Compiling PDDL3 Qualitative Preferences without Using Automata

...

Abstract

We address the problem of planning with preferences in propositional domains extended with the class of (preferred) temporally extended goals supported in PDDL3, that is part of the standard planning language PDDL since the 5th International Planning Competition (IPC5). Such preferences are useful to characterise plan quality by allowing the user to express certain soft constraints on the state trajectory of the desired solution plans. Starting from the work of Keyder and Geffner on compiling (reachability) soft goals, we propose a compilation scheme for translating a STRIPS problem enriched with qualitative PDDL3 preferences (and possibly also with soft goals) into an equivalent STRIPS problem with action costs. The proposed compilation, which supports all types of preferences in the benchmarks from IPC5, allows many existing STRIPS planners to immediately address planning with preferences. An experimental analysis presented in the paper evaluates the performance of state-of-the-art STRIPS planners supporting action costs using our compilation approach to deal with qualitative PDDL3 preferences. The results indicate that our approach is highly competitive with respect to current planners that natively support the considered class of preferences.

Introduction

Planning with preferences, also called "over-subscription planning" in [2, 5**?**], concerns the generation of plans for problems involving soft goals or soft state-trajectory constraints (called preferences in PDDL3), that it is desired a plan satisfies, but that do not have to be satisfied. The quality of a solution plan for these problems depends on the soft goals and preferences that are satisfied.

For instance, a useful class of preferences than can be expressed in PDDL3 [6] consists of *always preferences*, requiring that a certain condition should hold in *every* state reached by a plan. As discussed in [**?** **?** 6**?**], adding always preferences to the problem model can be very useful to express safety or maintenance conditions, and other desired plan properties. An simple example of such conditions is "whenever a building surveillance robot is outside a room, all the room doors should be closed".

PDDL3 supports other useful types of preferences, and in particular the qualitative preferences of types *at-end*, which are which are equivalent to soft goals, *sometime*, *sometime-before* and *at-most-once*, which are all the types used in the available benchmarks for planning with qualitative PDDL3 preferences [6]. Examples of preferences that can be expressed through these constructs in a logistics domain are: "sometime during the plan the fuel in the tank of every vehicle should be full", "a certain depots should be visited before another once", "every store should be visited at most once" (the reader can find additional examples in [6]).

In this paper, we study propositional planning with these types of preferences through a compilation approach.

Related Work

Our compilative approach is inspired by the work of Keyder and Geffner [**?**] on compiling soft goals into STRIPS with action costs (here denoted with STRIPS+). In this work the compilation scheme introduces, for each soft goal p of the problem, a dummy goal p' that can be achieved using two actions in mutual exclusion. The first one, which is called *collect(p)*, has cost equal to 0 and requires that p be true when it is applied; the second one, whic is called *forgo(p)*, has cost equal to the utlity of p and requires that p be false when it is applied.

Both of these action can be performed at the end of the plan and for each soft goal p but just one of $\{$*collect(p), forgo(p)*$\}$ can appear in the plan depending on whether the soft goal has been achived or not. This scheme has achieved good perfomance which can be improved with the use of an ad hoc admissible heuristic based on the reachability of soft goals [9].

The most prominent existing planners supporting PDDL3 preferences are HPlan-P [**?** **?**], which won the "qualitative preference" track of IPC-5, MIPS-XXL [**?** **?**] and the more recent LPRPG-P [**?**] and its extension in [**?**]. These (forward) planners represent preferences through automata whose states are synchronised with the states generated by the action plans, so that an accepting automaton state corresponds to preference satisfaction. For the synchronisation, HPlan-P and LPRPG-P use planner-specific techniques, while MIPS-XXL compiles the automata by modifying the domain operators and adding new ones modelling the automata transitions of the grounded preferences.

Our computation method is very different from the one of MIPS-XXL since, rather than translating automata into new

operators, the problem preferences are compiled by only modifying the domain operators, possibly creating multiple variants of them. Moreover, our compiled files only use STRIPS+, while MIPS-XXL also uses numerical fluents.[1]

The works on compiling LTL goal formulas by Cresswell and Coddington [**?**] and Rintanen [**?**] are also somewhat related to ours, but with important differences. Their methods handle *hard* temporally extended goals instead of preferences, i.e., every temporally extended goal must be satisfied in a valid plan, and hence there is no notion of plan quality referred to the amount of satisfied preferences. Rintanen's compilation considers only single literals in the always formulae (while we deal with arbitrary CNF formulas), and it appears that extending it to handle more general formulas requires substantial new techniques [**?**]. An implementation of Crosswell and Coddington's approach is unavailable, but Bayer and McIlaraith [**?**] observed that their approach suffers exponential blow up problems and performs less efficiently than the approach of HPlan-P.

IMPORTANTE, CONFRONTO LAVORO NEBEL: An important work about soft-trajectory constraints compilation, which is closely related to ours, has been recently proposed by Wright, Mattüller and Nebel in [**?**] (cerca riferimento articolo Nebel). This approach is a based on the compilation of the soft trajectory constraints into conditional effects and state dependent action costs using LTL_f and Büchi automata. There are some similarities between between our and their approach but at the same time there are also differences.

