (\overline{Bike_0} \vee \overline{Bus_0}) \wedge \overline{Car_0}$, which asks that at least one among $Bike_0$ and Bus_0 has to be assigned as in P , while Car_0 has to remain assigned as in P ; assume now the next computed plan is
3. $\pi_3 = \{Bike_0\}$, then $\psi_3 : \overline{Bike_0} \wedge \overline{Car_0} \wedge \overline{Bus_0}$, which would be satisfied only by a plan where all actions are not performed: given this plan does not satisfy (the constraints related to the transition relation of) (2) (which, intuitively, say “at least one action has to be performed”), π_3 is optimal. In PREF-DLL, this is achieved by means of no other plan is computed at line 4 of Figure 1, and the procedure eventually exits at line 3 with π_3 as π_{opt} .

It is clear from the example how the quality of the plan increases at each new solution found. The QL-PLAN-GNT-2 algorithm behaves similarly.

	SATPLAN(P)(W)	SATPLAN-GNT(W)	SATPLAN(P)()	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 1: Results on domains coming from IPCs. $x(y)$ stands for y instances solved with x secs of mean CPU time.

Implementation and experimental evaluation

As we already said in the introduction, we used SATPLAN (ver. of Feb. 2006) as underlying planning system. SATPLAN is the most famous SAT-based system and we already mentioned its good performances in IPCs. SATPLAN can only handle STRIPS domains. We thus extended SATPLAN in order to incorporate such ideas (i.e., to implement QL-PLAN-GNT-1/QL-PLAN-GNT-2 at each makespan of the SATPLAN’s approach, till the optimal makespan), and we called SATPLAN-GNT the overall system: it implements PREF-DLL in MINISAT which is also one of the solvers SATPLAN can use, and that we set as default for SATPLAN.²

Experiments for minimal-actions plans

We considered several STRIPS domains from the first five IPCs (the recent IPC-6 does not have basic STRIPS problems). Given $P := \{\bar{a} | a \in act(\Pi_n)\}$ defined above, we considered both the qualitative preference $\langle P, \emptyset \rangle$ and the quantitative preference $\langle P, c \rangle$ in which c is the constant function 1, i.e, the setting where a uniform cost is associated to “not to perform” each action: the related objective function is thus the minimization of the number of action involved in the plan. We used Warners encoding (Warners 1998) (denoted with “W”) to reduce to qualitative preferences (with a non-empty partial order): it showed the best performances on planning problems in (Giunchiglia and Maratea 2007a; 2007b), and it is thus the same used in SATPLAN(P). In Table 1 there are the results of our analysis. The first column is the domain of problem, then SATPLAN(P)(W) and SATPLAN-GNT(W) (resp. SATPLAN(P)() and SATPLAN-GNT()) denote the systems working on the quantitative (resp. qualitative) case. Results are presented as in the Max-SAT

Evaluations³ by $x(y)$, where y is the number of solved instances within the time limit (900s on a Linux box equipped with a Pentium IV 3.2GHz processor and 1GB of RAM), and x is the mean solving time of solved instances (used to break ties). “–” means that no instance is solved within the time limit. We can note that in the quantitative case SATPLAN-GNT(W) has an edge over SATPLAN(P)(W): it often solves more instances, and never less, while in the qualitative case results are mixed. In general, SATPLAN-GNT convergence to the optimal solution is effective, and only few iterations are needed: we performed a detailed analysis on selected benchmarks, and we noted that, in mean, only 2.5 iterations were needed, and the quality of the first plan was already very good. In the qualitative case, on same domains, splitting preferentially on action variables, without the burden introduced by the W-encoding, can efficiently lead to the optimal solution. Finally, note that often the differences in performances are in the order of one/few instances, or just in term of mean CPU time: this is in line with state-of-the-art results, given that often in optimization problems solving even one more, or just in a faster way, the available benchmarks can be a significant result (e.g., in the Max-SAT Evaluations, where often the domain winner is granted by only the mean CPU time). Considering the length of the plans returned by SATPLAN-GNT(W) (and thus SATPLAN(P)(W)⁴ and plain SATPLAN we want to mention that the reduction percentage for the domains in Table 1 goes from 54% for mprime to no reduction in 5 domains (this is because no reduction is possible given the structure of the problems, or because the approach stops at the optimal makespan).

