Planning as Satisfiability with Preferences *

Enrico Giunchiglia and **Marco Maratea**
DIST - University of Genova, Viale F. Causa 15, Genova, Italy
{enrico,marco@dist.unige.it}

Abstract

Planning as Satisfiability is one of the most well-known and effective technique for classical planning: SATPLAN has been the winning system in the deterministic track for optimal planners in the 4th International Planning Competition (IPC) and a co-winner in the 5th IPC.

In this paper we extend the Planning as Satisfiability approach in order to handle preferences and SATPLAN in order to solve problems with simple preferences. The resulting system, SATPLAN(P) is competitive with SGPLAN, the winning system in the category "simple preferences" at the last IPC. Further, we show that SATPLAN(P) performances are (almost) always comparable to those of SATPLAN when solving the same problems without preferences: in other words, introducing simple preferences in SATPLAN does not affect its performances. This latter result is due both to the particular mechanism we use in order to incorporate preferences in SATPLAN and to the relative low number of soft goals (each corresponding to a simple preference) usually present in planning problems. Indeed, if we consider the issue of determining minimal plans (corresponding to problems with thousands of preferences) the performances of SATPLAN(P) are comparable to those of SATPLAN in many cases, but can be significantly worse when the number of preferences is very high compared to the total number of variables in the problem. Our analysis is conducted considering both qualitative and quantitative preferences, different reductions from quantitative to qualitative ones, and most of the propositional planning domains from the IPCs and that SATPLAN can handle.

Introduction

Planning as Satisfiability (Kautz and Selman 1992) is one of the most well-known and effective technique for classical planning: SATPLAN (Kautz and Selman 1999) is a planner based on propositional satisfiability (SAT) and, considering the track for optimal propositional planner, it has been the winning system in the 4th International Planning Competition (IPC) (Hoffmann and Edelkamp 2005) and a co-winner in the last IPC (IPC-5). Given a planning problem Π, the basic idea of planning as satisfiability is to convert the problem of determining the existence of a plan for Π with a fixed makespan n into a SAT formula φ such that there is a one-to-one correspondence between the plans of Π with makespan n and the interpretations satisfying φ. Of course, for SATPLAN effectiveness, it is crucial the availability of very effective SAT solvers, like MINISAT (Eén and Sörensson 2003). MINISAT is based on the Davis-Logemann-Loveland procedure (DLL) like most of the state-of-the-art SAT checkers, and won the last SAT competition in 2005.

In this paper we extend the Planning as Satisfiability approach in order to handle both qualitative and quantitative preferences, and SATPLAN in order to solve problems with simple preferences. The basic idea, extending the results presented in (Giunchiglia and Maratea 2006) in the SAT setting, is to explore the search space of possible plans in accordance with the expressed preferences, i.e., to force the splitting of the SAT solver in order to follow the given partial order on preferences. Qualitative preferences are naturally handled in this way, while quantitative preferences need an encoding of the objective function to minimize/maximize. Our system, that we call SATPLAN(P), allows for handling both qualitative and quantitative preferences, the latter under two possible encodings of the objective function. We show that SATPLAN(P) is competitive with SGPLAN when considering simple preferences, and that SATPLAN(P) is as effective as SATPLAN on all the problems we consider, i.e., that introducing simple preferences in SATPLAN does not affect its performances. The first result is remarkable given that SGPLAN is the clear winner in the category "simple preferences" in the last IPC. The second result is somehow surprising, given that limiting the splitting of the SAT solver can cause an exponential degradation of its performances (Järvisalo, Junttila, & Niemelä 2005). We correlate the good behavior of SATPLAN(P) to the relative low number (in the order of tens) of soft goals (i.e., simple preferences) in the problems. To evaluate the impact of the number of preferences (wrt the total number of variables) on the performances of SATPLAN(P) we considered the problem of determining "minimal" plans: in this case, any action variable corresponds to a preference and it is thus common to have problems with several thousands of preferences. We show that the performances of SATPLAN are not affected even in this settings for many problems. However, SATPLAN(P) can be much slower than SATPLAN when considering problems with a very high ratio (i.e., $> 45\%$) of number of preferences (i.e., action variables) to the total number of variables. Of course, in most cases SATPLAN(P) re-

