Journal of Logic and Computation Advance Access published July 18, 2010

Constraints Corner

Introducing Preferences in Planning as Satisfiability

ENRICO GIUNCHIGLIA and MARCO MARATEA, *DIST - University of Genova, Viale F. Causa 15, Genova, Italy.*
E-mail: enrico@dist.unige.it; 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 article 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 given partially ordered preferences. We first prove that, at fixed makespan, our approach returns an 'optimal' plan, if any. Then, considering both classical planning problems and problems coming from IPC-5, we show that SATPLAN extended in order to deal with preferences: (i) returns optimal plans that 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 of the IPC-5. Notably, such results are often obtained without sacrificing efficiency.

Keywords: Planning, satisfiability, preferences.

1 Introduction

Planning as Satisfiability [42] is one of the most well-known and effective techniques for classical planning: SATPLAN [43, 44] 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] [38] and a co-winner in the IPC-5[2] [30] (together with another planner based on SAT, MAXPLAN [14, 16, 61]). 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 [24]. MINISAT is based on the Davis–Logemann–Loveland procedure (DLL) [17] like most of the state-of-the-art SAT checkers, and won the SAT competition in

[1] http://www.tzi.de/~edelkamp/ipc-4/
[2] http://zeus.ing.unibs.it/ipc-5/

doi:10.1093/logcom/exq023

2005[3] (together with the SAT/CNF minimizer SATELITE), the SAT race 2006 and 2008,[4] on industrial benchmarks.

In this article we extend the Planning as Satisfiability approach in order to handle both qualitative (in which the 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. section 2.3 of [30]. 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 [8] (in the context of CP-net [7]) and then in [13, 33] 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) returns optimal plans that 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 goals violation: even if SATPLAN(P) and SGPLAN solve a different problem, i.e. bounded versus 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 surprising, by correlating the good behaviour 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 [41]. 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

[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.

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 [26] 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 [52, 57]) is practically very important, we have considered some domains from the 'SimplePreferences' category of the IPC-5: given that on such domains some ADL [49] 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 [5, 27] 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 article

- 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 plans 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 non-conflicting and conflicting soft goals with uniform weights.
- We show that the performance 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 [60] leads to better performances than the one based on [3], at least in our setting. Remarkably, this fact does not agree with the good results of [3] presented in [3, 12].

The article 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 article 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 $\mathcal{F} \cap \mathcal{F}' = \emptyset$); and
- 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 a 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; and
- 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

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

in which the first formula corresponds to the initial state, the second to the transition relation and the third to the goal state. Equation (2) has 7 plans (i.e. satisfying interpretations), each corresponding to a non-empty subset of $\{Car_0, Bus_0, Bike_0\}$.

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 $\neg Bus_0$, and it amounts to prefer the plans satisfying $\neg Bus_0$ 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 $\neg Bike_0$, $\neg Bus_0$ and $\neg Car_0$ 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 *(w.r.t.* $\langle P, \prec\rangle$) 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 w.r.t. $\langle P, \prec\rangle$. A plan π is *optimal for Π_n (w.r.t. $\langle P, \prec\rangle$)* 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 $\pi' \prec \pi$ (w.r.t. $\langle P, \prec\rangle$). Other preference formalisms include [6, 8, 18, 54] and are discussed in Section 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 (w.r.t. $\langle P, c\rangle$)* 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 (respectively quantitative) preference:

- $\langle\{\neg Bike_0, \neg Bus_0, \neg Car_0\}, \emptyset\rangle$, (respectively $\langle\{\neg Bike_0, \neg Bus_0, \neg Car_0\}, c\rangle$, where c is the constant function 1) then there are three optimal plans, corresponding to $\{Bike_0\}$, $\{Bus_0\}$, $\{Car_0\}$.
- $\langle\{\neg Bike_0, \neg Bus_0, \neg Car_0\}, \{\neg Bike_0 \prec \neg Car_0\}\rangle$, (respectively $\langle\{\neg Bike_0, \neg Bus_0, \neg Car_0\}, c\rangle$, where $c(\neg Bike_0) = 2$ while $c(\neg Bus_0) = c(\neg Car_0) = 1$) then there are two optimal plans, corresponding to $\{Bus_0\}$, $\{Car_0\}$.