²SATPLAN’s default solver is SIEGE: we run SATPLAN with SIEGE and MINISAT and we have seen no significant differences in performances in terms of both CPU time and plans quality.

³See <http://www.maxsat07.udl.es/> for the last.

⁴Note that, in general, this is not the case for SATPLAN-GNT() and SATPLAN(P)(), given they return subset-minimal solutions.

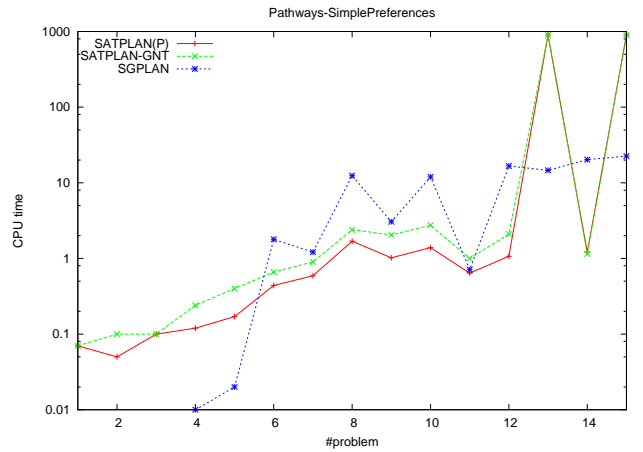
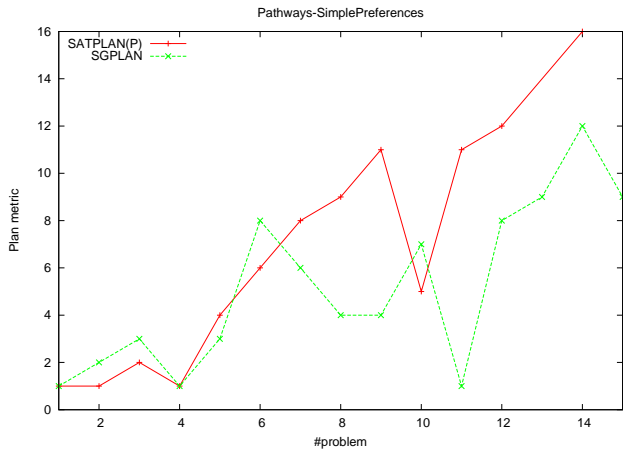


Figure 2: Pathways domain, “SimplePreferences” track of IPC-5. Left: Plan metric, i.e., number of unsatisfied soft goals, for SATPLAN-GNT(W) (and thus SATPLAN(P)(W)) and SGPLAN. Right: CPU time for SATPLAN-GNT(W), SATPLAN(P)(W) and SGPLAN (in log scale).

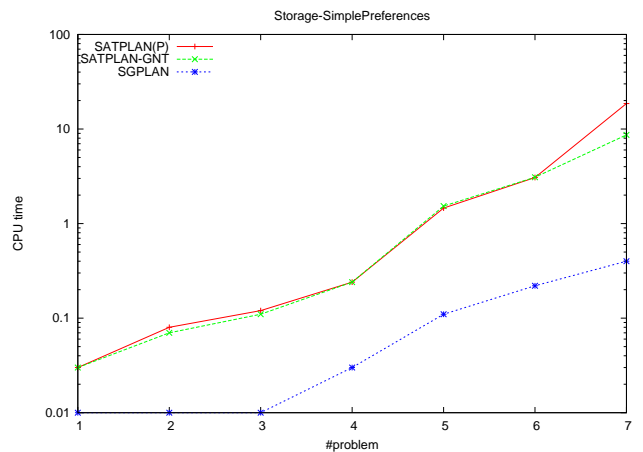
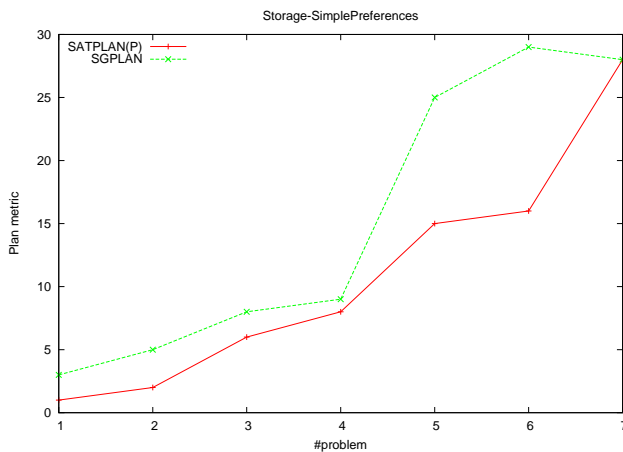


