A Pseudo-Boolean approach for Solving Planning Problems with IPC Simple Preferences

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

Abstract

Planning as Satisfiability (SAT) is the best approach to optimally solve classical planning problems in term of makespan, as witnessed by the results of past International Planning Competitions (IPCs). The language of the IPCs has evolved in the last two editions in order to include plan quality measures other than the makespan, e.g., to include "preferences" for the satisfaction of actions preconditions and/or goals: yet, the design, implementation and analysis of satisfiability-based approaches to cope with this issues is still at an early stage. In this paper, motivated by the recent availability of efficient systems to solve Pseudo-Boolean (PB) optimization problems, we present an approach to solve the instances in the "SimplePreferences" category of the IPC-5 by a reduction to a PB formula, and then use off-the-shelf PB solvers. Our approach thus returns plans with optimal plan metrics, at fixed makespan. We prove that the approach is correct, and then show that an implementation of our ideas based on SATPLAN yields to an effective method to solve these IPC-5 benchmarks.

Introduction

Planning as Satisfiability (SAT) is the best approach to optimally solve classical planning problems in term of makespan, as witnessed by the results of past International Planning Competitions (IPCs): two SAT-based planners, namely SATPLAN[1] (Kautz and Selman 1999; 2006) and MAXPLAN[2] (Xing, Chen, and Zhang 2006a; Chen et al. 2009), have been the winners in the "optimal" track of the deterministic part of the IPC-4[3] (Hoffmann and Edelkamp 2005) and IPC-5[4] (Gerevini et al. 2009). The language of the IPCs has evolved in the last two editions in order to include plan quality measures other than the makespan, e.g., to include "preferences" for the satisfaction of actions preconditions and/or goals. Instead, the work on satisfiability planning has mainly focused of enhancing the efficiency of the SAT-based approach with improved encodings, see, e.g., (Rintanen, Heljanko, and Niemelä 2006; Chen et al. 2009), and the exceptions to this trend are limited to somehow particular form of preferences and plan quality measures (e.g., soft goals with uniform costs (Giunchiglia and Maratea 2007), minimum-length plans (Büttner and Rintanen 2005), actions costs (Ramirez and Geffner 2007; Chen, Lv, and Huang 2008)). Thus, the design, implementation and analysis of satisfiability-based approaches to cope with the mentioned plan quality issues is still at an early stage.

In this paper, motivated by the recent availability of efficient systems to solve Pseudo-Boolean (PB) optimization problems, which is the result of a series of PB evaluations[5], we present an approach to solve the instances in the "SimplePreferences" category of the IPC- 5 by a reduction to a PB formula, and then using off-the-shelf PB solvers: such domains contain preferences on goals and/or actions preconditions, and there is a cost associated to the violation of such preferences. The reduction is carried out in two steps: given that such IPC-5 benchmarks are non-STRIPS, and some ADL constructs are used, we first compile IPC-5 benchmarks into STRIPS problems, by using the ADL2STRIPS tool[6], and then generate PB formulas from the STRIPS problems. A PB formulation in a very natural and concise way to express such optimization problems, and result in (much more) compact formulas than SAT.

Our approach thus returns plans with optimal plan metric, at fixed makespan. It has to be noted that there already exist in the context of optimal planning works that use optimization problems: a Max-SAT formulation has been used in, e.g., (Xing, Chen, and Zhang 2006b), but for the minimization of the makespan. Also, a modeling of planning problems with preferences, expressed through the PDDL3.0 (Gerevini et al. 2009) language, in 0-1 Integer Programming has been presented in (van den Briel, Kambhampati, and Vossen 2006). However, the first paper does not take into account the new features/metrics of PDDL, while in the second paper no implementation and experi-

[1] Available at http://www.cs.rochester.edu/~kautz/satplan/index.htm.

[2] Available at http://www.cse.wustl.edu/~chen/maxplan/.

[3] http://www.tzi.de/~edelkamp/ipc-4/.

[4] http://zeus.ing.unibs.it/ipc-5/.

[5] See http://www.cril.univ-artois.fr/PB09/ for the last.

[6] Specifically, we have used the version based on LPG (see, e.g., (Gerevini and Serina 1999; Gerevini, Saetti, and Serina 2003), provided by Alessandro Saetti, which is the one used in the IPC-5.

mental analysis are provided; moreover, both papers do not contain any formal result.

We first prove that the approach is correct. Then, we show that an implementation of our ideas based on SATPLAN yields to an effective method to solve the instances in the "SimplePreferences" category of the IPC-5, by taking as reference the results of SGPLAN (Hsu et al. 2006; 2007), the clear winner of the IPC-5 in this category.

Given that in the IPC-5 the impact of preferences violation on the plan metric is restricted to be linear, we focus on this case. Thus, all results we present could be simply adapted to work with an approach based on partial weighted Max-SAT, an extension of the well-known Max-SAT problem in which there are "hard" and "soft" clauses, and the goal is to find an assignment that satisfies all hard clauses and maximize the sum of the weights of satisfied soft clauses. Still, we present an analysis where a PB solver, namely MINISAT+ (Eén and Sörensson 2006), is the best, among a variety of PB and Max-SAT solvers, on the instances of interest. This is interesting, given that PB, in general, is a richer formalism than partial weighted Max-SAT.

Summing up, the major contributions of the paper are:

- We present an approach based on PB for handling IPC-5 "SimplePreferences" benchmarks within a satisfiability framework;
- We prove that the approach is correct;
- We implement these ideas in SATPLAN, and show that the approach is viable, i.e., it allows to widening the set of benchmarks can be dealt with a satisfiability-based approach.

