Introducing Preferences in Planning as Satisfiability[⋆]

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 techniques 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. Given a planning problem Π and a makespan n, the approach based on satisfiability (a.k.a., SAT-based) simply works by (i) constructing a SAT formula Π_n; and (ii) checking Π_n for satisfiability: if there is a model for Π_n then we have found a plan, otherwise n is increased. The approach guarantees that the makespan is optimal, i.e., minimum.

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. This allows, e.g., to take into consideration "plan quality" issues other than makespan, like number of actions and "soft" goals. The basic idea is to explore the search space of possible plans in accordance with the preferences expressed as a partial order, i.e., to force the splitting of the SAT solver in order to follow the given partial order on preferences. We first prove that, at fixed makespan, our approach returns an "optimal" plan, if any. We then show that the resulting system, SATPLAN(P) (i) returns optimal plans which are often of considerable better quality, i.e., with fewer actions or with a better plan metric on soft goals, than SATPLAN; and (ii) is overall competitive, in terms of plan quality, with SGPLAN, the winning system in the "SimplePreferences" category, which includes "soft" goals, at the IPC-5.

About the performance, we show that SATPLAN(P) is often as effective as SATPLAN when solving the same problems without preferences in the cases where the number of preferences in not very high: in other words, for these problems, introducing simple preferences in SATPLAN does not affect its performances. Indeed, if we consider planning problems where the number of preferences is very high, compared to the total number of variables in the problem, e.g., the issue of determining minimal length plans (corresponding to problems with thousands of preferences), the performance of SATPLAN(P) is comparable to those of SATPLAN in many cases, but can be significantly worse.

Our analysis is conducted considering both qualitative and quantitative preferences, different reductions from quantitative to qualitative ones, most of the propositional STRIPS planning domains from the first 5 IPCs, and some domains from the "SimplePreferences" category of the IPC-5.

[⋆] This paper extends previous work presented in [31, 32]. The authors would like to thank Chic-Wei Hsu for his help with SGPLAN, Jörg Hoffmann and Alessandro Saetti for providing their ADL2STRIPS tools; Alessandro Saetti also helped with IPC-5 benchmarks.

1 Introduction

Planning as Satisfiability [39] is one of the most well-known and effective techniques for classical planning: SATPLAN [40, 41] 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)[1] [35] and a co-winner in the IPC-5[2] [27] (together with another planner based on SAT, MAXPLAN [57]). Given a planning problem Π, the basic idea of Planning as Satisfiability (a.k.a., SAT-based) is to convert the problem of determining the existence of a plan for Π with a fixed makespan n into a SAT formula Π_n such that there is a one-to-one correspondence between the plans of Π with makespan n and the interpretations satisfying Π_n. The SAT-based approach thus works by (i) constructing a SAT formula Π_n; and (ii) checking Π_n for satisfiability: if there is a model for Π_n then we have found a plan, otherwise n is increased. The approach guarantees that the makespan is optimal, i.e., minimum. Of course, for SATPLAN effectiveness, it is crucial the availability of very effective SAT solvers, like MINISAT [21]. MINISAT is based on the Davis-Logemann-Loveland procedure (DLL) [14] like most of the state-of-the-art SAT checkers, and won the SAT competition in 2005[3] (together the the the SAT/CNF minimizer SATELITE), the SAT race 2006 and 2008[4], on industrial benchmarks.

In this paper we extend the Planning as Satisfiability approach in order to handle both qualitative (in which relative importance among preferences is described by, e.g., a partial order) and quantitative (using a reward-based approach) preferences, and SATPLAN in order to solve problems with simple preferences. Qualitative and quantitative approaches both received attention in planning, each having its pros and cons, as commented in, e.g., Sec. 2.3 of [27]. By "simple" we mean that the structure of the partial order is restricted to particular, e.g., linear, cases: despite this limitation, which nonetheless only holds at implementation level, we will see that simple preferences allow to express and solve important optimization problems.

The basic idea of our algorithm, extending the results first presented in the CSP setting in [6] (in the context of CP-net [5]) and then in [30] in the SAT setting, is to explore the search space of possible plans in accordance with the preferences expressed as a partial order, 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 be minimized/maximized. This allows, e.g., to take into consideration "plan quality" issues other than makespan, like number of actions and plan metrics defined on "soft" goals. We implemented this approach by modifying SATPLAN, and we call SATPLAN(P) the resulting system.

We first prove that, at fixed makespan, our approach returns an "optimal" plan, i.e., such that there is no a "better" plan under the expressed preferences, if any. Then, we show that SATPLAN(P)

[1] `http://www.tzi.de/~edelkamp/ipc-4/`.
[2] `http://zeus.ing.unibs.it/ipc-5/`.
[3] `http://www.satcompetition.org/2005/`.
[4] See `http://fmv.jku.at/sat-race-2006/` and `http://baldur.iti.uka.de/sat-race-2008/`, respectively.

1. returns optimal plans which are often of considerable better quality, i.e., with fewer actions or with a better plan metric defined on soft goals, than SATPLAN; and
2. is overall competitive, in terms of plan quality, with SGPLAN when considering benchmarks from IPC-5 related to optimization on the goals: even if SATPLAN(P) and SGPLAN solve a different problem, i.e., bounded vs. unbounded, they are still both targeted for optimization. This is a remarkable result given that SGPLAN is the clear winner in the "SimplePreferences" category, which includes "soft" goals, at the IPC-5.

About the performance, we show that SATPLAN(P) is often as effective as SATPLAN, in terms of CPU time, on the problems we consider where the number of preferences involved is not very high, i.e., that on these problems introducing simple preferences in SATPLAN does not affect its performances. Indeed, if we consider planning problems where the number of preferences is very high compared to the total number of variables in the problem, e.g., the issue of determining minimal length plans (corresponding to problems with thousands of preferences), the performance of SATPLAN(P) is comparable to those of SATPLAN in many cases, but can be significantly worse. Even if for the problems with "few" preferences this is not that surprising, by correlating the good behavior of SATPLAN(P) to the relative low number (in the order of tens) of preferences, this is instead quite surprising when we consider problems with "many" preferences, e.g., the problem of determining minimal length plans, where any action variable corresponds to a preference: in such case it is common to have problems with several thousands of preferences, and it is well-known that limiting the splitting of the SAT solver can cause an exponential degradation of its performances [38]. When the number of preferences is not high, only few (initial) branches are imposed. However, SATPLAN(P) can be much slower than SATPLAN when considering problems with a high ratio of number of preferences (e.g., action variables) to the total number of variables.

Our analysis is conducted considering both qualitative and quantitative preferences, different encodings of the objective function in the case of quantitative preferences, most of the propositional planning domains from the first 5 IPCs, and some domains from the "SimplePreferences" category of the IPC-5. About soft goals, we have considered different types of benchmarks. First, we have adapted the original STRIPS [23] planning instances in order to consider all goals as being "soft". 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. Then, given that the case in which not all goals can be satisfied (often called over-subscription planning [53, 48]) is practically very important, we have considered some domains from the "SimplePreferences" category of the IPC-5: given that on such domains some ADL constructs are used in order to express quantitative preferences on soft goals, we rely on a compilation into a STRIPS problem. The compilation is similar to the ones used in [3, 24], and is detailed in the experimental part. For such benchmarks, we have considered both the weighted and the unweighted (making all weights associated to goals violation to be uniform) related versions.

Summing up, in this paper

- 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 prove that, at fixed makespan, our approach returns an "optimal" plan, if any.
- We then show that SATPLAN(P) often returns plan of considerably better quality than SATPLAN. Moreover, SATPLAN(P) is overall competitive with the state-of-the-art system SGPLAN when considering plan quality issues related to soft goals, but for conflicting goals with non-uniform weights.
- We show that the performances of SATPLAN(P) are comparable to those of SATPLAN (when given the same problems without preferences) even when there is a high number of preferences, and that high deviations can only appear when the ratio between the number of preferences to the total number of variables is relatively high.

Moreover, as a side effect of our analysis, in the case of quantitative preferences, we show that the encoding based on [56] leads to better performances than the one based on [2], at least in our setting. Remarkably, this fact does not agree with the good results of [2] presented in [2, 10].