Figure 3: Storage domain, “SimplePreferences” track of IPC-5. Left: Plan metric, i.e., number of unsatisfied soft goals, for SATPLAN-GNT(W) (and thus SATPLAN(P)(W)) and SGPLAN. Right: CPU time for SATPLAN-GNT(W), SATPLAN(P)(W) and SGPLAN (in log scale).

Experiments with soft goals

For this part of the experiments we considered two types of problems. First, we evaluated SATPLAN-GNT on some of the instances from (Giunchiglia and Maratea 2007a; 2007b). Such instances were created from the original STRIPS instances of the domain we mentioned above, but considering all goals as being “soft” (but with the constraint that at least one goal has to be reached, otherwise the empty plan is always a solution). We do not show detailed results for this analysis, and we just summarize it. The vast majority of these benchmarks were already solved efficiently by SATPLAN(P). We have considered 10 problems (each from a different domain) where SATPLAN(P) took considerable time to solve, in the quantitative case: on such problems,

SATPLAN-GNT(W) took in mean around half a time wrt SATPLAN(P)(W) to solve them. In the qualitative case, all problems are solved quite easily by both SATPLAN-GNT() and SATPLAN(P)(). But, besides the fact that in the instances so far mentioned goals are precisely soft, i.e., they can be satisfied, or not, without affecting plan validity, such instances are not fully satisfactory because goals are non-conflicting, i.e., all soft goals can be (eventually) satisfied at the same time.

For this reason, given that the case in which not all the goals can be satisfied (often called over-subscription planning) is practically very important, we also evaluated some domains from the “SimplePreferences” track of the IPC-5, which include the possibility to express and reason on

conflicting soft goals. Given that such domains are non-STRIPS, and some ADL constructs are used, we have used the following compilation technique: the preferences (goals) in the IPC-5 problems are translated into preconditions of dummy actions, which achieve new dummy literals defining the new problem goals. Then, these new actions can be compiled into STRIPS actions by using an existing tool (we have used both Hoffmann’s tool for compiling ADL actions into STRIPS actions, namely ADL2STRIPS, and a modification of the same tool used in IPC-5, based on LPG). In our analysis we have included the domains where all goals are soft (but conflicting in general, changing all weights associated to goals violation to be the same), and preferences are only expressed on goals, i.e., the Storage and Pathways domains. Results are presented as in the reports of the IPC-5, considering, for each domain, both plan metric and CPU time to find the plan. In the analysis, we considered SATPLAN-GNT(W) and SATPLAN(P)(W) (given the metric is defined quantitatively in IPC-5 on soft goals) and, as a reference, SGPLAN, the clear winner of the “SimplePreferences” track at IPC-5.

For the Pathways domain in Figure 2 we can note (Figure 2 Right) that SATPLAN-GNT(W) and SATPLAN(P)(W) perform similarly (with SATPLAN(P)(W) being slightly better) and better than SGPLAN for non-easy (i.e., from problem #6, as numbered in the IPC-5) problems, but for two problems (#13 and #15) only solved by SGPLAN within the time limit. About the plan metric (Figure 2 Left), we can see that SGPLAN, overall, returns plans of slightly better quality, i.e., it can satisfy more soft goals. For the Storage domain, instead, in Figure 3 (Right) we can note that all systems solve all instances considered⁵, with SGPLAN being around one order of magnitude faster than the other systems, which nonetheless solve each problem in less than 20s, with SATPLAN-GNT(W) being faster of around a factor of 2. This fact is in line with all our results, given that this domain includes a high number of preferences on the biggest instances we considered. The reason for the performance gap wrt SGPLAN can be immediately explained by looking at Figure 3 (Left): SATPLAN-GNT(W) and SATPLAN(P)(W) return plans of much better quality than SGPLAN. The tradeoff between CPU performances and plan quality of SATPLAN-GNT (and SATPLAN(P)) seems to be very effective, at least on this domain, further considering that (i) it (unlike SGPLAN) is not tailored for finding general optimal solutions (but only bounded to the optimal makespan), and (ii) SGPLAN was by far the best systems on these domains in the “SimplePreferences” track at IPC-5.