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

function OPT-DLL(φ,S,P′,≺′)
  if (∅∈φ) return FALSE;
  if (φ=∅) return S;
  if ({l}∈φ_l) return OPT-DLL(φ_l,S∪{l},P′,≺′);
  l := ChooseLiteral(φ,S,P′,≺′);
  V := OPT-DLL(φ_l,S∪{l},P′,≺′);
  if (V≠FALSE) return V;
  return OPT-DLL(φ_l̄,S∪{l̄},P′,≺′).
```

FIGURE 1. The algorithm of QL-PLAN.

- $\langle\{Bike_0 \vee Bus_0\},\emptyset\rangle$, (respectively $\langle\{Bike_0 \vee Bus_0\},c\rangle$, where c is the constant function 1) then all the plans except for the one corresponding to $\{Car_0\}$ 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:

(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 $p \prec p'$; and
- $cnf(\varphi)$, 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 $cnf(\varphi)$ such that $\mu' \models cnf(\varphi)$ 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 $\mu' \models cnf(\varphi)$.

  There are well-known methods for computing $cnf(\varphi)$ in linear time by introducing additional variables, e.g. [40, 50, 55];
- 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; and
- $ChooseLiteral(\varphi,S,P',\prec')$ 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.

As it can be seen from the figure, OPT-DLL [33] 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: at 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 lines 2 and 3 do not hold, and there is no unit clause, then a new literal l is chosen according to the behaviour of *ChooseLiteral* explained above;
6: l is assigned and, at line 7, the optimal solution is returned if such branch has not encountered an inconsistency; and
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;
(2) looks for assignments extending $\{v(p_1), \neg v(p_2)\}$: if the search fails, then OPT-DLL backtracks;
(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 $p_1 = (\neg Bike_0 \wedge \neg Bus_0 \wedge \neg Car_0)$ and $p_2 = (\neg Bike_0 \wedge \neg Bus_0)$ with $p_1 \prec p_2$. QL-PLAN returns the plan corresponding to $\{Car_0\}$ 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 w.r.t. $\langle P, \prec \rangle$, otherwise.

PROOF. From [19], we know that OPT-DLL(φ,S,P',$\prec'$) returns

(1) FALSE if φ is unsatisfiable, and
(2) an ‘optimal solution’ for φ w.r.t. $\langle P', \prec' \rangle$, 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 vice versa;
- 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 vice versa; 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 $S \models \varphi$.

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

FIGURE 2. The algorithm of QT-PLAN.

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

Consider now a quantitative preference $\langle P,c\rangle$. The problem of finding an optimal plan for Π_n w.r.t. $\langle P,c\rangle$ 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 $b_{k-1},\ldots,b_0$ and then consider the qualitative preference $\langle\{b_{k-1},\ldots,b_0\},\{b_{k-1}\prec b_{k-2},\ldots,b_1\prec b_0\}\rangle$.

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 $b_{k-1},\ldots,b_0$; 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 $\Pi_n\wedge Bool(P,c)$; and
(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* $b_{k-1},\ldots,b_0$.

An example of the binary encoding is Warners [60], denoted with 'W-encoding' in the following. The encoding by Bailleux and Boufkhad [3], 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:

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

Both encodings can be realized in polynomial time, but the size of the W-encoding is linear in $|P|$ while the 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. [3, 12]).

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 $b_{k-1},\ldots,b_0$, $b(c)=\{b_{k-1},\ldots,b_0\}$ and $p(c)$ is the partial order $b_{k-1}\prec b_{k-2},\ldots,b_1\prec b_0$.

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;
(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;
(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 Π_n is unsatisfiable, and thus no plan exists for Π 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 Π_n be a planning problem Π with makespan n, and let $\langle P, c\rangle$ be a quantitative preference for Π_n. QT-PLAN(Π_n,P,c) returns

(1) FALSE if Π_n has no plans, and
(2) an optimal plan for Π_n w.r.t. $\langle P, c\rangle$, otherwise.

PROOF. The proof is similar to the one of Theorem 1. Consider $\varphi' = cnf(\Pi_n \wedge Bool(P, c))$. Again from [19], 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' w.r.t. $\langle b(c), p(c)\rangle$, 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 vice versa; 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 point holds by definitions from items 1 and 2b in the definition of *Bool*. The second point holds from the assumptions on *cnf*. ■

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, until the optimal makespan. Our approach thus stops at the first satisfiable formula, and guarantees that no 'better' plan exists in this optimal makespan under the expressed preferences. Of course, better plans can instead exist at higher makespan. At implementation level, we performed some optimizations of the algorithm presented in Figure 1: (i) we do not introduce an auxiliary variable when the preference is already an atom (e.g. in the case of preferences defined on actions), in this case (sometimes significantly) reducing the number of variables OPT-DLL has to deal with; and (ii) we substitute $\equiv$ with $\vee$ again in the OPT-DLL call at line 1, given it guarantees correctness and