The paper is structured as follows. We first introduce some basic notation and terminology (Section 2), then we introduce preferences and the concept of "optimal" plan (Section 3) and show how it is possible to incorporate them in planning as satisfiability (Section 4). The implementation and experimental analysis is presented in Section 5. We end the paper with the related work in Section 6 and some final remarks in Section 7.

2 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, 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 *initial state*;
- 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.

Of course, in our setting, we assume that the description of actions' effects is deterministic: 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).

As an example, consider 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 $\{\mathit{Car}_0, \mathit{Bus}_0, \mathit{Bike}_0\}$.

3 Preferences and Optimal Plans

Let Π_n be a planning problem Π with makespan n.

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 preferences $\neg \mathit{Bike}_0$, $\neg \mathit{Bus}_0$ and $\neg \mathit{Car}_0$ simultaneously.

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 $\pi_2 \not\models p_1$, i.e., π_1 satisfies a preference (p_1) which is preferred to p_2, and 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 $\langle P, \prec\rangle$). Other preference formalisms include [15, 6, 4, 50], and are discussed in Sec. 6.

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 in the IPC-5. 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[5]

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

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.

4 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

[5] We have seen that the types of functions defined by (3), i.e., additive, were the only ones used in IPC-5. Nonetheless, 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 $\sum_{p \in P' : \pi \models p} c'(p)$.

1. encoding the set P of preferences 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) such that $\mu' \models$ *cnf*(φ) it is true also that $\mu \models \varphi$, where μ is the interpretation μ' but restricted to the signature of φ; and
  - for any interpretation $\mu \models \varphi$ there exists an interpretation μ', $\mu' \supseteq \mu$, such that $\mu' \models$ *cnf*(φ).

  There are well-known methods for computing *cnf*(φ) in linear time by introducing additional variables, e.g., [51, 46, 37];
- 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 in P' are assigned, and an arbitrary literal in φ is selected, or $l \in P'$ and all the other variables $v(p) \in P'$ with $v(p) \prec l$ are assigned.

```
function QL-PLAN($\Pi_n$,$P$,$\prec$)
 1 return OPT-DLL(cnf($\Pi_n \wedge \wedge_{p \in P}(v(p) \equiv p)$),$\emptyset$,$v(P)$,$v(\prec)$)

function OPT-DLL($\varphi$,$S$,$P'$,$\prec'$)
 2 if ($\emptyset \in \varphi$) return FALSE;
 3 if ($\varphi = \emptyset$) return $S$;
 4 if ($\{l\} \in \varphi_l$) return OPT-DLL($\varphi_l, S \cup \{l\}, P', \prec'$);
 5 $l$ := ChooseLiteral($\varphi, S, P', \prec'$);
 6 $V$ := OPT-DLL($\varphi_l, S \cup \{l\}, P', \prec'$);
 7 if ($V \neq$ FALSE) return $V$;
 8 return OPT-DLL($\varphi_{\bar{l}}, S \cup \{\bar{l}\}, P', \prec'$).
```

Fig. 1. The algorithm of QL-PLAN.

As it can be seen from the figure, OPT-DLL [30] is the standard DLL except for the modification in the heuristic, i.e., *ChooseLiteral*, which initially selects literals according to the partial order $\prec'$. Thus, OPT-DLL performs the following steps; in line

2 : if there is a contradictory clause in φ, i.e., a clause C such that $\exists l \in C : \bar{l} \in S$, then backtracking occurs;
3 : if φ is satisfiable, then S is returned as optimal solution;
4 : if there is a unit clause in φ_l, i.e., a clause with just one literal, then the literal is assigned;
5 : if the base cases at line 2 and 3 do not hold, and there is no unit clause, then a new literal l is chosen according to the behavior of *ChooseLiteral* explained above;
6 : l is assigned and, in line 7, the optimal solution is returned if such branch has not encountered an inconsistency;
8 : the branch extending $\bar{l}$ is followed (if the branch extending l at line 6 returned FALSE).

As an example. if we have two preferences p_1 and p_2 with $p_1 \prec p_2$, then the algorithm works as follows: it

1. looks for assignments extending $\{v(p_1), v(p_2)\}$: if the search fails (meaning that no plan is found extending this assignment) then OPT-DLL backtracks, and
2. looks for assignments extending $\{v(p_1), \neg v(p_2)\}$: if the search fails, then OPT-DLL backtracks, and
3. looks for assignments extending $\{\neg v(p_1), v(p_2)\}$: if the search fails, then OPT-DLL backtracks, and
4. looks for assignments extending $\{\neg v(p_1), \neg v(p_2)\}$: if the search fails, then OPT-DLL returns FALSE. This corresponds to the case when Π_n is unsatisfiable, and thus no plan exists for Π at makespan n.

Of course, in the above, we have assumed that $v(p_2)$ has not been assigned as unit at line 4 of Figure 1.

Consider now the problem (2) with $p_1 = (\neg \mathit{Bike}_0 \wedge \neg \mathit{Bus}_0 \wedge \neg \mathit{Car}_0)$ and $p_2 = (\neg \mathit{Bike}_0 \wedge \neg \mathit{Bus}_0)$ with $p_1 \prec p_2$. QL-PLAN 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, as stated by the following Theorem.

Theorem 1. *Let Π_n be a planning problem Π with makespan n, and let $\langle P, \prec \rangle$ be a qualitative preference for Π_n.* 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.

Proof. From [16], we know that OPT-DLL(φ,S,P',$\prec'$) returns

1. FALSE if φ is unsatisfiable, and
2. an "optimal solution" for φ wrt $\langle P', \prec' \rangle$, otherwise.

Given these, in order for the actual Theorem to hold, we have to show that

- for each formula $p \in P$ there must be a corresponding variable $v(p) \in P'$, and viceversa;

- for each p_1 and p_2, $\{p_1, p_2\} \subseteq P$ such that $p_1 \prec p_2$, it must hold that $v(p_1) \prec' v(p_2)$, and viceversa; and
- for each assignment S in the signature of φ such that $S \models \varphi$, it must also hold that $S' \models \Pi_n$, where S' corresponds to S but reduced to the signature of Π_n, and for each assignment S' in the signature of Π_n such that $S' \models \Pi_n$, there exists an assignment S, $S \supseteq S'$, such that $S \models \varphi$.

The first two points hold by definition of $v(P)$ and $v(\prec)$. The third point holds from the assumptions on *cnf*.

$\square$

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. The basic idea is to encode the value of the objective function (3) as a sequence of bits $b_{k-1}, \dots, b_0$ and then consider the qualitative preference $\langle \{b_{k-1}, \dots, b_0\}, \{b_{k-1} \prec b_{k-2}, \dots, b_1 \prec b_0\}\rangle$.

The details of such encoding depend if we are using a *binary* or an *unary* encoding. For a binary encoding, let *Bool*(P, c) be a Boolean formula such that

1. if $k = \lceil log_2((\sum_{p\in P} c(p))+1)\rceil$, *Bool*$(P, c)$ contains k new variables $b_{k-1}, \dots, b_0$; 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) $\sum_{p\in P:\pi\models p} c(p) = \sum_{i=0}^{k-1} \mu(b_i) \times 2^k$, 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* $b_{k-1}, \dots, b_0$.

An example of the binary encoding is Warners [56], denoted with “W-encoding” in the following. The encoding by Bailleux and Boufkhad [2], denoted with “BB-encoding” in the following is, instead, an example for the unary case. The BB-encoding can only handle cardinality constraints, i.e., *Bool*$(P, 1)$, with

(i) $k = |P|$; and
(ii) $|\{p \in P : \pi \models p\}| = \sum_{i=0}^{k-1} \mu(b_i)$.

Both encodings can be realized in polynomial time, but 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 positive results (see, e.g., [2, 10]).

In the quantitative case, the resulting procedure, QT-PLAN, is represented in Figure 2 in which *Bool*(P, c) is a Boolean encoding of $\langle P, c\rangle$ with output $b_{k-1}, \dots, b_0$, $b(c) = \{b_{k-1}, \dots, b_0\}$ and $p(c)$ is the partial order $b_{k-1} \prec b_{k-2}, \dots, b_1 \prec b_0$.

```
function QT-PLAN($\mathit{\Pi}_n$,$P$,$c$)
1  return OPT-DLL($cnf(\mathit{\Pi}_n \wedge Bool(P, c))$,$\emptyset$,$b(c)$,$p(c)$)
```

Fig. 2. The algorithm of QT-PLAN.