The paper is structured as follows. We first present the needed preliminaries. Then, the following sections focus on how we model and solve the problem of interest. Details about the implementation, and the experimental evaluation, are then presented. The paper ends with discussion of related works and with conclusions and possible topics for future research.

Preliminaries

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

A *planning problem* is a triple $\langle I, tr, G\rangle$ where

- I is a Boolean formula over $\mathcal{F}$ and represents the *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*.

In the above definition of a planning problem, actions' effects on a state are described in an high-level action language like STRIPS (Fikes and Nilsson 1971), or PDDL. Given that the focus of our work is on classical planning, we make the assumption that the description 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.

In the planning as satisfiability approach (Kautz and Selman 1992), if n is an integer, the *planning problem* Π *with makespan* n is the Boolean formula Π_n defined as

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

and a *plan* is an interpretation satisfying (1). [7]

In a linear PB optimization problem, SAT clauses (i.e., set of literals) are extended to possibly contain integer coefficients, variables truth/falsity is interpreted as 0/1, and there is a bound on the value the constraint can assume. Given a set of coefficients $c_1, \ldots, c_k$, a set of boolean variables $\{x_1, \ldots, x_k\}$, and a positive integer number b, a PB constraint is of the form:

$$\sum_{i=1}^{k} c_i \times x_i \geq b \quad (2)$$

A PB formula is a conjunction of PB constraints. Moreover, an objective function can be applied[8] to the problem. If such objective function is specified, given a PB formula φ_n^{PB} the goal is to find an assignment to the variables of the problem that satisfies the formula (i.e., all PB constraints are satisfied), and optimizes the objective function.

Modeling IPC Simple Preferences problems as PB problems

As we said in the introduction, the reduction to a PB formula is carried out in two steps: SATPLAN, the planner we rely on, can only handle STRIPS problems, while IPC-5 benchmarks contain some ADL constructs to represent preferences. Thus, the first step is to adapt the IPC-5 benchmarks

[7] In the following, we can switch between plans and satisfying interpretations, intuitively meaning the same thing. In SAT, an interpretation is a set of literals; in PB, is an assignment of variables to 0 (i.e., falsity) or 1 (i.e., truth). It is easy to map interpretations into assignments and viceversa.

[8] In fact, in the PB evaluations the categories take into account if such function is specified (OPT), or not (DEC).

in a way that can be compiled into a STRIPS problem by state-of-the-art tools, e.g., ADL2STRIPS. For presenting our approach, we rely on the instance #1 of the TPP (Travelling and Purchase Problem) domain of the IPC-5, which contains preferences on both actions preconditions and goals, "GroundedPreferences" variant (referred as tpp1 below). In the following, we show the part related to the treatment of preferences, and how they have been modeled in PB. All the other SAT clauses are translated into PB constraints. In tpp1, action "drive" is represented as follows

```
(:action drive
:parameters (?t - truck ?from ?to - place)
:precondition (and (at ?t ?from) (connected ?from ?to)
            (preference p-drive (and
                        (ready-to-load goods1 ?from level0)
                        (ready-to-load goods2 ?from level0)
                        (ready-to-load goods3 ?from level0))))
:effect (and (not (at ?t ?from)) (at ?t ?to)))
```
(3)

(Soft) Goals and the metric are represented with

```
(:goal (and
      (preference p4A
                  (and (ready-to-load goods3 market1 level0)
                       (loaded goods3 truck1 level0)))
      ...
      (preference p0A (stored goods3 level1))
      ...
))

(:metric minimize (+ (* 1 (is-violated p0A))
                     ...
                     (* 16 (is-violated p4A))
                     (* 1 (is-violated p-drive))))
```
(4)

The semantic is defined as follows (Gerevini et al. 2009): in (3), action "drive" can be executed even if preference "p-drive" is violated, but then a cost 1 (from (4)) is paid each time this happens; in (4), the related cost in the metric is paid if any of the goal preference is violated.

We first modify the problem, in a way which is inspired by the approaches in (Gazen and Knoblock 1997; Benton, Kambhampati, and Do 2006), in order to avoid the use of the construct "preference": for each goal preference we introduce a (dummy) action whose precondition is the preference, and the effect is a (dummy) literal, e.g., for preference $p4A$

```
(:action dummy-p4A
:parameters ()
:precondition (and (ready-to-load goods3 market1 level0)
                  (loaded goods3 truck1 level0))
:effect (and (goal-p4A))).
```
(5)

The new (hard) goal of the problem is the conjunction of the dummy literals related to goal preferences introduced.
Note that, in PDDL3.0, several preferences can be tagged with the same preference name, and the same cost is associated to each single violation. This is indeed the case for the instances in the TPP domain ("GroundedPreferences" variant).

Actions containing preferences their preconditions, are split into two actions: the first, which we consider to maintain its name, consider the soft precondition in the action "drive" as hard; then, we introduce a second (dummy) action of the form[9]

```
(:action dummy_dr
:parameters (?t - truck ?from ?to - place)
:precondition (and (at ?t ?from) (connected ?from ?to)
                   (not (and(ready-to-load goods1 ?from level0)
                            (ready-to-load goods2 ?from level0)
                            (ready-to-load goods3 ?from level0))))
:effect (and (not (at ?t ?from)) (at ?t ?to) (goal-p-drive)))
```
(6)

where the soft precondition of the related original action is negated, and a dummy literal is added to the original effects. The intuition is to take into account if the original action is executed with, or without, its precondition formula satisfied: in this second case, $goal\text{-}p\text{-}drive$ is added as a further effect. The two actions are mutually exclusive: the second takes into account if action drive is executed with its soft precondition not satisfied.