Conclusions, related and future works

In this paper we have presented a generate-and-test approach for finding optimal plans which, differently to a previous SAT-based approach (i) does not constrain the heuristic, (ii) works in polynomial space, and (iii) often shows performance benefits. The most related approaches in planning are the ones in (Brafman and Chernyavsky 2005;

Büttner and Rintanen 2005). In (Brafman and Chernyavsky 2005), the authors show how to extend GP-CSP (Do and Kambhampati 2001) in order to planning with preferences expressed as a TCP-net (Boutilier et al. 2004). In the Boolean case, TCP-net can be expressed as Boolean formulas, and the problem they consider is the same we deal with. No implementation of the approach is provided, or at least that we are aware of, to which we can compare to. Though this work is not based on satisfiability, the problem they consider is the same we deal with: find an optimal plan wrt the given preferences among the plans with makespan n . We can also deal with quantitative preferences. However, both approaches can be (easily) extended in order to work without a bounded horizon, by simply using an iterative deepening approach, i.e., by successively incrementing n , each time using the previously found solutions to bound the search space, up to a point in which we are guaranteed to have an optimal solution independent from the bound n . This is the approach followed in (Büttner and Rintanen 2005), where the problem considered is to extend the planning as satisfiability approach in order to find plans with optimal sequential length. Interestingly, the authors use a Boolean formula to encode the function representing the sequential length of the plan. In their approach, for a given n , the search for an optimal solution is done by iteratively calling the SAT solver, each time adding a constraint imposing a smaller value for the objective function (using (Bailleux and Boufkhad 2003)): when the SAT formula becomes unsatisfiable, n is set to $n + 1$ and the process is iterated looking for a better plan than the one so far discovered. For a fixed n , the problem considered in (Büttner and Rintanen 2005) is exactly one of the problems we can deal with: finding a plan with the minimum number of actions for a planning problem Π with makespan n , thus using a quantitative approach. Our approach can also deal with “soft” goals, and with the qualitative approach to all problems. Another, algorithmic, difference between our work and (Büttner and Rintanen 2005) is that, iteratively calling the SAT solver as a black box, the nogoods computed during a call are not re-used by the following calls for the same n , while this is not the case for us. Also other, search-based, planners, e.g., LPG-QUALITY (Gerevini, Saetti, and Serina 2008), do anytime search and gradually improve the solution quality.

We are currently extending the analysis regarding soft goals, in order to include more domains from the IPC-5. In the future we want to compare our system with the IP-based planner OPTIPLAN (van den Briel and Kambhampati 2005). We then plan to relax the computation of makespan-optimal plans in order to find even better solutions when dealing with problems involving both optimizations wrt actions, e.g., in the quantitative case (Büttner and Rintanen 2005), and, more recently, (Chen, Lv, and Huang 2008), and optimizations wrt soft goals. We also plan to address non-uniform action costs (e.g., (Keyder and Geffner 2008; Chen, Lv, and Huang 2008)), by dealing with the “action costs” requirement introduced in IPC-6.

⁵We have considered all instances that the tools could compile. We are contacting the authors to be able to possibly compile bigger instances.

