On Compiling Away PDDL3 Soft Trajectory Constraints without Using Automata

Francesco Percassi, Alfonso E. Gerevini
Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia
Via Branze 38, I-25123 Brescia, Italy

Abstract

We address the problem of propositional planning extended with the class of soft temporally extended goals supported in PDDL3, also called qualitative preferences since IPC-5. 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. We propose and evaluate a compilation approach that extends previous work on compiling soft reachability goals and always goals to the full set of PDDL3 qualitative preferences. This approach directly compiles qualitative preferences into propositional planning without using automata to represent the trajectory constraints. Moreover, since no numeric fluent is used, it allows many existing STRIPS planners to immediately address planning with preferences without changing their algorithms or code. An experimental analysis presented in the paper evaluates the performance of state-of-the-art propositional planners using our compilation of qualitative preferences. The results indicate that the proposed approach is highly competitive with respect to current planners that natively support the considered class of preferences, as well as with a recent automata-based compilation approach.

Introduction

Soft state-trajectory constraints are temporally extended goals that in PDDL3 are also called preferences (Gerevini et al. 2009). In planning with preferences, the quality of the solution plans depends on the soft goals and preferences that are satisfied by the plans.

PDDL3 supports an useful set of types of preferences expressed through certain modal operators, and in particular the "qualitative preferences" of types *at-end* (soft goals), *sometime*, *sometime-before*, *at-most-once*, and *sometime-after*. One of the competition tracks of IPC-5 was centered on qualitative preferences, and since then several systems addressing this class of planning problems have been developed. Most of these systems represent state-trajectory constraints as automata, that are compiled into the problem operators and states or are handled by the planning algorithm.

In this paper we study propositional planning with qualitative preferences through a compilation approach. In particular, we extend previous work on compiling soft goals (Percassi, Gerevini, and Geffner 2017) and soft always goals (Ceriani and Gerevini 2015) to deal with the full set of PDDL3 qualitative preferences that don't involve explicit numeric time. Differently from most other systems, our method directly compiles qualitative preferences into propositional planning with action costs without using automata to represent the state-trajectory constraints, and without using numeric fluents, as commonly done in other approaches. Propositional planning with action costs is supported by many powerful planners, and the proposed compilation method allows them to immediately support (through the compiled problems) qualitative preferences with no change to their algorithms and code.

The paper also presents an experimental analysis evaluating the performance of state-of-the-art propositional planners supporting action costs using our compilation of soft state-trajectory constraints. The results indicate that the proposed approach is highly competitive with a state-of-the-art planner that natively supports qualitative preference, as well as with a recent (automata-based) compilation approach.

We start with a brief description of the main related work; then, after the necessary preliminaries, we describe the compilation method in detail; finally, we present the experimental analysis and give the conclusions.

Related Work

The structure of the proposed compilation was inspired by the work of Keyder and Geffner (2009) on compiling soft goals into STRIPS with action costs (in the following denoted with STRIPS+). Keyder and Geffner's compilation scheme is considerably simpler than ours because it does not consider the different kinds of interference between actions and the types of PDDL3 preferences treated in our compilation.

The most prominent existing planners supporting PDDL3 preferences are HPlan-P (Baier, Bacchus, and McIlraith 2009; Baier and McIlraith 2006), which won the "qualitative preference" track of IPC-5, MIPS-XXL (Edelkamp 2006; Edelkamp, Jabbar, and Naizih 2006) and the more recent LPRPG-P (Coles and Coles 2011) with its variants (Coles and Coles 2013). These 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 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 numeric fluents.

The compilation of LTL goal formulas by Cresswell and Coddington (2004) and Rintanen (2000) handle *hard* temporally extended goals, instead of preferences, i.e., every specified 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 formulae (while we deal with arbitrary CNF formulas), and it appears that extending it to handle more general formulas requires substantial new techniques. Bayer and McIlaraith (2006) observed that Crosswell and Coddington's approach suffers exponential blow up problems and is less efficient than HPlan-P.

Other works that are related to ours are the compilation schema in (Ceriani and Gerevini 2015), which however supports only soft goals and always preferences, and the recent schema by Wright, Mattmueller and Nebel (2018) (here abbreviated WMN), which supports a class of soft state-trajectory constraints richer than the PDDL3 qualitative preferences. WMN compiles soft trajectory constraints into conditional effects and state dependent action costs using $\mathsf{LTL_f}$ (De Giacomo, De Masellis, and Montali 2014) and automata. Besides the use of automata, other main differences are the following ones. Our compilation has less additional fluents (at most two for each preference against one for each automaton state in WMN), and WMN use numeric fluents while our compilation avoids them. Both WMN and our compilations introduce additional conditional effects, that in WMN are generated from the automaton transitions without considering the structure (preconditions and effects) of the operator to which they are added, and they are the same for every operator. On the contrary, our compilation exploits the operator structure to generate less conditional effects, each of which with a condition specialized for the particular operator where it is added. This leads to a more compact compiled problem, that hence can be easier to solve for a classical planner.

To help the planner search WMN's "action penalty compilation" introduces some artificial negative penalties (action costs) in the compilation, which however are supported by few planners. WMN's "positively shifted costs" technique allows to use only zero and positive costs, but it could loose optimality guarantees since an optimal compiled plan might not correspond to an optimal plan for the original problem.