*The authors would like to thank Chic-Wei Hsu for his help with SGPLAN. This work has been partially supported by MIUR.

turns plans with better quality than plain SATPLAN both in the case of planning with soft goals and in the case of determining minimal plans. Our analysis is conducted considering both qualitative and quantitative preferences, different encodings of the objective function in the case of quantitative preferences, and most of the propositional planning domains considered in the IPCs and that SATPLAN can handle. About the encodings, our experimental analysis points out that the one based on (Warners 1998) leads to better performances than that based on (Bailleux and Boufkhad 2003) on most problems. This last fact does not agree with the good results presented in (Bailleux and Boufkhad 2003; Büttner and Rintanen 2005). Summing up, in this paper:

1. We extend (i) the planning as satisfiability approach in order to deal with both qualitative and quantitative preferences, and (ii) SATPLAN in order to handle simple preferences. We use the technique presented in (Giunchiglia and Maratea 2006) in the SAT setting for solving MAXSAT and MINONE problems.
2. We show that SATPLAN(P) is competitive with the state-of-the-art system SGPLAN when considering soft goals, and as competitive as SATPLAN when the latter is given the same problems but without soft goals.
3. We show that the performances of SATPLAN(P) are comparable to those of SATPLAN (when given the same problems without preferences) even when there are a high number of preferences, and that large deviations are possible if the ratio between the number of preferences to the total number of variables is relatively high ($> 45\%$).
4. In the case of quantitative preferences, our analysis shows that the encoding based on (Warners 1998) leads to better performances than the one based on (Bailleux and Boufkhad 2003).

Basic 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* is an interpretation of the action signature. Intuitively, a complex action α models the concurrent execution of the actions satisfied by α.

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 preferred this formulation because the techniques we are going to describe are independent of 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

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

and a *plan for* Π_n is an interpretation satisfying (1).

For example, considering the planning problem of going to work from home. Assuming that we can use the car or the bus or the bike, this scenario can be easily formalized using a single fluent variable *AtWork* and three action variables *Car*, *Bus* and *Bike* with the obvious meaning. The problem with makespan 1 can be expressed by the conjunction of the formulas:

$$\begin{array}{c} \neg \mathit{AtWork}_0, \\ \mathit{AtWork}_1 \equiv \neg \mathit{AtWork}_0 \equiv (\mathit{Car}_0 \vee \mathit{Bus}_0 \vee \mathit{Bike}_0), \\ \mathit{AtWork}_1, \end{array} \tag{2}$$

in which the first formula corresponds to the initial state, the second to the transition relation, and the third to the goal state. (2) has 7 plans (i.e., satisfying interpretations), each corresponding to a non-empty subset of $\{\mathit{Car}_0, \mathit{Bus}_0, \mathit{Bike}_0\}$.

Preferences and Optimal Plans

Let Π_n be a planning problem Π with makespan n.

In addition to the goals, it may be desirable to have plans satisfying other conditions. For example, considering the problem (2), in addition to being at work at time 1, we may want to avoid taking the bus (at time 0). Formally this preference is expressed by the formula $\neg \mathit{Bus}_0$, and it amounts to prefer the plans satisfying $\neg \mathit{Bus}_0$ to those satisfying Bus_0. In general, there can be more than one preference, and it may not be possible to satisfy all of them. For example, in (2) it is not possible to satisfy the three possible preferences $\neg \mathit{Bike}_0$, $\neg \mathit{Bus}_0$ and $\neg \mathit{Car}_0$.

A "qualitative" solution to the problem of conflicting preferences is to define a partial order on them. A *qualitative preference (for* Π_n*)* is a pair $\langle P, \prec\rangle$ where P is a set of formulas (the preferences) whose atoms are in Π_n, and $\prec$ is a partial order on P. The partial order can be empty, meaning that all the preferences are equally important. The partial order can be extended to plans for Π_n. Consider a qualitative preference $\langle P, \prec\rangle$. 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

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 $\pi_1 \not\models p_2$, and this good reason is that $\pi_1 \models p_1$, $p_1 \prec p_2$ and $\pi_2 \not\models p_1$. 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 $\langle P, \prec\rangle$).