The resulting problem is then given to the ADL2STRIPS tool to be compiled into a STRIPS problem. Dummy actions introduced are thus compiled into STRIPS actions. Thus, there could be (multiple) STRIPS actions in place of the ones in (5) or (6), whose preconditions (resp. effects) are finite conjunction of atoms (resp. literals). Below, such conjunction can be (equivalently) considered as sets.

Consider a STRIPS problem (with variable names possibly equal to the non-STRIPS problem). Further, let φ_{dr} be the preference formula related to "p-drive", φ_h is the formula corresponding to the "hard" part of the preconditions of the action "drive", and φ_e its effects, the following PB constraints are added to the PB problem.

Given a non-STRIPS action A-p, in the following we write $A_{p,i}$ for (one of) the corresponding STRIPS action with subscript, or time-stamp, i. Given that in the $tpp1$ instance there are 5 ($0 \leq j \leq 4$) soft goals, for each STRIPS action related to goals (5) (in PB-like format, without "$\times$")

$$\wedge_{l \in \varphi_{jA,n-1}} -1\ dummy_{pjA,n-1} + 1\ l_{n-1}\ \geq 0,$$
$$-1\ dummy_{pjA,n-1} + 1\ goal_{pjA,n} \geq 0$$

Regarding action drive, the following PB constraints are added (we remind that its preconditions are all hard)

$$\wedge_{l \in \varphi_h} -1\ drive_i + 1\ l_i \geq 0,$$
$$\wedge_{l \in \varphi_{dr}} -1\ drive_i + 1\ l_i \geq 0$$

while, for the effects, the following constraints are added

$$\wedge_{l \in \varphi_e} -1\ drive_i + sign(l)\ var(l)_{i+1} \geq b_l$$

where $var(l)$ returns the (fluent) variable the literal is built on, and $sign(l)$ is 1 and b_l is 0 if the literal is positive, while

[9] We can introduce a single action given that only one preference formula is in action "drive". This is the case for all instances in the "SimplePreferences" category. In general, we have to consider their power set.

$sign(l)$ is -1 and b_l is -1, otherwise. The last PB constraint added is

$$-1\ drive_i + 1\ goal_{p-drive,i+1} \geq 0$$

Similarly for action $dummy_{dr}$ (5).

Now, it is left how to express the optimization function: in (4), the idea is to minimize the violation of preferences (expressed with constructs *(is-violated p)* in PDDL3.0, where, given a preference p, *(is-violated p)* takes value 1 if the preference is not satisfied, and 0 otherwise (Gerevini et al. 2009)). With our formulation, the new goal literals of introduced actions are reached when a preference is satisfied and this is "mimicked" by the related action's execution: thus, the characterization of the metric function in (4) can be expressed using both actions and goals, i.e. with the following (linear) optimization functions (without subscript "A" in the equations)

$$\textit{max: } +1\ \textit{(goal}_{p0}\textit{)} + \ldots + 16\ \textit{(goal}_{p4}\textit{)} + \textstyle\sum_{i=1}^{n} -1\ \textit{(goal}_{dr,i}\textit{)} \quad (7)$$

$$\textit{max: } +1\ \textit{(dummy}_{p0}\textit{)} + \ldots + 16\ \textit{(dummy}_{p4}\textit{)} + \textstyle\sum_{i=o}^{n-1} -1\textit{(dummy}_{dr,i}\textit{)} \quad (8)$$

with the meaning that $goal$ (resp. $dummy$) takes value 1 if it holds (resp. is executed) at that time stamp, and 0 otherwise. While literals referred the goal preferences, i.e., $\{goal_{p0A}, \ldots, goal_{p4A}\}$ in (7), (resp., dummy actions, i.e., $\{dummy_{p0A}, \ldots, dummy_{p4A}\}$) can hold only at time stamp n (resp. $n-1$), the ones related to action preconditions can, in general, hold at any time stamp (in (7) and (8) such subscripts are implicit). If we know that, instead, actions can be only executed once, the optimization function that uses goals is expressed with

$$\textit{max: } +1\ \textit{(goal}_{p0}\textit{)} + \ldots + 16\ \ \textit{(goal}_{p4}\textit{)} - 1\ \textit{(goal}_{dr}\textit{)} \quad (9)$$

and similarly for (8). Even if from one hand this hypothesis on (ground) actions execution underlying (9) can be seen as a further approximation (other than the makespan) of the (unbounded) optimal plan metric, such hypothesis hold in practice in various cases, e.g., on classical, real-world planning domain like blocks-world and logistics.

A PB-based Solving Algorithm

Consider a STRIPS problem Π, and an integer n; P is the set of atoms added as effects in the actions introduced ($P = \{goal_{p0A}, \ldots, goal_{p4A}, goal_{dr}\}$ in (9)), and c is a function that maps elements in P with their corresponding positive integer costs.[10]

In Figure 1 there is the solving algorithm, in which

function PBPLAN(Π,n,P,c)
1 **return** PBO(CNF2PB(CNF(Π_n)), μ, o)

Figure 1: The algorithm of PBPLAN.