Preliminaries, Background and Notation

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 set of positive literals 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 $s_0 = I$, $\mathit{Pre}(a_i) \subseteq s_i$ and $s_{i+1} = s_i \cup \mathit{Eff}(a_i)^+ \setminus \{p \mid \neg p \in \mathit{Eff}(a_i)^-\}$, 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.

$$
\begin{aligned}
&\langle s_0, s_1, \ldots, s_n\rangle \models (\textit{at end } \phi) \quad \text{iff} \quad s_n \models \phi\\
&\langle s_0, s_1, \ldots, s_n\rangle \models (\textit{always } \phi)\\
&\qquad \text{iff} \quad \forall i : 0 \le i \le n \cdot s_i \models \phi\\
&\langle s_0, s_1, \ldots, s_n\rangle \models (\textit{sometime } \phi)\\
&\qquad \text{iff} \quad \exists i : 0 \le i \le n \cdot s_i \models \phi\\
&\langle s_0, s_1, \ldots, s_n\rangle \models (\textit{at-most-once } \phi)\\
&\qquad \text{iff} \quad \forall i : 0 \le i \le n \cdot \text{ if } s_i \models \phi \text{ then}\\
&\qquad\qquad \exists j : j \ge i \cdot \forall k : i \le k \le j \cdot s_k \models \phi \text{ and}\\
&\qquad\qquad \forall k : k > j \cdot s_k \models \neg\phi\\
&\langle s_0, s_1, \ldots, s_n\rangle \models (\textit{sometime-after } \phi\ \psi)\\
&\qquad \text{iff} \quad \forall i \cdot 0 \le i \le n \cdot \text{if } s_i \models \phi \text{ then } \exists j : i \le j \le n \cdot s_j \models \psi\\
&\langle s_0, s_1, \ldots, s_n\rangle \models (\textit{sometime-before } \phi\ \psi)\\
&\qquad \text{iff} \quad \forall i \cdot 0 \le i \le n \cdot \text{if } s_i \models \phi \text{ then } \exists j : 0 \le j < i \cdot s_j \models \psi
\end{aligned}
$$

Figure 1: Semantics of the basic modal operators in PDDL3.

PDDL3 introduced state-trajectory constraints, which are modal logic expressions expressible using LTL that have to be true in the state trajectory produced by the execution of a plan. Let $\langle s_0, s_1, \ldots, s_n\rangle$ be the sequence of states in the state trajectory of a plan. Figure 1 defines PDDL3 qualitative state-trajectory constraints, i.e., constraints that do not involve numbers. Here ϕ and ψ are first-order formulae that, without loss of generality, we assume are translated by a preprocessing step into equivalent grounded CNF-formulae; e.g., $\phi = \phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_n$ where ϕ_i ($i = 1 \ldots n$) is a clause formed by literals over the problem fluents. These constraints can be either soft or hard. When they are soft they are called *qualitative preferences*.

We will use the following notation: $\mathcal{A}, \mathcal{SB}, \mathcal{SA}, \mathcal{ST}, \mathcal{AO}$, $\mathcal{G}$ denote the classes of qualitative preferences of type always, sometime-before, sometime-after, sometime, at-most-once and soft goal, respectively, for a given planning problem; A_ϕ, $\mathsf{SB}_{\phi,\psi}$, $\mathsf{SA}_{\phi,\psi}$, ST_ϕ, AO_ϕ, G_ϕ denote a particular preference over $\mathcal{A}$, $\mathcal{SB}$, $\mathcal{SA}$, $\mathcal{ST}$, $\mathcal{AO}$, $\mathcal{G}$, respectively, involving formulae ϕ and ψ; if a plan π satisfies a preference P, we write $\pi \models P$.

Definition 1. *A* STRIPS+ *problem with preferences* Π *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;*
- $\mathscr{P} = \{\mathscr{P}_\mathcal{A} \cup \mathscr{P}_\mathcal{SB} \cup \mathscr{P}_\mathcal{SA} \cup \mathscr{P}_\mathcal{ST} \cup \mathscr{P}_\mathcal{AO} \cup \mathscr{P}_\mathcal{G}\}$ *is the set of the preferences of* Π *where* $\mathscr{P}_\mathcal{A} \subseteq \mathcal{A}$, $\mathscr{P}_\mathcal{SB} \subseteq \mathcal{SB}$, $\mathscr{P}_\mathcal{SA} \subseteq \mathcal{SB}$, $\mathscr{P}_\mathcal{ST} \subseteq \mathcal{ST}$, $\mathscr{P}_\mathcal{AO} \subseteq \mathcal{AO}$ *and* $\mathscr{P}_\mathcal{G} \subseteq \mathcal{G}$*;*
- u *is an utility function* $u : \mathscr{P} \to \mathbb{R}_0^+$.

STRIPS+ with preferences will be indicated with STRIPS+P.

Definition 2. *Let* Π *be a* STRIPS+P *problem with preferences* $\mathscr{P}$. *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 (2009). 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 *violation cost* of a preference is the value of its utility.

In order to make the presentation of our compilation approach more compact, we introduce some further notation. 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 $L(\phi_i)$.

Given an operator o of a STRIPS+P problem, $Z(o)$ denotes the set of literals

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

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

The state where an operator o is applied is indicated with s and the state resulting from the application of o with s'.

Definition 3. *Given an operator o and a CNF formula* $\phi = \phi_1 \wedge ... \wedge \phi_n$, *the set* $C_\phi(o)$ *of clauses of* ϕ *that o makes* **certainly true** *in* s' *is defined as:*

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

Given a clause $\phi_i = l_1 \vee ... \vee l_{m_i}$ of ϕ, condition $|L(\phi_i) \cap Z(o)| > 0$ in Definition 3 requires that at least a literal of ϕ_i is in $Z(o)$, and thus that clause ϕ_i is true in s'.