FP: differenze approccio Nebel e nostro: 1) numero di fluenti introdotto nella compilazione diverso; 2) il nostro e' un approccio piu' specifico PDDL3-oriented, il loro piu' generale; 3) diverso meccanismo di aggiornamento dei costi, nel loro caso ricompensa e penalita', nel nosto solo penalit 4) il loro schema prevede quindi costi negativi e postivi, il nostro positivi; nel loro caso per avere solo costi positivi e' necessario perdere l'ottimalita' nel nostro caso no ed inoltre volendo potremmo introdurre dei costi negativi per quelle preferenze la cui violazione puo' essere testata solo alla fine del piano mantenendo l'ottimalita'; ho fatto inoltre menzione del fatto che i costi possono essere incrementanti il prima possibile (es. always)

1) In our approach, given a preference P, we introduce at most a pair of boolean fluents, typically one additional fluent to represent if a preference is violated or not (P-*violated*) and in some cases an additional fluent to correctly represent the status of a preference during the planning, while in their approach it is introduced a boolean variable for each state of the corresponding automaton of P. **2)** Their approach is more general while ours is focused on PDDL3 constraints and therefore it is more specific.

3) In their work the cost of the plan during the planning is updated by using rewards and penalties (negative and positive costs respectively). The cost of the plan is increased whenever a violation of a preference (also reversible) occurs. On the contrary, if an operator causes the satisfaction of a preference then the cost of the plan is decreased. In both cases the negative or positive cost is equal to the utility of the interested preference[2]. **4)** This type of cost update requires the use of negative costs while our compilation scheme produces a problem whose costs are monotonically increasing because, similarly to what was proposed in Keyder and Geffner, costs are realized only at the end of the planning. In our scheme some costs can be anticipated for those preferences whose violation is irreversible (e.g. always, sometime-before) and negative costs could be used for those preferences that may be "temporarily" violated (e.g. sometime, at-end). In both cases our scheme would maintain the optimality but in this work we have provided a compilation based only on positive costs in order to take advantage of a wider spectrum of classical planners (few planners still support negative costs).

we propose a compilation scheme for translating a STRIPS problem with PDDL3 qualitative preferences into an equivalent STRIPS+ problem. Handling action costs is a practically important, basic functionality that is supported by many powerful planners; the proposed compilation method allows them to immediately support (through the compiled problems) the considered class of preferences with no change to their algorithms and code.

Propositional Planning with Qualitative PDDL3 Preferences

A STRIPS problem is a tuple $\langle F, I, O, G\rangle$ where F is a set of fluents, $I \subseteq F$ and $G \subseteq F$ are the initial state and goal set, respectively, and O is a set of actions or operators defined over F as follows.

A STRIPS operator $o \in O$ is a pair $\langle \mathit{Pre}(o), \mathit{Eff}(o)\rangle$, where $\mathit{Pre}(o)$ is a sets of atomic formulae over F and $\mathit{Eff}(o)$ is a set of literals over F. $\mathit{Eff}(o)^+$ denotes the set of positive literals in $\mathit{Eff}(o)$, $\mathit{Eff}(o)^-$ the set of negative literals in $\mathit{Eff}(o)$. An action sequence $\pi = \langle a_0, \ldots, a_m\rangle$ is applicable in a planning problem Π if all actions a_i are in O and there exists a sequence of states $\langle s_0, \ldots, s_{m+1}\rangle$ such that $s_0 = I$, $prec(a_i) \subseteq s_i$ and $s_{i+1} = s_i \setminus \{p \mid \neg p \in \mathit{Eff}(a_i)^-\} \cup \mathit{Eff}(a_i)^+$, for $i = 0 \ldots m$. Applicable action sequence π achieves a fluent g if $g \in s_{m+1}$, and is a valid plan for Π if it achieves each goal $g \in G$ (denoted with $\pi \models G$).

A STRIPS+ problem is a tuple $\langle F, I, O, G, c\rangle$, where $\langle F, I, O, G\rangle$ is a STRIPS problem and c is a function mapping each $o \in O$ to a non-negative real number. The cost $c(\pi)$ of a plan π is $\sum_{i=0}^{|\pi|-1} c(a_i)$, where $c(a_i)$ denotes the cost of the ith action a_i in π and $|\pi|$ is the length of π.

[1] Another compilation scheme using numerical fluents is considered in [6] to study the expressiveness of PDDL3(without an implementation).

[2] Messo in footnote altrimenti era troppo lungo: Note that both the violation and the satisfaction of a preference may be temporary conditions depending on the type of interested preference. For example an always preference can be irreversibly violated and its satisfaction can only be evaluated at the end of the plan considering the whole trajectory of the states; on the contrary a sometime-after preference can be satisfied and violated several times during the execution of the plan.

Starting from a STRIPS+ problem we define the notion of state trajectory generatered by a plan. Given a STRIPS+ $\Pi = \langle F, I, O, G, c\rangle$ problem, a plan π generates the trajectory $\langle (S_0, 0), (S_1, t_1), ..., (S_n, t_n)\rangle$ iif $S_0 = I$ and for each happening h generated by π, with h at time t, there is some i such that $t_i = t$ and S_i is the result of applying the happening h to S_{i-1}, and for every $j \in \{1 \dots n\}$ there is a happening π at t_j.