[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.

completeness as well. In the case of soft goals, the variables in $v(P)$ play the roles of 'goal selectors': they indicate if a certain soft goal is satisfied or not.

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. Given they are based on SATPLAN, we remind that these planners can find an optimal plan, at fixed makespan, under the expressed preferences.

In our experiments, we consider SATPLAN(w)/(b)/(s), SGPLAN [39] 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 only deal with the qualitative case and cannot guarantee optimality in the quantitative case: its results are presented, or most of the time commented in the text, to show that often are of the same quality of SATPLAN(w)/(b), or close, and often obtained in shorter time (at least in the domains we consider). Thus, it could be often 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 fewer soft goals but to have performance 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 (and the returned plan would be the empty one). In order to avoid this situation, which is inherited from the SATPLAN 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. In practice, we added a clause with the disjunction of all goal selectors in $v(P)$. Because of this, we discarded the problems whose original version has only one goal because they would have no soft goal. Also note that SGPLAN rejects these instances. Moreover, we also discarded 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 3.2 GHz processor and 1 GB RAM. Timeout has been set to 900 s.

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 sections 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 section 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, blocks-world, 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

[7]Note that the IPC-6 does not contain basic STRIPS problems.

TABLE 1. Number of soft goals satisfied on STRIPS domains coming from IPCs

DOMAIN	SGPLAN	SATPLAN	SATPLAN(w)	SATPLAN(b)	SATPLAN(s)
pipesworld-notankage	10	43	**102**	99	96
pipesworld-tankage	45	29	**108**	92	100
satellite	67	27	**75**	74	**75**
airport	0	31	**66**	63	51
promela-philosophers	55	29	**464**	**464**	**464**
promela-optical	52	13	**104**	**104**	**104**
psr-small	152	48	**232**	**232**	**232**
depots	23	21	**59**	**59**	56
driverLog	**229**	20	97	97	82
zenoTravel	**216**	19	67	60	66
freeCell	**53**	7	14	14	13
logistics	**243**	28	123	123	122
blocks-world	**269**	35	78	78	73
mprime	4	10	**12**	11	11
mystery	4	11	**12**	**12**	**12**
openstacks	0	5	**5**	**5**	**5**
pathways	0	29	64	64	**83**
storage	**84**	26	30	30	30
tpp	14	19	**82**	**82**	**82**
trucks	0	16	**16**	**16**	**16**

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 [31].

In Tables 1 and 2 we present results for solving problems with non-conflicting soft goals. In both tables, each row and column corresponds to a domain of instances and a system, respectively; the content is the sum of satisfied soft goals in all solved instances (Table 1) and the total number of timeouts, memory out or segmentation faults and the mean CPU time of solved instances in the domain (Table 2). Note that the first two numbers are the ones used to evaluate planning systems in the 'SimplePreferences' category of the IPC-5, and the way of presenting results used in Table 2 is customary in, e.g. Max-SAT evaluations. Regarding SGPLAN, it has been run on instances adapted from the propositional STRIPS instances and, when available, we have used the ADL version of the problems that, according to the authors,[8] may lead to less segmentation faults. For each domain, the best result is specified in bold, but for SATPLAN.

Looking at the tables, we see that SATPLAN(w)/(b)(s) behave comparatively well with SGPLAN: in Table 1 there are 12 domains in which SATPLAN(w) manages to satisfy more soft goals than SGPLAN, while for SGPLAN this happens in 6 domains. In Table 2, instead, SATPLAN(w) is the 'best' solver on 5 domains, SATPLAN(s) in others 4, while for SGPLAN this happens in 11 domains. It has to be noted that the results for SGPLAN would be much better in few domains by considering the STRIPS formulation, i.e. the numbers for the airport and openstacks domains would be 174 and 6/46.39, and 84 and 3/0.04, respectively, Nonetheless, at the same time, a STRIPS formulation leads to much worse results in other domains, i.e. SGPLAN does not solve any of the promela-philosophers, promela-optical and pathways instances.

[8]Personal communication with Chic-Wei Hsu.

TABLE 2. Results on STRIPS domains coming from IPCs. x/y stands for x timeouts or segmentation faults, and y seconds of solved instances (at mean)

DOMAIN	SGPLAN	SATPLAN	SATPLAN(w)	SATPLAN(b)	SATPLAN(s)
pipesworld-notankage	**0/0.14**	7/6.74	7/10.51	8/8.26	7/10
pipesworld-tankage	**4/0.81**	21/25.24	21/61.28	24/54.48	22/38.2
satellite	**5/1.01**	9/3.29	9/10.03	10/8.55	9/7.07
airport	50/–	13/21.8	**19/42.73**	20/39.8	21/21.47
promela-philosophers	0/88.27	0/3.28	**0/3.61**	0/3.64	0/3.65