Definition 4. *Given an operator o and a CNF formula* $\phi = \phi_1 \wedge ... \wedge \phi_n$, *we say that o* ***can make*** ϕ ***true*** *if* $|C_\phi(o)| > 0$ *and, for each clause* ϕ_i *of* ϕ *not in* $C_\phi(o)$, $\overline{L}(\phi_i) \not\subseteq Z(o)$.

The first condition in Definition 4 requires that there is at least a clause of ϕ that is certainly true in s' independently from s, while the second requires that the clauses that are not certainly true in s' are not falsified by o according to $Z(o)$.

Definition 5. *Given an operator o and a CNF formula* ϕ, *we say that o* ***can make*** ϕ ***false*** *in* s' *if there is a clause* ϕ_i *of* ϕ *such that*

1. $|\overline{L}(\phi_i) \cap Z(o)| > 0 \wedge \overline{L}(\phi_i) \subset Z(o)$
2. $|L(\phi_i) \cap Z(o)| = 0$
3. $\overline{L}(\phi_i) \not\subseteq Pre(o)$.

The conditions of Definition 5 require that at least a clause of ϕ (1) has some (but not all) literals which are falsified after the execution of o, (2) has no literal that is certainly true in s', and (3) is not already false in s.

Operator-Preference interferences

Operators and preferences may have different kinds of interference, that we have to deal with in their compilation. We say that an operator o is *neutral* for a preference P if its execution in a plan can not affect the satisfaction of P in the state trajectory of the plan. Otherwise, depending on the preferences type of P, o can behave as a *violator*, a *threat* or a *potential support* of P. Informally, a violator falsifies the preference, a threat may falsify it (depending on s), and a potential support may satisfy it over the full state trajectory of the plan. In the following, more formal definitions of these interferences are given for each type of preference.

Operators Affecting Always Preferences

An always preference A_ϕ is violated if ϕ is false in any state on the plan state trajectory. Hence, if ϕ is false in every state s' generated by an operator o, then o is a violator of A_ϕ.

Definition 6. *Given an operator o and an always preference* A_ϕ *of a* STRIPS+P *problem, o is a* ***violator*** *of* A_ϕ *if there is a clause* ϕ_i *of* ϕ *such that: (1)* $\overline{L}(\phi_i) \subseteq Z(o)$, *and (2)* $\overline{L}(\phi_i) \not\subseteq Pre(o)$.

Operator o is a threat of A_ϕ if it is not a violator, its effects make false at least a literal of a clause ϕ_i of ϕ, and its preconditions don't entail $\neg\phi_i$ (otherwise A_ϕ would be already false in s). Such clause ϕ_i is a *threatened clause* of A_ϕ.

Definition 7. *Given an operator o and an always preference* A_ϕ *of a* STRIPS+P *problem, o is a* ***threat*** *of* A_ϕ *if it is not a violator and it can make* ϕ *false.*

The set of clauses of a preference A_ϕ threatened by o is denoted with $TC(o, \phi)$.

An operator is neutral for A_ϕ if it makes ϕ true, does not falsify any clause of ϕ, or it can be applied only in states where ϕ is false.

Definition 8. *Given an operator o and an always preference* A_ϕ *of a* STRIPS+P *problem, o is* ***neutral*** *for* A_ϕ *if:*

1. *for all clauses* ϕ_i *of* ϕ, $|L(\phi_i) \cap Z(o)| > 0$ *or* $|\overline{L}(\phi_i) \cap Z(o)| = 0$, *or*
2. *there exists a clause* ϕ_i *of* ϕ *such that* $\overline{L}(\phi_i) \subseteq Pre(o)$.

Example. Operator $o = \langle\{\}, \{\neg a, \neg b\}\rangle$ is a threat for A_{ϕ_1}, a violator of A_{ϕ_2} and neutral for A_{ϕ_3} where $\phi_1 = (b \vee c) \wedge d$, $\phi_2 = a \vee b$ and $\phi_3 = d$.

Operators Affecting Sometime Preferences

A sometime preference ST_ϕ is violated if ϕ is never true on the plan state trajectory. Hence, if the state s' generated by an operator o makes ϕ true, then o is a potential support of ST_ϕ.

Definition 9. *Given an operator o and a sometime preference* ST_ϕ *of a* STRIPS+P *problem, o is a* ***potential support*** *of* ST_ϕ *if o can make true* ϕ, *otherwise the operator is* ***neutral*** *for* ST_ϕ.

Example. Operator $o = \langle\{\}, \{\neg b\}\rangle$ is a potential support of ST_{ϕ_1} and neutral for ST_{ϕ_2} where $\phi_1 = (c \vee \neg b) \wedge a$ and $\phi_2 = c$.

Operators Affecting Sometime-before Preferences

A sometime-before preference $\mathsf{SB}_{\phi,\psi}$ is violated if ϕ becames true before ψ has been made true on the plan state trajectory. Hence, if operator o can make ψ true in s', then

o is a potential support of $\mathsf{SB}_{\phi,\psi}$. Depending on the state s where such operator is applied, it can be an actual support or neutral for $\mathsf{SB}_{\phi,\psi}$.

Definition 10. *Given an operator o and a sometime-before preference $P = \mathsf{SB}_{\phi,\psi}$ of a* STRIPS+P *problem, o is a **potential support** of P if o can make ψ true.*

If an operator o can make ϕ true in s', then o is a threat of $\mathsf{SB}_{\phi,\psi}$. Depending on the state s where it is applied, such operator can be a violator or neutral for $\mathsf{SB}_{\phi,\psi}$.