- CNF(φ), where φ is a formula, is a set of clauses 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., (Tseitin 1970; Plaisted and Greenbaum 1986; Jackson and Sheridan 2005);
- CNF2PB(C), where $C = \{l_1 \ldots l_m\}$ is a clause, is a PB constraint $p = c_1 \times x_1 + \ldots + c_m \times x_m \geq b$ where, for each k, $1 \leq k \leq m$, $c_k = +1$ and $x_k = l_k$ if l_k is a positive literal, while $c_k = -1$ and $x_k = var(l_k)$ if l_k is a negative literal, with $b = 1 - count(c_1, \ldots, c_m)$, where $count(c_1, \ldots, c_m)$ returns the number of negative coefficients; p is such that
  - for any interpretation μ, $\mu \models C$, there is an assignment to integer variables in p such that p is satisfied; and
  - for any assignment to the integer variables in p that satisfies the constraint, there is an interpretation μ such that $\mu \models C$.

  CNF2PB(ϕ), where ϕ is a set of clauses, is

  $$\bigwedge_{C \in \phi} \text{CNF2PB}(C)$$

- $o(\varphi)$ is the optimization function, of the form (7)-(9), thus

  $$max : \sum_{p \in P'} c'(p) \times p.$$

  where
  - $P' = P$ and $c' = c$, if we deal with an optimization function of type (9); or
  - $P' = P \cup_{i=1}^{n-1} \{goal_{dr,i}\}$ and c' extends c by assigning atoms in $P' \setminus P$ the weight $c(goal_{dr,n})$, if we deal with an optimization function of type (7) (atoms in P are implicitly with subscript n). Similarly for (8).

Consider Π_n^{PB} to be a PB formula, and PBO a generic PB solver, which returns FALSE if Π_n^{PB} is unsatisfiable, or a solution μ, which satisfies Π_n^{PB} and maximizes o, otherwise.

We are now ready to state the following Theorem.

Theorem 1 *Let Π be a planning problem, n the makespan, P the set of variables involved in the metric, with function c.* PBPLAN*(Π,n,P,c) returns*

[10] We can restrict to positive integer costs given that: (i) IPC-5 benchmarks do not contain negative weights, and (ii) positive real costs of the IPC-5 can be represented with integer numbers by multiplying them by 10^d, where d is the maximum number of (significant) decimal digits in the problem. However, dealing with negative costs is not a problem: 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 $\langle P, c\rangle$, 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. FALSE, *if* Π *has no plans at makespan* n*, and*
2. *a plan for* Π *at makespan* n *optimal wrt* $\langle P, c\rangle$*, otherwise.*

Proof. Let $\varphi_n^{PB} :=$ CNF2PB(CNF(Π_n)), from the assumptions on PBO we know that PBO(φ_n^{PB},μ,o) returns

1. FALSE if φ_n^{PB} is unsatisfiable, and
2. a solution μ, that maximizes o, otherwise.

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

- for each assignment μ in the signature of φ_n such that μ satisfies φ_n^{PB}, it must also hold that $\mu' \models \Pi_n$, where μ' corresponds to μ but reduced to the signature of Π_n; and
- for each assignment μ' in the signature of Π_n such that $\mu' \models \Pi_n$, there exists an assignment μ, $\mu \supseteq \mu'$, such that μ satisfies φ_n^{PB}.

The point holds from the assumptions on CNF, and by construction of CNF2PB.

Then, we need also to show that there is a correspondence between function o and the pair $\langle P, c\rangle$. Such correspondence holds by construction as well.

□

Implementation and experimental evaluation

We have evaluated the instances in all domains of the "SimplePreferences" category of the IPC-5, i.e., the TPP, Pathways, Storage, Trucks and Openstacks, with preferences grounded (when available). For TPP, which is the only domain with preferences on both actions preconditions and goals, we have used the optimization function (9).

At implementation level, we have modified SATPLAN in order to implement our PBPLAN algorithm, creating formulas in the format of the PB evaluations, instead of the DIMACS format for SAT problems in Conjunctive Normal Form (CNF), for each makespan up to the optimal, starting from 0. PBPLAN is also the name of our overall system. As we noticed in the introduction, given that in IPC-5 benchmarks the impact of preferences is restricted to be linear, the results we have presented hold also with an approach based on partial weighted Max-SAT problem: we run an analysis on the first satisfiable and last unsatisfiable formulas for each instance of the domains considered, and show the results in Fig. 2. We have used the best solvers that have participated to Max-SAT and PB evaluations along the years, with emphasis on the (more recent) "partial weighted" and "OPT-SMALL-INT" categories, the last being part of PB evaluations, and where: (i) PB constraints correspond to SAT clauses; (ii) there is no constraint with a sum of coefficients greater than 2^{20} (20 bits), and (iii) the objective function is linear.

The solvers are: MINIMAXSAT ver. 1.0 (Heras, Larrosa, and Oliveras 2008), based on MINISAT ver. 1.13, WMAXSATZ ver. 2.5, INCWMAXSATZ,[11] MSUNCORE (Marques-Silva and Manquinho 2008) ver. 1.2 and ver. 4.0; MINISAT ver. 1.14 (Eén and Sörensson 2006), GLPPB ver. 0.2, (by the same authors of PUEBLO (Sheini and Sakallah 2005)), BSOLO ver. 3.0.17 (Manquinho and Marques-Silva 2006), SAT4J ver. 2.1 and SCIPSPX ver. 1.2.0.[12] MINIMAXSAT and SAT4J read instances in both formats, and we show the results for their best formulation. Regarding MSUNCORE, the results of the two versions are very similar, thus in the analysis we show only ver. 1.2 . The timeout has been set to 900s on a Linux box equipped with a Pentium IV 3.2GHz processor and 1GB of RAM. Fig. 2, ordered according to MINISAT+ performances, shows that it is the best overall solver on such benchmarks