promela-optical	**0/15.25**	0/14.77	0/17.22	0/17.09	0/15.95
psr-small	11/0.48	0/0.61	**0/0.48**	0/0.7	0/0.69
depots	7/98.04	1/2.33	1/19.7	1/19.79	**1/17.93**
driverLog	**0/88.55**	0/0.61	1/25.6	1/25.69	2/0.49
zenoTravel	**0/1.55**	1/10.44	1/15.23	2/10.26	1/14.62
freeCell	**6/25.24**	13/27.58	13/42.32	15/41.84	14/0.49
logistics	**0/0.02**	0/0.03	0/0.7	0/0.7	0/0.25
blocks-world	0/78.5	0/0.2	0/1.01	0/1.01	**0/0.35**
mprime	**0/3.6**	2/12.16	2/17.54	3/9.56	3/9.43
mystery	**0/3.14**	2/10.52	2/13.35	2/11.59	2/11.87
openstacks	30/–	5/13.45	**5/11.66**	5/57.92	5/67.29
pathways	**0/0.02**	0/1.44	11/33.89	11/34.08	5/15.89
storage	0/0.58	0/0.23	0/0.29	0/0.31	**0/0.28**
tpp	15/0.01	0/2.59	0/9.05	0/9.07	**0/8.15**
trucks	16/–	0/1.61	**0/1.88**	0/1.92	0/1.9

About the mean CPU time for solving instances in the domains, we can note that it is the case that the relative good results of SATPLAN(w)/(b)/(s) w.r.t. SGPLAN can come with a price: this is a well-known fact given that there can be a big difference in performance between optimal and non-optimal planning; with respect to SATPLAN, we can note that 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 in 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 section, 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 performance of SGPLAN is 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 3a shows the number of soft goals each planner does not satisfy, while Figure 3b 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 Figure 3 contains only the instances solved by SATPLAN within the time limit: there are around 50 instances not showed that 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).

As expected, from Figure 3a, 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 cases equal, while in theory this is not necessary the case. From Figure 3b we can see that the performance of SATPLAN is in overall not affected when adding preferences, i.e.

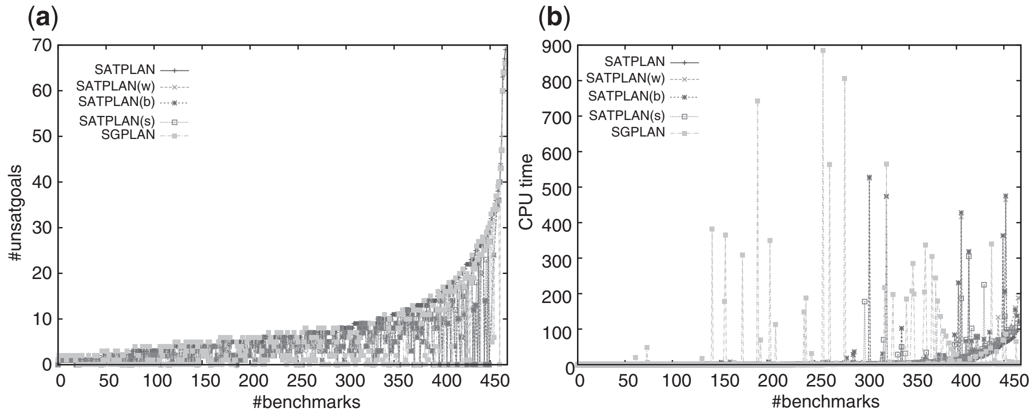

FIGURE 3. Non-conflicting soft goals. (**a**) Number of unsatisfied soft goals for SATPLAN/(w)/(b)/(s) and SGPLAN. (**b**) CPU time for SATPLAN/(s)/(w)/(b) and SGPLAN.

when imposing an ordering to the variables to be used for splitting in the SAT solver. There are just very few exceptions to this behaviour, in order of less than 10 (respectively 5) for SATPLAN(w)/(b) (respectively 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 performance [41]. We correlate the good performances of SATPLAN(w)/(b) to the fact that in all the problems considered there are at most 70 soft goals, and that the vast majority of the instances analysed 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 Section 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 oversubscription planning [52, 57]) 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 [49] 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 [5, 30]), we have relied on the following compilation technique, similar to the one used in [4, 5] to deal with PDDL3 benchmarks in YOCHAN^{PS} and in [27] 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,

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

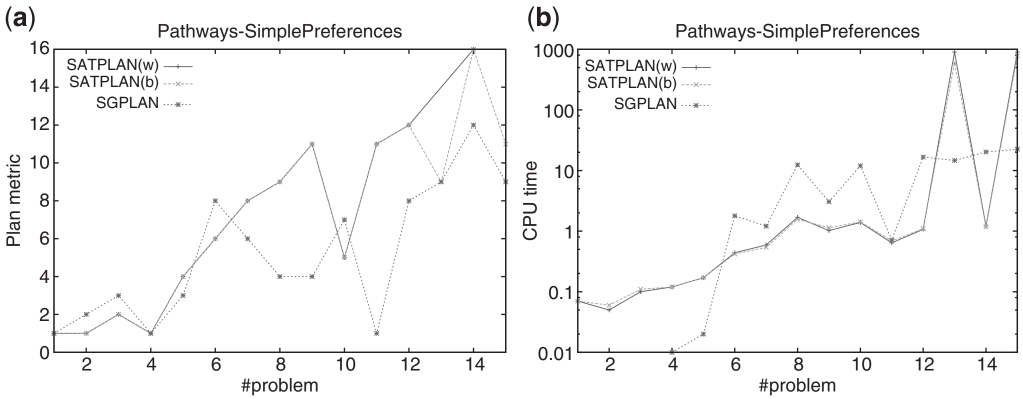

FIGURE 4. Pathways domain, 'SimplePreferences' category of IPC-5 with uniform weights. (**a**) Plan metric, i.e. number of unsatisfied soft goals, for SATPLAN(w), SATPLAN(b) and SGPLAN. (**b**) CPU time for SATPLAN(w), SATPLAN(b) and SGPLAN (in log scale).

we have included the pathway, storage, openstacks and trucks domains from IPC-5, i.e. the domains where plan metric is defined only on goals and there exist 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.

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 goals 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 instance, both plan metric and CPU time to find the plan. We remind that, on such benchmarks, the goal is to minimize the plan metric. In the analysis, we considered SATPLAN(w), SATPLAN(b) (given the metric is defined quantitatively) and, as a reference, SGPLAN, winner of the 'SimplePreferences' category at IPC-5. The next two paragraphs analyse separately the two cases.

5.2.1 Conflicting soft goals with uniform weights

For the Pathways domain in Figure 4b we can note 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 to the time limit, while SATPLAN(w) runs in timeout on both. This behaviour is consistent with the fact that the BB-encoding is effective, and can be more effective that the W-encoding, when the number of preferences is not high. About the plan metrics in Figure 4a, 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 5b 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

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

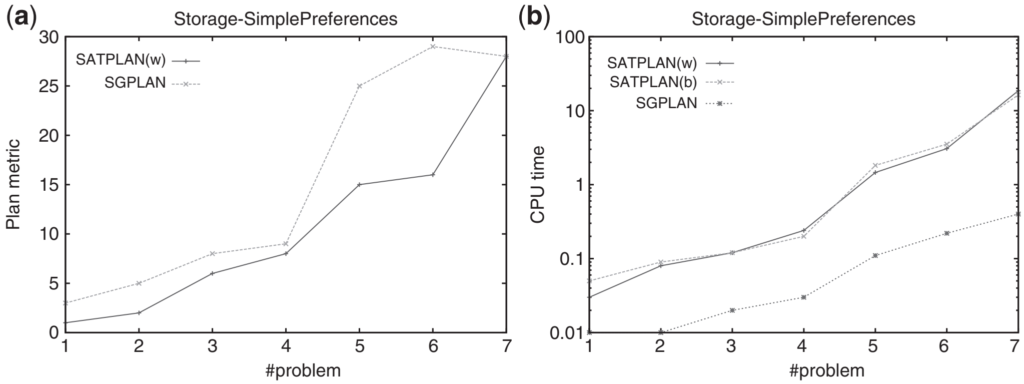

FIGURE 5. Storage domain, 'SimplePreferences' category of IPC-5 with uniform weights. (**a**) Plan metric, i.e. number of unsatisfied soft goals, for SATPLAN(w) (the same holds for SATPLAN(b)) and SGPLAN. (**b**) CPU time for SATPLAN(w), SATPLAN(b) and SGPLAN (in log scale).