Definition 11. *Given an operator o and a sometime-before preference $P = \mathsf{SB}_{\phi,\psi}$ of a* STRIPS+P *problem, o is a **threat** of P if o can make true ϕ.*

Definition 12. *Given an operator o and a sometime-before preference $P = \mathsf{SB}_{\phi,\psi}$ of a* STRIPS+P *problem, o is **neutral** for P if o is neither a potential support nor a threat for P.*

Example. Operator $o = \langle\{\}, \{b\}\rangle$ is a potential support of $\mathsf{SB}_{\phi_1,\psi_1}$, a threat of $\mathsf{SB}_{\phi_2,\psi_2}$ and a operator neutral for $\mathsf{SB}_{\phi_3,\psi_3}$ where $\phi_1 = c$, $\psi_1 = (a \vee b) \wedge d$, $\phi_2 = (c \vee b) \wedge d$, $\psi_2 = e$, $\phi_3 = d$ and $\psi_3 = e$.

Operators Affecting At-most-once Preferences

A preference AO_ϕ is violated if ϕ if ϕ becames true more than once in the state trajectory. Thus, if an operator o can make ϕ true in s', then o is a threat of AO_ϕ.

Definition 13. *Given an operator o and an at-most-once preference $P = \mathsf{AO}_\phi$ of a* STRIPS+P *problem, o is a **threat** of P if o can make true ϕ, otherwise o is **neutral** for AO_ϕ.*

Example. Operator $o = \langle\{\}, \{\neg b\}\rangle$ is a threat of AO_{ϕ_1} and neutral for AO_{ϕ_2} where $\phi_1 = (c \vee \neg b) \wedge d$ and $\phi_2 = d$.

Operators Affecting Sometime-After Preferences

A sometime-after preference $\mathsf{SA}_{\phi,\psi}$ is violated if ϕ becomes true in a state without ψ becoming true in a succeeding state on the plan state trajectory. Hence, if the state s' generated by an operator o can make ϕ true, then o threats $\mathsf{SA}_{\phi,\psi}$ because ψ could be false in s'; o threats the preference also if it can make ψ false, because ϕ could be true in s'.

Definition 14. *Given an operator o and a sometime-after preference $P = \mathsf{SA}_{\phi,\psi}$ of a* STRIPS+P *problem, o is a **threat** of P if o can make ϕ true or ψ false.*

If an operator o can make ψ true in s', then it is a potential support of $\mathsf{SA}_{\phi,\psi}$ because if the preference is temporarily violated in the plan state trajectory up to s, then o could make it satisfied.

Definition 15. *Given an operator o and a sometime-after preference $P = \mathsf{SA}_{\phi,\psi}$ of a* STRIPS+P *problem, o is a **potential support** of P if o can make ψ true.*

Definition 16. *Given an operator o and a sometime-after preference $P = \mathsf{SA}_{\phi,\psi}$ of a* STRIPS+P *problem, o is **neutral** for P if o is neither a threat nor a potential support of P.*

Example. Operator $o = \langle\{\}, \{\neg a, b\}\rangle$ is a potential support of $\mathsf{SA}_{\phi_1,\psi_1}$, a threat of $\mathsf{SA}_{\phi_2,\psi_2}$ and $\mathsf{SA}_{\phi_3,\psi_3}$ and neutral for $\mathsf{SA}_{\phi_4,\psi_4}$, where $\phi_1 = c$, $\psi_1 = \neg a \wedge c$, $\phi_2 = \neg a \wedge c$, $\psi_2 = c$, $\phi_3 = \neg d$, $\psi_3 = \neg b \vee c$, $\phi_4 = c$ and $\psi_4 = d$.

Compilation of Qualitative Preferences

In this section we describing the general compilation scheme of a STRIPS+P Π problem. First we compile Π into a problem with conditional effects (and possibly also disjunctive preconditions), which can then be compiled away obtaining a STRIPS+ problem equivalent to Π, where problem equivalence is defined as in (Keyder and Geffner 2009). We will use $O_{neutral}$ to denote the set of the problem operators that are neutral for all preferences, and $P_{affected(o)}$ to denote the set of all problem preferences for which o is not neutral.

Since the compilation of soft goals is the same as in (Keyder and Geffner 2009), we omit its description, and we focus on the other types of preferences. Moreover, we use a preprocessing step to: (a) filter out from $\mathscr{P}$ all preferences of type $\mathcal{A}$ and $\mathcal{SB}$ that are falsified in the initial state and all preferences of type $\mathcal{ST}$ that are satisfied in it; (b) initialize the plan cost as the sum of the costs of the removed unsatisfied preferences.

For a STRIPS+P problem $\Pi = \langle F, I, O, G, \mathscr{P}, c, u\rangle$, the compiled problem of Π is $\Pi' = \langle F,' I', O', G', c'\rangle$ where:

- $F' = F \cup V \cup S \cup C' \cup \overline{C'} \cup \{\textit{normal-mode}, \textit{end-mode}\}$
- $I' = I \cup \overline{C'} \cup V_{\mathcal{ST}} \cup V_{\mathcal{SA}} \cup S_{\mathcal{AO}} \cup \{\textit{normal-mode}\}$
- $G' = G \cup C'$
- $O' = \{\textit{collect}(P), \textit{forgo}(P) \mid P \in \mathscr{P}\} \cup \{\textit{end}\} \cup \{\textit{comp}(o, \mathscr{P}) \mid o \in O\}$
- $\textit{forgo}(P) = \langle\{\textit{end-mode}, P\textit{-violated}, \overline{P'}\}, \{P', \neg\overline{P'}\}\rangle$
- $\textit{collect}(P) = \langle\{\textit{end-mode}, \neg P\textit{-violated}, \overline{P'}\}, \{P', \neg\overline{P'}\}\rangle$
- $\textit{end} = \langle\{\textit{normal-mode}\}, \{\textit{end-mode}, \neg\textit{normal-mode}\}\rangle$
- $\textit{comp}(o, \mathscr{P})$ is the function translating operator o according to Definition 17
- $c'(o') = \begin{cases} u(P) & \text{if } o' = \textit{forgo}(P) \\ c(o') & \text{if } o' = \textit{comp}(o, \mathscr{P}) \\ 0 & \text{otherwise} \end{cases}$
- $V = \{P\textit{-violated} \mid P \in \mathscr{P}\}$
- $S = \{\phi\textit{-seen} \mid \mathsf{AO}_\phi \in \mathscr{P}_{\mathcal{AO}}\} \cup \{\psi\textit{-seen} \mid \mathsf{SB}_{\phi,\psi} \in \mathscr{P}_{\mathcal{SB}}\}$
- $S_{\mathcal{AO}} = \{\phi\textit{-seen} \mid \mathsf{AO}_\phi \in \mathscr{P}_{\mathcal{AO}} \wedge I \models \phi\}$
- $V_{\mathcal{ST}} = \{P\textit{-violated} \mid P = \mathsf{ST}_\phi, I \models \neg\phi\}$
- $V_{\mathcal{SA}} = \{P\textit{-violated} \mid P = \mathsf{SA}_{\phi,\psi}, I \models \phi \wedge \neg\psi\}$
- $C' = \{P' \mid P \in \mathscr{P}\}$ and $\overline{C'} = \{\overline{P'} \mid P \in \mathscr{P}\}$.

The *collect* and *forgo* actions can only appear at the end of the plan. For each preference P the compilation of Π into Π' adds a dummy hard goal P' that is false in the initial state I'; P' can be achieved either by action *collect*(P), that has cost 0 but requires P to be satisfied, or by action *forgo*(P), that has cost equal to the utility of P and can be performed only if P is false (P*-violated* is true in the goal state). For each P, exactly one of *collect*(P) and *forgo*(P) appears in the plan.

The P*-violated* literals in the compiled initial state I' are used to consider each $\mathcal{ST}$ and $\mathcal{SA}$ preference that is not satisfied in I violated until an operator supporting them is inserted into the plan; the ϕ*-seen* literals in I' are necessary to capture the violation of the corresponding $\mathcal{AO}$ preference when an operator makes the preference formula true for the second time in the state trajectory.

Function $comp(o, \mathscr{P})$ transforms an original operator o into the equivalent compiled operator o' with an additional precondition forcing it to appear before the *forgo* and *collect* operators. Regarding the effects of o', if $o \in O_{neutral}$, they are the same of o; otherwise, $comp(o, \mathscr{P})$ extends the effects of o in o' with a set of conditional effects for each preference affected by o. The definition of such additional effects depends on the type of the affected preference, and on how o interferes with it; this is detailed below.

Definition 17. *Given an operator o the corresponding compiled operator is defined using the following function:*

$$Pre(o') = Pre(o) \cup \{normal\text{-}mode\}$$
$$Eff(o') = Eff(o) \cup \bigcup_{P \in P_{affected}(o)} \mathcal{W}(o, P)$$

where $\mathcal{W}(o, P)$ is the set of conditional effects concerning the affected preference P (if any).

In the following, ϕ_i denotes a clause of ϕ and ψ_i a clause of ψ; a set of formulas is interpreted as their conjunction.

Conditional Effects for $\mathcal{A}$ Preferences

The conditional effects for a compiled operator affecting a preference A_ϕ are defined as follows, where $\overline{AA}(o)_{\phi_i}$ is the literal-complement of the subset of literals in $L(\phi_i)$ that are *not* falsified by o, i.e., $\overline{AA}(o)_{\phi_i} = \overline{L}(\phi_i) \setminus \{\overline{L}(\phi_i) \cap Z(o)\}$.

Definition 18. *Given a preference $P = \mathsf{A}_\phi$ and an operator o affecting it, the conditional effect set $\mathcal{W}(o, P)$ in the compiled version o′ of o (according to Definition 17) is:*

$$\mathcal{W}(o,P) = \begin{cases} \{when\ (cond(o,P))\ (P\text{-}violated)\} \\ \quad \text{if } o \text{ is a threat of } P \\ \{when\ (\top)\ (P\text{-}violated)\} \\ \quad \text{if } o \text{ is a violator for } P \end{cases}$$

where $cond(o, P) = \bigvee_{\phi_i \in TC(o,P)} (l_1 \wedge \ldots \wedge l_q)$
and $\{l_1, \ldots, l_q\} = \overline{AA}(o)_{\phi_i}$ ($TC(o, P)$ is defined above).

For each affected preference $P = \mathsf{A}_\phi$, o' has a (conditional) effect P-*violated* with a condition depending on how A_ϕ is affected: if o is a violator, then the condition is always true; if o is a threat, the condition checks that there exists at least a clause of ϕ that is certainly false in s' – this is the case if there is at least a threatened clause whose literals that are not falsified in s' are false in s.

Example. Consider the operator $o = \langle\{b\}, \{\neg a, \neg c\}\rangle$ and preference A_ϕ with $\phi = (a \vee b) \wedge (c \vee d \vee e)$. The second clause of ϕ is threatened by o, and $cond(o, \mathsf{A}_\phi) = \overline{AA}(o)_{c \vee d \vee e} = \{\neg d, \neg e\}$.

Conditional Effects for $\mathcal{ST}$ Preferences

The conditional effects for a compiled operator affecting a preference ST_ϕ are defined as follows.

