Planning as Satisfiability with IPC Simple Preferences and Action Costs *

Marco Maratea [a,**]

[a] *DIBRIS, University of Genova, Viale F. Causa 15, Genova, Italy*
E-mail: marco@dist.unige.it

Planning as Satisfiability (SAT) is currently the best approach for optimally (wrt makespan) solving classical planning problems and the extension of this framework to include preferences is nowadays considered the reference approach to compute "optimal" plans in SAT-based planning. It includes reasoning about soft goals and plans length as introduced in the 2006 and 2008 editions of the International Planning Competitions (IPCs). Despite the fact that the planning as satisfiability with preferences framework has helped to enhance the applicability of the SAT-based approach in planning, the actual approach used within the framework somehow suffers from some main limitations: the metrics, i.e. linear optimization functions defined over goals and/or actions, which account for plan quality issues, are fully reduced to SAT formulas, further increasing the size of (often already) big formulas; moreover, the search for optimal solutions is performed by forcing a heuristic ordering.

In this paper we address these issues by reducing the IPC planning problems with soft goals (from IPC-5) and/or action costs (from IPC-6) to optimization problems extending SAT and that can naturally handle the integer "weights" of the metrics, i.e. to Max-SAT and Pseudo-Boolean (PB) problems. Our idea is partially motivated by the approach followed by IPPLAN in the deterministic part of the IPC-5 and by the recent availability of efficient Max-SAT and PB solvers. First, we prove that our approach is correct; then, we implement these ideas in SATPLAN and run a wide experimental analysis on planning problems from IPC-5 and IPC-6, taking as references state-of-the-art planners on these competitions and the previous SAT-based approach. Our analysis shows that our approach is competitive and helps to further widen the set of benchmarks that a SAT-based framework can efficiently deal with. At the same time, as a side effect of this analysis, challenging Max-SAT and PB benchmarks have been identified, as well as the Max-SAT and PB solvers performing best on these planning problems.

Keywords: Planning, satisfiability.

1. Introduction

Planning as Satisfiability (SAT) [KS92] is currently the best approach for optimally (wrt makespan) solving classical planning problems. The SAT-based planner SATPLAN [KS99,KS06] was the winner of the deterministic track for optimal planners in the 4th International Planning Competition (IPC-4)[1][HE05] and co-winner in the IPC-5[2][GHL+09] (together with another SAT-based planner, MAXPLAN [XCZ06a,CXZ07]). Then, the work on satisfiability planning has mainly focused on enhancing the efficiency of the SAT-based approach by e.g. improved encodings [RHN06,CHXZ09, RGPS09,HCZ10] and the exceptions to this trend are limited to particular forms of preferences and plan quality measures (e.g., soft goals with uniform costs [GM07], minimum-length plans [BR05], action costs [RG07,CLH08]). In this context the recent extension of the planning as satisfiability framework to include preferences [GM07,GM11] is nowadays considered the reference approach to compute "optimal" plans in SAT-based planning, which includes reasoning about soft goals and plan length as (part of the features) introduced in IPC-5 and IPC-6.[3] It helped to enhance the applicability of the SAT-based approach in planning, by allowing to deal with plan quality issues other than the makespan, outside the "traditional" domains of SAT-based technologies.

Despite the previously mentioned success story, the actual approach employed within the framework suf-

*This paper is an extended and revised version of [GM10]. The author would like to thank Enrico Giunchiglia for fruitful discussions and comments on the topic of the paper. He also would like to thank Josep Argelich, Han Lin and Peter Kissmann for providing, and getting support to, their solvers. Alessandro Saetti and Malte Helmert are, instead, thanked for their help with IPC-5 and IPC-6 benchmarks, and Menkes van den Briel for insights on the approach based on Integer Programming.

**Corresponding author

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

AI Communications
ISSN 0921-7126, IOS Press.

fers from the following limitations, somehow diminishing its value and applicability: (i) the metric of the problem, i.e. an objective function (limited to be linear in IPC-5 and IPC-6 benchmarks) defined over the goal and/or action variables of the problem for taking into account plan quality, is fully reduced to a SAT formula, and (ii) the solving method is based on imposing an ordering on the heuristic of the underlying SAT solver, to be followed while branching, which can cause some problems [JJN05].

In this paper we address these issues by reducing planning problems at fixed makespan to optimization problems providing a more natural and concise representation, e.g. by natively handling integer coefficients of the metrics[4], i.e. to Max-SAT[5] and Pseudo-Boolean (PB) problems. Our idea is partially motivated by the approach employed by IPPLAN [vdBK05,vdBKV06b] in the deterministic part of the IPC-5 (further improved and shown effective in [vdBVK08], but only in the optimal planning case). This approach reduces STRIPS problems to 0-1 Integer Programming (IP) problems and then calls CPLEX [ILO02]. It is also motivated by the recent availability of efficient Max-SAT and PB systems, owing to Evaluations and Competitions[6] held during the last few years. In particular, we consider all domains in the "SimplePreferences" track of the IPC-5 to have preferences defined on action preconditions and/or goals, and the STRIPS domains with "simple" action costs in the "net benefit" optimization track of the IPC-6, i.e. with a single global cost, which is increased by a positive integer when actions with costs are executed. Given that our idea is to rely on classical SAT-based encodings of STRIPS problems to generate optimization problems, non-STRIPS[7] IPC-5 and IPC-6 problems are compiled into STRIPS [FN71] problems with a compilation technique similar to that one used in [BKD06] (based on a technique for dealing with conditional effects presented in [GK97]), and techniques used in the FF pre-processor [HN01]. Planning benchmarks are then reduced to (a series of) Weighted Partial Max-SAT and PB problems: the first one is from the categories of the recent Max-SAT Evaluations, while for PB the resulting reduction falls into the "OPT-SMALL-INT" category of the PB evaluations, which restricts PB problems to having (i) no constraint with a sum of coefficients greater than 2^{20} (20 bits), and (ii) linear objective functions. With (ii) PB problems correspond to 0-1 IP problems. A modeling of IPC-5 planning problems with preferences, expressed through the PDDL3 [GHL$^+$09] (and [McD00] for its original version) language, in 0-1 Integer Programming has been presented in [vdBKV06a]. However no implementation and experimental analysis are provided,[8] as well as no formal results.

First, we prove that our approach is correct and returns plans with optimal plan metrics, at fixed makespan, i.e. such that there is no plan with a "better" metric at this makespan. Then, we implement our ideas in SATPLAN and run a wide experimental analysis on planning problems from IPC-5 and IPC-6, taking as references the state-of-the-art planners SGPLAN [HWHC06,HWHC07] and GAMER [EK09], i.e. the winners of the "SimplePreferences" track of the IPC-5 and of all optimizations tracks of the IPC-6, and the previous SAT-based proposal SATPLAN(P) [GM11]. Differently from our proposal and [GM11], SGPLAN and GAMER can find sequential plans with unbounded horizon, but they were the clear winners of the competition tracks of interest, and thus are used as references. Our analysis shows that our approach is competitive and helps to widen the set of benchmarks that can be dealt with efficiently using SAT-based technology, at the same time relying on a more natural representation of the planning problem at hand. We also evaluate the anytime performance of our planner, by not stopping at the optimal makespan but letting the planner run for all the allotted time limit: the quality of the plans returned significant increases, approaching the results of state-of-the-art planners SGPLAN and GAMER.

As a side effect of our analysis, we identify the solvers performing best on these planning problems, as well as challenging Max-SAT and PB benchmarks.

The paper is structured as follows. First, we present some basic preliminaries about planning (as satisfiability), Max-SAT and PB problems in Section 2. In Section 3 we show how we model the problems of interest as PB/Max-SAT problems. We go on showing the

[4] In IPC-5 and IPC-6 benchmarks not all weights applied to goals violation and/or action costs are integers. Nonetheless, without loss of generality, we can consider all weights to be integers.

[5] A Max-SAT formulation has been already used in the context of optimal STRIPS planning [XCZ06b], but with a different usage, i.e. to minimize "directly" the number of time stamps (i.e., the makespan) instead of using the basic incremental scheme as in the original SATPLAN algorithm.

[6] See `http://maxsat.ia.udl.cat/` and `http://www.cril.univ-artois.fr/PB11/` for the last editions.

[7] Some constructs, e.g., the ADL construct ":preferences" in IPC-5 and ":actions-costs" requirements in IPC-6, are used.

[8] This fact has been confirmed by personal communications with Menkes van den Briel.

new solving algorithms in Section 4, along with some formal results, whose implementation and experimental evaluation is presented in Section 5. The last part of the paper discusses related work in Section 6 and draws some conclusions in Section 7.

2. Preliminaries

A *fluent* is a propositional variable that encodes information about the state of the world. Let $\mathcal{F}$ be the set of all fluents, i.e. the fluent signature. A *state* is an interpretation of the fluent signature. An *action* is a propositional variable that corresponds to an operator that can change the state of the world. Let $\mathcal{A}$ be the set of all actions, i.e., the action signature. A *complex action* α is an interpretation of the action signature and models the concurrent execution of the actions satisfied by α, i.e. it is a set of actions that can be executed in parallel.

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

- I is a Boolean formula over $\mathcal{F}$ (more precisely, a conjunctions of fluents thus having exactly one satisfying assignment) 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*.

Given that our focus is on classical planning, we thus 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;
- tr_i is the formula obtained from tr by substituting each symbol $p \in \mathcal{F} \cup \mathcal{A}$ with p_{i-1} and each $f \in \mathcal{F}'$ with f_i.