Results for planning domains Pathways, Storage and TPP are instead presented as in the IPC-5, in Fig. 3-5, in terms of both plan metrics, as defined in the original instances, and CPU time for PBPLAN and SGPLAN. The ones for the Trucks and Openstacks domains were only mentioned, given that few instances could be compiled and/or solved by PBPLAN. In the evaluation of the results, we have to underline that PBPLAN and SGPLAN solve two different problems: SGPLAN is targeted for sequential, unbounded planning, thus we expect to be usually much faster. Nonetheless, it is added as reference, in particular for plan metrics, given it has been the clear winner in IPC-5 on the category considered. Fig. 3 contains results for the Storage domain on the first 7 instances (as numbered in the IPC-5), i.e. the ones ADL2STRIPS could compile. On these instances, we can see that SGPLAN is, indeed, much faster than PBPLAN of about 1 order of magnitude (Fig.3 Right); nonetheless, PBPLAN solves the instances in less than (around) $30s$ while, notably, having better plan metrics than SGPLAN on most instances. The results for the first 20 instances of the Pathways domain are in Fig. 4. On these instances, the CPU times, Fig. 4 (Right), for PBPLAN and SGPLAN are comparable but on 5 instances, where PBPLAN runs in timeout, while are solved by SGPLAN, even if in tens of seconds. Regarding plan metrics, in Fig. 4 (Left), on the instances solved by both systems the results are comparable, but on 3 instances where SGPLAN (#8, #9 and #20) gives back plans of better quality. Among the instances from #21 to #30, not showed in Fig. 4, PBPLAN solves two instances, namely #23 and #29, in few seconds, and with plan metrics of 25.5 and 26.7, respectively. On the same instances, SGPLAN has metrics of 18 and 22, respectively, while it solves the remaining instances with a mean time of around $100s$. Results for the TPP domain are presented in Fig. 5. On the 10 instances showed, the behavior is similar to the Pathways domain, but the plan

[11] Both WMAXSATZ and INCWMAXSATZ versions we have used are slightly different from the ones used in the last evaluation, because the evaluation versions have caused some memory problems if the test instance is large (due to the fact that clause number is set statically in the code), i.e., storage6 and storage7. Personal communications by Joseph Argelich and Han Lin.

[12] Solvers have been downloaded from `http://www.lsi.upc.edu/~fheras/docs/m.tar.gz`, `http://www.minisat.se/MiniSat+.html`, `http://www.eecs.umich.edu/~hsheini/pueblo`, `http://forge.ow2.org/projects/sat4j/`, `http://www.csi.ucd.ie/staff/jpms/soft/soft.php`, `http://scip.zib.de/`, or obtained on request to the authors. We have used the version submitted to the evaluations, or the last available.