A "quantitative" approach to solve the problem of conflicting preferences is to assign weights to each of them, and then minimize/maximize a given objective function involving the preferences and their weights. In most cases the objective function is the weighted sum of the preferences: this has been the case of each planning problem with preferences in the last IPC. With this assumption, a *quantitative preference (for* Π_n*)* can be defined as a pair $\langle P, c\rangle$ where P is a set of formulas in Π_n signature (as before) and c is a function associating an integer to each preference in P. Without loss of generality, we can further assume that $c(p) \geq 0$ for each $p \in P$ and that we are dealing with a maximization problem. Thus, a plan is *optimal (wrt* $\langle P, c\rangle$*)* if it maximizes[1]

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

For instance, considering the planning problem (2), if we have the qualitative (resp. quantitative) preference

- $\langle\{\neg \mathit{Bike}_0, \neg \mathit{Bus}_0, \neg \mathit{Car}_0\}, \emptyset\rangle$, (resp. $\langle\{\neg \mathit{Bike}_0, \neg \mathit{Bus}_0, \neg \mathit{Car}_0\}, c\rangle$, where c is the constant function 1) then there are three optimal plans, corresponding to $\{\mathit{Bike}_0\}, \{\mathit{Bus}_0\}, \{\mathit{Car}_0\}$.
- $\langle\{\neg \mathit{Bike}_0, \neg \mathit{Bus}_0, \neg \mathit{Car}_0\}, \{\neg \mathit{Bike}_0 \prec \neg \mathit{Car}_0\}\rangle$, (resp. $\langle\{\neg \mathit{Bike}_0, \neg \mathit{Bus}_0, \neg \mathit{Car}_0\}, c\rangle$, where $c(\neg \mathit{Bike}_0) = 2$ while $c(\neg \mathit{Bus}_0) = c(\neg \mathit{Car}_0) = 1$) then there are two optimal plans, corresponding to $\{\mathit{Bus}_0\}, \{\mathit{Car}_0\}$.
- $\langle\{\mathit{Bike}_0 \vee \mathit{Bus}_0\}, \emptyset\rangle$, (resp. $\langle\{\mathit{Bike}_0 \vee \mathit{Bus}_0\}, c\rangle$, where c is the constant function 1) then all the plans except for the one corresponding to $\{\mathit{Car}_0\}$ are optimal.

Planning as Satisfiability with preferences

Let Π_n be a planning problem with makespan n. Consider a qualitative preference $\langle P, \prec\rangle$. In planning as satisfiability, plans for Π_n are generated by invoking a SAT solver on Π_n. Optimal plans for Π_n can be obtained by

1. Encoding the preference P as a formula to be conjoined with Π_n; and
2. Modifying DLL in order to search first for optimal plans, i.e., to branch according to the partial order $\prec$.

The resulting procedure is reported in Figure 1 in which:

- 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 $p \prec p'$;
- *cnf*(φ), where φ is a formula, is a set of clauses (i.e., set of sets of literals) such that for any interpretation μ in the signature of *cnf*(φ), $\mu \models$ *cnf*(φ) iff $\mu \models \varphi$: there are well known methods for computing *cnf*(φ) in linear time by introducing additional variables.
- S is an *assignment*, i.e., a consistent set of literals. An assignment S corresponds to the partial interpretation mapping to true the literals $l \in S$.
- 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;
- *ChooseLiteral*$(\varphi, S, P', \prec')$ returns an *unassigned* literal l (i.e., such that $\{l, \bar{l}\} \cap S = \emptyset$) in φ such that either all the variables is P' are assigned, or $l \in P'$ and all the other variables $v(p) \in P'$ with $v(p) \prec l$ are assigned.

```
function QL-PLAN(Π_n,P,≺)
  1 return OPT-DLL(cnf(Π_n ∧ ∧_{p∈P}(v(p) ≡ p)),∅,v(P),v(≺))

function OPT-DLL(φ,S,P',≺')
  2 if (∅ ∈ φ) return FALSE;
  3 if (φ = ∅) return S;
  4 if ({l} ∈ φ) return OPT-DLL(φ_l, S ∪ {l}, P', ≺');
  5 l := ChooseLiteral(φ, S, P', ≺');
  6 V := OPT-DLL(φ_l, S ∪ {l}, P', ≺');
  7 if (V ≠ FALSE) return V;
  8 return OPT-DLL(φ_l̄, S ∪ {l̄}, P', ≺').
```