References

- Bailleux, O., and Boufkhad, Y. 2003. Efficient CNF encoding of boolean cardinality constraints. In Rossi, F., ed., *Proc. of the 9th International Conference on Principles and Practice of Constraint Programming (CP)*, volume 2833 of *Lecture Notes in Computer Science*, 108–122. Springer.
- Boutillier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research* 21:135–191.
- Brafman, R. I., and Chernyavsky, Y. 2005. Planning with goal preferences and constraints. In Biundo, S.; Myers, K. L.; and Rajan, K., eds., *Proc. of the 15th International Conference on Automated Planning and Scheduling (ICAPS)*, 182–191. AAAI Press.
- Büttner, M., and Rintanen, J. 2005. Satisfiability planning with constraints on the number of actions. In Biundo, S.; Myers, K. L.; and Rajan, K., eds., *Proc. of the 15th International Conference on Automated Planning and Scheduling (ICAPS)*, 292–299. AAAI Press.
- Castell, T.; Cayrol, C.; Cayrol, M.; and Berre, D. L. 1996. Using the Davis and Putnam procedure for an efficient computation of preferred models. In *Proc. of the 12th European Conference on Artificial Intelligence (ECAI)*, 350–354. John Wiley and Sons, Chichester.
- Chen, Y.; Lv, Q.; and Huang, R. 2008. Plan-A: A cost-optimal planner based on SAT-constrained optimization. In *Proc. of 6th International Planning Competition, ICAPS-08*.
- DiRosa, E.; Giunchiglia, E.; and Maratea, M. 2008. A new approach for solving satisfiability problems with qualitative preferences. In *Proc. of ECAI-08*, 510–514. IOS Press.
- Do, M. B., and Kambhampati, S. 2001. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artificial Intelligence* 132(2):151–182.
- Gavanelli, M. 2002. An algorithm for multi-criteria optimization in CSPs. In van Harmelen, F., ed., *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI)*, 136–140. IOS Press.
- Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the 5th IPC: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence* 173(5-6):619–668.
- Gerevini, A.; Saetti, A.; and Serina, I. 2008. An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artificial Intelligence* 172(8-9):899–944.
- Giunchiglia, E., and Maratea, M. 2007a. Planning as satisfiability with preferences. In *Proc. of the 22nd AAAI Conference on Artificial Intelligence*, 987–992. AAAI Press.
- Giunchiglia, E., and Maratea, M. 2007b. SAT-based planning with minimal-#actions plans and "soft" goals. In Basili, R., and Pazienza, M. T., eds., *Proc. of the 10th Congress of the Italian Association for Artificial Intelligence*, volume 4733 of *Lecture Notes in Computer Science*, 422–433. Springer.
- Hoffmann, J., and Edelkamp, S. 2005. The deterministic part of IPC-4: An overview. *Journal of Artificial Intelligence Research* 24:519–579.
- Järvisalo, M.; Junttila, T. A.; and Niemelä, I. 2005. Unrestricted vs restricted cut in a tableau method for boolean circuits. *Annals of Mathematics and Artificial Intelligence* 44(4):373–399.
- Kautz, H., and Selman, B. 1992. Planning as satisfiability. In Neumann, B., ed., *Proc. of the 10th European Conference on Artificial Intelligence (ECAI)*, 359–363. IOS Press.
- Kautz, H., and Selman, B. 1999. Unifying SAT-based and graph-based planning. In Dean, T., ed., *Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, 318–325. Morgan-Kaufmann.
- Keyder, E., and Geffner, H. 2008. Heuristics for planning with action costs revisited. In Ghallab, M.; Spyropoulos, C. D.; Fakotakis, N.; and Avouris, N. M., eds., *Proc. of 18th European Conference on Artificial Intelligence (ECAI)*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, 588–592. IOS Press.
- Tseitin, G. 1970. On the complexity of proofs in propositional logics. *Seminars in Mathematics* 8.
- van den Briel, M., and Kambhampati, S. 2005. Optiplan: Unifying ip-based and graph-based planning. *Journal of Artificial Intelligence Research* 24:919–931.
- Warners, J. P. 1998. A linear-time transformation of linear inequalities into CNF. *Information Processing Letters* 68(2):63–69.
- Xing, Z.; Chen, Y.; and Zhang, W. 2006. Maxplan: Optimal planning by decomposed satisfiability and backward reduction. In *Proc. of 5th International Planning Competition, ICAPS-06*, 53–55.