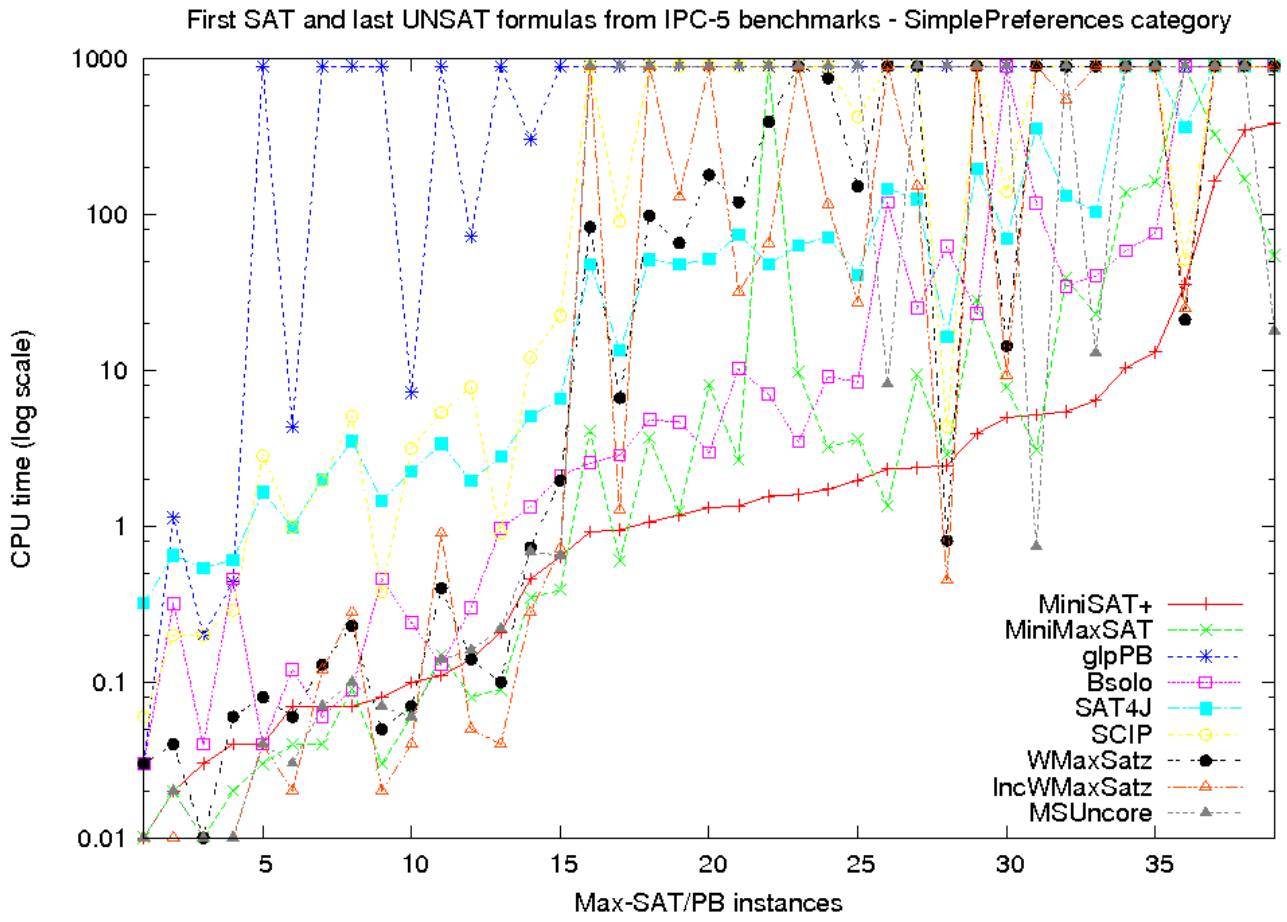

Figure 2: Results for Partial Weighted Max-SAT and PB solvers.

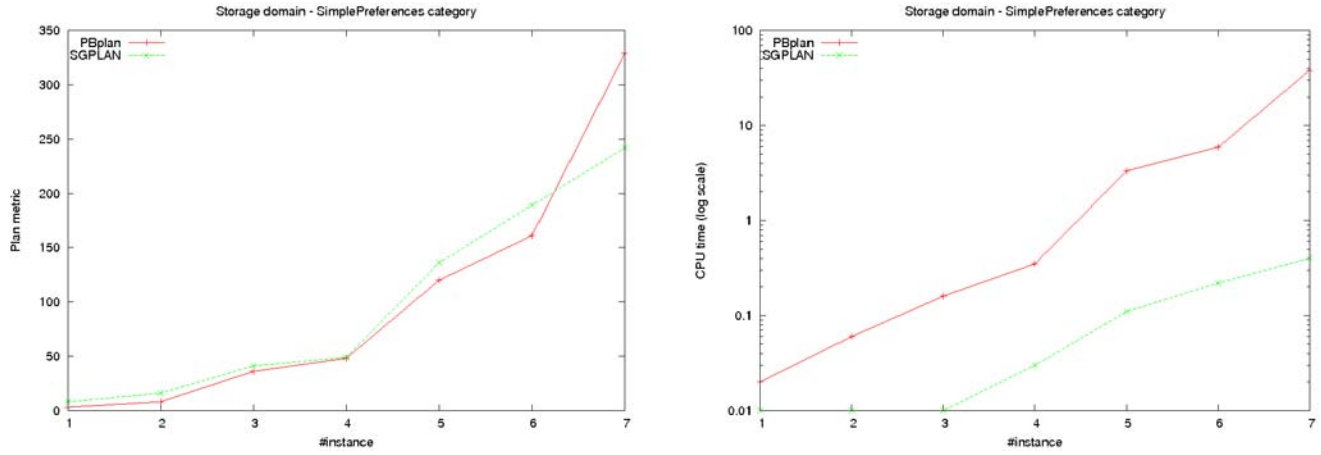

Figure 3: Plan metrics (Left) and CPU time (Right) for PBPLAN and SGPLAN on Storage instances.

quality of SGPLAN is better than the one of PBPLAN, in particular on instances #9 and #10. Instances from #11 to #15 can be compiled by ADL2STRIPS but not solved by PBPLAN, while instances from #16 to #20 can not be com-

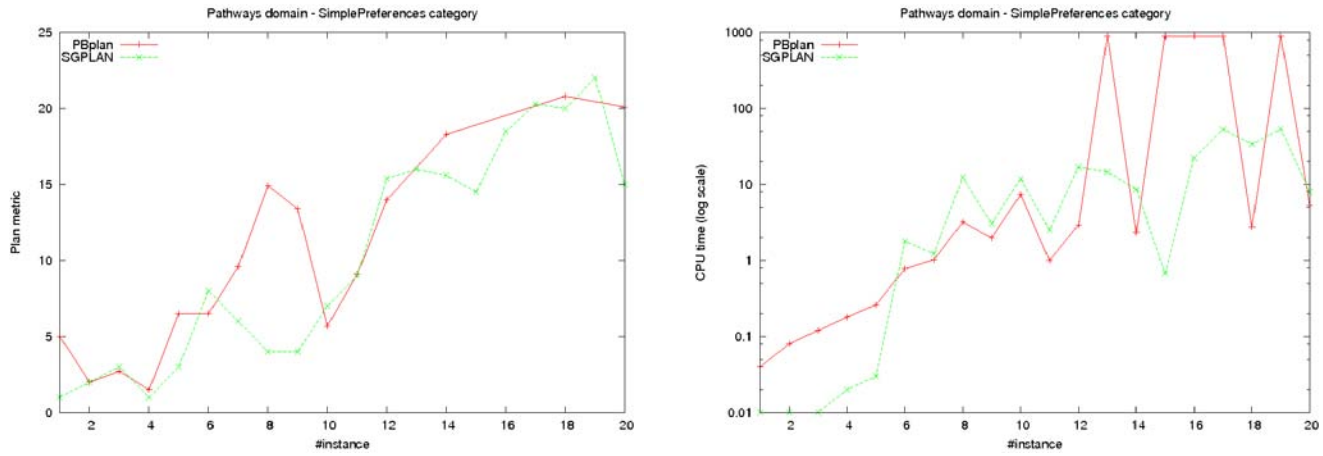

Figure 4: Plan metrics (Left) and CPU time (Right) for PBPLAN and SGPLAN on Pathways instances.