Figure 1: The algorithm of QL-PLAN

As it can be see from the figure, OPT-DLL is the standard DLL except for the modification in the heuristic, i.e., *ChooseLiteral* which initially selects literals according to the partial order $\prec'$. If we have two preferences $p_1 = (\neg \mathit{Bike}_0 \wedge \neg \mathit{Bus}_0 \wedge \neg \mathit{Car}_0)$ and $p_2 = (\neg \mathit{Bike}_0 \wedge \neg \mathit{Bus}_0)$ with $p_1 \prec p_2$ and we consider the problem (2), OPT-DLL returns the plan corresponding to $\{\mathit{Car}_0\}$ determined while exploring the branch extending $\{\neg v(p_1), v(p_2)\}$. This plan is optimal, and, in general, it can be proved that QL-PLAN(Π_n,P,$\prec$) returns

1. FALSE if Π_n has no plans, and
2. an optimal plan for Π_n wrt $\langle P, \prec\rangle$, otherwise.

Consider now a quantitative preference $\langle P, c\rangle$. The problem of finding an optimal plan for Π_n wrt $\langle P, c\rangle$ can be solved again using OPT-DLL in Figure 1 as core engine.

[1] Assuming that $c(p) < 0$ for some $p \in P$, we can replace p with $\neg p$ in P and define $c(\neg p) = -c(p)$: the set of optimal plans does not change. Given $\langle P, c\rangle$ and assuming we are interested in minimizing the objective function (3), we can consider the quantitative preference $\langle P', c'\rangle$ where $P' = \{\neg p : p \in P\}$ with $c'(\neg p) = c(p)$, and then look for a plan maximizing $\sum_{p \in P' : \pi \models p} c'(p)$.

function QT-PLAN(Π_n,P,c)
1 **return** OPT-DLL($cnf(\Pi_n \wedge Bool(P,c))$,$\emptyset$,$b(c)$,$p(c)$)

Figure 2: The algorithm of QT-PLAN

The basic idea is to 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 $\langle \{b_{n-1}, \ldots, b_0\}, \{b_{n-1} \prec b_{n-2}, \ldots, b_1 \prec b_0\} \rangle$. In more details, let $Bool(P,c)$ be a Boolean formula such that:

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

If the above conditions are satisfied, we say that $Bool(P,c)$ is a *Boolean encoding of* $\langle P,c \rangle$ *with output* $b_{n-1}, \ldots, b_0$. $Bool(P,c)$ can be realized in polynomial time in many ways, see, e.g., (Warners 1998).

The resulting procedure is represented in Figure 2 in which $Bool(P,c)$ is a Boolean encoding of $\langle P,c \rangle$ with output $b_{n-1}, \ldots, b_0$, $b(c) = \{b_{n-1}, \ldots, b_0\}$ and $p(c)$ is the partial order $b_{n-1} \prec b_{n-2}, \ldots, b_1 \prec b_0$.

If we have two preferences $p_1 = (\neg Bike_0 \wedge \neg Bus_0 \wedge \neg Car_0)$ and $p_2 = (\neg Bike_0 \wedge \neg Bus_0)$ with $c(p_1) = 2$ and $c(p_2) = 1$, then two bits b_1 and b_0 are sufficient as output of $Bool(\{p_1, p_2\}, c)$. Further, if we consider the problem (2), OPT-DLL returns the plan corresponding to $\{Car_0\}$ determined while exploring the branch extending $\neg b_1, b_0$. This plan is optimal, and, as in the qualitative case, it can be proved that QT-PLAN(Π_n,P,c) returns

1. FALSE if Π_n has no plans, and
2. an optimal plan for Π_n wrt $\langle P,c \rangle$, otherwise.

Implementation and Experimental Analysis

As we already said in the introduction, we use SATPLAN as underlying planning system. SATPLAN is the most famous system using the planning as satisfiability approach and we already mentioned its good performances in IPCs. SATPLAN can only handle propositional domains, and, among them, we considered the pipesworld, satellite, airport, promela philosophers and optical, psr, depots, driverLog, zenoTravel, freeCell, logistic, blocks, mprime and mistery domains from the first 4 IPCs, and pathway, storage, tpp and trucks from IPC-5. Notice that we do not consider the domains in the "simple preferences" track in IPC-5 because they are not handled by SATPLAN: this task would lead to a main reimplementation of the system. Consequently, we do not use such domains even for SGPLAN, otherwise the results would be incomparable. These are standard planning problems in which the goal corresponds to a set S of literals and without soft goals. We modified these problems in order to interpret all the literals in S as "soft goals". More precisely, we considered both the qualitative preference $\langle S, \emptyset \rangle$ and the quantitative preference $\langle S, c \rangle$ in which c is the constant function 1. These preferences and constraint encode the fact that the goals in S are now "soft" in the sense that it is desirable but not necessary to achieve them.