Assuming we have the quantitative preference $\langle P, c\rangle$ with $P = \{p_1, p_2\}$, $c(p_1) = 2$ and $c(p_2) = 1$, $Bool(P, c)$ for the binary case is simply $(b_1 \equiv p_1) \wedge (b_0 \equiv p_0)$ and the assignments generated by our approach are in the following order: QT-PLAN

1. looks for assignments extending $\{b_1, b_0\}$: this corresponds to search for plans with cost 3; if no such plan is found, then OPT-DLL backtracks, and
2. looks for assignments extending $\{b_1, \neg b_0\}$: this corresponds to search for plans with cost 2; if no such plan is found, then OPT-DLL backtracks, and
3. looks for assignments extending $\{\neg b_1, b_0\}$: this corresponds to search for plans with cost 1; if no such plan is found, then OPT-DLL backtracks, and
4. looks for assignments extending $\{\neg b_1, \neg b_0\}$: if no plan is found, then OPT-DLL returns FALSE. Again, this corresponds to the case when $\mathit{\Pi}_n$ is unsatisfiable, and thus no plan exists for $\mathit{\Pi}$ at makespan n.

Consider again problem (2), and the two preferences

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

with $c(p_1) = 2$ and $c(p_2) = 1$: we have seen that two bits b_1 and b_0 are sufficient as output of $Bool(\{p_1, p_2\}, c)$. On this example, QT-PLAN returns the plan corresponding to $\{Car_0\}$ determined while exploring the branch extending $\neg b_1, b_0$. This plan is optimal, as stated by the following Theorem.

Theorem 2. *Let $\mathit{\Pi}_n$ be a planning problem $\mathit{\Pi}$ with makespan n, and let $\langle P, c\rangle$ be a quantitative preference for $\mathit{\Pi}_n$.* QT-PLAN*($\mathit{\Pi}_n$,P,c) returns*

1. FALSE *if $\mathit{\Pi}_n$ has no plans, and*
2. *an optimal plan for $\mathit{\Pi}_n$ wrt $\langle P, c\rangle$, otherwise.*

Proof. The proof is similar to the one of Theorem 1. Consider $\varphi' = cnf(\mathit{\Pi}_n \wedge Bool(P, c))$.

Again from [16], we know that OPT-DLL(φ',$\emptyset$,$b(c)$,$p(c)$) returns

1. FALSE if φ' is unsatisfiable, and
2. an "optimal solution" for φ' wrt $\langle b(c), p(c)\rangle$, otherwise.

Given these, in order for the actual Theorem to hold, we have to show that

- for each pair $\langle P, c\rangle$ of quantitative preferences, there must be a corresponding qualitative pair $\langle b(c), p(c)\rangle$, and viceversa;

- for each assignment S in the signature of φ' such that $S \models \varphi'$, it must also hold that $S' \models \Pi_n$, where S' corresponds to S but reduced to the signature of Π_n, and for each assignment S' in the signature of Π_n such that $S' \models \Pi_n$, there exists an assignment S, $S \supseteq S'$, such that $S \models \varphi'$.

The first point holds by definitions from items 1. and 2.(b) in the definition of *Bool*. The second point holds from the assumptions on *cnf*.

$\Box$

5 Implementation and Experimental Analysis

We implemented OPT-DLL in MINISAT,[6] which is also one of the solvers SATPLAN can use, and that we set as default for SATPLAN: thus, QL-PLAN is implemented at each makespan of the SATPLAN's approach, till the optimal makespan. At implementation level, we performed some optimizations of the algorithm presented in Fig. 1: (i) we do not introduce an auxiliary variable when the preference is already an atom itself (e.g., in the case of actions), in this case (sometimes significantly) reducing the number of variables OPT-DLL has to deal with; and (ii) we substituted $\equiv$ with $\vee$ again in the OPT-DLL call, given it guarantees correctness and completeness as well.

In the case of quantitative preferences, we considered the W- and BB-encoding. 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 experiments, we consider SATPLAN(w)/(b)/(s), SGPLAN [36], and plain SATPLAN. SGPLAN has been the clear winner in the "SimplePreferences" category, which includes metrics defined on soft goals, in the IPC-5 and is thus the reference system for the problems that we consider, even if we have to take into account that it solves a different problem, i.e., unbounded and sequential. SATPLAN(s) is a variation which can not guaranty optimality in the quantitative case: its results are presented, or most of the time commented in the text, to show that often are on the same quality of SATPLAN(w)/(b), or close, and often obtained in shorter time (at least in the domains we consider). It often could be thus used for computing approximated results efficiently. SATPLAN has been included in order to evaluate the differences in the performance 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: in the cases where 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, which is inherited from the SAT-based approach, 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. Note that also SGPLAN rejects these instances. Moreover, we also discarded

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

instances where SATPLAN exits with a syntax error, i.e., some tpp and trucks instances, and one promela optical instance.

All the tests have been run on a Linux box equipped with a Pentium IV 3.2GHz processor and 1GB of RAM. Timeout has been set to 900s.

We will first consider the problems with soft goals, which have received a lot of attention since the IPC-5 in 2006: in the next three subsections we present analysis for three "types" of benchmarks of increasing difficulty, ranging from a modification of classical STRIPS planning problems, to benchmarks with conflicting goals with uniform weights to the ones used in the "SimplePreferences" category of the IPC-5. The last subsection is then devoted to the analysis on planning problems where the goal is the minimization of the plan length.

5.1 Non-conflicting soft goals.

For this analysis, given that SATPLAN can only directly handle STRIPS domains, we considered the pipesworld tankage and notankage, satellite, airport, promela philosophers and optical, psr small, depots, driverLog, zenoTravel, freeCell, logistics, blocksworld, mprime and mistery domains from the first 4 IPCs, and openstacks, pathway, storage, tpp and trucks from IPC-5, in their STRIPS formulation.[7] In these problems the goal corresponds to a set G of literals and without soft goals. We modified these problems in order to interpret all the literals in G as "soft goals". More precisely, we considered both the qualitative preference $\langle G, \emptyset \rangle$ and the quantitative preference $\langle G, c \rangle$ in which c is the constant function 1. These preferences encode the fact that the goals in G are now "soft", in the sense that it is desirable but not necessary to achieve them. For SGPLAN, we also straightforwardly translated the planning problems in the language of PDDL3 [28].

In Table 1 we present results for solving problems with non-conflicting soft goals. Each row and column corresponds to a domain of instances and a system, respectively; the first number reports the total number of timeouts, memory out or segmentation faults; the second number is the sum of the satisfied soft goals in all solved instances, and the third number is the mean CPU time of solved instances in the domain. Note that the first two parameters are the ones used to evaluate planning systems in the "SimplePreferences" category of the last IPC-5. Looking at the table, we see that SATPLAN(w)/(b)(s) behave comparatively well with SGPLAN: there are 7 domains in which they manage to solve more problems and satisfy more soft goals than SGPLAN, while for SGPLAN this happens in 5 domains. Moreover, they manage to satisfy, in sum, more soft goals than SGPLAN (and SATPLAN). SGPLAN, on these instances adapted from propositional STRIPS instances, has several time outs/segmentation faults on some domains, due to the high number of grounded operators in the problems.[8] The authors of SGPLAN also suggested that the ADL version of the problems may lead to less segmentation faults. We tried the ADL versions for the domains where it is available: on the optical and philosopher domains the numbers that we got are 0/52/15.25 and 0/55/88.27,

[7] Note that the IPC-6 does not have basic STRIPS problems.
[8] Personal communication with Chic-Wei Hsu.