The following is a fragment of the grammar of PDDL3, describing the new modalities of PDDL3 for expressing these soft state-trajectory constraints (indicated with `con-GD`) (the full BNF grammar is given in [**? ?**]) where `<GD>` is a goal description (a first order logic formula):

```
<con-GD> ::= (at end <GD>) | (always <GD>) |
             (sometime <GD>) |
             (at-most-once <GD>) |
             (sometime-after <GD> <GD>) |
             (sometime-before <GD> <GD>)
```

Let ϕ and ψ be atomic formulae over the predicates of a planning problem, then interpretation of the modal operators considered in this work is specified in Figure **??** [COMMENTATA]. We write $\pi \models P$ to indicate that the state-trajectory generated by π satisfies a preference P and we indicate with $\mathcal{A}$, $\mathcal{SB}$, $\mathcal{ST}$, $\mathcal{AO}$, $\mathcal{SG}$ the classes of all prefer-ecense of always, sometime-before, sometime, at-most-once and soft goal respectively for a given STRIPS+ problem.

But in the following, without loss of generality, we will assume in the discussion that the formula ϕ and ψ involved in a preference P is expressed in conjuctive normal form $\phi = \phi_1 \wedge \phi_2 \wedge ... \wedge \phi_n$ where each ϕ_i ($i \in \{1 \dots n\}$) is a clause of P formed by literals over the problem fluents.

Definition 1 *A* STRIPS+ *problem with preferences is a tuple* $\langle F, I, O, G, \mathscr{P}, c, u\rangle$ *where:*

- $\langle F, I, O, G, c\rangle$ *is a* STRIPS+ *problem;*
- $\mathscr{P} = \{\mathscr{P}_{\mathcal{A}} \cup \mathscr{P}_{\mathcal{SB}} \cup \mathscr{P}_{\mathcal{ST}} \cup \mathscr{P}_{\mathcal{AO}} \cup \mathscr{P}_{\mathcal{SG}}\}$ *is the set of the preferences of* Π *where* $\mathscr{P}_{\mathcal{A}} \subseteq \mathcal{A}$, $\mathscr{P}_{\mathcal{SB}} \subseteq \mathcal{SB}$, $\mathscr{P}_{\mathcal{ST}} \subseteq \mathcal{ST}$, $\mathscr{P}_{\mathcal{AO}} \subseteq \mathcal{AO}$ *and* $\mathscr{P}_{\mathcal{SG}} \subseteq \mathcal{SG}$*;*
- u *is an utility function mapping each* $P \in \mathscr{P}$ *to a value in* $\mathbb{R}_0^+$.

In the following the class of STRIPS+ problems with a set of preferences is indicated with STRIPS+P.

Definition 2 *Let* Π *be a* STRIPS+P *problem. The utility* $u(\pi)$ *of a plan* π *solving* Π *is the difference between the total amount of utility of the preferences by the plan and its cost:*

$$u(\pi) = \sum_{P \in \mathscr{P} : \pi \models P} u(P) - c(\pi).$$

The definition of plan utility for STRIPS+P is similar to the one given for STRIPS+ with soft goals by Keyder and Geffner [**?**]. A plan π with utility $u(\pi)$ for a STRIPS+P problem is optimal when there is no plan π' such that $u(\pi') > u(\pi)$. The definitions below are introduced to simplify the notation in the discussion.

Operator-Preference Interactions

The definitions below are introduced to simplify the notation in the discussion.

Definition 3 *Given a preference clause* $\phi_i = l_1 \vee l_2 \vee ... \vee l_m$*, the set* $L(\phi_i) = \{l_1, l_2, ..., l_m\}$ *is the equivalent set-based definition of* ϕ_i *and* $\overline{L}(\phi_i) = \{\neg l_1, \neg l_2, ..., \neg l_m\}$ *is the literal complement set of* $L(\phi_i)$.

Definition 4 *Given an operator* $o \in O$ *of a* STRIPS+P *problem,* $Z(o)$ *is the set of literal defined as:*

$$Z(o) = (\mathit{Pre}(o) \setminus \{p \mid \neg p \in \mathit{Eff}(o)^-\}) \cup \mathit{Eff}(o)^+ \cup \mathit{Eff}(o)^-.$$

Note that the literals in $Z(o)$ hold in any reachable state resulting from the execution of operator o.

To shorten the following definitions and explanation we indicate the state where the operator o is applied with s and the state resulting from the application of o with s'.

Definition 5 *Given an operator* o *and CNF formula* $\phi = \phi_1 \wedge ... \wedge \phi_n$*, we define the set of clauses which will surely be true in the resulting state from the application of* o *as:*

$$C_\phi(o) = \{\phi_i : |L(\phi_i) \cap Z(o)| > 0,\ i \in \{1 \dots n\}\}$$

We can also define the complementary set of the remaining clauses of ϕ *which does not satisfy the previous condition as* $\overline{C}_\phi(o) = \{\phi_i : \phi_i \notin C_\phi(o),\ i \in \{1 \dots n\}\}$

Definition 6 *Given a preference clause* $\phi_i = l_1 \vee l_2 \vee ... \vee l_m$*, the set* $L(\phi_i) = \{l_1, l_2, ..., l_m\}$ *is the equivalent set-based definition of* ϕ_i *and* $\overline{L}(\phi_i) = \{\neg l_1, \neg l_2, ..., \neg l_m\}$ *is the literal complement set of* $L(\phi_i)$.

Definition 7 *Given an operator* $o \in O$ *of a* STRIPS+P *problem,* $Z(o)$ *is the set of literal defined as:*

$$Z(o) = (\mathit{Pre}(o) \setminus \{p \mid \neg p \in \mathit{Eff}(o)^-\}) \cup \mathit{Eff}(o)^+ \cup \mathit{Eff}(o)^-.$$

Note that the literals in $Z(o)$ hold in any reachable state resulting from the execution of operator o.