Definition 19. *Given a preference $P = \mathsf{ST}_\phi$ and an operator o that potentially supports it, the conditional effect set in the compiled version o′ of o (according to Definition 17) is:*

$$\mathcal{W}(o, P) = \{when(cond(o, P))\ (\neg P\text{-}violated)\}$$

where $cond(o, P) = \{\phi_i \mid \phi_i \notin C_\phi(o)\}$.

As described above, for each $P = \mathsf{ST}_\phi$, P-*violated* holds in the compiled initial state, and a potential support o of P makes ϕ true when all clauses of ϕ not in $C_\phi(o)$ hold in s, where $C_\phi(o)$ is the set of clauses of ϕ that are certainly true in s'. If this condition holds in s, then o' falsifies P-*violated*.

Conditional Effects for $\mathcal{SB}$ Preferences

The conditional effects for a compiled operator affecting a preference $\mathsf{SB}_{\phi,\psi}$ are defined as follows.

Definition 20. *Given a preference $P = \mathsf{SB}_{\phi,\psi}$ and an operator o affecting it, the conditional effect set $\mathcal{W}(o, P)$ in the compiled version o′ of o (according to Definition 17) is:*

$$\mathcal{W}(o,P) = \begin{cases} \{when\ (cond_S(o,P))\ (\psi\text{-}seen)\} \\ \quad \text{if } o \text{ is a potential support of } P \\ \{when\ (cond_T(o,P))\ (P\text{-}violated)\} \\ \quad \text{if } o \text{ is a threat of } P \end{cases}$$

where:

- $cond_S(o, P) = \{\psi_i \mid \psi_i \notin C_\psi(o)\}$
- $cond_T(o, P) = \{\neg\psi\text{-}seen\} \cup \{\phi_i \mid \phi_i \notin C_\phi(o)\}$.

An operator o affecting a preference $P = \mathsf{SB}_{\phi,\psi}$ can behave as a (a) potential support of P, (b) a threat of P, or (c) both. These cases are captured by the two conditional effects of o' in Definition 20.

In case (a), if all clauses that are not certainly true in s' (i.e., $cond_S(o, P)$) hold in s, then ψ is true in s', and o' keeps track of this by making ψ-*seen* true. In case (b), if ψ has never been true in the state-trajectory up to s and all clauses of ϕ that are not certainly true (i.e., $cond_T(o, P)$) hold in s, then P is violated by o and o' makes P-*violated* true in s'. In case (c), if the conditions of both conditional effects hold, P is violated because ψ is made true simultaneously with ϕ.

Conditional effects for $\mathcal{AO}$ Preferences

The conditional effects for a compiled operator threatening a preference AO_ϕ are defined as follows.

Definition 21. *Given a preference $P = \mathsf{AO}_\phi$ and an operator o that threats P, the conditional effect set $\mathcal{W}(o, P)$ in the compiled version o′ of o (according to Definition 17) is:*

$$\mathcal{W}(o,P) = \{when\ (cond_N(o,P))\ (\phi\text{-}seen)) \\ when\ (cond_T(o,P)))\ (P\text{-}violated)\}$$

where:

- $cond_N(o, P) = \{\neg\phi\text{-}seen\} \cup \{\phi_i \mid \phi_i \notin C_\phi(o)\}$
- $cond_T(o, P) = \{\phi\text{-}seen\} \cup \{\phi_i \mid \phi_i \notin C_\phi(o)\} \cup \{\bigvee_{\phi_i \in C_\phi(o)} (\neg l_1 \wedge ... \wedge \neg l_q) \mid \{l_1, ..., l_q\} = L(\phi_i)\}$.

If an operator o affecting $P = \mathsf{AO}_\phi$ makes ϕ true for the first time in the state trajectory (i.e., $\text{cond}_N(o, P)$ holds in s), then the first conditional effect of o' keeps track that ϕ has become true. Otherwise, if (1) ϕ was true in any state before s', (2) the execution of o in s makes ϕ true , and (3) ϕ was false before (i.e., the three condition sets in $\text{cond}_T(o, P)$), then o violates P and o' makes P-*violated* true.

Conditional effects for $\mathcal{SA}$ Preferences

The conditional effects for a compiled operator affecting a preference SA_ϕ preference are defined as follows.

Definition 22. *Given a preference $P = \mathsf{SA}_\phi$ and an operator o that affects P, the conditional effect set $\mathcal{W}(o, P)$ in the compiled version o' of o (according to Definition 17) is:*

$$\mathcal{W}(o,P) = \begin{cases} \{when\ (cond_T(o,P))\ (P\text{-}violated)\} \\ \quad \textit{if } o \textit{ is a threat of } P \\ \{when\ (cond_S(o,P))\ (\neg P\text{-}violated)\} \\ \quad \textit{if } o \textit{ is a support of } P \end{cases}$$

where:

- $cond_T(o,P) = \{R(o,\phi), R(o,\neg\psi)\}$
- $cond_S(o,P) = \{R(o,\psi)\}$
- $R(o,\phi) = \begin{cases} \top \textit{ if } \forall \textit{ clause } \phi_i \textit{ of } \phi, |Z(o) \cap L(\phi_i)| > 0, \\ \bigwedge_{\phi_i \notin C_\phi(o)} (l_1 \vee ... \vee l_q) \textit{ where } \{l_1, ..., l_q\} \\ \quad \textit{is the set of literals of } \phi_i \textit{ not falsified by } o \\ \textit{otherwise.} \end{cases}$

An operator o affecting $P = \mathsf{SA}_{\phi,\psi}$ can behave as (a) a threat of P, (b) a potential support of P, or (c) both. In case (a) the effect condition captures the fact that o generates a state s' where ϕ is true and ψ false, (temporarily) violating P. In (b) the condition captures the fact that o generates s' in which ψ is true, and so s' cannot violate P. In (c) only one of the two conditional effects can hold in s' because their conditions are mutually exclusive. Note that $R(o,\phi)$ is the condition that has to hold in s to have ϕ true in s'.