	SGPLAN	SATPLAN	SATPLAN(w)	SATPLAN(b)	SATPLAN(s)
pipesworld-notankage	0/10/0.14	7/43/6.74	7/102/10.51	8/99/8.26	7/96/10
pipesworld-tankage	4/45/0.81	21/29/25.24	21/108/61.28	24/92/54.48	22/100/38.2
satellite	5/67/1.01	9/27/3.29	9/75/10.03	10/74/8.55	9/75/7.07
airport	6/174/46.39	13/31/21.8	19/66/42.73	20/63/39.8	21/51/21.47
promela-philosophers	29/0/–	0/29/3.28	0/464/3.61	0/464/3.64	0/464/3.65
promela-optical	12/0/–	0/13/14.77	0/104/17.22	0/104/17.09	0/104/15.95
psr-small	11/152/0.48	0/48/0.61	0/232/0.48	0/232/0.7	0/232/0.69
depots	7/23/98.04	1/21/2.33	1/59/19.7	1/59/19.79	1/56/17.93
driverLog	0/229/88.55	0/20/0.61	1/97/25.6	1/97/25.69	2/82/0.49
zenoTravel	0/216/1.55	1/19/10.44	1/67/15.23	2/60/10.26	1/66/14.62
freeCell	6/53/25.24	13/7/27.58	13/14/42.32	15/14/41.84	14/13/0.49
logistics	0/243/0.02	0/28/0.03	0/123/0.7	0/123/0.7	0/122/0.25
blocks-world	0/269/78.5	0/35/0.2	0/78/1.01	0/78/1.01	0/73/0.35
mprime	0/4/6.85	2/10/12.16	2/12/17.54	3/11/9.56	3/11/9.43
mystery	1/4/3.63	2/11/10.52	2/12/13.35	2/12/11.59	2/12/11.87
openstacks	3/45/0.04	5/5/13.45	5/5/11.66	5/5/57.92	5/5/67.29
pathways	29/0/–	0/29/1.44	11/64/33.89	11/64/34.08	5/83/15.89
storage	0/84/0.58	0/26/0.23	0/30/0.29	0/30/0.31	0/30/0.28
tpp	15/14/0.01	0/19/2.59	0/82/9.05	0/82/9.07	0/82/8.15
trucks	16/0/–	0/16/1.61	0/16/1.88	0/16/1.92	0/16/1.9

Table 1. Results on STRIPS domains coming from IPCs. $x/y/z$ stands for x time outs or segmentation faults, y soft goals satisfied, and z seconds of solved instances (at mean).

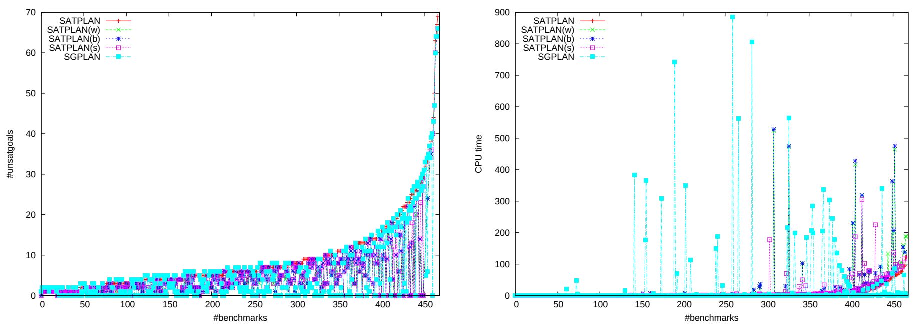

Fig. 3. Non-conflicting soft goals. Left: Number of unsatisfied soft goals for SATPLAN/(w)/(b)/(s) and SGPLAN. Right: CPU time for for SATPLAN/(s)/(w)/(b) and SGPLAN.

respectively; on the mprime and mystery domains we got 0/4/3.6 and 0/4/3.14, respectively; on the pathways domain we got 0/0/0.02, while on the airport and openstacks domains we got 50/0/$-$ and 30/0/$-$, respectively. Qualitatively, the picture thus does not change significantly. About the mean CPU time for solving instances in the domains, we can note that it is the case that the good results of SATPLAN(w)/(b)/(s) wrt SGPLAN can come with a price: this is a well-known fact given that there can be a big difference in performance between optimal planning and non-optimal planning; with respect to SATPLAN, we can note that often this is often not the case, thus adding simple preferences seems not to hit performances in most cases.

Table 1 gives an overview and helps to understand that our solution is overall competitive with the state-of-the-art (even considering that SGPLAN and SATPLAN are targeted for solving different problems as we saw before), but it loses a punctual analysis on each benchmark, and gives only an indication about the CPU times. In the remaining part of this subsection, we focus on SATPLAN and SATPLAN-based systems: the goal is to evaluated the impact that our ideas have on SATPLAN performances (in terms of both CPU time and quality of the plan returned). The performances of SGPLAN are included as a reference. Figure 3 shows the performances of SATPLAN/(w)/(b)/(s) and SGPLAN on the problems considered, ordered according to SATPLAN's performances. Figure 3 (Left) shows the number of soft goals each planner does not satisfy, while Figure 3 (Right) presents CPU times, again ordered according to SATPLAN's results. This way of presenting the data has the feature that is possible to compare the results about the same instance, i.e., the results for all the systems at a given x-coordinate refer to the same instance. It is to be noted that (i) Figure 3 contains only the instances solved by SATPLAN within the time limit: there are around 50 instances not showed which are solved by SGPLAN and not by SATPLAN/(w)/(b) (the majority of such instances are from the freeCell, pipesworld-tankage and airport domains, cf. Table 1); and (ii) for SGPLAN, when available, we have chosen the formulation, between STRIPS and ADL, which leads to the best results.

As expected, from Figure 3 (Left), we can see that SATPLAN does not satisfy many of the soft goals, while SATPLAN(w)/(b)/(s) manage to satisfy all or almost all of them in many cases, and thus, a significantly higher number than SATPLAN. Interestingly, the number of soft goals not satisfied by SATPLAN(w)/(b) and SATPLAN(s) are in most case equal, while in theory this is not necessary the case. From Figure 3 (Right) we can see that the performance of SATPLAN are overall not affected when adding preferences, i.e., when imposing an order on the variables to be used for splitting in the SAT solver. There are just very few exceptions to this behavior, in order of less than 10 (resp. 5) for SATPLAN(w)/(b) (resp. SATPLAN(s)), where adding preferences leads to a significant performance degradation.

The last result is somehow surprising given that limiting the splitting of the SAT solver can cause an exponential degradation of its performances [38]. We correlate the good performances of SATPLAN(w)/(b) with the fact that in all the problems considered there are at most 70 soft goals, and that the vast majority of the instances analyzed contain at most 30 soft goals. This means that OPT-DLL branching heuristic is forced for at most the initial few of tens 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.

5.2 Conflicting soft goals.

Besides the fact that in the instances considered in Subsec. 5.1 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 goals can be satisfied (often called over-subscription planning [53, 48]) is practically very important, we also evaluated some domains from the "SimplePreferences" category of the IPC-5, which include the possibility to express conflicting soft goals. Given that such domains are non-STRIPS, with some ADL constructs used, and that the impact of preferences violation on the plan metric is restricted to be linear (i.e., metric is the sum of weighted preference expressions) (as noted in [3, 27]) we have relied on the following compilation technique, similar to the one used in [3] to deal with PDDL3 benchmarks in YOCHAN^{PS}, and in [24] for dealing with conditional effects: 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[9]). In our analysis we have included the pathway, storage, openstacks and trucks domains from IPC-5, i.e., the domains where plan metric in defined only on the goals and there are instances with integer weights associated to goals violation, and that thus we can deal with our quantitative approach. These domains contain, in general, both hard and soft goals.

[9] Kindly provided by Jörg Hoffmann and Alessandro Saetti, respectively.

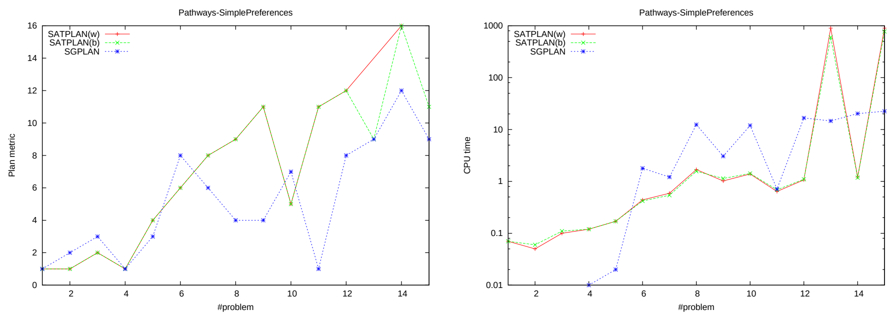

Fig. 4. Pathways domain, "SimplePreferences" category of IPC-5 with uniform weights. Left: Plan metric, i.e., number of unsatisfied soft goals, for SATPLAN(w), SATPLAN(b) and SGPLAN. Right: CPU time for SATPLAN(w), SATPLAN(b) and SGPLAN (in log scale).