To shorten the following definitions and explanation we indicate the state where the operator o is applied with s and the state resulting from the application of o with s'.

Definition 8 *Given an operator* o *and CNF formula* $\phi = \phi_1 \wedge ... \wedge \phi_n$*, we define the set of clauses which will surely be true in the resulting state from the application of* o *as:*

$$C_\phi(o) = \{\phi_i : |L(\phi_i) \cap Z(o)| > 0,\ i \in \{1 \dots n\}\}$$

We can also define the complementary set of the remaining clauses of ϕ *which does not satisfy the previous condition as* $\overline{C}_\phi(o) = \{\phi_i : \phi_i \notin C_\phi(o),\ i \in \{1 \dots n\}\}$.

With reference to Definition 8, given a clause $\phi_i = l_1 \vee ... \vee l_{m_i}$ of ϕ, the condition $|L(\phi_i) \cap Z(o)| > 0$ requires that exists at least a literal l_j, with $j \in \{1 \dots m_i\}$, that, belonging to the set $Z(o)$, will be certainly true in the resulting state from the application of the operator o thus making the clause ϕ true in s'.

Definition 9 *Given an operator* o *and a CNF formula* $\phi = \phi_1 \wedge ... \wedge \phi_n$*, we say that* o ***can make true*** ϕ *if*

1. $|C_\phi(o)| > 0$*;*

2. *for each clause* $\phi \notin \overline{C}_\phi(o)$, $\overline{L}(\phi) \not\subseteq Z(o)$.

The first condition in Definition 9 requires that exists at least a clause of the formula which contains some literals which will certainly be true true in the state resulting from the execution of o. The second condition in Definition 9 requires that the remaining clauses of ϕ, which do not belong to $C_\phi(o)$, are certainly not falsified in s', because otherwise the ϕ formula will be certainly falsified in s'.

An operator o that *can make* ϕ does not not guarantees a switch from a state $s \models \neg\phi$ to a state $s' \models \phi$, but it only guarantees $s' \models \phi$ without imposing any condition on the truth value of ϕ in s.

In our compilation scheme of a STRIPS+P problem we have to distinguish, for each kind of preference, different class of operators in order to specialize the operators compilation based on how they interact with the preferences of the problem.

Generally speaking, we can identify three classes of operators regardless of the type of preference considered. An operator is a *threat* for a preference P if in case it is executed it may violate P. An operator is a (potential) *support* for a preference P if in case it is executed it could satisfy P. Finally, an operator o is *neutral* for a preference P if its execution can not influence the current state of the preference. In the following Subsection **??–??** we will decline these generic definitions of classes of operators for each type of considered preference providing for each of them a formal definition.

In the following definitions of operators classes we will assume that ϕ (and ψ in the case of dual preferences) are arbitrary CNF formulae $\phi = \phi_1 \wedge ... \wedge \phi_n$ where each clause $\phi_i = l_1 \vee ... \vee l_{m_i}$ for each $i \in \{1 \dots n\}$.

Compilation of Qualitative Preferences

Experimental Results

Experiments Description

We implemented the proposed compilation scheme and have evaluating it by two sets of experiments with different purposes. On the one hand we evaluated the scheme in a satisficing planning context in which we focused on the search for sub-optimal plans using different planning systems, while in the other we focuse on the search of optimal plans using admissible heuristics.

Regarding the comparison in the context of the satisficing planning we have considered the following STRIPS+ planning system LAMA[10], Mercury [7], MIPlan [8], IBaCoP2 [3], which are some of the best performing planning system in IPC8 [11], and Fast Downward Stone Soup 2018[**?**], Fast Downward Remix[**?**] (abbreviated with FDRemix), which are some of the best performing planning system in the last IPC9 [1] which have been compared with LPRPG-P [4], which is one of the performing planner which supports PDDL3 preferences. Moreover have considered our specifically enhanced version of LAMA for planning with soft goal, which is $\text{LAMA}_\text{P}(h_\text{R})$, which makes use of admissibile heuristic h_R to test the reachability of the soft goals of the problem [9].

As benchmark we have considered the five domains of the qualitative preference track of IPC5 [6] which involve always, sometime, sometime-before, at-most-once and soft goal preferences, i.e Rovers, TPP, Trucks, Openstacks and Storage.

For each original problem all preferences and each original utility were kept. The the classical planners were runned on the compiled problems while LPRPG-P was runned on the original problems of the competition. All the experiments were conducted on a 2.00GHz Core Intel(R) Xeon(R) CPU E5-2620 machine with CPU-time and memory limits of 30 minutes and 8GiB, respectively, for each run of every tested planner. We have tested 8 planners for 5 domains each of which consists of 20 instances for a total of 800 runs.

Table 1 shows the performances of the considered planning system in term of plans quality. As quality measure we have used the IPC quality score, a popular metric of which we have reported a brief description [**?**].

Given a planner p and a task i we assign, if p solves i, the following score to p:

$$score(p,i) = \frac{cost_{best}(i)}{cost(p,i)}$$

where $cost_{best}(i)$ is the cost of the best know solution for the task i found by any planner, and $cost(p,i)$ is the cost of the solution found by the considered planner p in 30 minutes. In our case our reference for $cost_{best}(i)$ is equal to the cost of the best solution among the tested planners within 30 minutes. If p did not find a solution within the time assigned, then $score(p,i)$ is equal to 0 in order to reward both quality and coverage.