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

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

and a *plan* is an interpretation (or, equivalently, a set of literals) satisfying (1). [9]

A Boolean formula is in Conjunctive Normal Form (CNF) if it is a set of clauses, a clause being a set of literals. Given a Boolean formula ϕ, we can always produce an equisatisfiable CNF formula efficiently, i.e. in linear time in the size of ϕ, by introducing additional variables, see, e.g. [Tse70,PG86,JS05]. An assignment π is a consistent set of literals. An assignment π corresponds to the partial interpretation mapping to true the literals $l \in \pi$. Given a formula ϕ, we say that ϕ is *satisfiable* if there exists a *satisfying* assignment π for ϕ.

Consider a CNF formula $\varphi := \varphi_h \cup \varphi_s$, where φ_h and φ_s define the set of *hard* and *soft* clauses respectively. Hard clauses *must* be satisfied, while soft clauses do not need to be satisfied, but their satisfaction is preferred. A Max-SAT problem is defined as having $\varphi_h = \emptyset$, and the goal is to find an assignment satisfying as many as possible of the clauses in φ_s. In a SAT problem, $\varphi_s = \emptyset$. The Partial Max-SAT problem is an extension of the Max-SAT problem where there are both hard and soft clauses: in this case the goal is to find an assignment satisfying all the clauses in φ_h and as many as possible of the clauses in φ_s. The Weighted Partial Max-SAT problem is a further extension of the Max-SAT problem: in order to characterize the problem, consider a function w that assigns a positive integer to each clause in φ_s. Thus the goal of the Weighted Partial Max-SAT problem is to find an assignment π satisfying all the clauses in φ_h and maximizing

$$\sum_{C \in \varphi_s : \pi \models C} w(C) \tag{2}$$

i.e. the sum of the weights of satisfied soft clauses.

In a (linear) PB optimization problem, a PB constraint extends a CNF clause to possibly contain integer coefficients (c_i), variables (x_i) truth/falsity is interpreted as $0/1$ and there is a bound (b) on the value the constraint can assume, i.e. linear PB constraints are of the form

[9] In the following, we can switch between plans and satisfying interpretations, intuitively having the same meaning.

$$\sum_i c_i \, x_i \geq b \tag{3}$$

A PB formula is a conjunction of PB constraints. Moreover, an objective function can be applied[10] to the problem. If such an 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., satisfies all PB constraints) and minimizes the objective function. In the objective function, coefficients may be negative and thus minimization/maximization are exchangeable and will be used both later on.

3. Modeling IPC problems as Boolean optimization problems

Given that SATPLAN can only handle STRIPS problems while IPC-5 and IPC-6 benchmarks can contain constructs to represent preferences and action costs, the overall reduction is, in general, carried out in two steps. The first step adapts the original problem to compile preferences away.

Here we need a more formal definition of actions. An action A is a triple $\langle HPre, SPre, Eff \rangle$ where $HPre$, $SPre$ and Eff are Boolean formulas representing hard preconditions, soft preconditions and effects of A, denoted by $HPre(A)$, $SPre(A)$ and $Eff(A)$ respectively. The set G of goals is partitioned into *hard* and *soft* goals represented by the Boolean formulas HG and SG respectively. w is a function mapping actions with soft preconditions and goals to positive integer numbers. In the following $\top$ denotes the empty formula.

The semantic is defined as follows [GHL$^+$09]: A can be executed even if $SPre$ is not satisfied, but then a cost is paid each time this happens. Hard goals HG need to be satisfied while soft goals are not mandatory, but if a soft goal g is reached a reward $w(g)$ is gained.

Each action A having soft preconditions (i.e. having $SPre(A) \neq \top$)[11] is split into two actions:

- A_1:=$\langle HPre(A) \wedge SPre(A), \top, Eff(A) \rangle$; and
- A_2:=$\langle HPre(A) \wedge \neg SPre(A), \top, Eff(A) \wedge goal_A \rangle$

A_1 and A_2 are mutually exclusive: the second takes into account whether the original action A is executed with its soft preconditions not satisfied and the introduced effect ($goal_A$) takes into account this fact, and $w(A_2) = w(A)$. This technique of splitting actions is similar to the ideas employed in [KG09] for STRIPS problems.

Then, for each goal $g \in SG$ we define a (dummy) action whose precondition is the goal, and the effect is a (dummy) literal, i.e.

$$A_g := \langle g, \top, goal_{Ag} \rangle$$

This is inspired by the approaches in [GK97,BKD06].

All added actions can be non-STRIPS in general: given that we want to rely on classical STRIPS satisfiability planning encoding to generate Boolean optimization formulas we exploit classical methods (i.e. the ones in FF's pre-processor [HN01]) to compile into STRIPS actions. At this point we have a STRIPS formulation of our problem. G corresponds to HG.

Given a STRIPS planning problem Π and a makespan n, Boolean optimization formulas, in particular PB formulas, are defined by

1. the clauses of classical SAT-based encodings, expressed as PB constraints; and
2. an optimization function.

Regarding 1.,

– for the set of literals $\{l_1, \ldots, l_m\}$ of the initial state I the following PB constraints are added: for each $j = 1 \ldots m$

$$sign(l_j) \; var(l_j)_0 \geq b$$

where $var(l)$ returns the (fluent) variable the literal l is built on, $sign(l)$ is 1 if the literal l is positive, and -1 otherwise, and b is 1 if l is positive, and 0 otherwise.

– for each $i = 0 \ldots n-1$, action executability at time i is encoded in SAT as follows

$$A_i \rightarrow \bigwedge_{p \in HPre(A)} p_i \wedge \bigwedge_{l \in Eff(A)} l_{i+1} \tag{4}$$

where p is an atom, l a literal, and $HPre(A)$ (resp. $Eff(A)$) are now (equivalently) considered as the set of preconditions (resp. effects) of A. Thus, for each STRIPS action A the PB formula will contain

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

[11] For simplicity, we restrict to action having at most one soft precondition formula: this is the case for all instances in the "SimplePreferences" track of IPC-5. In general, we have to consider their power set.

* for each $p \in HPre(A)$ a PB constraint of the form

$$-A_i + p_i \geq 0$$

* for each $l \in Eff(A)$ a PB constraint of the form

$$-A_i + sign(l)\ var(l)_{i+1} \geq b$$

and now b is 0 if l is a positive literal and -1 otherwise.

The transition relation is composed of other PB constraints, i.e. the ones corresponding to the clauses arising from *frame* and *exclusion* axioms.

– for G (now equivalently considered as a set of clauses), the following groups of PB constraints are added: let $g = \{l_1, \ldots, l_k\}$ be a set of literals, for each $g \in HG$ the PB constraint

$$\sum_{i=1}^{k} sign(l_i)\ var(l_i)_n \geq 1 - neg(g)$$

where $neg(g)$ is the number of negated literals in g; and

Consider a plan π and let SA be the set of all actions having soft action preconditions. In the formulation above, if $A \in SA$ we consider all its instantiations A_i, $i = 0 \ldots n-1$ to be in SA.

The metric of the problem is defined as follows

$$max : \sum_{g \in SG, \pi \models g} w(g) - \sum_{A_i \in SA, A_{2i} \in \pi} w(A_2)$$

IPC-6 benchmarks contain also action costs, i.e. a cost associated with its execution. Considering action costs in this work we restrict ourselves to what we called "simple" action costs, i.e. we consider problems with a single total cost increased by a positive integer number if an action is executed. In IPC-6 benchmarks this fact is expressed by using a construct of the form

(increase (total-cost) (stack-cost)) (5)

as effect of an action A. The semantic of (5) is simple: if A is executed "*total-cost*" is increased by "*stack-cost*". Both *total-cost* and *stack-cost* are initialized in the initial state. A dummy literal is added in place of (5) as effect of A (*goal-act*). w is extended to be defined over the set of actions with action costs, called AC: the metric of the problem becomes

$$max : \sum_{g \in SG, \pi \models g} w(g) - \sum_{A_i \in \pi, A_i \in AC} w(A) - \sum_{A_i \in AS, A_{2i} \in \pi} w(A_2)$$

A Weighted Partial Max-SAT formulation is also possible. Consider Π_n as a set of clauses, the formulation is a pair

$$\langle \Pi_n \wedge pref(\Pi_n), w' \rangle \quad (6)$$

where $pref(\Pi_n)$ is the formula

$$\wedge_{g \in SG} (\vee_{l \in g} l) \wedge \wedge_{A_i \in AC} \neg A_i \wedge \wedge_{A_i \in SA} \neg A_{2i}$$

that encodes the metric, and where w' is a (partial) function mapping clauses to positive integer numbers. w is defined as follows: a soft clause corresponding to

– a soft goal g has the related weights $w(g)$;
– for each $i = 0 \ldots n-1$
  * the (non-) execution of an action $A \in AS$ with its soft precondition not satisfied has a weight $w'(\neg A_{2i}) = w(A_2)$; and
  * the (non-) execution of an action $A \in AC$ has a weight $w'(\neg A_i) = w(A)$.