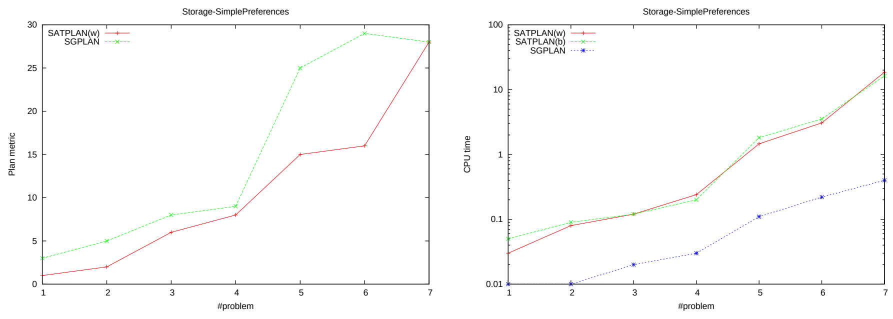

Fig. 5. Storage domain, "SimplePreferences" category of IPC-5 with uniform weights. Left: Plan metric, i.e., number of unsatisfied soft goals, for SATPLAN(w) (the same holds for SATPLAN(b)) and SGPLAN. Right: CPU time for SATPLAN(w), SATPLAN(b) and SGPLAN (in log scale).

Further, we have considered two cases: in the first we have changed all weights associated to goals violation to be the same, while the second fully exploits the weights in the original problems. We have decided to present both analysis because we wanted to evaluate (separately) what is the impact of considering conflicting goals instead of non-conflicting, and then the impact of adding weights to goal violation. SGPLAN has been run on the original IPC-5 problems (modified with uniform weights in the first case).

Results are here 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(w), SATPLAN(b) (given the metric is defined quantitatively) and, as a reference, SGPLAN. The next two paragraphs analyze separately the two cases.

Conflicting soft goals with uniform weights. For the Pathways domain in Figure 4 we can note (Right) that SATPLAN(w) and SATPLAN(b) perform often similarly 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), which are solved by SGPLAN in few tens of seconds, by SATPLAN(b) even if close the time limit, while SATPLAN(w) runs in timeout for both. This behavior is consistent with the fact that the BB-enconding is effective, and can be more effective that the W-encoding, when the number of preferences in not high. About the plan metric (Figure 4 Left), we can see that SGPLAN, overall, returns plans of slightly better quality, i.e., it can satisfy more soft goals, than SATPLAN(w)/(b).

For the Storage domain, instead, in Figure 5 (Right) we can note that all systems solve all the instances considered[10], with SGPLAN being around one order of magnitude faster than the other systems, which nonetheless solve each problem in less than 20s. The reason for the performance gap of SATPLAN(w)/(b) wrt SGPLAN can be immediately explained by looking at Figure 5 (Left): SATPLAN(w)/(b) return plans of much better quality than SGPLAN. The tradeoff between CPU performances and plan quality of SATPLAN(w)/(b) seems to be very effective, at least on this domain, further considering that SATPLAN(P) (i) (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 system on these domains in the "SimplePreferences" category at IPC-5.

For the trucks domain, the compilation technique used could compile 7 instances. Out of these 7 instances, our method could find an optimal solution on just one, in 6.93s and 4.91s by SATPLAN(w) and SATPLAN(b), respectively, with a very good plan metric of 1: on the same instance, SGPLAN returns very quickly a solution with the same plan metric. The other instances were already very hard to solve even for SATPLAN: problem #2 took 396s, and from problem #3 it could not solve the instances. Concerning the openstacks domain, we could compile only 1 instance, on which SATPLAN/(w)/(b) run in time out. Given that, we remind, the implementation of OPT-DLL is based on MINISAT, the performance of SATPLAN has been a major limitation in solving the instances of these two domains.

Conflicting soft goals with non-uniform weights. In this paragraph we comment on the analysis on problems with conflicting soft goals, where the weights associated to goal violation are fully taken into account as in the original IPC-5 benchmarks. As we said before, we restrict to integer weights for goal violation: only the W-encoding is used, given that BB-encoding can only deal with cardinality constraints.

For the Pathways domain, only problem #1 and #2 (as numbered in the competition) have only integer weights: these instances are solved very fast (< 0.2s) by SATPLAN(w) and SGPLAN; the plan metrics are 5 and 1 on problem #1 for SATPLAN(w) and SGPLAN, respectively, while is 2 on problem #2 for both systems.

For the Storage domain, until problem #4 the plan metrics of SATPLAN(w) and SGPLAN are comparable, and both systems solve the instances in less than 1s. Then, starting from problem #5, the size of the *Bool* formula starts to be of very big size, and difficult to manage. About SATPLAN(s), it behaves similarly to SATPLAN(w) and SG-

[10] We have considered all instances that the tools could compile, after also some interactions with the related tool's authors.

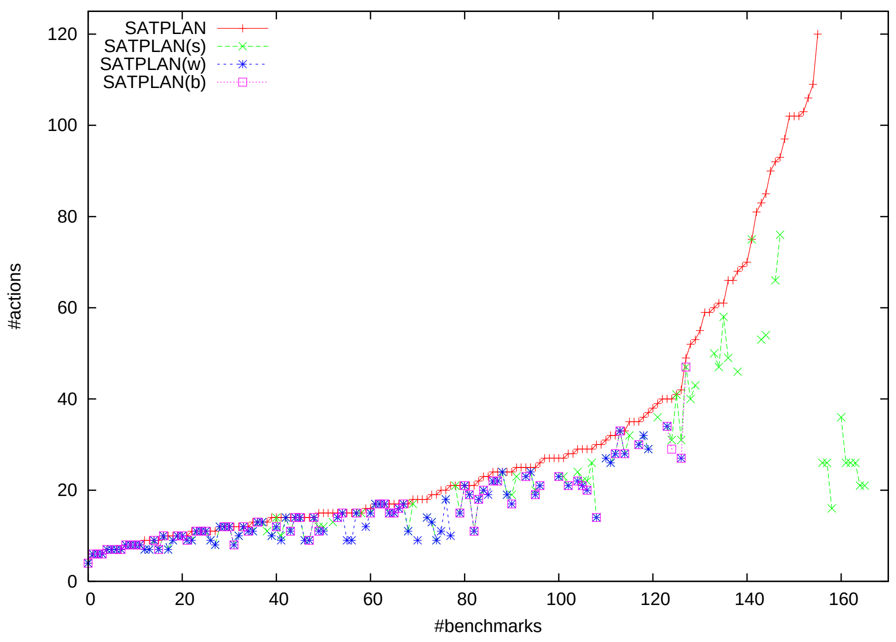

Fig. 6. Number of actions in the returned plan for SATPLAN and SATPLAN(w)/(s)/(b).

PLAN on the first instances, and also has a decent behavior on bigger instances, solving up to problem #7: nonetheless, on these instances the quality of the plans returned by SGPLAN is much better than the corresponding of SATPLAN(s), and the performances gap goes up to more than 1 order of magnitude. About plan quality, on bigger instances, to stop at the optimal makespan seems to be a severe limitation, e.g., problem #7 is solved by SATPLAN(s) at makespan 5 with a plan metric of 530: we run SATPLAN(s) at makespan 6 and the quality of the plan returned (in around 10s) was 397, thus sensibly better.

Regarding the Trucks domain, as we noted before, only problem #1 is solved by SATPLAN and SATPLAN(w). This instance is solved by SGPLAN very fast with a plan quality of 13, while it is solved by SATPLAN(w) in few seconds with a plan quality of 5.

