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.

Cciccio

Introduction

Planning with preferences, also called "over-subscription planning" in (**?**; **?**; **?**), 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 (**?**) consists of *always preferences*, requiring that a certain condition should hold in *every* state reached by a plan. As discussed in (**?**; **?**; **?**; **?**), 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 (**?**). 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 (**?**)).

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 (**?**).

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

Operator-Preference Interactions

Compilation of Qualitative Preferences

Experimental Results

Experiments Description

We implemented the proposed compilation scheme and have evaluated 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, while in the other we focuse on the search of optmial plans.

Regarding the comparison in the context of the satisficing planning we have considered the following STRIPS+ planning system LAMA(**?**), Mercury (**?**), MIPlan (**?**), IBaCoP2 (**?**), which are some of the best performing planning system in IPC8 (**?**), and Fast Downward Stone Soup 2018(**?**), Fast Downward Remix(**?**) (abbreviated with FD-Remix), which are some of the best performing planning system in the

[1] Another compilation scheme using numerical fluents is considered in (**?**) 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.

last IPC9 (**?**) which have been compared with LPRPG-P (**?**), which is one of the performing planner which supports PDDL3 preferences.

As benchmark we have considered the five domains of the qualitative preference track of IPC5 (**?**) 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. The time for the preferences goal compilation was included in the 30 minutes.

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

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

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

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 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 test problems:

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

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 benchmark, while in the remaining subtables we have splitted the calculation of IPC quality score considering each kind of preference seperately. In the the header of each subtable we have reported the class of considered preferences.

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

Figures 1—**??** show the qualitative comparison in a more detailed way. In these histograms we have reported, for each istance of each domain, another quality measure, which we have denoted with α_{cost}. In particular we have evaluated the best performing planners according to the results showed in Table 1 which are LAMA, IBaCoP2 and LPRPG-P respectively. Each figure is associated with one of the considered domains.

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).

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

In Table 1, considering all kinds of preferences, we can observe that all the classical planning system, except for MIPlan and Mercury, perform overall better than LPRPG-P in term of IPC score while MIPlan and Mercury performs overall worse. The compilative approach seems at glance to be 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 while LAMA in Trucks. Also MIPlan get a well performance in Trucks but is penalized due to coverage (it solves only 15 instances out of 20).

NOTA (FP): commento TPP - punto di debolezza.

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 because, as shown in (**?**) (articolo pruning+softgoal), the compilation of soft goals can be sometime problematic. Indeed the part of Table 1, which concerns soft goals, cleary shows that LPRPG-P is overall more performing than the classical planners especially in TPP in term of satisfied soft goal.

Regarding Opentacks the tested planners achieve a comparable performance even if the classical planners are slightly penalized compared to LPRPG-P. Also in this case the classical planners have difficulty to satisfy soft goal as shown in Table 1.

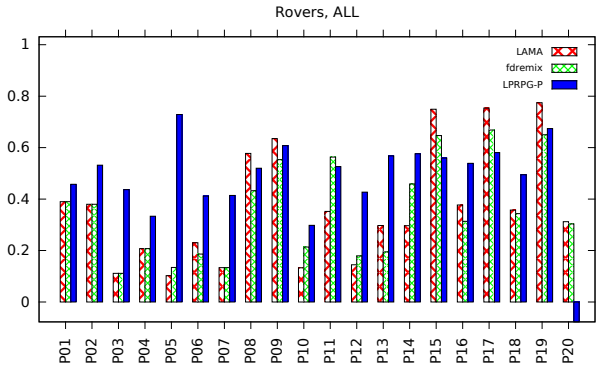
Looking at Figure 1a we can observe that LAMA and FD-Remix generally compute higher quality plans than LPRPG-P in Rovers, in particular they find better plan in 15 and 16 instances out of 20 respectively. Looking at Figure 1b we can observe that LAMA and FD-Remix compute lower quality plans that LPRPG-P for more than half of the instances. Both classical system work better than LPRPGP in

$\mathcal{P}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
LAMA	17.37	8.61	15.57	19.26	18.34	79.14
FD-Remix	17.24	7.1	17.76	18.96	16.26	77.32
FDSS 2018	16.95	7.03	17.16	18.66	17.19	76.99
IBaCoP2	18.9	9.68	10.0	17.82	15.78	72.19
LPRPG-P	10.81	18.74	7.07	19.68	12.95	69.25
MIPlan	16.9	8.8	9.23	17.32	14.47	66.72
Mercury	15.61	8.57	7.82	18.02	14.56	62.59
$\mathcal{A}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
LAMA	15.09	20.0	15.0	20.0	20.0	90.09
IBaCoP2	16.19	20.0	13.0	19.0	19.0	87.19
FDSS 2018	14.19	17.0	18.0	17.83	20.0	87.02
MIPlan	14.71	20.0	12.0	19.0	20.0	85.71
FD-Remix	12.64	15.0	15.0	18.5	19.0	80.14
Mercury	13.8	20.0	4.0	20.0	20.0	77.8
LPRPG-P	14.46	7.0	0.0	19.5	11.0	51.96
$\mathcal{SG}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
LPRPG-P	—	19.45	16.48	19.39	15.02	70.34
FD-Remix	—	14.73	17.12	18.59	18.63	69.07
FDSS 2018	—	14.66	16.49	18.36	18.49	68.01
LAMA	—	14.41	14.52	18.68	19.05	66.66
IBaCoP2	—	16.03	10.7	17.19	18.67	62.59
MIPlan	—	15.08	9.85	16.67	19.38	60.97
Mercury	—	14.83	7.78	17.36	18.35	58.32
$\mathcal{AO}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
Mercury	16.3	20.0	19.0	—	20.0	75.3
FDSS 2018	16.1	17.0	20.0	—	19.0	72.1
FD-Remix	16.07	16.0	20.0	—	18.0	70.07
LAMA	15.35	15.0	19.0	—	20.0	69.35
MIPlan	13.3	16.0	15.0	—	20.0	64.3
IBaCoP2	12.9	15.0	16.0	—	19.0	62.9
LPRPG-P	13.0	1.0	19.0	—	12.0	45.0
$\mathcal{SB}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
LAMA	18.09	20.0	18.0	—	19.0	75.09
Mercury	16.35	20.0	14.5	—	20.0	70.85
MIPlan	18.53	20.0	12.0	—	20.0	68.53
FD-Remix	15.83	16.0	18.5	—	20.0	68.33
FDSS 2018	15.8	17.0	18.5	—	19.0	68.3
IBaCoP2	18.01	20.0	12.0	—	18.0	68.01
LPRPG-P	7.41	14.0	15.5	—	7.0	43.91
$\mathcal{ST}$						
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
	15.76	11.0	—	—	20.0	46.76
FDSS 2018	16.15	8.0	—	—	20.0	44.15
IBaCoP2	16.78	10.0	—	—	17.0	43.78
LPRPG-P	9.53	17.0	—	—	15.0	41.53
FD-Remix	15.42	8.0	—	—	17.0	40.42
MIPlan	14.88	9.0	—	—	15.0	38.88
Mercury	12.76	4.0	—	—	13.0	29.76