nonetheless solve each problem in <20 s. The reason for the performance gap of SATPLAN(w)/(b) w.r.t. SGPLAN can be immediately explained by looking at Figure 5a: SATPLAN(w)/(b) return plans of much better quality than SGPLAN. The trade-off 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 benchmarks 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 6.93 s and 4.91 s 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 396 s, and from problem #3 it could not solve the instances within the time limit. Concerning the openstacks domain, we could compile only 1 instance, on which SATPLAN/(w)/(b) run in timeout. 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.

5.2.2 Conflicting soft goals with non-uniform weights

Here, 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 goals violation: only the W-encoding is used, given that BB-encoding can only deal with cardinality constraints.

For the Pathways domain, only problems #1 and #2 (as numbered in the competition) have only integer weights: these instances are solved very fast (<0.2 s) 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 <1 s. Then, starting from problem #5, the size of the *Bool* formula starts to be of very big size and difficult to manage.

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 ∼10 s 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:

(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 subset-minimal plans, 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 (i) a reduction in the plan length for SATPLAN(w)/(b)/(s) versus SATPLAN and (ii) a degradation in the CPU performances of SATPLAN(w)/(b)/(s) w.r.t. SATPLAN. We anticipate that, while for point (i) results meet expectations, this is only partially the case for point (ii).

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 allow optimizations in plan length. SATPLAN(w) in many cases returns plans of considerable better quality than SATPLAN. We can 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 is 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, similar to the case of soft goals, the quality of the plan returned 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 [36] 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 log scale.

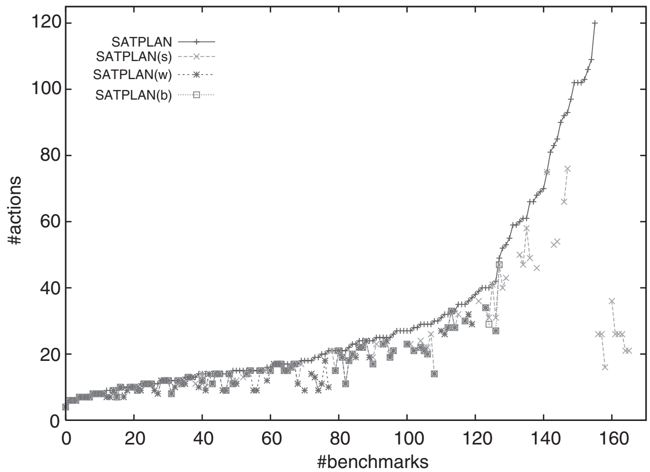

FIGURE 6. Number of actions in the returned plan for SATPLAN and SATPLAN(w)/(s)/(b).

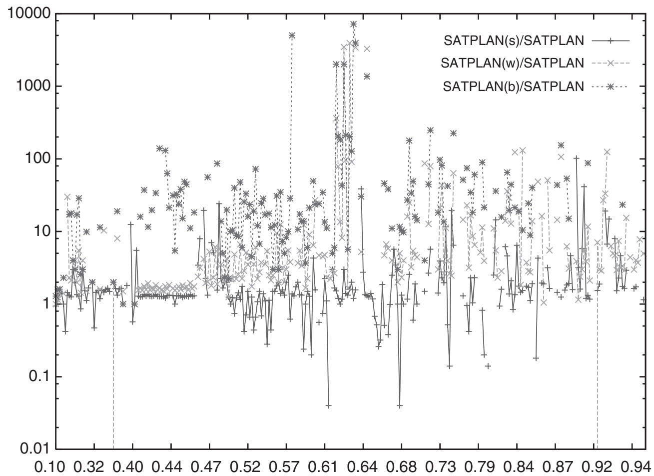

FIGURE 7. Performances degradation for SATPLAN(s)/(w)/(b).

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. The bad performances of SATPLAN(b) can be easily explained given 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 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 1.28 s/2.17 s/57.14 s by

TABLE 3. Summing up of results for minimizing plan length

Percent ratio	Problem	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

SATPLAN/(w)/(b). Note also how SATPLAN(s) instead maintains its good behaviour for almost all the spectra of instances; it is also in several cases more efficient than SATPLAN.

Finally, in Table 3 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, but that contain at least a certain percentage of action variables, i.e. at least 40 of the total variables of the instance. The table is structured as follows: the first column contains the percentage of action 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 of preferences and with a certain range of performance (as in the y-axis of Figure 7). We also consider the 'special' case '$r \leq 1$': first notice that SATPLAN(w)/(b) add an, even small, overhead due to the function *Bool*. It is thus very difficult that their CPU times do not exceed SATPLAN CPU time: in fact, only 5 (respectively 2) instances are such that '$r \leq 1$' for SATPLAN(w) (respectively SATPLAN(b)).

6 Related work

We have seen that in the deterministic part of the IPC-5, SATPLAN was the winner together with another SAT-based planner, i.e. MAXPLAN [14, 16, 61]. 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 then (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 of 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+ [25] has been the most effective solver in the 2005 Pseudo-Boolean evaluation[11] [46], and among the best in the 2006 and 2007 evaluations.[12] It is based on a reduction to SAT. In [33], 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 article and show that the resulting system is competitive with MINISAT+ on MAXSAT and MINONE problems. Our work extends [33], and differs

[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.

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' [7]. From an experimental point of view, here we focus on planning problems generated by SATPLAN, and also analyse the impact on the performance 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 performance of the underlying SAT solver.