The quality score assigned to each tested planner p is set equal to sum of the quality scores assigned to p over all the considered instances:

$$score(p) = \sum_{i \in tasks} score(p,i)$$

Table 1 reports the quality comparison using the IPC score described above. It is splitted into six parts, at top we have reported the qualitative comparision considering all kinds of preferences together in the computation of the IPC score, while in the remaining subtables we have splitted the have considered each considered class of preference seperately (which is indicated in the header of each subtable).

NOTA (FP): selezionare un numero di pianificatori significativi da includere nella tabella per alleggerire la tabella.

Figures 1a—1e and **??**—2a show the qualitative comparison in a different and detailed way. In these histograms we have reported, for each istance of each domain, another quality measure denoted with α_{cost}. Each figure is associated with one of the considered domains and involve two planners, the best performing planner, in term of IPC score and according to Table 1, among the classical planner and LPRPG-P which is the competitor.

We reported a brief description of the metric α_{cost} that we used. Given a planner p and a task i we assign, if p solves i, the following score to p:

$$\alpha_{cost}(p,i) = cost(p,i)/cost_{total}(i) = \frac{\sum_{P \in \mathscr{P}(i) \,:\, \pi \not\models P} c(P)}{\sum_{P \in \mathscr{P}(i)} c(P)}$$

where $cost(p, i)$ is the cost of the solution found by planner p for the task i within 30 minutes and $cost_{total}(i)$ is the sum of the costs of all the preferences involved in the task i (note that $\mathscr{P}(i)$ denote the set of the preferences of the task i).

If we want to restric the calculation of α_{cost} to a single type of preference, for example just always preferences, we denote the cost as $\alpha_{cost}(\mathcal{A})$ while if nothing is indicated, it means that we have considered all the classes of preferences.

From the previous definition, $\alpha_{cost}(p, i)$ could vary between 0 and 1. If $\alpha_{cost}(p, i) = 0$, then it means that the numerator $cost(p, i)$ is equal to 0 and that p has found an optimal plan for i which satisfies all the preferences of the problem. On the contrary, if $\alpha_{cost}(p, i) = 1$, then it means that p has found the worst plan for i where all the preferences of the problem are violated.

More generally given an instance i, the ratio $\alpha_{cost}(p, i)$, comparing plans produced by different systems, tell us which planner has achieved the satisfaction of the most useful subset of preferences in absolute terms. In particular, the planner with the lowest ratio is the planner who got the best performance on that particular instance.

Satisficing Planning Results

The results obtained comparing the planners considered above show that the compilative approach is almost always preferable since the tested classical planners obtain an higher IPC score than LPRPG-P except for Mercury and MIPlan.

With reference to Table 1 the compilative approach seems at glance to be particularly preferable in Rovers, Trucks and Storage. In these domains each classical planner performs better or at least comparable than LPRPG-P (except for Mercury in Trucks); IBaCoP2 performs particularly well in Rovers, FDRemix in Trucks and $\text{LAMA}_{\text{P}}(h_{\text{R}})$ in Storage. Also MIPlan works well in Trucks but it is penalized due to coverage (it solves only 15 instances out of 20).

The planning system from the more recent IPC9 FDRemix and Fast Downward Stone Soup 2018 perform overall better in this benchmark than the planning system from the previous IPC8 but the enhanced version $\text{LAMA}_{\text{P}}(h_{\text{R}})$ is better than everyone else indeed it improves the performance of LAMA in all the considered domains except in Rovers (where there is no soft goal).

Figures 1a–1e show the α_{cost} comparison for each domain comparing the best perfoming classical planner in term of IPC score in such domain with LPRPG-P.

Looking at Figure 1a, where we have reported the α_{cost} comparison for the best performing planners in Rovers in term of IPC score, we can say that the compilative approach, combined with the use of IBaCoP2, achieves to satisfy a better subset of preferences in all the compared instances compared to LPRPG-P.

In TPP the compilative approach seems to be very ineffective, each classical planner achieves an extremely lower quality performance compared to LPRPG-P. The bad performances in this domain are probably due to the many soft goals and sometime preferences because, as shown in [9], the compilation of soft goals can be sometime problematic and neither the use of the reachability heuristic h_{R} in $\text{LAMA}_{\text{P}}(h_{\text{R}})$ can compensate this weakness.

Indeed the part of Table 1, which concerns soft goals and sometime preferences, cleary shows that LPRPG-P is overall more performing than the classical planners especially in TPP in term of these kinds of preferences achieved. This aspect is clearly observable Figures 2a and 2b which report the $\alpha_{\text{cost}}(\mathcal{SG})$ and $\alpha_{\text{cost}}(\mathcal{ST})$ comparison respectively between $\text{LAMA}_{\text{P}}(h_{\text{R}})$ and LPRPG-P in TPP. In these figures we can see that the compilative approach fails to achieve a better subset of soft goals and sometime preferences compared to what LPRPG-P in half of the instances does. But looking at Table 1 we can also observe that $\text{LAMA}_{\text{P}}(h_{\text{R}})$ achieves a better result of always, sometime-before and at-most-once preferences compared to LPRPG-P in term of IPC score, but this is not significant because apparently it happens at the expense of soft-goal and sometime preferences which are clearly more expensive to violate (or equivalently more useful to satisfy).