3.1. Example

We consider instance #1 of the Traveling and Purchase Problem (TPP) domain containing preferences on both action preconditions and goals, "Grounded-Preferences" variant (referred as "tpp1" below). In the following we show the part of interest and how they have been modeled. In the tpp1 instance there is an action "drive" 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)))
```
(7)

(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))
   ...
))                                                    (8)

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

For the preference $p4A$ we introduce the following action

```
(:action dummy-p4A
:parameters ()
:precondition (and
      (ready-to-load goods3 market1 level0)          (9)
      (loaded goods3 truck1 level0))
:effect (and (goal-p4A))).
```

and similarly for the other soft goals.

Regarding the action precondition we split action "drive" into two actions as we have explained before

```
(:action drive
:parameters (?t - truck ?from ?to - place)
:precondition (and
      (at ?t ?from) (connected ?from ?to)
      (ready-to-load goods1 ?from level0)            (10)
      (ready-to-load goods2 ?from level0)
      (ready-to-load goods3 ?from level0))
:effect (and (not (at ?t ?from)) (at ?t ?to)))

(: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)              (11)
    (ready-to-load goods2 ?from level0)
    (ready-to-load goods3 ?from level0))))
:effect (and (not (at ?t ?from)) (at ?t ?to)
          (goal-p-drive)))
```

The new goal of the problem is the conjunction of the dummy literals related to goal preferences and action preconditions and costs introduced (all goals are soft in this instance).

The resulting problem is then given in input to the ADL2STRIPS tool to be compiled into a STRIPS problem. As a consequence the (dummy) actions introduced are compiled into STRIPS actions. There could be (multiple) STRIPS actions in place of those in the original formulation.

Consider the simple case in which a single STRIPS action is in place of each action and consider that the related STRIPS action has the same name. With reference to our working example and with fixed makespan n, for each $0 \leq j \leq 4$ the PB constraints resulting from action executability are

$$\wedge_{p \in HPre(\textit{dummy-pjA})} -\textit{dummy-pjA}_{n-1} + p_{n-1} \geq 0 \quad (12)$$

$$-\textit{dummy-pjA}_{n-1} + \textit{goal-pjA}_n \geq 0 \quad (13)$$

Then for each i, $1 \leq i \leq n$, regarding action drive, the following PB constraints are added because of its precondition

$$\wedge_{p \in HPre(drive)} \; -drive_i + \; p_i \geq 0 \quad (14)$$

while the following constraints are added for its effects

$$\wedge_{l \in Eff(drive)} -drive_i + sign(l)\; var(l)_{i+1} \geq b \quad (15)$$

where b is 0 if the literal is positive and -1 otherwise. The last PB constraint added is

$$-drive_i + \; \textit{goal-p-drive}_{i+1} \geq 0 \quad (16)$$

Similarly for action $dummy_{dr}$ from (11). Now the question remains of how to express the optimization function: in (8) the idea is to minimize the violation of preferences (expressed with (*is-violated p*) in PDDL3 having the following meaning: given a preference p, (*is-violated p*) takes value 1 if the preference is not satisfied and 0 otherwise [GHL+09]). 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 (8) can be expressed using both fluents and actions, i.e. with the (linear) optimization functions (17) and (18), to be maximized, where π is a satisfying interpretation, and $\pi(p)$ is 1 if p is true and 0 otherwise. If actions are used, the weights are associated with action executions. The characterization with fluents is instead more similar to PDDL3 syntax, where the metric is mostly defined on states (but for action costs).

$$\textstyle\sum_{j=0}^{4} 2^j \; \pi(\textit{goal-pjA}) - \sum_{i=1}^{n} \pi(\textit{goal-p-drive}_i) \quad (17)$$

$$\textstyle\sum_{j=0}^{4} 2^j \; \pi(\textit{dummy-pjA}) - \sum_{i=0}^{n-1} \pi(\textit{dummy}_{dr,i}) \quad (18)$$

$\sum_{j=0}^{4} 2^j$ *π(goal-pjA) − π(goal-p-drive)* (19)

In general, action preconditions and costs can hold at any time stamp. If we know that instead actions can be only executed once, we can add a single fluent *goal-p-drive* for all instantiations and the optimization function is expressed with (19) (similar changes apply to (18)). Even if on the one hand this hypothesis on (ground) action executions underlying (19) can be seen as a further approximation (other than the makespan) of the (unbounded) optimal metric, such a hypothesis holds in various cases, e.g. on a classical, real-world planning domain like blocks-world and logistics.

3.2. Algorithms for finding optimal plans

Now we define approaches for finding "optimal" plans, compiling the problem at fixed makespan into an optimization problem as shown before. Consider a STRIPS problem Π, and a makespan n.
In the following figures

1. *cnf*(φ), where φ is a formula, is a set of clauses such that
   - for any interpretation π' in the signature of *cnf*(φ) such that $\pi' \models$ *cnf*(φ) it is true also that $\pi \models \varphi$, where π is the interpretation π' but restricted to the signature of φ; and
   - for any interpretation $\pi \models \varphi$ there exists an interpretation π', $\pi' \supseteq \pi$, such that $\pi' \models$ *cnf*(φ).

   There are well-known methods for computing *cnf*(φ) in linear time by introducing additional variables, e.g., [Tse70,PG86,JS05];
2. *cnf2pb*(φ, P, w) is *cnf*(φ) and then following Section 3
   a. each clause C in *cnf*(φ) is expressed as the corresponding PB constraint, and
   b. the optimization function is built;
3. *cnf2wcnf*(φ, P, w, top) corresponds to Eq. (6);
4. PBO and PWMAXSAT are generic Weighted Partial Max-SAT and PB solvers: differently from SAT, in these research fields there is no a single underlying basic algorithm like DPLL for SAT[12], but a wide range of effective approaches exist.

[12] In this case, instead of DPLL, we could refer to the CDCL (Conflict Driven Clause Learning) algorithm [Mit05] given that SATPLAN(P) relies on MINISAT as SAT solver.

function QT-PLAN-PB(Π,P,w)
1 n := 1
2 **while** (true)
3 **if** (PBO(*cnf2pb*$(\Pi_n, P, w), \pi$))
4 **return** π
5 **else**
6 n := n + 1

Fig. 1. The algorithm of QT-PLAN-PB.

Assume now that PBO returns FALSE if a satisfying assignment does not exist, or TRUE with an optimal solution π, i.e. we assume that the underlying algorithm is sound and complete. Moreover, we assume that a plan always exists.

Now we are ready to state the following theorem.

Theorem 1 *Let* Π *be a planning problem,* n *a makespan, and let* $\langle P, w\rangle$ *be a quantitative preference.* QT-PLAN-PB*(*Π*,*P*,*w*) returns an optimal plan for* Π *wrt* $\langle P, w\rangle$ *at minimum makespan* n *for which a plan exists.*

Proof. Given ϕ^{PB}=*cnf2pb*(Π_n, P, w), from the assumptions on PBO we know that PBO(ϕ^{PB},P,w) returns

- FALSE if ϕ^{PB} is unsatisfiable, or
- an optimal solution π which maximizes (17) otherwise.

Given these, in order for the actual theorem to hold, we have first to show that for each assignment π in the signature of ϕ^{PB} such that π satisfies ϕ^{PB}, it must also hold that $\pi' \models \Pi_n$, where π' corresponds to π but reduced to the signature of Π_n, and for each assignment π' in the signature of Π_n such that $\pi' \models \Pi_n$, there exists an assignment π, $\pi \supseteq \pi'$, such that π satisfies ϕ^{PB}.

The point holds from

- the assumptions on *cnf* in 1. above ; and
- the assumption in $2a$ above about the one-to-one mapping between the clauses in *cnf*(Π_n) and the PB constraints in ϕ^{PB}, i.e. an assignment π satisfying $C \in$ *cnf*(Π_n) satisfies also the related constraints *cnf2pb*(C, P, w), and vice-versa.