Considering the literature on preferences in planning, some recent papers on planning with preferences particularly related to our work are [6, 8, 12, 51, 53]. 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' that can then be 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 [8], the authors show how to extend GP-CSP [20] in order to find plans in planning problems with preferences expressed as a TCP-net [9, 10]. 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 w.r.t. 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, independently from the bound n. This is the approach followed in [12], 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 posing a constraint imposing a smaller value for the objective function (using [3]): 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 [12] is exactly the same we deal with in Section 5: finding a 'minimum' plan for Π_n using a quantitative approach. The fundamental difference between our approach and [12] 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 clauses learned during a call are not reused by the following calls for the same n. Our approach can also deal with qualitative preferences. In [51], the authors present an algorithm very similar to the one we have presented in this article, and that thus first appeared in [34]. 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 [53] the authors have extended PDDL3 with Hierarchical Task Network (HTN) [32] preference constructs. The related system, HTNPLAN-P, which is based on the HTN planner SHOP2 [48], 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 goals violation, that use different approaches. We already presented

SGPLAN [39]. 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 subproblems, one for each (soft) goal (considered as hard), solve the subproblems individually by a modified version of an existing planner i.e. METRIC-FF [37] and then resolve inconsistencies across subproblems. Other planners include YOCHAN^{PS} [4, 5] and MIPS-XXL [22]. 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 metrics is linear (i.e. a 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 [27] and then uses the SAPA^{PS} PSP planner [21] 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 [37], forcing plans to have decreasing metric. In our analysis, we have used SGPLAN because the results of the IPC-5 clearly demonstrate its superior performance.

We have seen that the last planner is able to deal with TEP. Another important approach is the one presented in [1, 2], 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 [47] 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) into SAT.

Considering the approaches based on logic programming, two of the main works are those in [6, 54], which can deal with qualitative TEP. They are based on the *PP* preference language, which can express different types of preferences, e.g. over states, actions and trajectories. The concept of preferred plans is defined similarly to our article. In [54], the authors use a compilation into a logic program to be solved under the answer set semantics (ASP) [28, 29] by using an ASP solver. The work in [6] 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 towards the achievement of preferences. In the same area, the work in [18] 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 [11].

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 [34]: it is actually considered the reference SAT-based approach to deal with preferences, see e.g. [30], the AAAI 2007 tutorial on 'Planning and Scheduling with Over-Subscribed Resources, Preferences, and Soft Constraints'[13] by Do, Zimmermann and Kambhampati, and the joint KR/ICAPS 2008 tutorial on 'Preferences, Planning and Control'[14] by 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 analysed 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 considerably better quality than SATPLAN, and competitive with state-of-the-art results showed by SGPLAN on most problems with non-conflicting and conflicting soft goals with uniform weights. Instead, the same do not hold on most problems with conflicting soft goals and non-uniform weights. This fact can suggest to evaluate an approach based on compilation into a Pseudo-Boolean formula, which naturally handles linear metric functions defined over Boolean variables with integer coefficients (similar to [56, 58, 59]), and to relax the computation of makespan-optimal plans, which has been shown to be another possible source of inefficiency for plan quality. We also plan to directly deal with 'action costs' (e.g. [15, 45]), also introduced as a requirement in IPC-6 (see e.g. [23] for one of the best approaches).

Acknowledgements

This article extends previous work presented in [34, 35]. 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. We would also like to thank the anonymous reviewers for their useful suggestions.

References

[1] J. A. Baier, F. Bacchus, and S. A. McIlraith. A heuristic search approach to planning with temporally extended preferences. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, M. M. Veloso, ed., pp. 1808–1815, 2007.

[2] J. A. Baier, F. Bacchus, and S. A. McIlraith. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence*, **173**, 593–618, 2009.

[3] O. Bailleux and Y. Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP 2003)*, F. Rossi, ed. Vol. 2833 of *Lecture Notes in Computer Science*, pp. 108–122. Springer, 2003.

[4] J. Benton, M. B. Do, and S. Kambhampati. Anytime heuristic search for partial satisfaction planning. *Artificial Intelligence*, **173**, 562–592, 2009.

[5] J. Benton, S. Kambhampati, and M. B. Do. YochanPS: PDDL3 simple preferences and partial satisfaction planning. In *5th International Planning Competition Booklet*, pp. 23–25, 2006. Available at http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf.

[6] M. Bienvenu, C. Fritz, and S. A. McIlraith. Planning with qualitative temporal preferences. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation*