Looking at Figure 1c, where we have reported the α_{cost} comparison for the best performing planners in Trucks in term of IPC score, we can say that the compilative approach, combined with the use of FDRemix, achieves to satisfy a better, worst and equal subset of preferences in all instances compared to LPRPG-P in 15, 4 and 1 respectively.

Regarding Opentacks the tested planners achieve a comparable performance even if the classical planners are slightly penalized compared to LPRPG-P. Looking at Figure 1d, where we have reported the α_{cost} comparison between $\text{LAMA}_{\text{P}}(h_{\text{R}})$ and LPRPG-P calculated considering all kind of preferences, we can say that our classical planner achieves to satisfy a better, worst and equal subset of preferenes in 12, 7 and 1 instances respectively

NOTA (FP): scrivi commento Storage.

$\mathscr{P}_{\mathcal{ALL}}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
$\text{LAMA}_{\text{B}}(h_r)$	16.98	8.34	15.32	19.28	**18.47**	**78.39**
FDRemix	17.89	7.1	**17.67**	18.99	16.2	77.86
FDSS 2018	17.6	7.03	17.08	18.7	17.11	77.52
LAMA(2011)	17.01	7.53	13.04	18.42	17.81	73.82
IBaCoP2	**19.62**	9.68	10.0	17.85	15.72	72.87
LAMA(2018)	16.44	7.63	13.34	16.03	17.78	71.22
LPRPG-P	11.36	**18.74**	6.99	**19.71**	12.87	69.66
MIPlan	17.65	8.8	9.23	17.35	14.42	67.45
Mercury	16.07	6.57	7.78	18.06	14.5	62.97

$\mathscr{P}_{\mathcal{A}}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
$\text{LAMA}_{\text{B}}(h_r)$	13.46	20.0	15.0	20.0	19.0	87.46
FDSS 2018	13.51	17.0	18.0	17.83	20.0	86.34
LAMA(2011)	13.8	20.0	13.0	20.0	19.0	85.8
IBaCoP2	14.73	20.0	13.0	19.0	19.0	85.73
MIPlan	14.03	20.0	12.0	19.0	20.0	85.03
LAMA(2018)	15.64	20.0	10.0	15.0	19.0	79.64
FDRemix	11.96	15.0	15.0	18.5	19.0	79.46
Mercury	13.07	20.0	4.0	20.0	20.0	77.07
LPRPG-P	13.78	7.0	0.0	19.5	11.0	51.28

$\mathscr{P}_{\mathcal{SG}}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
LPRPG-P	—	19.45	16.48	19.43	14.94	70.3
FDRemix	—	14.73	17.12	18.63	18.57	69.05
FDSS 2018	—	14.66	16.49	18.4	18.44	67.99
$\text{LAMA}_{\text{B}}(h_r)$	—	14.82	14.36	18.7	18.9	66.78
LAMA(2011)	—	13.95	13.57	17.78	18.29	63.6
IBaCoP2	—	16.03	10.7	17.23	18.61	62.57
LAMA(2018)	—	14.82	13.17	15.84	18.3	62.13
MIPlan	—	15.08	9.85	16.7	19.3	60.92
Mercury	—	14.83	7.78	17.39	18.29	58.29

$\mathscr{P}_{\mathcal{AO}}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
Mercury	16.61	20.0	19.0	—	20.0	75.61
FDSS 2018	16.4	17.0	20.0	—	19.0	72.4
LAMA(2018)	14.92	17.0	20.0	—	20.0	71.92
$\text{LAMA}_{\text{B}}(h_r)$	14.54	18.0	20.0	—	19.0	71.54
LAMA(2011)	14.36	17.0	20.0	—	20.0	71.36
FDRemix	16.37	16.0	20.0	—	18.0	70.37
MIPlan	14.03	16.0	15.0	—	20.0	65.03
IBaCoP2	13.21	15.0	16.0	—	19.0	63.21
LPRPG-P	13.42	1.0	19.0	—	12.0	45.42

$\mathscr{P}_{\mathcal{SB}}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
$\text{LAMA}_{\text{B}}(h_r)$	18.25	20.0	18.0	—	19.0	75.25
LAMA(2011)	18.18	20.0	15.5	—	18.0	71.68
Mercury	17.07	20.0	14.5	—	20.0	71.57
MIPlan	18.24	20.0	12.0	—	20.0	70.24
FDRemix	17.49	16.0	16.5	—	20.0	69.99
FDSS 2018	17.46	17.0	16.5	—	19.0	69.96
IBaCoP2	19.72	20.0	12.0	—	18.0	69.72
LAMA(2018)	17.21	20.0	13.33	—	17.0	67.54
LPRPG-P	8.21	14.0	15.5	—	7.0	44.71

$\mathscr{P}_{\mathcal{ST}}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
LAMA(2018)	13.23	11.0	—	—	20.0	44.23
$\text{LAMA}_{\text{B}}(h_r)$	14.28	10.0	—	—	19.0	43.28
FDSS 2018	16.17	8.0	—	—	19.0	43.17
IBaCoP2	16.81	10.0	—	—	16.0	42.81
LAMA(2011)	14.41	9.0	—	—	19.0	42.41
LPRPG-P	9.56	17.0	—	—	14.0	40.56
FDRemix	15.44	8.0	—	—	16.0	39.44
MIPlan	14.91	9.0	—	—	14.0	37.91
Mercury	12.78	4.0	—	—	12.0	28.78