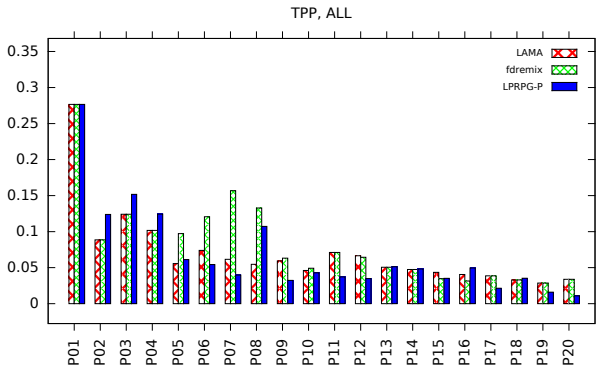
Table 1: Temp caption

smaller instances but they get worse as the size increases. Looking at Figure 1c we can say that LAMA and FD-Remix performs better for more than half of the instances, in particular they find better plan in 13 and 16 instances out of 20. Note that the classical planners get the optimal solution for some of the first seven instances. Looking at Figure 1d we can say that both approaches achieve a comparable performance even if LPRPG-P generally finds slightly better solutions than both classical competitors in 13 instances out of 20.

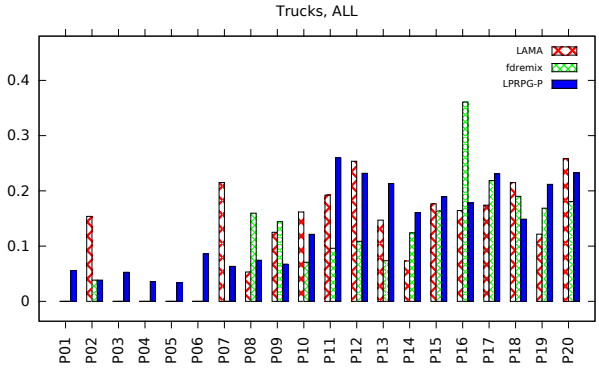
NOTA (FP): scrivi commento Storage.



(a) Rovers



(b) TPP

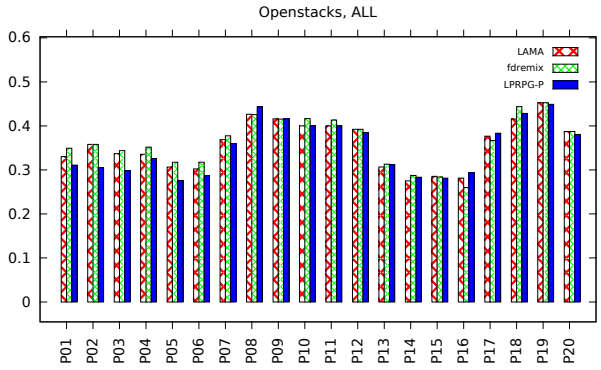


(c) Trucks

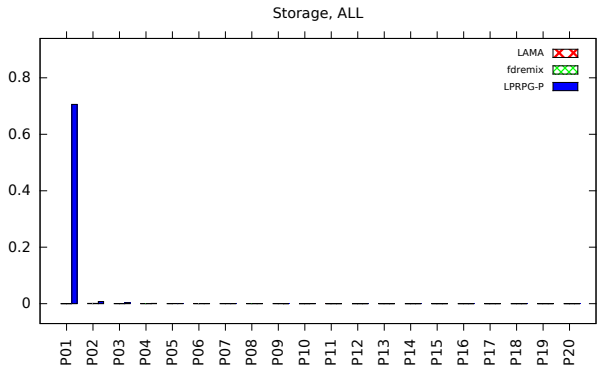
Optimal Planning Results

Similarly to what to what has been done in **??** (articolo NEBEL), we have tested our scheme using some admissible heuristics which are h^{blind}, $h^{\max}$ an $h^{\text{M\&S}}$ which guarantee us the optimality of the solution found. Starting from the IPC5 domains, we generated 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 (excpet for 100 %), five



(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.

sampled instances in order to average the obtained results. The results about this experiment 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

Domain	h^{blind}	h^{max}	$h^{\text{m\&s}}$
Storage	49.0	41.0	23.0
Rovers	23.0	23.0	23.0
TPP-p	36.0	33.0	35.0
Trucks	23.0	28.0	23.0
TOTAL	33.0	31.0	26.0

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