The minimality of n holds simply from the iterative deepening approach of the algorithm in Fig.1.

□

function QT-PLAN-MAXSAT(Π,P,w,top)
1 n := 1
2 **while** (true)
3 **if** (PWMAXSAT(*cnf2wcnf*$(\Pi_n, P, w, top), \pi)$)
4 **return** π
5 **else**
6 n := n + 1

Fig. 2. The algorithm of QT-PLAN-MAXSAT.

Now we move to the Max-SAT formulation. Fig. 2 presents the solving procedure based on Max-SAT: in the algorithm top is defined as

$$top := \sum_{p \in P} w(p) + 1 \tag{20}$$

and it is used to define hard clauses in the Weighted Partial Max-SAT problem (details on the format are presented in the next section).

We assume that, similarly to PBO, the PWMAXSAT is sound and complete.

We are now ready to state the following theorem, similar to Theorem 1.

Theorem 2 *Let* Π *be a planning problem,* n *a makespan, and let* $\langle P, w\rangle$ *be a quantitative preference.* QT-PLAN-MAXSAT*(*Π,P,w,top*) returns an optimal plan for* Π *wrt* $\langle P, w\rangle$ *at minimum makespan* n *for which a plan exists.*

The proof is similar to Theorem 1.

Before ending the section, we would like to underline a few things. An alternative way to present our approach would be to rely on a unique characterization of the underlying solvers for Boolean optimization instead of using PWMAXSAT and PBO separately. There are encodings mapping Max-SAT to PB problems and vice-versa. In addition since the PB competition 2010 there has been even a track on "soft PB constraints' called "Weighted Boolean Optimization" making an attempt to combine the two problems (see also [MSP09]). However we preferred to keep the presentations separated: this is because the Max-SAT and PB formalisms are well-known and established in the literature with their own formats, and often the related systems use different techniques for solving the two problems.

4. Comparison to SAT-based encoding

In this section we compare the actual solutions with the framework of planning as satisfiability with preferences first presented in [GM07].

instance	PB		[GM11]	
	#VAR	#CL	#VAR	#CL
storage1	227	933	291	1321
storage2	541	3819	661	4554
storage3	728	8073	976	9642
storage4	902	22535	1212	24488
storage5	8789	92748	14340	128129
storage6	13725	235711	22821	294077
storage7	20646	350862	35915	449197
tpp1	763	4510	967	5819
tpp2	1031	6046	1331	7971
tpp3	1238	7572	1586	9805
tpp4	1438	9712	1810	12099
tpp5	3409	42247	4151	47070
tpp6	3736	47728	4548	53006
tpp7	4156	55852	5052	61676
tpp8	4590	64557	5598	71109
tpp9	8535	197084	10103	207374
tpp10	9022	215879	10670	226694
pegsol1	245	909	377	1602
pegsol2	245	909	635	3407
pegsol9	699	4120	831	4813
pegsol10	699	4120	1089	6618
pathways1	415	3583	450	3797
pathways17	15249	534526	16040	539775
trucks1	3113	47403	3182	47831
trucks2	8213	160654	8396	161800
trucks3	13374	375755	13467	376343
open1-ipc2008	1318	12664	1390	13084
open2-ipc2008	1936	24097	2029	24658
open1	3643	43983	3883	45453

Table 1
Sizes of the evaluated formulas. The pegsol and pathways domains contain more than 20 instances: for such domains we only show the data for the smallest and biggest instances in the pool.

The approach described in [GM07,GM11] is completely based on a compilation into a SAT problem: non-uniform weights, described by a non-constant function w, are dealt by reducing w to a SAT formula by using the encoding in [War98], to be conjoined with the SAT-based encoding of classical planning explained before.

Given that, this approach can further increase the sizes of the formulas to be evaluated. Moreover, from a knowledge representation viewpoint, integers are not naturally represented with SAT.

The last two columns of Table 1 report the number of variables and clauses of the first satisfiable resulting CNF formula. This is done for most of the instances evaluated in this work (whose details will be given in the next section).

In this work we are interested in optimization problems that can more naturally and in a concise way represent the problems of interest, from a knowledge representation point of view: PB and Max-SAT formalisms. The second and third columns of Table 1 report the number of variables and clauses of the resulting PB formula as defined in the previous section. We can note that, in general, the new encodings are not that much smaller than those in [GM11]: the size of the added formula depends on how many variables are involved in w, and on the coefficients. Several benchmarks are described by a function w having relatively few actions and goals involved, and with relatively low coefficients. On the other hand, there are cases, e.g. some pegsol instances, in which the approach in [GM11] produces formulas up to a factor of 3 (resp. 4) bigger in the number of variables (resp. clauses) than the approach in this paper. Moreover, as we said before, the approach in [GM11] relies on an ordering of the heuristic that can hit performance, while the optimization solvers employed do not rely on such heuristics.

5. Implementation and experimental evaluation

As we already mentioned in the introduction, we have evaluated the instances of all domains of the "SimplePreferences" track of the IPC-5 i.e. the TPP, Pathways, Storage, Trucks and Openstacks, with preferences grounded (when available), and the STRIPS domains of the "net benefit" optimization track of the deterministic part of the IPC-6 having "simple" (as defined in this paper) action costs. When preferences are expressed on action preconditions and/or costs we have used the optimization function (19).

At implementation level, we used (i) ADL2STRIPS for compiling non-STRIPS actions into STRIPS actions: ADL2STRIPS was introduced in IPC-4 and is based on FF's pre-processor [HN01], while in our analysis we employed the version used in IPC-5 based on LPG [GS02,GSS03,GSS08], and (ii) SATPLAN for the encoding: we modified SATPLAN at each makespan of the SATPLAN's approach, in order to implement QT-PLAN-PB/QT-PLAN-MAXSAT. Thus the main changes in SATPLAN were related to the adaptation of the plain CNF creation to the new formats.

Regarding the back-end solvers, we used the best solvers which participated in Max-SAT and PB evaluations and competitions, in particular to the Weighted Partial and OPT-SMALL-INT categories, of interest in our work. Specifically the solvers we used are: MINIMAXSAT ver. 1.0 [HLO08], based on MINISAT+ ver. 1.13, WMAXSATZ [LMMP09] ver. 2.5 submitted to the 2010 Max-SAT evaluation (a first version of the solver has been presented in [ALM07]), INCWMAXSATZ [LS07,LSL08],[13] MSUNCORE ver. 1.2 and ver. 4 [MSM08,MSP08]; MINISAT+ ver. 1.14 [ES06], GLPPB ver. 0.2 (by the same authors of PUEBLO [SS05]), as submitted to the 2007 evaluation,[14] BSOLO ver. 3.0.17 [MMS06], SAT4J ver. 2.1 and SCIPSPX ver. 1.2.0 [ABKW08].[15] MINIMAXSAT was the winner of the 2007 Max-SAT evaluation in the Partial Max-SAT category. WMAXSATZ and INCWMAXSATZ extend the MAXSATZ solver [LS07], winner of the Max-SAT category at the 2007 Max-SAT evaluation and themselves have achieved very good results in the 2009 Max-SAT evaluation. In fact, INCWMAXSATZ was the winner on random and crafted benchmarks of the Weighted Partial category and WMAXSATZ was the second and third best on the benchmarks of the same categories. MSUNCORE was the winner on industrial Max-SAT benchmarks at the 2010 Max-SAT evaluation. MINISAT+ was the solver able to prove unsatisfiability and optimality on a larger number of instances than all the other solvers which entered into the PB evaluation 2005 [MR06] and the best performing solver (together with BSOLO) also in the 2006 PB evaluation, category OPT-SMALLINT-

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

[14] `http://www.eecs.umich.edu/~hsheini/pueblo/`