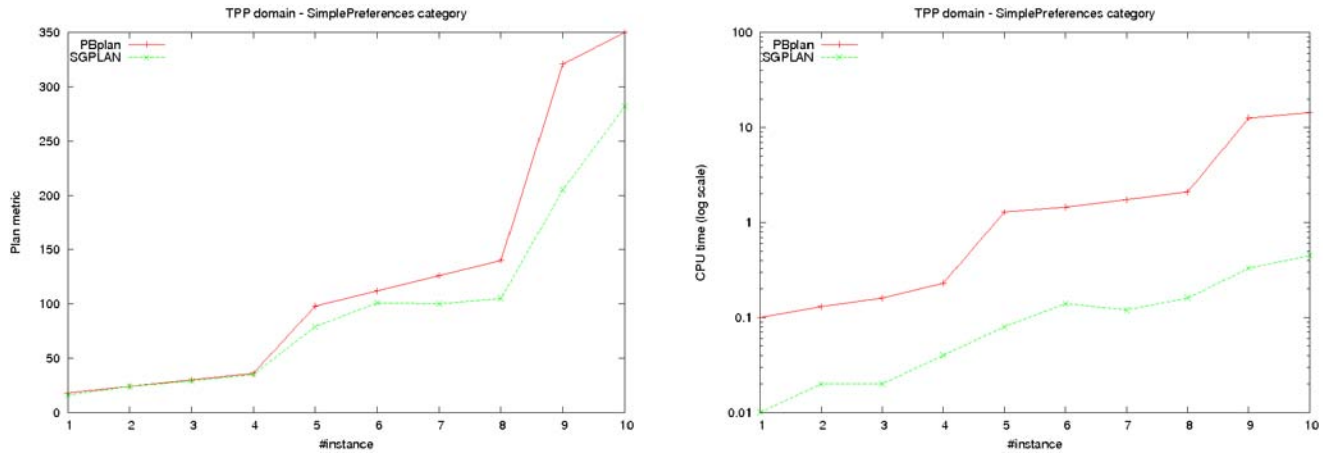

Figure 5: Plan metrics (Left) and CPU time (Right) for PBPLAN and SGPLAN on TPP instances.

piled with ADL2STRIPS. Note that these last instances contain more than 20000 variables and 800000 clauses, and the mean solving time for SGPLAN is more than $100s$. About the last two domains, Trucks and Openstacks, of the first 7 instances of the Trucks domain that can be compiled by ADL2STRIPS, only two instances can be solved by PBPLAN in 10 and 800 seconds, approximately, with plan metrics of 1 and 2, respectively. The same instances are solved very fast by SGPLAN, but with plan metric of 13 and 52, thus much higher than PBPLAN. On the same domain, finally note that for instances #3 to #7 even checking satisfiability on the first satisfiable formula generated by PBPLAN is difficult for MINISAT. The same holds for the Openstacks domain, where only 1 instance is compiled by ADL2STRIPS.

Summing up, SGPLAN returns plans faster than PBPLAN, and can solve instances that are not solved by PBPLAN: this is expected given they solve different problems. About plan metrics, the two planners return plans having, in general, comparable quality: sometimes, it is the case that the one returned by PBPLAN is better. This is remarkable, given that SGPLAN has been the clear winner in IPC-5.

Related work

The co-winner of the IPC-5, MAXPLAN, works as follows: it estimates an upper bound of the optimal makespan, and than decreases the makespan until an unsatisfiable SAT instance is found. (Rintanen, Heljanko, and Niemelä 2006; Chen et al. 2009) optimize the SATPLAN encoding in order to gain further efficiency. About "plan quality", the paper closer to our is the one in (van den Briel and Kambhampati 2005), where a compilation into 0-1 Integer Programming is proposed: a wider set of IPC-5 domains is analyzed, with more constructs, i.e., not only the ones in the "SimplePreferences" category, but no implementation, experimental evaluation and formal results are presented. As we have noticed, our encoding is similar to the one used

in YOCHANPS (Benton, Kambhampati, and Do 2006; Benton, Do, and Kambhampati 2009): it compiles a PDDL3.0 problem into a PSP planning problem (van den Briel et al. 2004), which allows to add the cost of an action within its definition. The compilation is, however, different from our in the sense that it generates actions that are all applicable in states where all preferences are met, and actions that have cost may be inappropriately included in the plan at such states (from (Benton, Kambhampati, and Do 2006)), while for us this is not the case, by negating (soft) preconditions in one of the added actions. This would mean that, in principle, the PSP compilation could produce incorrect metric values, that have to be adjusted. Moreover, we also provide a characterization of the metric functions defined on states, thus closer to the one of PDDL3.0 . The works in (Brafman and Chernyavsky 2005; Giunchiglia and Maratea 2007) deal with our same problems, i.e., optimality at fixed makespan, but they are targeted for preferences defined qualitatively ((Brafman and Chernyavsky 2005) is based on CP-nets (Boutilier et al. 2004), and (Giunchiglia and Maratea 2007) deals only with unary weights in case of quantitative preferences, like (Büttner and Rintanen 2005), where, using a SAT-based approach, they solve the problem of finding parallel plans with as few actions as possible). (Ray and Ginsberg 2008) provides theoretical results for the framework presented in (Giunchiglia and Maratea 2007). The problem of planning with "action costs", also introduced as a requirement in IPC-6, has been dealt with in, e.g., (Keyder and Geffner 2008; Chen, Lv, and Huang 2008), and the approach in (Edelkamp and Kissmann 2009) is one of the best, as witnessed by the results of the IPC-6. Regarding the use of (partial, weighted) Max-SAT in planning, other than (Xing, Chen, and Zhang 2006b), in (Yang, Wu, and Jiang 2007) it is used in case-based planning, while in a recent paper (Russell and Holden 2010) to handle goal utilities dependencies.