5.3 Plans length.

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 G is now the set of action variables in Π_n, we consider

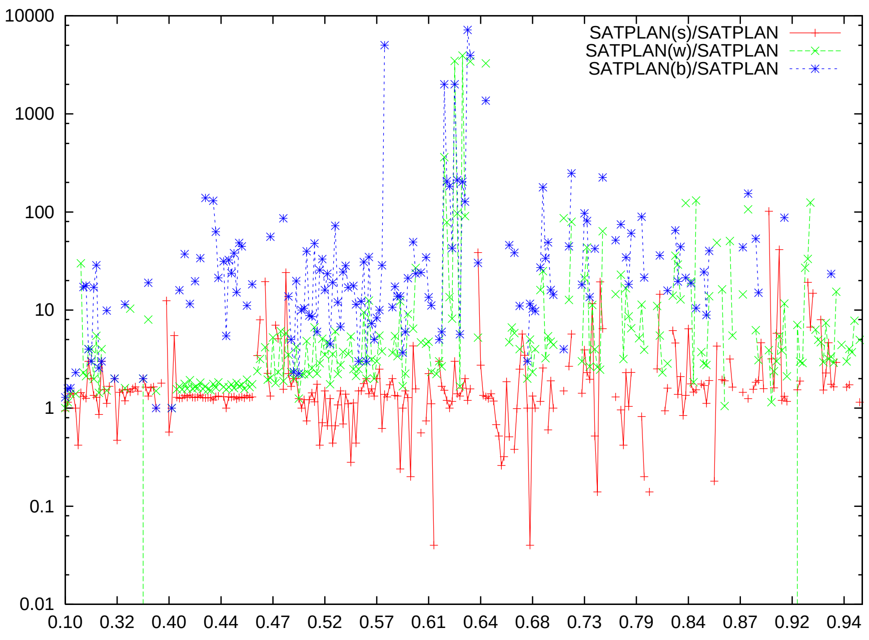

Fig. 7. Performances degradation for SATPLAN(s)/(w)/(b).

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

In the quantitative case, these preferences encode the fact that we prefer plans with as few actions as possible; in the qualitative case that we prefer plans with subset-minimal solutions, i.e., no subset of the plan's actions achieve the goals. 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 both (a) a reduction in the plan length for SATPLAN(w)/(b)/(s) vs. SATPLAN, and (b) a degradation in the CPU performances of SATPLAN(w)/(b)/(s) wrt SATPLAN. We anticipate that, while for point (a) results meet expectations, this is only partially the case for point (b).

Considering the "quality" of the plan returned in terms of number of actions, the results are in Figure 6, ordered according to SATPLAN's performances.

In the Figure, for sake of readability, we do not show the results for the airport, promela philosopher and optical, psr and trucks domains: for each instance in these domains SATPLAN/(w)/(b) (and also SATPLAN(s)) return plans with the same length: this is likely due to the structure of the problems, which do not permit optimizations in plan length. SATPLAN(w) in many cases returns plans of considerable better quality than SATPLAN. We can also further note that even if there are points missing in the Figure, which indicate instances that do not terminate within the time limit, they are just a few for SATPLAN(w), but this in not the case for SATPLAN(b), that in several cases runs out of time or memory when instances are big. Thus, its representative line is exactly the same of SATPLAN(w), but with a significant fraction of the points in the Figure missing.

About SATPLAN(s) we first note that, similarly to the case of soft goals, the quality of the plan returned by SATPLAN(s) is in most cases equal to that of SATPLAN(w), and it then could be used to effectively compute good approximation. Moreover, there are also instances, in the final part of the plot, solved uniquely by SATPLAN(s): if compared to SATPLAN, this reminds the observation in [33] that preferential splitting on action variables can lead to significant speed-ups.

Figure 7 takes into account the results for the problems of minimizing plan length by considering CPU time. In this Figure,

- on the x-axis there are the instances, 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(s)/(w)/(b)/ and SATPLAN, in logarithmic scale.

Differently from what we expected, we see that SATPLAN(w) is as efficient as SATPLAN for a significant initial portion of the plot, though on a few instances it does not terminate within the time limit. Thus, SATPLAN(w) can be very effective (or, at least, as effective as SATPLAN) even when the number of preferences is very high (e.g., several thousands). For ratio ≥ 0.45 SATPLAN(w) performances degrade, and several instances are not solved, or solved with a considerably higher CPU time in comparison to SATPLAN. SATPLAN(b) bad performances are easily explained by the fact that the quadratic BB-encoding leads to very big SAT instances: considering the relatively small "optical1" problem, the first satisfiable SAT instance has 2228 variables, of which 1050 correspond to preferences (action variables), and 34462 clauses, respectively; the same numbers when considering the W-encoding and the BB-encoding are 9348 and 56091, 13768 and 1157016, respectively: "optical1" is solved in 1.28s/2.17s/57.14s by SATPLAN/(w)/(b). Note also how SATPLAN(s) instead maintains its good behavior for almost all the spectrum of instances; it is also in several cases more efficient than SATPLAN.

Finally, in Table 2 we summarize the results presented in this paragraph, partly presented in Figure 7. In the table we also include instances that, for easy of presentation, we have not included in the figure. The table is structured as follows: the first column contains the percentage of actions variables over variables considered (as in the x-axis of Figure 7); the second column contains the number of instances in the percentage range specified in the first column, while the last three columns contain the results for SATPLAN(w)/(b)(s) over SATPLAN, further divided into three subcolumns, where each number is the instances that are solved in the respective percentage range and with a certain ratio range (as in the y-axis of Figure 7). Table 2 presents the analysis considering problems that contain at least a certain percentage of action variables, i.e., at least 40# of the total variables of the instance.

% ratio	#pb	r =SATPLAN(b)/SATPLAN			r =SATPLAN(w)/SATPLAN			r =SATPLAN(s)/SATPLAN		
		$r \leq 4$	$4 < r \leq 10$	$r > 10$	$r \leq 4$	$4 < r \leq 10$	$r > 10$	$r \leq 4$	$4 < r \leq 10$	$r > 10$
0.4−0.5	52	40	9	3	3	2	47	43	4	5
0.51−0.6	41	24	7	10	1	12	28	41	0	0
0.61−0.7	51	5	15	31	1	4	46	41	2	8
0.71−0.8	42	7	9	26	1	0	41	20	2	20
0.81−0.9	43	7	4	32	0	1	42	25	5	13
0.91−0.99	37	16	9	12	0	0	37	17	4	16

Table 2. Summing up of results for minimizing plan length.

6 Related work

We have seen that in the IPC-5 SATPLAN was the winner together with another SAT-based planner, i.e., MAXPLAN [57, 13, 11]. The question is whether the new features we have proposed can be simply integrated into MAXPLAN. MAXPLAN works by first estimating an upper bound n for the optimal makespan, and than (i) generating a SAT formula Π_n for the fixed makespan n, and (ii) checking Π_n for satisfiability, by using a modified version of MINISAT. The algorithm stops if Π_n is unsatisfiable (and in this case $n + 1$ is the optimal makespan), otherwise n is decreased. Given this, it should be relatively easy to integrate the proposed features for SATPLAN into MAXPLAN.

Considering the SAT literature, the problem of finding an optimal solution for Π_n in the presence of a quantitative constraint can be formulated as a Pseudo-Boolean optimization problem. MINISAT+ [22] has been the most effective solver in the 2005 Pseudo-Boolean evaluation[11] [43], and among the best in the 2006 and 2007 evaluations[12]. It is based on a reduction to SAT. In [30] the authors consider the problem of optimally solving SAT problems with preferences on literals, show how this can be done by modifying DLL along the same lines used in this paper, and show that the resulting system is competitive with MINISAT+ on MAXSAT and MINONE problems. Our work extends [30], and differs from it both theoretically and experimentally. From a theoretical point of view, we provide a simpler and more general treatment of preferences: for us, a preference is a formula and not just a literal. Formulas allow, e.g., to model conditional preferences like “If my jacket and pants have different color then I prefer to wear a white shirt” [5]. From an experimental point of view, here we focus on planning problems generated by SATPLAN, and also analyze the impact on the performances of the planner depending on the number of preferences. We show that when there are a few preferences, the performances of SATPLAN are not affected, which in turn implies that the same holds for the performances of the underlying SAT solver.

[11] See `http://www.cril.univ-artois.fr/PB05/`.

[12] See `http://www.cril.univ-artois.fr/PB06/` and `http://www.cril.univ-artois.fr/PB07/`, respectively.