Compilation equivalence

It can be proved that the original STRIPS+P problem has a solution plan with a certain total cost (sum of its action costs and of the violated preference costs) if and only if the compiled plan has a solution with the same total cost.

Proposition 1. *Let Π' be the compiled problem with conditional effects of a* STRIPS+P *problem Π. From any plan π solving Π we can derive a plan π' solving Π', and viceversa, such that the total costs of π and π' are the same.*

Proof. (Sketch). The proof has the same structure of the plan-correspondence proof for Keyder and Geffner's compilation of soft goals (Keyder and Geffner 2009), with $\pi' = \langle \pi'', end, \pi''' \rangle$ in which π'' is obtained from π by replacing the original operators with the compiled ones involving conditional effects, and the rest of π' is defined as in Keyder and Geffner's proof (π''' involves only *collect* and *forgo* actions). Since the conditional effects in π'' affect only the additional fluents of the compiled problem, all original operator preconditions remain satisfied in the state trajectory of π''. Moreover, by construction of the conditional effects for the compiled operators, it can be proved that in the state where *end* is applied, for each preference P, P-*violated* holds if and only if P is violated in π. Viceversa, from a valid plan π' we can obtain a plan for π by replacing the compiled operators with their original version, and removing *end* and all *collect*/*forgo* actions. By construction of the conditional effects in the compiled operators and of π and π', π violates a preference if and only if π''' contains the corresponding *forgo* action. It follows that π and π' have the same total costs. □

Compilation of Conditional Effects and Violators

In the literature, there are two main general methods for compiling conditional effects away. In the first method, proposed by Gazen and Knoblock (1997), each plan of the compiled problem preserves the length of the corresponding plan for the original problem, but an exponential number of compiled operators are generated. In the second method, proposed by Nebel (2000), a polynomial number of new operators are generated, but each plans for the compiled problem increases polynomially the length.

In our context, we use Nebel's method because, depending on the operators' structure and the input preferences, many conditional effects can be generated, making Gazen and Knoblock's method impractical. Moreover, Nebel's method can be optimised for our conditional effects because of their particular structure. Specifically, the effects of our conditional effects concerning different preferences can never conflict, while those referring to the same preference can be resolved in the compilation by imposing a particular order of evaluation for their conditions. This allows us to simplify the compilation by omitting the so called "copy-operators" of Nebel's method. Another optimisation concerns the ordering of the set of operator pairs "activating" the conditional effects, which in Nebel's original compilation is unordered, while in our context they can be ordered as a sort of macro operators. For lack of space, in this paper we don't give a detailed description of these optimisations, which leads to a revised method for compiling conditional effect away similar to the technique described in (Ceriani and Gerevini 2015) for always constraints, but extended to deal with every class of PDDL3 qualitative preferences.

Another optimization that we implemented is the following one. For a compiled action that is a violator of a preference P, we assign the violation cost of P to this action, instead of to the forgo action for P, that is removed from the compilation together with goal P' in G'.

Experimental Analysis

We implemented the proposed compilation scheme, and we compared the performance (plan quality) of several propositional planners using it and of LPRPG-P, a the state-state-of-the art system for satisficing planning that "natively" supports PDDL3 preferences.[1] Moreover, we have evaluated the effectiveness of using our compilation for optimal planning with preferences against an alternative recent compilation based on automata.

We used a selection of the best performing planners from IPC8 and IPC9: LAMA (Richter and Westphal 2010), Mercury (Katz and Hoffmann 2014), MIPlan (Núnez, Borrajo, and Linares López 2014), IBaCoP2 (Cenamor, De La Rosa,

[1]For our benchmarks we observed that LPRPG-P performs better than the other available systems supporting PDDL3 preferences. OPTIC (Benton, Coles, and Coles 2012) and the best variant of LPRPG-P in (Coles and Coles 2013) often gave runtime errors.

and Fernández 2014), Fast Downward Stone Soup 2018 (Seipp and Röger 2018), abbreviated FDSS, and Fast Downward Remix (Seipp 2018), abbreviated FDRemix. In addition, we used a recent version of LAMA, called $\text{LAMA}_{\text{P}}(h_{\text{R}})$, exploiting admissibile heuristic h_{R} for testing soft goals reachability during search (Percassi, Gerevini, and Geffner 2017).

As benchmarks we used all (100) original problems of the qualitative preference track of IPC-5 (Gerevini et al. 2009), which has five domains: Rovers, TPP, Trucks, Openstacks and Storage. The problems in Storage and TPP have no hard goal. In these benchmarks there is no preference of type sometime-after. The propositional planners were run on the compiled problems, while LPRPG-P was run on the original problems. All experiments were conducted on a 2.00GHz Core Intel(R) Xeon(R) CPU E5-2620 machine with CPU-time/memory limits of 30 minutes/8GB, respectively. The average compilation time was from 0.2 to few seconds for all domains except Trucks for which it was 86 seconds.

The compared planners are evaluated using the IPC quality score (introduced in IPC-6). Given a planner p and a planning instance i, if p solves i, the following score is assigned to p: $\mathit{score}(p,i) = \frac{\mathit{cost}_{\mathit{best}}(i)}{\mathit{cost}(p,i)}$, where $\mathit{cost}_{\mathit{best}}(i)$ is the cost of the best known solution for i found by any planner, and $\mathit{cost}(p,i)$ is the cost of the best solution found by p, using at most 30 CPU minutes. If p does not find a solution, then $\mathit{score}(p,i) = 0$. We also consider another metric for plan quality evaluation, that we call α_{cost}: if planner p solves instance i, we assign the following score to p

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

where $\mathscr{P}(i)$ is the set of the preferences in i. If for two planners p and p' we have $\alpha_{\mathit{cost}}(p,i) < \alpha_{\mathit{cost}}(p',i)$, then p performs better than p' for i, and the difference between these metric values quantifies the performance discrepancy. The α-cost is minimum when the plan π found by p for i is (preference-wise) optimal.