[15] 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. For the ones downloaded, we have used the version submitted to the evaluations or the last available.

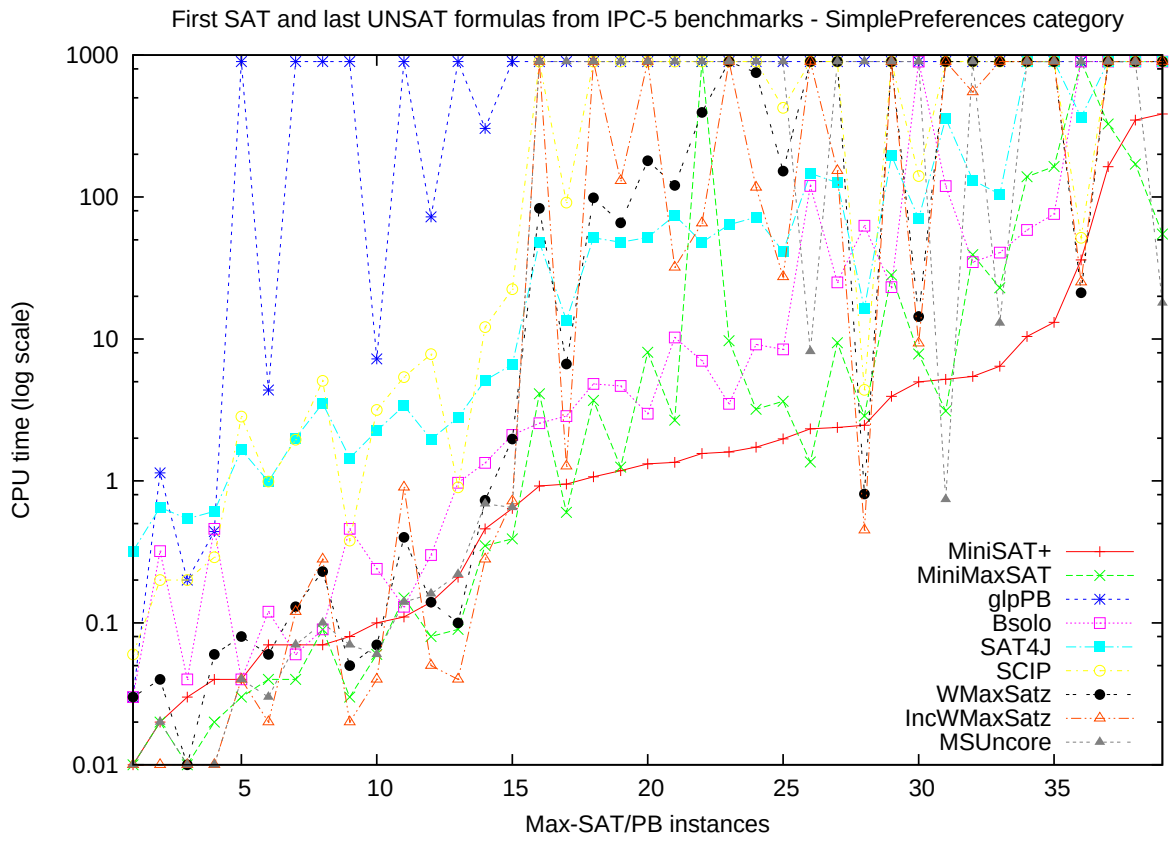

Fig. 3. Results for Weighted Partial Max-SAT and PB solvers on IPC-5 instances.

LIN. BSOLO and GLPPB have been the best performing PB solvers in the OPT-SMALLINT-LIN category of the 2007 PB evaluation. SCIPSPX and SAT4J were the winners of the OPT-SMALL-INT and Industrial Weighted Partial category of the respective evaluations in 2009.

As input format, MINIMAXSAT and SAT4J accept instances in the format of both Max-SAT and PB evaluations, and the best results are presented; MINISAT+ reads instances in the format of the PB evaluations with a minor exception, i.e. the symbol "*" has to be added between coefficients and variables like in the first evaluation. About MSUNCORE, we will not show results for both versions given that they perform very similarly on our benchmarks.

The timeout has been set to 900s on a Linux box equipped with a Pentium IV 3.2GHz processor and 1GB of RAM and memory limit to 900MB.

The next two subsections show detailed results about the IPC-5 and IPC-6 benchmarks respectively, and are organized as follows: at first it is presented an experimental analysis among all Max-SAT and PB solvers mentioned above on the first satisfiable and last unsatisfiable formulas of all instances created following the SATPLAN approach, in order to choose the "best" overall back-end system. Note that satisfiable instances are the vast majority in both cases. We will see that overall MINISAT+ (resp. MINIMAXSAT on problems expressed in PB format) is the best solver on IPC-5 (resp. IPC-6) benchmarks. The resulting planner is called PBPLAN.

For each planning domain both plan metrics of and CPU times to find plans are shown. In the analysis, we considered PBPLAN, SATPLAN(P) [GM11] and, as a reference, SGPLAN on IPC-5 benchmarks and GAMER on IPC-6 benchmarks.

5.1. IPC-5

Figure 3 reports the analysis mentioned above, on Max-SAT and PB instances, where results are pre-

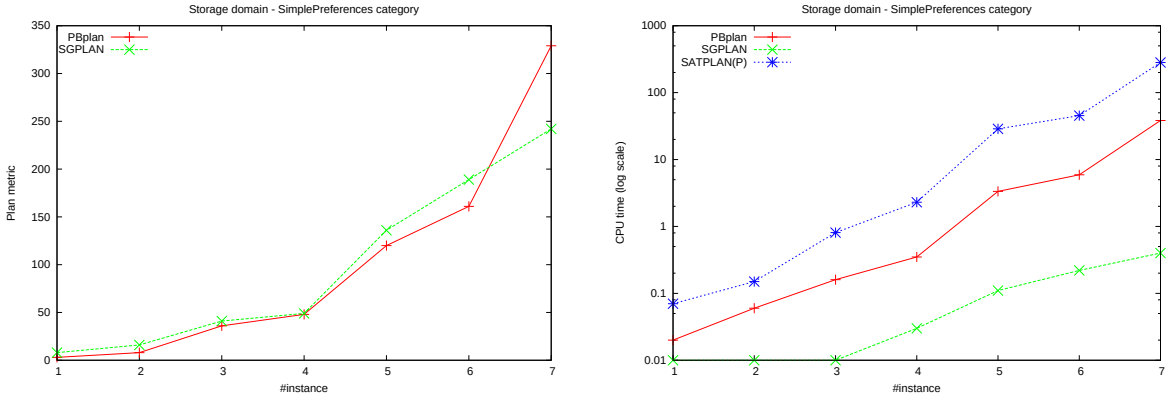

Fig. 4. Plan metrics (Left) and CPU time (Right) for PBPLAN, SGPLAN and SATPLAN(P) on Storage instances.

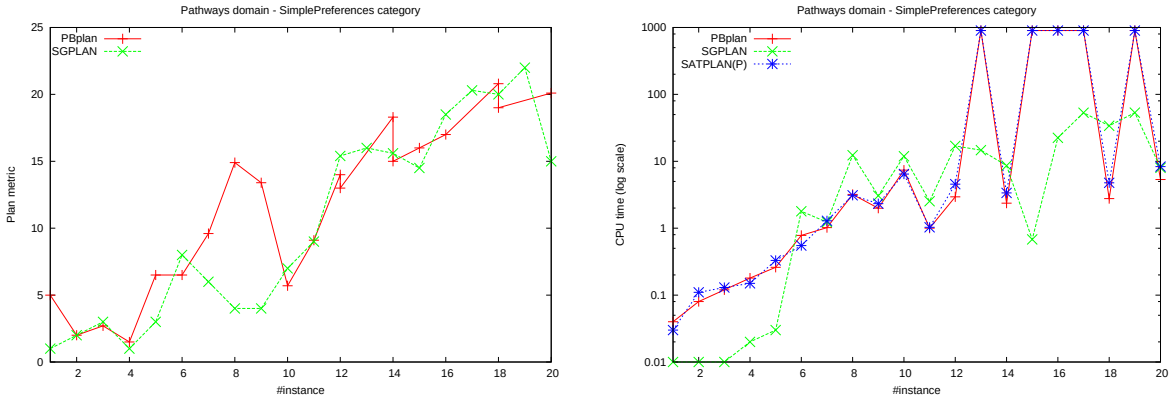

Fig. 5. Plan metrics (Left) and CPU time (Right) for PBPLAN, SGPLAN and SATPLAN(P) on Pathways instances.

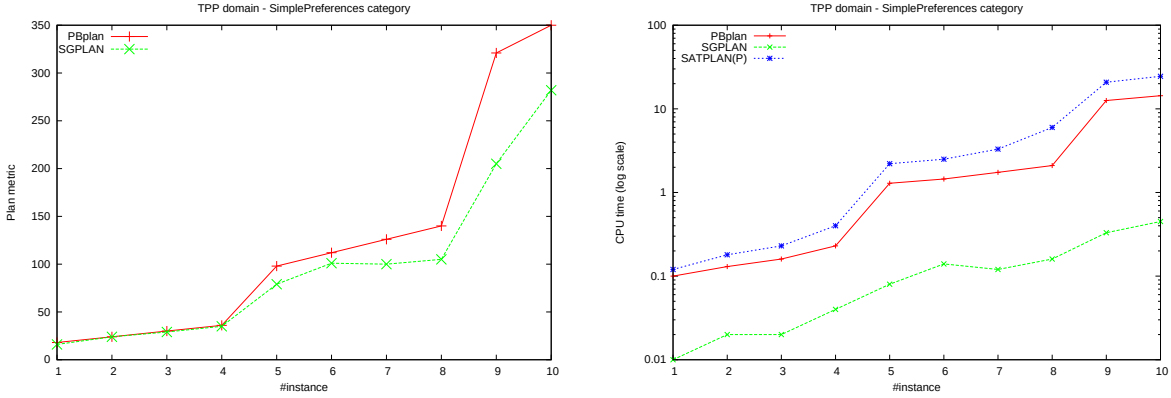

Fig. 6. Plan metrics (Left) and CPU time (Right) for PBPLAN, SGPLAN and SATPLAN(P) on TPP instances.

sented according to MINISAT+ performance, which is the best solver on these benchmarks, i.e. instances are put on x-axis according to increasing CPU times of MINISAT+, which is used as back-end of PBPLAN on IPC-5 problems. Other ways of presenting the results are possible, e.g. the one used in Max-SAT evaluations. We rely on the current form, which allows the results of all solvers on each single instance along the x-axis to be compared.

Results for the planning domains Pathways, Storage and TPP are instead presented as in the IPC-5 in Figures 4-6 in terms of both plan metrics and CPU time for PBPLAN, SATPLAN(P) and SGPLAN. About plan metrics, the results of PBPLAN also refer to the results of SATPLAN(P), and the goal here is to minimize the metric. The ones for the Trucks and Openstacks domains are only mentioned given that few instances could be compiled by ADL2STRIPS. Most of the pathways instances contain weights, related to goals violation, corresponding to real numbers: we made it integer numbers by multiplying each weight by 10^n, where n is the maximum number of (significant) decimal digits of the instance. 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 faster. Nonetheless it is added as reference, in particular for plan metrics, given it has been the clear winner at the IPC-5 on the track considered. Figure 4 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 more than 1 order of magnitude (Right), while PBPLAN is faster than SATPLAN(P) of almost 1 order of magnitude; PBPLAN solves the instances in less than (around) $30s$ while notably both PBPLAN and SATPLAN(P) have better plan metrics than SGPLAN on most instances. The results for the first 20 instances of the Pathways domain are presented in Figure 5. On these instances the CPU times in Figure 5 (Right) for PBPLAN, SATPLAN(P) and SGPLAN are comparable, except for 5 instances, which are only solved by SGPLAN (even if in tens of seconds). Results of PBPLAN and SATPLAN(P) are comparable, PBPLAN being slightly better on the biggest instances solved. Regarding plan metrics, in Figure 5 (Left), on the instances solved by all systems the results are comparable, except for few instances where SGPLAN (#1, #8, #9, #15 and #20) gives back plans of better quality. Among the instances from #21 to #30, not shown in Figure 5, PBPLAN and SATPLAN solve two instances, #23 and #29, in a few seconds and with plan metrics of 25.5 and 26.7 respectively. On the same instances, SGPLAN has metrics of 18 and 22, while it solves the remaining instances with a mean time of around $100s$. Results for the TPP domain are presented in Figure 6. On the 10 instances shown the behavior is similar in terms of CPU times to the Pathways domain, with PBPLAN having slightly better performance than SATPLAN(P) up to a factor of 2; the plan quality of SGPLAN is better than the one of PBPLAN (and SATPLAN(P)) in particular on instances #9 and #10. Instances from #11 to #15 can be compiled by ADL2STRIPS but not solved by both PBPLAN and SATPLAN(P): these instances contain more than 20000 variables and 800000 constraints, up to 3000000 constraints, and the solving time for SGPLAN is more than $100s$. Instances from #16 to #20 can not be compiled. About the last two domains Trucks and Openstacks, on the first 7 instances of the Trucks domain that can be compiled by ADL2STRIPS, only the first two instances can be solved by PBPLAN in 10 and 800 seconds, approximately, with plan metrics of 1 and 2. SATPLAN(P) solves only the first instance. The same two instances are solved very fast by SGPLAN, but with plan metrics of 13 and 52, thus much higher. On the same domain, finally note that for instances #3 to #7 even checking satisfiability of the first satisfiable instance is difficult for MINISAT. The same holds for the Openstacks domain, where only 1 instance is compiled by ADL2STRIPS.

5.2. IPC-6

We focus on the STRIPS domains of "net benefit" optimization track of the deterministic track at the IPC-6 that contain "simple" action costs, i.e. the pegsol and Openstacks[16] domains. We remind the reader that here the goal is to maximize the plan metric. All 30 instances of the pegsol domain can be compiled and evaluated, except for instances #7 and #8. About the Openstacks domain, only 1 instance can be solved by PBPLAN: note that this domain is overall the hardest for IPC-6 planners (according to the results of the competition considering the number of solved instances). Figure 7 reports the analysis on Max-SAT and PB instances: results are presented according to MINIMAXSAT performance, which is the overall best solver

[16]Precisely, the instances of this domain contains negated action preconditions, thus the ADL2STRIPS tool has still to be used.

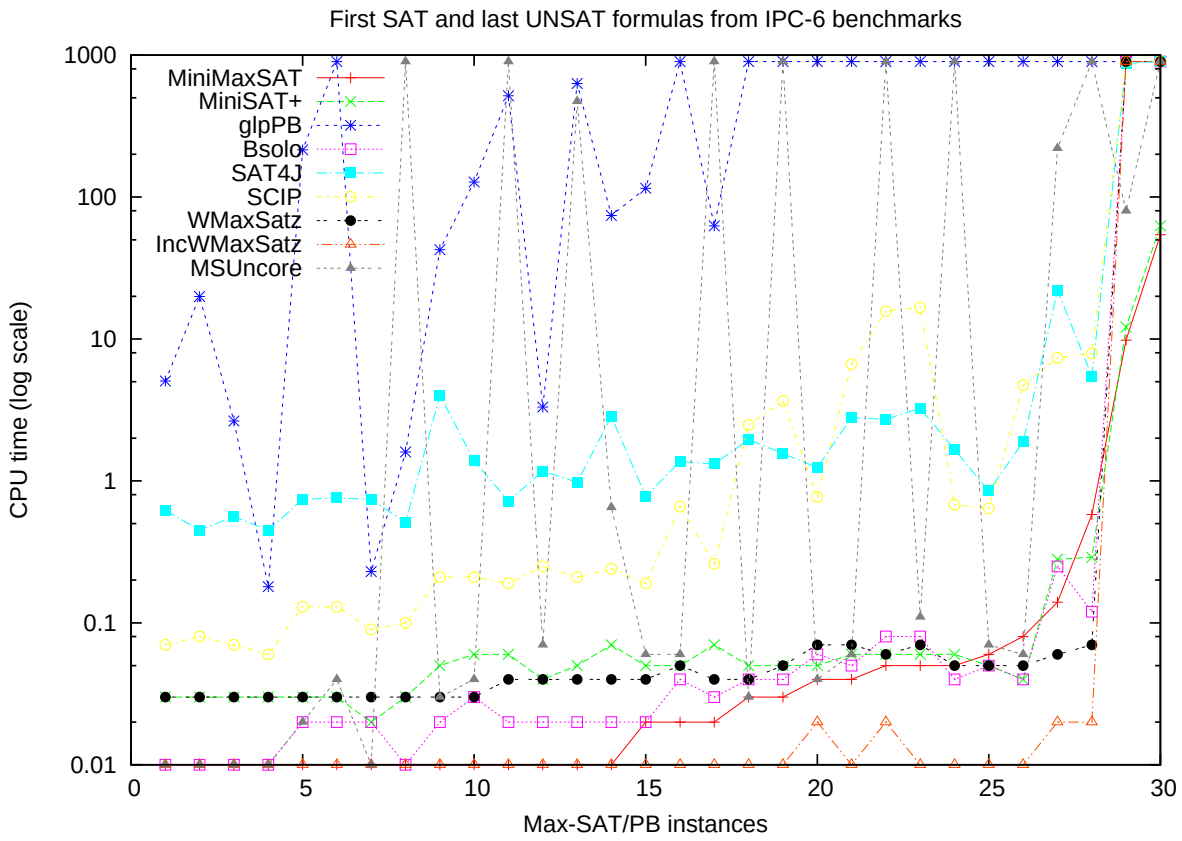

Fig. 7. Results for Weighted Partial Max-SAT and PB solvers on IPC-6 instances.

on these benchmarks[17] slightly better than MINISAT+ and it is used as back-end of PBPLAN on IPC-6 problems.

Results for the planning domain pegsol are presented in Figure 8. Here the results of SATPLAN(P) are not added for sake of readability: all instances are solved very easily by SATPLAN(P) and with results similar to PBPLAN. It is easy to see that on this domain PBPLAN is much faster than GAMER, while GAMER returns plans of much better quality. GAMER runs out of memory on the last instances (from #27 to #30), solved instead quite easily by PBPLAN.