Considering the literature on preferences in planning, some recent papers on planning with preferences particularly related to our work are [4, 6, 10, 47, 49]. In the first paper, the authors define a simple language for expressing temporally extended preferences and implement a forward search planner, called PPLAN. For each plan length $n \geq 0$, 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 [6], the authors show how to extend GP-CSP [17] in order to find plans in planning problems with preferences expressed as a TCP-net [7, 8]. 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 our 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, independently from the bound n. This is the approach followed in [10], 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 (using [2]): 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 [10] is exactly the same we deal with in Section 5: finding an optimal "minimal" plan for Π_n using a quantitative approach. The fundamental difference between our approach and [10] 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. In [47], the authors present an algorithm very similar to the one we have presented in this paper, and that thus first appeared in [31]. The algorithm is used to solve "optimal planning" problems, which can be easily captured by our framework. In addition, the authors provide theoretical results related to the framework presented, in particular to the complexity of "optimal planning". Finally, very recently, in [49] the authors have extended PDDL3 with Hierarchical Task Network (HTN) [29] preference constructs. The related system, HTNPLAN-P, which is based on the HTN planner SHOP2 [45], shows good results on IPC-5 benchmarks.

In the IPC-5, there were different planners able to compete in the "SimplePreferences" category, which include metrics on goal violation, that use different approaches.

We already presented SGPLAN [36]. In more details, SGPLAN ver. 5 extends ver. 4 for PDDL2.2 in order to deal with the new constructs of PDDL3. Its basic idea is to partition a problem into sub-problems, one for each (soft) goal (considered as hard), solve the sub-problems individually by a modified version of an existing planner i.e., METRIC-FF [34], and then resolve inconsistencies across sub-problems. Other planners include YOCHAN^{PS} [3] and MIPS-XXL [19]. YOCHAN^{PS} is a heuristic planner based on relaxed graph to obtain a heuristic estimation. Given that in PDDL3 benchmarks the impact of preferences violation on the plan metric is linear (i.e., metric is the sum of weighted preference expressions), preferences can be reduced to additive action costs: thus, YOCHAN^{PS} reduces a PDDL3 problem into a PSP (Partial Satisfaction Planning) problem, by compiling the conditional effects into multiple action instances [24], and then uses the SAPA^{PS} PSP planner [18] to solve it. It explicitly selects a subset of preferences to achieve, which can be costly in the presence of many preferences. In our work, we have used a compilation technique from non-STRIPS to STRIPS problems inspired by the YOCHAN^{PS}'s approach. MIPS-XXL, on the other hand, uses a very different approach based on buchi automata and it is able to plan with (quantitative) Temporally Extended Preferences (TEP) expressed with LTL formulas. It iteratively invokes (a modified version of) the METRIC-FF planner [34], forcing plans to have decreasing metric. In our analysis we have used SGPLAN because the results of the IPC-5 clearly demonstrate its superior performances.

We have seen that the last planner is able to deal with TEP. Another important approach is the one presented in [1], where the authors propose a method for compiling planning problems with TEP into problems containing only final-state preferences and a metric function, and exploit a variety of heuristic functions for the last. They also identify conditions under which a plan is optimal: thus, differently to our approach, they are not guaranteed to return always an optimal plan. Moreover, they look for such a plan using an incremental algorithm similar in spirit to the one used in YOCHAN^{PS}. Their resulting system, HPLAN-P, is shown to have good performances, even if not as good as SGPLAN. In the specific case of planning as satisfiability, in [44] the authors generalize classical planning to deal with planning with Temporal Extended Goals (TEG), expressed as formulas in LTL, by means of a reduction for planning with LTL goals (without the next-time operator) can be translated into SAT.

Considering the approaches based on logic programming, two of the main works are those in [4, 50], which can deal with qualitative TEP. They are based on the PP preference language, which can express different type of preferences, e.g., over states, actions and trajectories. The concept of preferred plans is defined similarly to our paper. In [50], the authors use a compilation into a logic programs to be solved under the answer set semantics (ASP) [25, 26] by using an ASP solver. The work in [4] further extends the PP language with quantification, conditional constructs and aggregators. The related systems are known to be less efficient than other planners mainly because they do not use suited methods, e.g., heuristics, to guide search toward the achievements of preferences. In the same area, the work in [15] presents a framework where preferences can be compiled into logic programs (to be again solved by an ASP solver). Different preference strategies are presented, which include other approaches captured by the framework, e.g., the one presented in [9].

7 Conclusions, final remarks and future works

We have shown how it is possible to easily extend the Planning as Satisfiability approach in order to handle both qualitative and quantitative preferences, and SATPLAN to handle simple (i.e., where the structure of the partial order is restricted to particular, e.g., linear, cases) preferences. This work is an extension of the first attempt based on SAT which can effectively deal with this important problem [31]: it is actually considered the reference SAT-based approach to deal with preferences, see, e.g., [27], the AAAI 2007 tutorial on "Planning and Scheduling with Over-Subscribed Resources, Preferences, and Soft Constraints"[13] by Mihn B. Do, Terry Zimmermann and Subbarao Kambhampati, and the joint KR/ICAPS 2008 tutorial on "Preferences, Planning and Control"[14] by Ronen Brafman. Our experimental analysis points out that with a few preferences (as it is the case for problems with soft goals), the performances in term of CPU time of SATPLAN(w)/(b) are roughly the same as SATPLAN's ones, and that the same happens for SATPLAN(w) even in the presence of thousands of preferences, at least on some domains: significant degradations only show up in a reduced portion of the analyzed instances (results are even better for SATPLAN(s): it solves a different problem but often its results are the same of SATPLAN(w), and thus it can be used to compute good approximated results). Remarkably, such results are reached while SATPLAN(P) returns plan of considerable better quality than SATPLAN, and in line with state-of-the-art results showed by SGPLAN on most problems with non-conflicting and conflicting soft goals, but on problems with conflicting goals with non-uniform weights. This fact can suggest to evaluate in the future an approach based on compilation into a PB formula, which naturally handle linear metric functions defined over boolean variables with integer coefficients (similarly to [52, 55, 54]), and to relax the computation of makespan-optimal plans, which has been showed to be another source of inefficiency, in order to find even better solutions when dealing with problems involving both optimizations wrt actions, e.g., in the quantitative case [10], and, more recently, [12], and optimizations wrt soft goals. We also plan to directly deal with "action costs" (e.g., [42, 12]), also introduced as a requirement in IPC-6 (see, e.g., [20] for one of the best approaches).

References

1. Jorge A. Baier, Fahiem Bacchus, and Sheila A. McIlraith. A heuristic search approach to planning with temporally extended preferences. In Manuela M. Veloso, editor, *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1808–1815, 2007.
2. Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In Francesca Rossi, editor, *Proc. of the 9th International Conference on Principles and Practice of Constraint Programming (CP 2003)*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2003.

[13] The related slides are available at `http://rakaposhi.eas.asu.edu/aaai2007-psp-tut/osp.pdf`.

[14] The related slides are available at `http://www.cs.bgu.ac.il/~brafman/kr+icaps08.pdf`.

3. Josh Benton, Subbarao Kambhampati, and Minh B. Do. YochanPS: PDDL3 simple preferences and partial satisfaction planning. 5th Internation Planning Competition Booklet, pages 23-25. Available at `http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf`, 2006.
4. Meghyn Bienvenu, Christian Fritz, and Sheila A. McIlraith. Planning with qualitative temporal preferences. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proc. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 134–144. AAAI Press, 2006.
5. Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.
6. Ronen I. Brafman and Yuri Chernyavsky. Planning with goal preferences and constraints. In Susanne Biundo, Karen L. Myers, and Kanna Rajan, editors, *Proc. of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 182–191. AAAI Press, 2005.
7. Ronen I. Brafman and Carmel Domshlak. Introducing variable importance tradeoffs into cp-nets. In Adnan Darwiche and Nir Friedman, editors, *Proc. of the 18th Conference in Uncertainty in Artificial Intelligence (UAI 2002)*, pages 69–76. Morgan Kaufmann, 2002.
8. Ronen I. Brafman, Carmel Domshlak, and Solomon Eyal Shimony. On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research*, 25:389–424, 2006.
9. Gerhard Brewka and Thomas Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2):297–356, 1999.
10. Markus Büttner and Jussi Rintanen. Satisfiability planning with constraints on the number of actions. In Susanne Biundo, Karen L. Myers, and Kanna Rajan, editors, *Proc. of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 292–299. AAAI Press, 2005.
11. Yixin Chen, Ruoyun Huang, Zhao Xing, and Weixiong Zhang. Long-distance mutual exclusion for planning. *Artificial Intelligence*, 173(2):365–391, 2009.
12. Yixin Chen, Qiang Lv, and Ruoyun Huang. Plan-A: A cost-optimal planner based on SAT-constrained optimization. In *6th International Planning Competition. Available at `http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=Plan-A.pdf`*, 2008.
13. Yixin Chen, Zhao Xing, and Weixiong Zhang. Long-distance mutual exclusion for propositional planning. In Manuela M. Veloso, editor, *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1840–1845, 2007.
14. Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Journal of the ACM*, 5(7), 1962.
15. James P. Delgrande, Torsten Schaub, and Hans Tompits. A framework for compiling preferences in logic programs. *Theory and Practice of Logic Programming*, 3(2):129–187, 2003.
16. Emanuele DiRosa, Enrico Giunchiglia, and Marco Maratea. Solving satisfiability problems with preferences. *Accepted to Constraints, Special issue on Constraint-based approaches to preference modelling and reasoning*, 2009. Available at `http://www.star.dist.unige.it/~marco/Data/09constraints.pdf`.
17. Minh B. Do and Subbarao Kambhampati. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artificial Intelligence*, 132(2):151–182, 2001.
18. Minh B. Do and Subbarao Kambhampati. Partial satisfaction (over-subscription) planning as heuristic search. In *Proc. of the International Conference of Knowledge Based Computer Systems (KBCS 2004)*, 2004.
19. Stefan Edelkamp, S. Jabber, M. Nazih Kambhampati, and Minh B. Do. Large-scale optimal PDDL3 with MIPS-XXL. 5th Internation Planning Competition Book-