We implemented OPT-DLL in MINISAT which is also one of the solvers SATPLAN can use, and that we set as default for SATPLAN.[2] In the case of quantitative preferences, we considered Warners' (1998), and also Bailleux and Boufkhad's (2003) encodings of $Bool(P,c)$, denoted with W-encoding and BB-encoding respectively. The size of W-encoding is linear in $|P|$ while BB-encoding is quadratic. However, the latter has better computational properties and it has been reported in the literature to lead to good results (Bailleux and Boufkhad 2003; Büttner and Rintanen 2005). In the following, we use SATPLAN(s), SATPLAN(w) and SATPLAN(b) to denote SATPLAN modified in order to handle qualitative, quantitative with W-encoding and quantitative with BB-encoding preferences, respectively.

In our first experiments, we consider SATPLAN(w)/(b)/(s), SGPLAN (Hsu *et al.* 2007), and plain SATPLAN. SGPLAN has been the clear winner in the category "Simple Preferences Domains" in the recent IPC-5 and is thus the reference system for the problems that we consider. SATPLAN has been included in order to evaluate the differences in the performances between our systems and SATPLAN itself: we expect SATPLAN to satisfy less soft goals but to have performances no worse than SATPLAN(w)/(b)/(s). A final crucial observation: Since there are no "hard" goals, the various versions of SATPLAN would always find a valid plan, even when the makespan n is 0 (in which case the returned plan would be the empty one). In order to avoid this situation, for SATPLAN/(w)/(b)/(s) we added a constraint saying that at time n at least one of the soft goals has to be satisfied. Because of this, we discarded the problems whose original version has only one goal because they would have no soft goal.

All the tests have been run on a Linux box equipped with a Pentium IV 2.4GHz processor and 512MB of RAM. Timeout has been set to 300s. The results are shown in Table 1.

In the Table, each row and column is an abbreviation for the corresponding class of problems and system respectively; the first (resp. second) number reports the total number of soft goals satisfied (resp. timeouts or segmentation faults) in that class of problems. Looking at the table, we see that SATPLAN(w)/(b) never time out and that on 11 out of the 19 classes of problems we considered, they manage to satisfy more soft goals than SGPLAN (and SATPLAN). SGPLAN on the other hand, has several time outs/segmentation faults, the latter due to the high number of grounded operators in the problems.[3]

Figure 3 shows the performances of the systems. In the figure, each point (x, y) means that in y seconds, x problems are solved. The figure shows the very good performances

[2] SATPLAN's default solver is SIEGE: we run SATPLAN with SIEGE and MINISAT and we have seen no significant differences in SATPLAN's performances.

[3] Personal communication with the authors. The authors also suggested that the ADL version of the problems may lead to less segmentation faults. We tried the ADL versions of the optical and philosopher problems and the numbers that we got are 0/52 and 12/22 respectively.

	sgP	SATP	SATP(w)	SATP(b)	SATP(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

Table 1: Results on domains coming from IPCs. x/y stands for x time outs or segmentation faults, y soft goals satisfied.

of SATPLAN/(w)/(b)/(s) if compared to those of SGPLAN. Moreover, it also shows that the performances of SATPLAN are not affected when adding preferences, i.e., when imposing an order on the variables to be used for splitting in the SAT solver. This result is somehow surprising given that limiting the splitting of the SAT solver can cause an exponential degradation of its performances (Järvisalo, Junttila, & Niemelä 2005). Such exponential degradation never shows up for SATPLAN(w)/(b), and SATPLAN(s) does not solve only one instance.