[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`.

and Reasoning (KR 2006), P. Doherty, J. Mylopoulos, and C. A. Welty, eds, pp. 134–144. AAAI Press, 2006.

[7] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, **21**, 135–191, 2004.

[8] R. I. Brafman and Y. Chernyavsky. Planning with goal preferences and constraints. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, S. Biundo, K. L. Myers, and K. Rajan, eds, pp. 182–191. AAAI Press, 2005.

[9] R. I. Brafman, C. Domshlak, and S. Eyal Shimony. On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research*, **25**, 389–424, 2006.

[10] R. I. Brafman and C. Domshlak. Introducing variable importance tradeoffs into CP-nets. In *Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence (UAI 2002)*, A. Darwiche and N. Friedman, eds, pp. 69–76. Morgan Kaufmann, 2002.

[11] G. Brewka and T. Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, **109**, 297–356, 1999.

[12] M. Büttner and J. Rintanen. Satisfiability planning with constraints on the number of actions. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, S. Biundo, K. L. Myers, and K. Rajan, eds, pp. 292–299. AAAI Press, 2005.

[13] T. Castell, C. Cayrol, M. Cayrol, and D. Le Berre. Using the Davis and Putnam procedure for an efficient computation of preferred models. In *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI 1996)*, pp. 350–354. John Wiley and Sons, 1996.

[14] Y. Chen, R. Huang, Z. Xing, and W. Zhang. Long-distance mutual exclusion for planning. *Artificial Intelligence*, **173**, 365–391, 2009.

[15] Y. Chen, Q. Lv, and R. 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.

[16] Y. Chen, Z. Xing, and W. Zhang. Long-distance mutual exclusion for propositional planning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, M. M. Veloso, ed., pp. 1840–1845, 2007.

[17] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, **5**, 394–397, 1962.

[18] J. P. Delgrande, T. Schaub, and H. Tompits. A framework for compiling preferences in logic programs. *Theory and Practice of Logic Programming*, **3**, 129–187, 2003.

[19] E. Di Rosa, E. Giunchiglia, and M. Maratea. Solving satisfiability problems with preferences. *Accepted to Constraints, Special issue on Constraint-based approaches to preference modelling and reasoning*, 2010. Available at http://www.star.dist.unige.it/∼marco/Data/09constraints.pdf.

[20] M. B. Do and S. Kambhampati. Planning as constraint satisfaction: solving the planning graph by compiling it into CSP. *Artificial Intelligence*, **132**, 151–182, 2001.

[21] M. B. Do and S. Kambhampati. Partial satisfaction (over-subscription) planning as heuristic search. In *Proceedings of the International Conference of Knowledge Based Computer Systems (KBCS 2004)*, 2004.

[22] S. Edelkamp, S. Jabber, M. N. Kambhampati, and M. B. Do. Large-scale optimal PDDL3 with MIPS-XXL. In *5th International Planning Competition Booklet*, pp. 28–30, 2006. Available at http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf.

[23] S. Edelkamp and P. Kissmann. Optimal symbolic planning with action costs and preferences. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, C. Boutilier, ed., pp. 1690–1695, 2009.

[24] N. Eén and N. Sörensson. An extensible SAT-solver. In *Proceedings of the 6th International Conference on the Theory and Applications of Satisfiability Testing (SAT 2003)*, E. Giunchiglia and A. Tacchella, eds. Vol. 2919 of *Lecture Notes in Computer Science*, pp. 502–518. Springer, 2003.

[25] N. Eén and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, **2**, 1–26, 2006.

[26] R. Fikes and N. J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, **2**, 189–208, 1971.

[27] B. C. Gazen and C. A. Knoblock. Combining the expressivity of UCPOP with the efficiency of Graphplan. In *Proceedings of the 4th European Conference on Planning (ECP 1997): Recent Advances in AI Planning*, Vol. 1348 of *Lecture Notes in Computer Science*, S. Steel and R. Alami, eds, pp. 221–233. Springer, 1997.

[28] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference on Logic Programming (ICLP/SLP 1988)*, R. Kowalski and K. Bowen, eds, pp. 1070–1080, 1988.

[29] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, **9**, 365–385, 1991.

[30] A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos. Deterministic planning in the 5th IPC: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, **173**, 619–668, 2009.

[31] A. Gerevini and D. Long. Plan constraints and preferences in PDDL3. In *Proceedings of the ICAPS-2006 Workshop on Preferences and Soft Constraints in Planning*, pp. 46–53, 2006.

[32] M. Ghallab, D. Nau, and P. Traverso. Hierarchical task network planning In *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2003.

[33] E. Giunchiglia and M. Maratea. Solving optimization problems with DLL. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, Vol. 141 of *Frontiers in Artificial Intelligence and Applications*, G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, eds, pp. 377–381. IOS Press, 2006.

[34] E. Giunchiglia and M. Maratea. Planning as satisfiability with preferences. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pp. 987–992. AAAI Press, 2007.