Conclusions and future works

In this paper we have presented a PB-based approach for solving planning problems from the "SimplePreferences" category of the IPC-5, which involves preferences on actions preconditions and/or goals. At the best of our knowledge, this is the first time that a PB approach is implemented, comparatively evaluated, and the correctness is proved, in this context. Given that in the IPC-5 the impact of preferences violation on the plan metric is restricted to be linear, all results can be equivalently obtained using an approach based on partial weighted Max-SAT problems. An implementation based on this approach shows that our ideas are viable, and allow to widening the range of benchmarks that satisfiability-based approaches can effectively deal with. Future works include the ability of solving other IPC-5, and IPC-6 benchmarks, within the framework. Regarding IPC-6, in the research note (Keyder and Geffner 2009), it is shown that IPC-6 benchmarks, i.e., STRIPS planning with soft goals and action costs, can be efficiently reduced to STRIPS problem with action costs.

References

Benton, J.; Do, M. B.; and Kambhampati, S. 2009. Anytime heuristic search for partial satisfaction planning. *Artif. Intell.* 173(5-6):562–592.

Benton, J.; Kambhampati, S.; and Do, M. B. 2006. 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`.

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

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

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

Chen, Y.; Huang, R.; Xing, Z.; and Zhang, W. 2009. Long-distance mutual exclusion for planning. *Artificial Intelligence* 173(2):365–391.

Chen, Y.; Lv, Q.; and Huang, R. 2008. Plan-A: A cost-optimal planner based on SAT-constrained optimization. In *IPC-6. Available at* `http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=Plan-A.pdf`.

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

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

Fikes, R., and Nilsson, N. J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3-4):189–208.

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

Gerevini, A., and Serina, I. 1999. Fast planning through greedy action graphs. In *Proc. of AAAI/IAAI*, 503–510.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the 5th IPC: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence* 173(5-6):619–668.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research* 20:239–290.

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

Heras, F.; Larrosa, J.; and Oliveras, A. 2008. MiniMaxSat: A new weighted Max-SAT solver. *Journal of Artificial Intelligence Research (JAIR)* 31:1–32.

Hoffmann, J., and Edelkamp, S. 2005. The deterministic part of IPC-4: An overview. *Journal of Artificial Intelligence Research* 24:519–579.

Hsu, C.-W.; Wah, B. W.; Huang, R.; and Chen, Y. 2006. New features in SGPlan for handling preferences and constraints in PDDL3.0. In *5th Internation Planning Competition Booklet, pages 39-41. Available at* `http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf`.

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

Jackson, P., and Sheridan, D. 2005. Clause form conversions for boolean circuits. In Hoos, H. H., and Mitchell, D. G., eds., *Proc. of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT 2004)*, volume 3542 of *Lecture Notes in Computer Science*, 183–198. Springer.

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

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

Kautz, H., and Selman, B. 2006. 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`.

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

Keyder, E., and Geffner, H. 2009. Soft goals can be compiled away. *Journal of Artificial Intelligence Research* 36:547–556.

Manquinho, V. M., and Marques-Silva, J. P. 2006. On using cutting planes in pseudo-Boolean optimization. *Journal on Satisfiability, Boolean Modeling and Computation* 2:209–219.

Marques-Silva, J., and Manquinho, V. M. 2008. Towards more effective unsatisfiability-based maximum satisfiability algorithms. In Büning, H. K., and Zhao, X., eds., *Proc. of 11th International Conference on Theory and Applications of Satisfiability Testing (SAT 2008)*, volume 4996 of *Lecture Notes in Computer Science*, 225–230. Springer.

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

Ramirez, M., and Geffner, H. 2007. Structural relaxations by variable renaming and their compilation for solving mincostsat. In Bessiere, C., ed., *Proc. of 13th International Conference on Principles and Practice of Constraint Programming*, volume 4741 of *Lecture Notes in Computer Science*, 605–619. Springer.

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

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence* 170(12-13):1031–1080.

Russell, R., and Holden, S. 2010. Handling goal utility dependencies in a satisfiability framework. In *Proc. of ICAPS 2010. To appear.*

Sheini, H. M., and Sakallah, K. A. 2005. Pueblo: A modern pseudo-boolean sat solver. In *Proc. of Design, Automation and Test in Europe Conference and Exposition (DATE 2005)*, 684–685. IEEE Computer Society.

Tseitin, G. 1970. On the complexity of proofs in propositional logics. *Seminars in Mathematics* 8.

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

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

van den Briel, M.; Kambhampati, S.; and Vossen, T. 2006. Planning with preferences and trajectory constraints through integer programming. In *Proc. of the ICAPS Workshop on Planning with Preferences and Soft Constraints*, 19–22.

Xing, Z.; Chen, Y.; and Zhang, W. 2006a. 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`, 53–55.

Xing, Z.; Chen, Y.; and Zhang, W. 2006b. Optimal strips planning by maximum satisfiability and accumulative learning. In Long, D.; Smith, S. F.; Borrajo, D.; and

McCluskey, L., eds., *Proc. of the 16th International Conference on Automated Planning and Scheduling (ICAPS 2006)*, 442–446. AAAI.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence* 171(2-3):107–143.