The good performances of SATPLAN(w)/(b)/(s) can be explained by the fact that in all the problems considered there are at most 30 soft goals. This means that OPT-DLL branching heuristic is forced for at most the initial 30 branches: while it is known in SAT that the first branches are very important, they are just a few. Further, for the quantitative case, the burden introduced by the Boolean encoding of the objective function is negligible.

In order to evaluate the effectiveness of SATPLAN(w)/(b)/(s) compared to SATPLAN when the number of preferences is high (i.e., when the heuristic is highly constrained) we considered the problem of finding a "minimal" plan. More precisely, if Π_n is the given planning problem with makespan n, and S is the set of action variables in Π_n, we consider

1. the qualitative preference $\langle\{\neg p : p \in S\}, \prec\rangle$ where $\neg p \prec \neg p'$ if p precedes p' according to the lexicographic ordering, and
2. the quantitative preference $\langle\{\neg p : p \in S\}, c\rangle$ in which c is the constant function 1.

These preferences encode the fact that we prefer plans with as few actions as possible. The qualitative preference has also been set in order to further constraint the heuristic up to the point to make it static on the action variables. In this setting, we are expecting a significant degradation in the CPU performances of SATPLAN(w)/(b)/(s) wrt SATPLAN's ones. The results are in the right plot in Figure 3, where,

- on the x-axis there are the problems, sorted according to the ratio between the number of preferences (i.e., action variables) and the total number of variables in the instance for which a plan is found, and
- on the y-axis there is the ratio between the performances of SATPLAN(w)/(b)/(s) and SATPLAN's ones, in logarithmic scale.

Differently from what we expected, we see that SATPLAN(w)/(s) is as efficient as SATPLAN for a significant initial portion of the plot, though on a few instances SATPLAN(w) does not terminate within the timeout. SATPLAN(s) is also in many cases more efficient than SATPLAN, and this reminds the observation in (Giunchiglia *et al.* 1998) that preferential splitting on action variables can lead to significant speed-ups. Thus, SATPLAN(w)/(s) can be very effective (or, at least, as effective as SATPLAN) even when the number of preferences is very high (e.g., several thousands). SATPLAN(b) bad performances are easily explained by the fact that the quadratic BB-encoding leads to very big SAT instances: considering for instance the relatively small "optical1" problem, the first satisfiable SAT instance has 2228 and 34462 variables and clauses respectively, while the same numbers when considering the W-encodings and the BB-encodings are 9500 and 56091, 13768 and 1157016 respectively; "optical1" is solved in 1.28s/2.17s/57.14s/1.6s by SATPLAN/(w)/(b)/(s). Considering the problems with ratio ≤ 0.45 on the x-axis, they include all the airport, promela philosophers and optical and some of the psr, storage and tpp problems. Of course, SATPLAN(w)/(b)/(s) often return plans with fewer actions than SATPLAN.

Related works

Considering the literature on preferences in planning, three recent papers on planning with preferences are (Bienvenu *et al.* 2006; Brafman and Chernyavsky 2005; Büttner and Rintanen 2005), but see also the proceedings of the ICAPS'06 workshop on preferences and soft constraints. In the first paper, the authors define a simple language for expressing temporally extended preferences and implement a forward search planner, called PPLAN integrating them. For each $n \geq 0$ PPLAN is guaranteed to be n-optimal, where this intuitively means that there is no better plan of sequential length $\leq n$. The basic language for expressing preferences (called "basic desire formulas (BDFs)") is based on Linear Temporal Logic (LTL). BDFs are then ranked according to a total order to form "atomic preference formulas" which can then combined to form "general and aggregated preference formulas". It is well known how to compile LTL with a bounded makespan into propositional logic and thus in the language of Π_n. It seems thus plausible that BDFs can be expressed as preferences in our setting, and we believe that the same holds for the preference formulas. In (Brafman and Chernyavsky 2005), the authors show how to extend GPCSP (Do and Kambhampati 2001) in order to plan with pref-