Table 1: Temp caption

DOMINO	h^{blind}		h^{max}		$h^{\text{m\&s}}$		h^{cpdb}	
	Nebel	Our	Nebel	Our	Nebel	Our	Nebel	Our
Storage	24.78	57.0	29.2	45.0	32.50	24.0	23.10	57.0
Rovers	17.4	24.0	41.0	25.0	16.67	26.0	15.17	23.0
Trucks	18.84	24.0	23.19	25.0	n/a	25.0	n/a	25
TPP	—	47.0	—	45.0	—	47.0	–	40.0

Table 2: Coverage of our and Nebel compilation scheme on the IPC5 benchmarks set with additional instances with random sampled soft-trajectory constraints, A* search for optimal solution. Our results concerning the sampled instances are averaged.

Optimal Planning Results

Similarly to what to what has been done in [12], we have tested our scheme using some admissible heuristics which are h^{blind}, which assign 1 to all states except for goal states to which assign 0, the maximum heuristic h^{max}, the merge and shrink heuristic $h^{\text{M\&S}}$, and the canonical pattern database heuristic h^{cpdb} [**?**]. These heuristitcs guarantee the optimality of the solution found. Starting from the IPC5 domains, we generated, similar to what was done [12], simpler instances by randomly sampling subsets of the soft trajectory constraints. Starting from each instance we have generated five new instances with 1%, 5%, 10%, 20% and 40% of the (grounded) soft trajectory constraints while the hard goals have remained unchanged.

Since we do not have the instances that have been used in the aforementioned paper, we have generated, for each percentage of sampling preferences (except for 100 %), 3 sampled instances in order to average the obtained results and be able to compare with their results. All our experiments were conducted on a 2.00GHz Core Intel(R) Xeon(R) CPU E5-2620 machine with CPU-time and memory limits of 30 minutes and 8GiB, respectively, while the experiments reported in [12] are conducted on a Intel(R) Xeon(R) E5-2650v2 2.60GHz processors with 64GiB with one hour of CPU-time for the search and then our approach is penalized.

The results about this experiment and the comparison with the automata approach are shown Table 2. The results inherent to Openstacks have been excluded because it was not possible to find optimal plans even for the simplest instances.

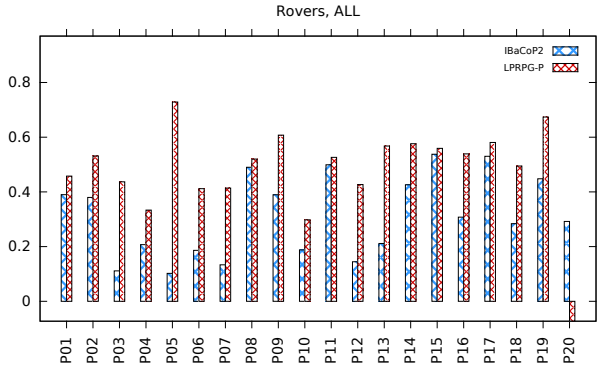
Conclusions

References

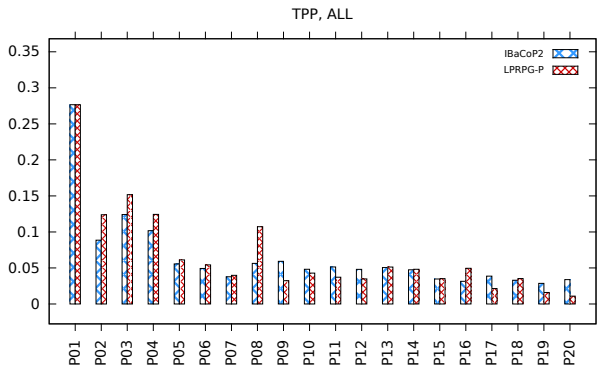
[1] The 2018 international planning competition (IPC) classical tracks. `https://ipc2018-classical.bitbucket.io`. Accessed: 2018-10-12.

[2] M. Briel, R. Sanchez, M. Do, and S. Kambhampati. Effective approaches for partial satisfaction (over-subscription) planning. In *Proceedings of Nineteenth National Conference on Artificial Intelligence (AAAI04)*, pages 562–569, 2004.

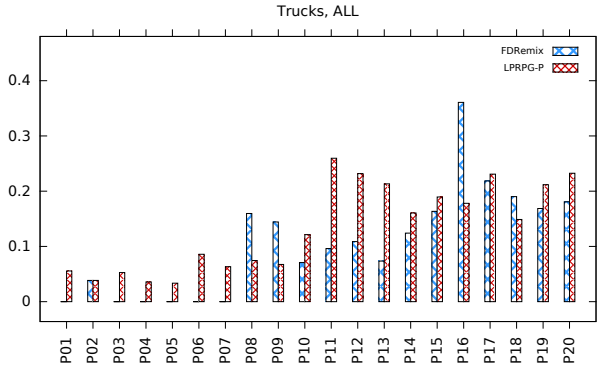
[3] I. Cenamor, T. De La Rosa, and F. Fernández. IBaCoP and IBaCoP planner. In *In Eighth International Plan-*