[35] E. Giunchiglia and M. Maratea. SAT-based planning with minimal-#actions plans and 'soft' goals. In *Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence (AI*IA 2007)*, Vol. 4733 of *Lecture Notes in Computer Science*, R. Basili and M. T. Pazienza, eds, pp. 422–433. Springer, 2007.

[36] E. Giunchiglia, A. Massarotto, and R. Sebastiani. Act, and the rest will follow: exploiting determinism in planning as satisfiability. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI 1998)*, pp. 948–953. AAAI Press, 1998.

[37] J. Hoffmann. The Metric-FF planning system: translating 'ignoring delete lists' to numeric state variables. *Journal of Artificial Intelligence Research*, **20**, 291–341, 2003.

[38] J. Hoffmann and Stefan Edelkamp. The deterministic part of IPC-4: an overview. *Journal of Artificial Intelligence Research*, **24**, 519–579, 2005.

[39] C.-W. Hsu, B. W. Wah, R. Huang, and Y. Chen. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, M. M. Veloso, ed., pp. 1924–1929, 2007.

[40] P. Jackson and D. Sheridan. Clause form conversions for Boolean circuits. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT 2004)*,

Vol. 3542 of *Lecture Notes in Computer Science*, H. H. Hoos and D. G. Mitchell, eds, pp. 183–198. Springer, 2005.

[41] M. Järvisalo, T. A. Junttila, and I. Niemelä. Unrestricted vs restricted cut in a tableau method for Boolean circuits. *Annals of Mathemathics and Artificial Intelligence*, **44**, 373–399, 2005.

[42] H. Kautz and B. Selman. Planning as satisfiability. In *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI 1992)*, B. Neumann, ed., pp. 359–363. IOS Press, 1992.

[43] H. Kautz and B. Selman. Unifying SAT-based and graph-based planning. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 1999)*, T. Dean, ed., pp. 318–325. Morgan-Kaufmann, 1999.

[44] H. Kautz and B. Selman. SATPLAN04: planning as satisfiability. In *5th International Planning Competition Booklet*, pp. 45–47, 2006. Available at http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf.

[45] E. Keyder and H. Geffner. Heuristics for planning with action costs revisited. In *Proc. of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, Vol. 178 of *Frontiers in Artificial Intelligence and Applications*, M. Ghallab, C. D. Spyropoulos, N. Fakotakis, and N. M. Avouris, eds, pp. 588–592. IOS Press, 2008.

[46] V. M. Manquinho and O. Roussel. The first evaluation of pseudo-Boolean solvers (PB'05). *Journal on Satisfiability, Boolean Modeling and Computation*, **2**, 103–143, 2006.

[47] R. Mattmüller and J. Rintanen. Planning for temporally extended goals as propositional satisfiability. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2009)*, M. M. Veloso, ed., pp. 1966–1951, 2007.

[48] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: an HTN planning system. *Journal of Artificial Intelligence Research*, **20**, 379–404, 2003.

[49] E. Pednault. ADL: exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR 1989)*, R. Brachman, H. Levesque, and R. Reiter, eds, pp. 324–332, 1989.

[50] D. A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, **2**, 293–304, 1986.

[51] K. Ray and M. L. Ginsberg. The complexity of optimal planning and a more efficient method for finding solutions. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008)*, J. Rintanen, B. Nebel, J. C. Beck, and E. A. Hansen, eds, pp. 280–287. AAAI, 2008.

[52] D. E. Smith. Choosing objectives in over-subscription planning. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS 2004)*, S. Zilberstein, J. Koehler, and S. Koenig, eds, pp. 393–401. AAAI, 2004.

[53] S. Sohrabi, J. A. Baier, and Sheila A. McIlraith. HTN planning with preferences. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, C. Boutilier, ed., pp. 1790–1797, 2009.

[54] T. C. Son and E. Pontelli. Planning with preferences using logic programming. *Theory and Practice of Logic Programming*, **6**, 559–607, 2006.

[55] G. Tseitin. On the complexity of proofs in propositional logics. *Seminars in Mathematics*, **8**, 115–125, 1970.

[56] M. H. L. van den Briel, T. Vossen, and S. Kambhampati. Loosely coupled formulations for automated planning: an integer programming perspective. *Journal of Artificial Intelligence Research*, **31**, 217–257, 2008.

[57] M. van den Briel, R. S. Nigenda, M. B. Do, and S. Kambhampati. Effective approaches for partial satisfaction (over-subscription) planning. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI 2004)*, D. L. McGuinness and G. Ferguson, eds, pp. 562–569. AAAI Press/The MIT Press, 2004.

[58] M. van den Briel and S. Kambhampati. Optiplan: unifying IP-based and graph-based planning. *Journal of Artificial Intelligence Research*, **24**, 919–931, 2005.

[59] M. van der Briel, S. Kambhampati, and T. Vessen. IPPLAN: planning as integer programming. In *5th Internation Planning Competition Booklet*, pp. 26–28, 2006. Available at http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf.

[60] J. P. Warners. A linear-time transformation of linear inequalities into CNF. *Information Processing Letters*, **68**, 63–69, 1998.

[61] Z. Xing, Y. Chen, and W. Zhang. MaxPlan: optimal planning by decomposed satisfiability and backward reduction. In *5th Internation Planning Competition Booklet*, pp. 53–55. Available at http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf, 2006.

Received 16 June 2010