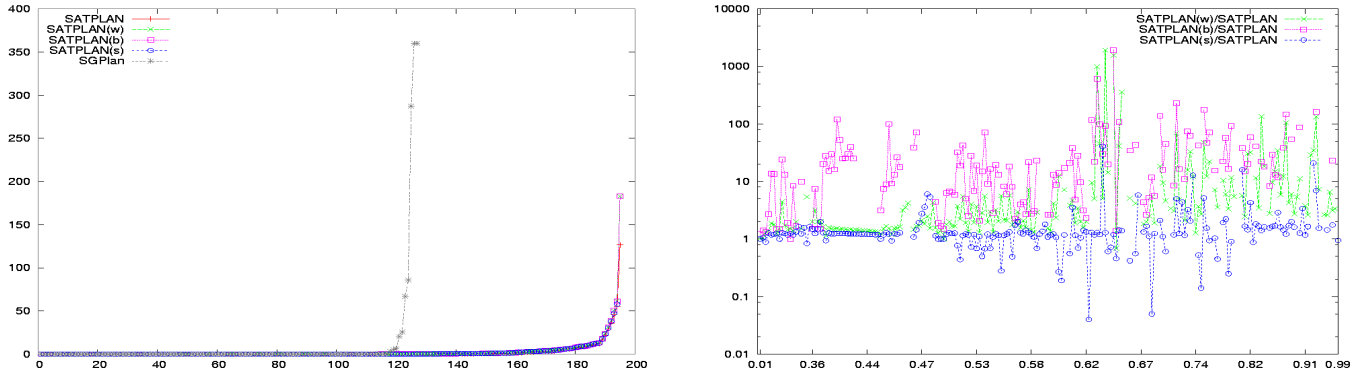

Figure 3: Left: Performances of SGPLAN, SATPLAN/(w)/(b)/(s). Right: Performances of SATPLAN(w)/(b)/(s) wrt SATPLAN as a function of the ration between the number of preferences and the number of variables.

erences expressed as a TCP-net (Boutilier *et al.* 2004). In the Boolean case, TCP-net can be expressed as Boolean formulas. Though these two works are 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. However, these approaches and ours 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 posting a constraint imposing a smaller value for the objective function: 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 the same we deal with in Section : finding an optimal "minimal" plan for Π_n using a quantitative approach. The fundamental difference between our approach and theirs is that we look for an optimal solution by imposing an ordering on the heuristic of the DLL solver, while they iteratively call the SAT solver as a black box. The disadvantage of their approach is that, e.g., the nogoods computed during a call are not re-used by the following calls for the same n. Our approach can also deal with qualitative preferences.

References

O. Bailleux and Y. Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In *Proc. CP*, pages 108–122, 2003.

M. Bienvenu, C. Fritz, and S. McIlraith. Planning with qualitative temporal preferences. In *Proc. KR*, 2006.

C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. Preference-based constrained optimization with CP-nets. *Computational Intelligence*, 20(2):137–157, 2004.

R. Brafman and Y. Chernyavsky. Planning with goal preferences and constraints. In *Proc. ICAPS*, pages 182–191, 2005.

M. Büttner and J. Rintanen. Satisfiability planning with constraints on the number of actions. In *Proc. ICAPS*, pages 292–299, 2005.

M.B. Do and S. Kambhampati. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artif. Intell.*, 132(2), 2001.

N. Eén and N. Sörensson. An extensible SAT-solver. In *Proc. SAT*, pages 502–518, 2003.

A. Gerevini and D. Long. Plan constraints and preferences in PDDL3. In *Proc. ICAPS-2006 Workshop on Preferences and Soft Constraints in Planning*, pp. 46-53, 2006.

E. Giunchiglia and M. Maratea. Solving optimization problems with DLL. In *Proc. ECAI*, 2006.

E. Giunchiglia, A. Massarotto, and R. Sebastiani. Act, and the rest will follow: Exploiting determinism in planning as satisfiability. In *Proc. AAAI*, 1998.

J. Hoffmann and S. Edelkamp. The deterministic part of IPC-4: An overview. *JAIR*, 2005.

C. Hsu, B. Wah, R. Huang, and Y. Chen Constraint Partitioning for Solving Planning Problems with Trajectory Constraints and Goal Preferences. In *Proc. IJCAI*, 2007.

Henry Kautz and Bart Selman. Planning as satisfiability. In *Proc. ECAI*, 1992.

Henry A. Kautz and Bart Selman. Unifying SAT-based and graph-based planning. In *Proc. IJCAI*, 1999.

Järvisalo, M.; Junttila, T.; 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.

Joost P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Inf. Process. Lett.*, 68(2):63–69, 1998.