let, pages 28-30. Available at `http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf`, 2006.

20. Stefan Edelkamp and Peter Kissmann. Optimal symbolic planning with action costs and preferences. In Craig Boutilier, editor, *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1690–1695, 2009.
21. Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Proc. of the 6th International Conference on the Theory and Applications of Satisfiability Testing (SAT 2003)*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.
22. Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into SAT. *Journal of Satisfiability, Boolean Modeling and Computation*, 2(1-4):1–26, 2006.
23. Richard Fikes and Nils J. Nilsson. STRIPSe: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.
24. B. Cenk Gazen and Craig A. Knoblock. Combining the expressivity of UCPOP with the efficiency of Graphplan. In Sam Steel and Rachid Alami, editors, *Proc. of the 4th European Conference on Planning (ECP 1997): Recent Advances in AI Planning*, volume 1348 of *Lecture Notes in Computer Science*, pages 221–233. Springer, 1997.
25. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Proc. 5th International Conference on Logic Programming (ICLP/SLP 1998)*, pages 1070–1080, 1988.
26. Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
27. Alfonso Gerevini, Patrick Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the 5th IPC: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009.
28. Alfonso Gerevini and Derek Long. Plan constraints and preferences in PDDL3. In *Proc. of the ICAPS-2006 Workshop on Preferences and Soft Constraints in Planning*, pages 46–53, 2006.
29. Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*, chapter Hierarchical Task Network Planning. Morgan Kaufmann, 2003.
30. Enrico Giunchiglia and Marco Maratea. Solving optimization problems with DLL. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *Proc. of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 377–381. IOS Press, 2006.
31. Enrico Giunchiglia and Marco Maratea. Planning as satisfiability with preferences. In *Proc. of the 22nd AAAI Conference on Artificial Intelligence*, pages 987–992. AAAI Press, 2007.
32. Enrico Giunchiglia and Marco Maratea. SAT-based planning with minimal-#actions plans and ”soft” goals. In Roberto Basili and Maria Teresa Pazienza, editors, *Proc. of the 10th Congress of the Italian Association for Artificial Intelligence (AI*IA 2007)*, volume 4733 of *Lecture Notes in Computer Science*, pages 422–433. Springer, 2007.
33. Enrico Giunchiglia, Alessandro Massarotto, and Roberto Sebastiani. Act, and the rest will follow: Exploiting determinism in planning as satisfiability. In *Proc. of 15th National Conference on Artificial Intelligence (AAAI 1998)*, pages 948–953. AAAI Press, 1998.
34. Jörg Hoffmann. The Metric-FF planning system: Translating “ignoring delete lists” to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341, 2003.
35. Jörg Hoffmann and Stefan Edelkamp. The deterministic part of IPC-4: An overview. *Journal of Artificial Intelligence Research*, 24:519–579, 2005.
36. Chih-Wei Hsu, Benjamin W. Wah, Rouyun Huang, and Yixin Chen. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In Manuela M. Veloso, editor, *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1924–1929, 2007.

37. Paul Jackson and Daniel Sheridan. Clause form conversions for boolean circuits. In Holger H. Hoos and David G. Mitchell, editors, *Proc. of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT 2004)*, volume 3542 of *Lecture Notes in Computer Science*, pages 183–198. Springer, 2005.
38. Matti Järvisalo, Tommi A. Junttila, and Ilkka Niemelä. Unrestricted vs restricted cut in a tableau method for boolean circuits. *Annals of Mathemathics and Artificial Intelligence*, 44(4):373–399, 2005.
39. Henry Kautz and Bart Selman. Planning as satisfiability. In Bernd Neumann, editor, *Proc. of the 10th European Conference on Artificial Intelligence (ECAI 1992)*, pages 359–363. IOS Press, 1992.
40. Henry Kautz and Bart Selman. Unifying SAT-based and graph-based planning. In Thomas Dean, editor, *Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI 1999)*, pages 318–325. Morgan-Kaufmann, 1999.
41. Henry Kautz and Bart Selman. SATPLAN04: Planning as satisfiability. In *5th Internation Planning Competition Booklet, pages 45-47. Available at* `http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf`, 2006.
42. Emil Keyder and Hector Geffner. Heuristics for planning with action costs revisited. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikolaos M. Avouris, editors, *Proc. of 18th European Conference on Artificial Intelligence (ECAI 2008)*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 588–592. IOS Press, 2008.
43. Vasco M. Manquinho and Olivier Roussel. The first evaluation of pseudo-boolean solvers (PB05). *Journal on Satisfiability, Boolean Modeling and Computation*, 2:103–143, 2006.
44. Robert Mattmüller and Jussi Rintanen. Planning for temporally extended goals as propositional satisfiability. In Manuela M. Veloso, editor, *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1966–1951, 2007.
45. Dana S. Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.
46. David A. Plaisted and Steven Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2:293–304, 1986.
47. Katrina Ray and Matthew L. Ginsberg. The complexity of optimal planning and a more efficient method for finding solutions. In Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric A. Hansen, editors, *Proc. of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008)*, pages 280–287. AAAI, 2008.
48. David E. Smith. Choosing objectives in over-subscription planning. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *Proc. of 14th International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 393–401. AAAI, 2004.
49. Shirin Sohrabi, Jorge A. Baier, and Sheila A. McIlraith. HTN planning with preferences. In Craig Boutilier, editor, *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1790–1797, 2009.
50. Tran Cao Son and Enrico Pontelli. Planning with preferences using logic programming. *Theory and Practice of Logic Programming*, 6(5):559–607, 2006.
51. G. Tseitin. On the complexity of proofs in propositional logics. *Seminars in Mathematics*, 8, 1970.
52. Menkes van den Briel and Subbarao Kambhampati. Optiplan: Unifying ip-based and graph-based planning. *Journal of Artificial Intelligence Research*, 24:919–931, 2005.
53. Menkes van den Briel, Romeo Sanchez Nigenda, Minh Binh Do, and Subbarao Kambhampati. Effective approaches for partial satisfaction (over-subscription) planning. In Deborah L. McGuinness and George Ferguson, editors, *Proc. of 19th National Conference on Artificial Intelligence (AAAI 2004)*, pages 562–569. AAAI Press / The MIT Press, 2004.

54. Menkes H. L. van den Briel, Thomas Vossen, and Subbarao Kambhampati. Loosely coupled formulations for automated planning: An integer programming perspective. *Journal of Artificial Intelligence Research*, 31:217–257, 2008.
55. Menkes van der Briel, Subbarao Kambhampati, and Thomas Vessen. IPPLAN: Planning as integer programming. 5th Internation Planning Competition Booklet, pages 26-28. Available at `http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf`, 2006.
56. Joost P. Warners. A linear-time transformation of linear inequalities into CNF. *Information Processing Letters*, 68(2):63–69, 1998.
57. Zhao Xing, Yixin Chen, and Weixiong Zhang. Maxplan: Optimal planning by decomposed satisfiability and backward reduction. In *5th Internation Planning Competition Booklet, pages 53-55. Available at `http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf`*, pages 53–55, 2006.