[17]These results are reached with the PB formulation. Note that for most of the "simple" instances in the figure INCWMAXSATZ is the fastest, except for the (only) two "hard" instances in which INCWMAXSATZ runs out of memory. On these two instances MINIMAXSAT is the best and further solves the other instances in less than 0.5s. For these reasons MINIMAXSAT has been employed in this analysis.

For the Openstacks domain only 6 instances are solved by GAMER and only 1 by PBPLAN and SATPLAN(P), with plan metrics of 16 and 9 respectively. GAMER is also much faster on this instance (around 5 and 60 seconds respectively).

5.3. *Anytime results*

We then performed a further analysis with the focus on plan metrics. This is because the fixed makespan approach followed can significantly limit the quality of the plans returned by PBPLAN: considering an instance where there is a high gap between the plan quality of PBPLAN/SATPLAN(P) and GAMER, e.g. instance #24 of pegsol domain. This instance is solved at (optimal) makespan 3 with plan metric 44. Running the instance at makespan 4 leads to a plan metric of 105 thus much better than the previous and very close to the global optimal plan metric of GAMER.

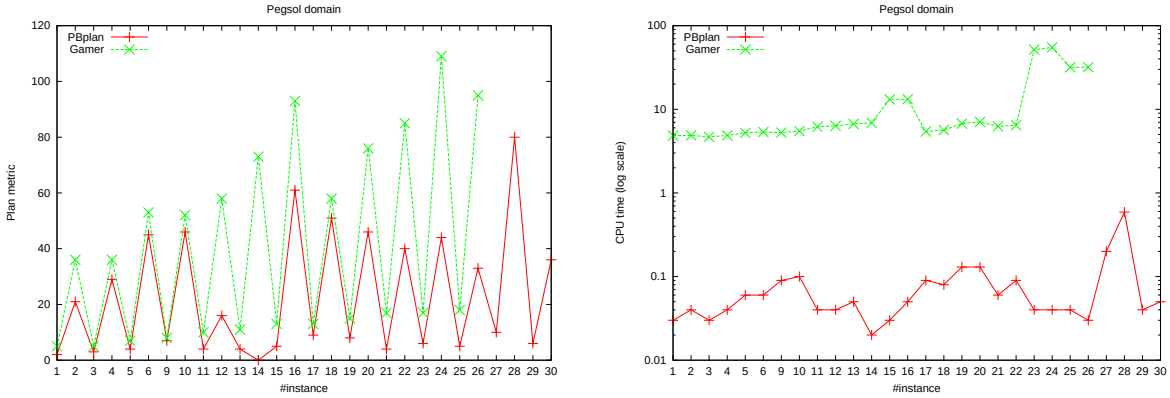

Fig. 8. Plan metrics (Left) and CPU time (Right) for PBPLAN, GAMER and SATPLAN(P) on pegsol instances.

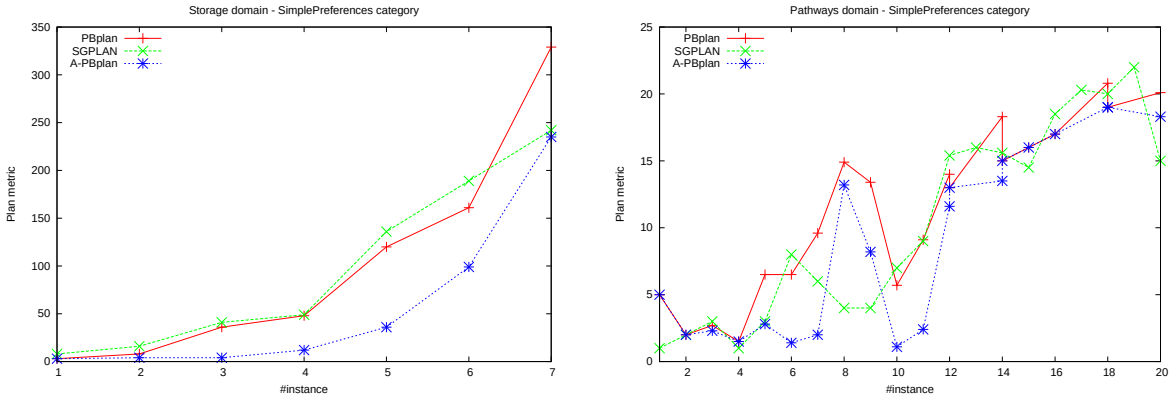

Fig. 9. Plan metrics for PBPLAN, A-PBPLAN and SGPLAN on (Left) Storage and (Right) Pathways instances.

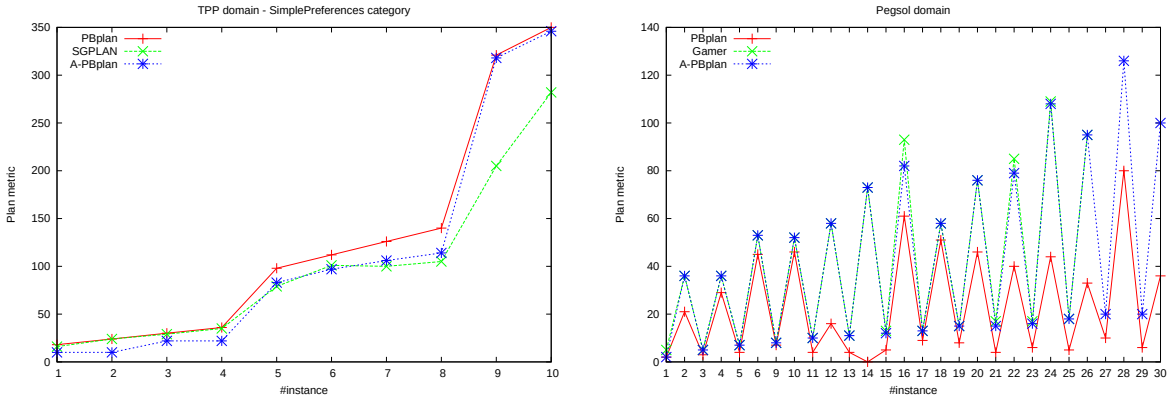

Fig. 10. Plan metrics for PBPLAN, A-PBPLAN and SGPLAN on (Left) TPP and (Right) pegsol instances.

Given this precise result, we decided to test in this direction all the planning problems analyzed: instead of stopping our procedure at the first makespan for which a plan exists, we continue to run PBPLAN for the full allowed CPU time and return the best solution found within the time limit. We call the resulting procedure A-PBPLAN.

The results are presented in Figures 9 and 10 that are organized as Fig. 4- 8 (Left). We can immediately see that the plan quality improves substantially on most of the instances: the plan metric returned by A-PBPLAN is now consistently better than SGPLAN in the Storage domain, in the first instance of the TPP domain, and on many instances of the Pathways, and the gap with SGPLAN is often diminished in the other instances. In particular the Pegsol instances were the more "problematic" on this side: now the plan metric returned by A-PBPLAN is almost always the same of GAMER (see Fig. 10 (Right)), being slightly worse on few instances. On the remaining domains, A-PBPLAN returns the same plan metric as of PBPLAN on the three instances solved by PBPLAN, given that no problems with higher makespan than the optimal are solved within the time limit.

5.4. Summary of the results.

A first contribution of our analysis is a wide comparison of Max-SAT and PB solvers on benchmarks coming from compilation of IPC-5/IPC-6 benchmarks (with a fixed makespan). Later we analyze why particular solvers perform well on particular instances. About PBPLAN, there are some very positive results that have to be underlined, further considering that SGPLAN and GAMER are, often by far, the best planning systems in the categories of interest in this work. On the Storage and Pathways domains, the plan metrics returned by PBPLAN are comparable, and sometimes better than, the ones of SGPLAN. The same holds for the smallest instances of the Trucks domain. On the pegsol domain, PBPLAN is faster than GAMER, and can even solve some instances in the domain not solved by GAMER. This holds at the price of a lower plan quality, but we have seen that PBPLAN can get close to the results of GAMER by increasing the makespan. Finally, note that when PBPLAN does not solve an instance within the time limit, it is often the case that the instance is hard to solve even by SGPLAN, or GAMER, and/or by SATPLAN (e.g., Trucks and IPC-5 Openstacks). As far as efficiency is concerned, we expect PBPLAN to become (much) faster thanks to the availability of new/improved solvers in the next Max-SAT/PB evaluations.

As far as Max-SAT and PB solvers performance is concerned, from Figures 3 and 7 we can note that generally MINISAT+ performs well. By analyzing the results, we noticed that the vast majority of the shown instances do not exceed 150K constraints, which is a reasonable size to be efficiently handled by MINISAT+. Thus this fact seems to be a major factor for its good results. About the two "exceptions", they refer to instances #36 and #37 of Figure 3 (coming from the Storage and Openstacks domains respectively) where the dimensions of the formulas are about 500K and 300K constraints respectively. About instances from IPC-6 benchmarks in Figure 7, we have to remember that MINIMAXSAT is based on (a version of) MINISAT+: on these instances its additional simplification techniques help to improve slightly the overall performance. Interestingly, the best solvers rely on a PB formulation: PB systems can solve problems with a more general representation of constraints, while the instances in our analysis involve clauses, thus in principle "best suited" for a Max-SAT formulation.

Finally, as far as anytime performance are concerned, we have seen that PBPLAN, when allotted all available time and not stopped at the optimal makespan, often improves significantly the quality of the returned plans.

6. Related work

The literature about planning based on constraint satisfaction techniques is vast and subject to a number of research papers and a series of workshop (e.g. Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems). In this section we refer just to the papers related to our approach more closely and with other approaches implemented in planners having distinguished performance in the "SimplePreferences" track of the IPC-5 and on the optimization tracks of the IPC-6.

Two of the most related approaches in planning based on constraint satisfaction techniques are the ones in [BC05,BR05]. In [BC05] the authors show how to extend GP-CSP [DK01] in order to plan with preferences expressed as a TCP-net [BBD+04]. In the Boolean case, TCP-net can be expressed as Boolean formulas. Though this work is not based on satisfiability, the problem they consider is the same as that we deal with but with qualitative preferences: find

an optimal plan wrt the given preferences among the plans with makespan n. An iterative deepening approach, similar to what we did in Subsection 5.3, is the approach followed in [BR05], where the considered problem is to extend the planning as satisfiability approach in order to find plans with optimal sequential length. Even if they solve a different problem to us, the overall goal is the same as in our paper, i.e. to allow the planning as satisfiability approach to take into account plan quality issues other than the makespan. It is interesting to note that the authors use a Boolean formula to encode the function representing the sequential length of the plan. In their approach, for a given n, the search for an optimal solution is done by iteratively calling the SAT solver, each time adding a constraint imposing a smaller value for the objective function (using [BB03]): 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 [BR05] is related to finding a plan with the minimum number of actions for a planning problem Π with makespan n. Our approach can also deal with "soft" goals and non-uniform weights. Non-uniform weights associated with the action's executions are taken into account in PLAN-A [CLH08], which relies on ideas and results obtained by the author of this paper in [GM06,GM07].