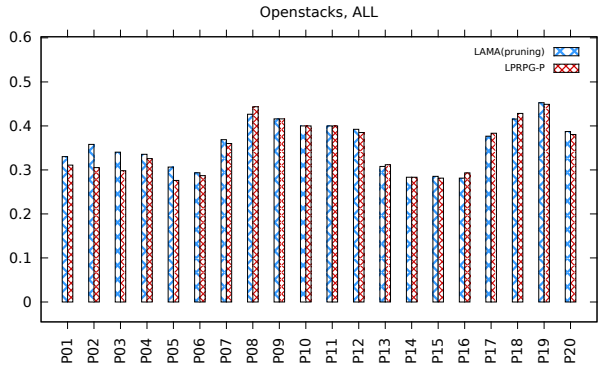
(a) Rovers



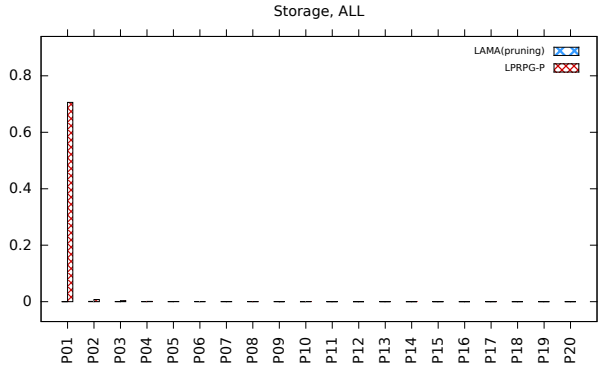
(b) TPP



(c) Trucks



(d) Openstacks



(e) Storage

Figure 1: Quality Comparison using α_{cost} for each domain. Each bar represents the α_{cost} of the best plan produced by the considered planner. The negative bar represents an instance which has not been solved or that has no preferences of that kind. From the top to bottom we have provided the results about α_{cost} calculated considering each kind of preferences for Rovers, TPP, Trucks, Openstacks and Storage.

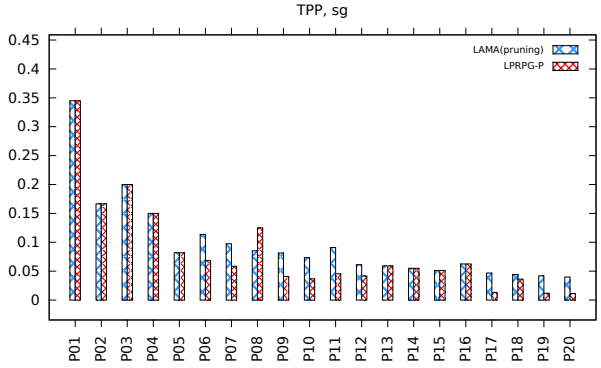
ning Competition Booklet (ICAPS-14), pages 35–38, 2014.

[4] Amanda Jane Coles and Andrew Coles. Lprpg-p: Relaxed plan heuristics for planning with preferences. In *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011)*, pages 26–33, 2011.

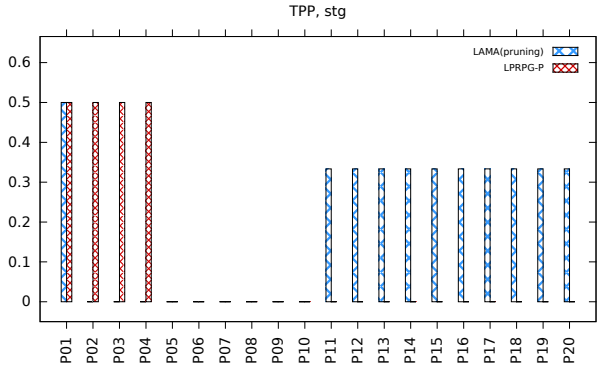
[5] M.B. Do and S. Kambhampati. Partial satisfaction (over-subscription) planning as heuristic search. In *Proceedings of Fifth International Conference on Knowledge Based Computer Systems (KBCS04)*, page mancanti, 2004.

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

[7] M. Katz and J. Hoffmann. Mercury planner: Push-



(a) $\mathcal{SG}$



(b) $\mathcal{ST}$

Figure 2: Each bar represents the α_{cost} of the best plan produced by the considered planner. The negative bar represents an instance which has not been solved or that has no preferences of that kind. From left to right we have provided the results about the α_{cost} calculated considering each kind of preferences, always, at-end (or soft goals), at-most-once, sometime-before and sometime preferences.

ing the limits of partial delete relaxation. In *In Eighth International Planning Competition Booklet (ICAPS-14)*, pages 43–47, 2014.

[8] S. Núnez, D. Borrajo, and C. Linares López. MIPlan and DPMPlan. In *In Eighth International Planning Competition Booklet (ICAPS-14)*, pages 13–16, 2014.

[9] F. Percassi, A. Gerevini, and H. Geffner. Improving plan quality through heuristics for guiding and pruning the search: A study using LAMA. In *Proceedings of the Tenth International Symposium on Combinatorial Search (SoCS 2017)*, pages 144–148, 2017.

[10] S. Richter and M. Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1), 2010.

[11] M. Vallati, L. Chrpa, M. Grześ, T. L. McCluskey, M. Roberts, and S. Sanner. The 2014 international planning competition: Progress and trends. *AI Magazine*, 36(3):90–98, 2015.

[12] Mattmller R. Wright B. and Nebel B.. Compiling away soft trajectory constraints in planning. In *In Proceedings of the Sixteenth Conference on Principles of Knowledge Representation and Reasoning (KR18)*, 2018.