IPPLAN participated in the deterministic part of the IPC-5 on classical domains. It reduces STRIPS benchmarks to 0-1 IP problems and then calls CPLEX [ILO02]. In [vdBKV06a] the same authors of IPPLAN presented a compilation of PDDL3 benchmarks into 0-1 IP, on a wider set of domains and features wrt the ones we have dealt with in this paper. On the other hand no implementation and experimental analysis are provided, as well as any formal result. Moreover, even if it should be easy, in [vdBKV06a] there is no indication on how weights are treated and we are not aware of compilations of IPC-6 benchmarks in their framework.

SGPLAN ver. 5 [HWHC06,HWHC07] extends ver. 4 for PDDL2.2 [HE05] in order to deal with the new constructs of PDDL3. Its basic idea is to partition a problem into sub-problems, one for each (soft) goal (considered as hard), to solve the sub-problems individually by a modified version of an existing planner i.e. METRIC-FF [Hof03] and then to resolve inconsistencies across sub-problems. GAMER has been the winner of the optimization tracks of the IPC-6; it uses BDDs and relies on "multi-actions" for tackle conditional effects and soft goals. The advantages of BDDs wrt to SAT formulas is that their sizes do not increase with plan horizon, but the representation through BDDs "easily" runs out of memory.

In [KG09] it is shown and formally proved that adding soft goals with linear impact on plan metrics does not increase the expressive power of STRIPS planning problems with action costs and the same problems can be (equivalently) tackled considering only action costs. We follow a similar approach also in the treatments of action preconditions and costs, which extends and adapts the approaches in [GK97, BKD06]. [KG09] also shows that "classical" planners with action costs can perform better than IPC-6 planners on IPC-6 benchmarks with action costs and soft goals, when run on compiled (similarly as presented in the paper) IPC-6 benchmarks,

7. Conclusions and future works

In this paper we presented a new approach for finding plans with "optimal" metrics in satisfiability-based planning, at a fixed makespan, based on compilation into Max-SAT and PB problems. We also proved that our approaches return an "optimal" plan, at a fixed makespan. We have shown, on planning problems of the IPC-5 and IPC-6, that the approach is viable and indeed can help to widen the set of benchmarks that can be effectively solved with a SAT-based approach, often computing good quality plans. Moreover, our analysis (i) reveals what are the Max-SAT/PB solvers performing best on these planning domains, giving hints on why this is the case, and (ii) individuates some weakness of the approach, in particular where the difference in plan metric wrt SGPLAN/GAMER is high due to the fixed makespan. In this respect we have conducted a further analysis in which our planner is not stopped at the fixed makespan, but all the allotted time is used for searching for high quality plans: results show that plan quality increased significantly. Moreover, differently from the other planners, our approach can easily rely on future improvements of Max-SAT/PB solvers in order to reduce the overall CPU times.

In the near future we plan to evaluate whether our approach can also effectively deal with other IPC-6 benchmarks having more "complex" action costs and to consider methods to trade-off among plan quality measures such as makespan and plan metric.

References

[ABKW08] Tobias Achterberg, Timo Berthold, Thorsten Koch, and Kati Wolter. Constraint integer programming: A new approach to integrate CP and MIP. In Laurent Perron and Michael A. Trick, editors, *Proc. of the 5th International Conference Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2008)*, volume 5015 of *Lecture Notes in Computer Science*, pages 6–20. Springer, 2008.

[ALM07] Joseph Argelich, Chu M. Li, and Felip Manyà. An improved exact solver for partial Max-SAT. In *Proc. of the International Conference on Nonconvex Programming: Local and Global Approaches (NCP-2007)*, pages 230–231, 2007.

[BB03] Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In Francesca Rossi, editor, *Proc. of the 9th International Conference on Principles and Practice of Constraint Programming (CP 2003)*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2003.

[BBD+04] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

[BC05] Ronen I. Brafman and Yuri Chernyavsky. Planning with goal preferences and constraints. In Susanne Biundo, Karen L. Myers, and Kanna Rajan, editors, *Proc. of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 182–191. AAAI Press, 2005.

[BKD06] Josh Benton, Subbarao Kambhampati, and Minh B. Do. YochanPS: PDDL3 simple preferences and partial satisfaction planning. 5th International Planning Competition Booklet, pages 23-25. Available at [IPC06], 2006.

[BR05] Markus Büttner and Jussi Rintanen. Satisfiability planning with constraints on the number of actions. In Susanne Biundo, Karen L. Myers, and Kanna Rajan, editors, *Proc. of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 292–299. AAAI Press, 2005.

[CHXZ09] Yixin Chen, Ruoyun Huang, Zhao Xing, and Weixiong Zhang. Long-distance mutual exclusion for planning. *Artificial Intelligence*, 173(2):365–391, 2009.

[CLH08] Yixin Chen, Qiang Lv, and Ruoyun Huang. 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`, 2008.

[CXZ07] Yixin Chen, Zhao Xing, and Weixiong Zhang. Long-distance mutual exclusion for propositional planning. In Manuela M. Veloso, editor, *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1840–1845, 2007.

[DK01] Minh B. Do and Subbarao Kambhampati. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artificial Intelligence*, 132(2):151–182, 2001.

[EK09] Stefan Edelkamp and Peter Kissmann. Optimal symbolic planning with action costs and preferences. In Craig Boutilier, editor, *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1690–1695, 2009.

[ES06] Niklas Eén and Niklas Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.

[FN71] Richard Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[GHL+09] Alfonso Gerevini, Patrick Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the 5th IPC: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009.

[GK97] B. Cenk Gazen and Craig A. Knoblock. Combining the expressivity of UCPOP with the efficiency of Graphplan. In Sam Steel and Rachid Alami, editors, *Proc. of the 4th European Conference on Planning (ECP 1997): Recent Advances in AI Planning*, volume 1348 of *Lecture Notes in Computer Science*, pages 221–233. Springer, 1997.

[GM06] Enrico Giunchiglia and Marco Maratea. Solving optimization problems with DLL. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *Proc. of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 377–381. IOS Press, 2006.

[GM07] Enrico Giunchiglia and Marco Maratea. Planning as satisfiability with preferences. In *Proc. of the 22nd AAAI Conference on Artificial Intelligence*, pages 987–992. AAAI Press, 2007.

[GM10] Enrico Giunchiglia and Marco Maratea. A pseudo-boolean approach for solving planning problems with ipc simple preferences. In *Proc. of COPLAS'2010: ICAPS 2010 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems*, pages 23–32, 2010.

[GM11] Enrico Giunchiglia and Marco Maratea. Introducing prefenreces in planning and satisfiability. *Journal of Logic and Computation*, 21(2):205–229, 2011.

[GS02] Alfonso Gerevini and Ivan Serina. LPG: A planner based on local search for planning graphs with action costs. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *Proc. of the the 16th International Conference on Artificial Intelligence Planning Systems (AIPS 2002)*, pages 13–22. AAAI, 2002.

[GSS03] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research*, 20:239–290, 2003.

[GSS08] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artificial Intelligence*, 172(8-9):899–944, 2008.

[HCZ10] Ruoyun Huang, Yixin Chen, and Weixiong Zhang. A novel transition based encoding scheme for planning as satisfiability. In *Proc. of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*. AAAI Press, 2010.

[HE05] Jörg Hoffmann and Stefan Edelkamp. The deterministic part of IPC-4: An overview. *Journal of Artificial Intelligence Research*, 24:519–579, 2005.

[HLO08] Federico Heras, Javier Larrosa, and Albert Oliveras. MiniMaxSAT: A new weighted Max-SAT solver. *Journal of Artificial Intelligence Research*, 31:1–32, 2008.

[HN01] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Hof03] Jörg Hoffmann. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341, 2003.

[HWHC06] Chih-Wei Hsu, Benjamin W. Wah, Rouyun Huang, and Yixin Chen. New features in SGPlan for handling preferences and constraints in PDDL3.0. 5th International Planning Competition Booklet, pages 39-41. Available at [IPC06], 2006.

[HWHC07] Chih-Wei Hsu, Benjamin W. Wah, Rouyun Huang, and Yixin Chen. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In Manuela M. Veloso, editor, *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1924–1929, 2007.

[ILO02] ILOG Inc. *Mountain Wiew, CA. ILOG CPLEX 8.0 users' manual*, 2002.

[IPC06] IPC-5. `http://zeus.ing.unibs.it/ipc-5/booklet/i06-ipc-allpapers.pdf`, 2006.

[JJN05] Matti Järvisalo, Tommi A. Junttila, and Ilkka Niemelä. Unrestricted vs restricted cut in a tableau method for boolean circuits. *Annals of Mathematics and Artificial Intelligence*, 44(4):373–399, 2005.

[JS05] Paul Jackson and Daniel Sheridan. Clause form conversions for boolean circuits. In Holger H. Hoos and David G. Mitchell, editors, *Proc. of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT 2004)*, volume 3542 of *Lecture Notes in Computer Science*, pages 183–198. Springer, 2005.

[KG09] Emil Keyder and Hector Geffner. Soft goals can be compiled away. *Journal of Artificial Intelligence Research*, 36:547–556, 2009.

[KS92] Henry Kautz and Bart Selman. Planning as satisfiability. In Bernd Neumann, editor, *Proc. of the 10th European Conference on Artificial Intelligence (ECAI 1992)*, pages 359–363. IOS Press, 1992.

[KS99] Henry Kautz and Bart Selman. Unifying SAT-based and graph-based planning. In Thomas Dean, editor, *Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI 1999)*, pages 318–325. Morgan-Kaufmann, 1999.

[KS06] Henry Kautz and Bart Selman. SATPLAN04: Planning as satisfiability. 5th International Planning Competition Booklet, pages 45-47. Available at [IPC06], 2006.

[LMMP09] Chu Min Li, Felip Manyà, Nouredine Ould Mohamedou, and Jordi Planes. Exploiting cycle structures in Max-SAT. In Oliver Kullmann, editor, *Proc. of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT 2009)*, volume 5584 of *Lecture Notes in Computer Science*, pages 467–480. Springer, 2009.

[LS07] Han Lin and Kaile Su. Exploiting inference rules to compute lower bounds for Max-SAT solving. In Manuela M. Veloso, editor, *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2334–2339, 2007.

[LSL08] Han Lin, Kaile Su, and Chu Min Li. Within-problem learning for efficient lower bound computation in Max-SAT solving. In Dieter Fox and Carla P. Gomes, editors, *Proc. of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008)*, pages 351–356. AAAI Press, 2008.

[McD00] Drew V. McDermott. The 1998 AI planning systems competition. *AI Magazine*, 21(2):35–55, 2000.

[Mit05] David G. Mitchell. A SAT solver primer. *Bulletin of the European Association for Theoretical Computer Science*, 85:112–132, 2005.

[MMS06] Vasco M. Manquinho and Joao Marques-Silva. On using cutting planes in pseudo-Boolean optimization. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:209–219, 2006.

[MR06] Vasco M. Manquinho and Olivier Roussel. The first evaluation of pseudo-Boolean solvers (PB'05). *Journal on Satisfiability, Boolean Modeling and Computation*, 2:103–143, 2006.

[MSM08] João Marques-Silva and Vasco M. Manquinho. Towards more effective unsatisfiability-based maximum satisfiability algorithms. In Hans Kleine Büning and Xishun Zhao, editors, *Proc. of 11th International Conference on Theory and Applications of Satisfiability Testing (SAT 2008)*, volume 4996 of *Lecture Notes in Computer Science*, pages 225–230. Springer, 2008.

[MSP08] João Marques-Silva and Jordi Planes. Algorithms for maximum satisfiability using unsatisfiable cores. In *Proc. of Design, Automation and Test in Europe (DATE 2008)*, pages 408–413. IEEE, 2008.

[MSP09] Vasco M. Manquinho, João Marques Silva, and Jordi Planes. Algorithms for weighted boolean optimization. In Oliver Kullmann, editor, *Proc. of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT 2009)*, volume 5584 of *Lecture Notes in Computer Science*, pages 495–508. Springer, 2009.

[PG86] David A. Plaisted and Steven Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2:293–304, 1986.

[RG07] Miquel Ramirez and Hector Geffner. Structural relaxations by variable renaming and their compilation for solving mincostsat. In Christian Bessiere, editor, *Proc. of 13th International Conference on Principles and Practice of Constraint Programming*, volume 4741 of *Lecture Notes in Computer Science*, pages 605–619. Springer, 2007.

[RGPS09] Nathan Robinson, Charles Gretton, Duc Nghia Pham, and Abdul Sattar. SAT-based parallel planning using a split representation of actions. In Alfonso Gerevini, Adele E. Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*. AAAI, 2009.

[RHN06] Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, 170(12-13):1031–1080, 2006.

[SS05] Hossein M. Sheini and Karem A. Sakallah. Pueblo: A modern pseudo-boolean sat solver. In *Proc. of Design, Automation and Test in Europe Conference and Exposition (DATE 2005)*, pages 684–685. IEEE Computer Society, 2005.

[Tse70] G. Tseitin. On the complexity of proofs in propositional logics. *Seminars in Mathematics*, 8:115–125, 1970.

[vdBK05] Menkes van den Briel and Subbarao Kambhampati. Optiplan: Unifying IP-based and graph-based planning. *Journal of Artificial Intelligence Research*, 24:919–931, 2005.

[vdBKV06a] Menkes van den Briel, Subbarao Kambhampati, and Thomas Vossen. Planning with preferences and trajectory constraints through integer programming. In *Proc. of the ICAPS Workshop on Planning with Preferences and Soft Constraints*, pages 19–22, 2006.

[vdBKV06b] Menkes van der Briel, Subbarao Kambhampati, and Thomas Vessen. IPPLAN: Planning as integer programming. 5th Internation Planning Competition Booklet, pages 26-28. Available at [IPC06], 2006.

[vdBVK08] Menkes van den Briel, Thomas Vossen, and Subbarao Kambhampati. Loosely coupled formulations for automated planning: An integer programming perspective. *Journal of Artificial Intelligence Research*, 31:217–257, 2008.

[War98] Joost P. Warners. A linear-time transformation of linear inequalities into CNF. *Information Processing Letters*, 68(2):63–69, 1998.

[XCZ06a] Zhao Xing, Yixin Chen, and Weixiong Zhang. MaxPlan: Optimal planning by decomposed satisfiability and backward reduction. 5th Internation Planning Competition Booklet, pages 53-55. Available at [IPC06], 2006.

[XCZ06b] Zhao Xing, Yixin Chen, and Weixiong Zhang. Optimal STRIPS planning by maximum satisfiability and accumulative learning. In Derek Long, Stephen F. Smith, Daniel Borrajo, and Lee McCluskey, editors, *Proc. of the 16th International Conference on Automated Planning and Scheduling (ICAPS 2006)*, pages 442–446. AAAI, 2006.