Concerning optimal planning, similarly to what done by WMN (Wright, Mattmüller, and Nebel 2018), we tested our scheme using Fast Downward (Helmert 2006) with three admissible heuristics: h^{blind}, assigning zero to goal states and 1 to every other state, $h^{\max}$ (Bonet and Geffner 2001), and h^{cpdb} (Haslum et al. 2007). For this analysis we generated a set of benchmarks using the same methodology of WMN. Starting from the IPC-5 problems, we created simpler instances by randomly sampling subsets of the soft trajectory constraints in the original instances: from each original instance, five additional instances are generated, each of which has, respectively, 1%, 5%, 10%, 20% and 40% of the original (grounded) soft trajectory constraints; the hard goals are all unchanged, if they exist. Since the sampled instances used by WMN are not available, we generated, for each sampling percentage (except for 100 %), 3 sampled instances and considered the average performance over them.

Experimental Results

Tables 1 and 2 give the performances of the compared planners in term of IPC quality score aggregated by domain. The results in Table 1 concerns plan quality considering all preferences (i.e., the original plan metrics in these benchmarks), while those in Table 2 concerns plan quality when only the class of preferences indicated in each subtable is considered for the plan metric (using the original violation costs).

Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
$\text{LAMA}_{\text{P}}(h_{\text{R}})$	16.98	8.34	15.43	19.28	**18.48**	**78.51**
FDRemix	17.89	7.1	**17.8**	18.99	16.21	78.0
FDSS 2018	17.6	7.03	17.21	18.7	17.12	77.66
LAMA(2011)	17.01	7.53	13.16	18.42	17.83	73.94
IBaCoP2	**19.62**	9.68	10.0	17.85	15.73	72.88
LPRPG-P	11.36	**18.74**	7.1	**19.71**	12.88	69.78
MIPlan	17.65	8.8	9.23	17.35	14.42	67.46
Mercury	16.07	6.57	7.84	18.06	14.51	63.04

Table 1: IPC scores of the compared planners using the original plan metrics of the IPC-5 benchmarks. The best scores are indicated in bold.

			$\mathscr{P}_{\mathcal{A}}$			
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
$\text{LAMA}_{\text{P}}(h_{\text{R}})$	14.91	**20.0**	15.0	**20.0**	19.0	**88.91**
FDSS 2018	14.75	17.0	**18.0**	17.83	**20.0**	87.59
MIPlan	**15.27**	**20.0**	12.0	19.0	**20.0**	86.27
LPRPG-P	15.02	7.0	0.0	19.5	11.0	52.52
			$\mathscr{P}_{\mathcal{G}}$			
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
LPRPG-P	—	**19.45**	16.48	**19.57**	14.96	**70.47**
FDSS 2018	—	14.66	**16.49**	18.55	18.46	68.16
$\text{LAMA}_{\text{P}}(h_{\text{R}})$	—	14.82	14.36	18.85	18.92	66.94
MIPlan	—	15.08	9.85	16.84	**19.32**	61.09
			$\mathscr{P}_{\mathcal{AO}}$			
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
FDSS 2018	**17.22**	18.0	**20.0**	—	19.0	**74.22**
$\text{LAMA}_{\text{P}}(h_{\text{R}})$	15.33	**19.0**	**20.0**	—	19.0	73.33
MIPlan	14.76	17.0	15.0	—	**20.0**	66.76
LPRPG-P	14.11	2.0	19.0	—	12.0	47.11
			$\mathscr{P}_{\mathcal{SB}}$			
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
$\text{LAMA}_{\text{P}}(h_{\text{R}})$	18.6	**20.0**	**18.0**	—	19.0	**75.6**
MIPlan	**18.66**	**20.0**	12.0	—	**20.0**	70.66
FDSS 2018	17.92	17.0	16.5	—	19.0	70.42
LPRPG-P	8.63	14.0	15.5	—	7.0	45.13
			$\mathscr{P}_{\mathcal{ST}}$			
Planner	Rovers	TPP	Trucks	Openstacks	Storage	TOTAL
$\text{LAMA}_{\text{P}}(h_{\text{R}})$	15.3	10.0	—	—	**19.0**	**44.3**
FDSS 2018	**17.2**	8.0	—	—	**19.0**	44.2
LPRPG-P	10.42	**17.0**	—	—	14.0	41.42
MIPlan	15.86	9.0	—	—	14.0	38.86

Table 2: IPC scores of a selection of the tested planners whose plans are evaluated considering each kind of preferences separately. "—" means that no preferences of this class are present. The best scores are indicated in bold.

The analysis in Table 2 considers a subset of the planners in Table 1. Overall the compilation approach performs better than LPRPG-P, with five planners obtaining better total IPC scores. The comparison considering each preference class separately shows good performance as well for every preference class except for soft goals.

In Rovers, Trucks and Storage each considered planner performs better than, or at least similarly to, 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 its inferior coverage (it solves only 15 instances out of 20). The tested planners from IPC9, FDRemix and FDSS 2018, perform overall better than those from IPC8, and $\text{LAMA}_{\text{P}}(h_{\text{R}})$ is better than everyone else (it improves the performance of LAMA in all the considered domains

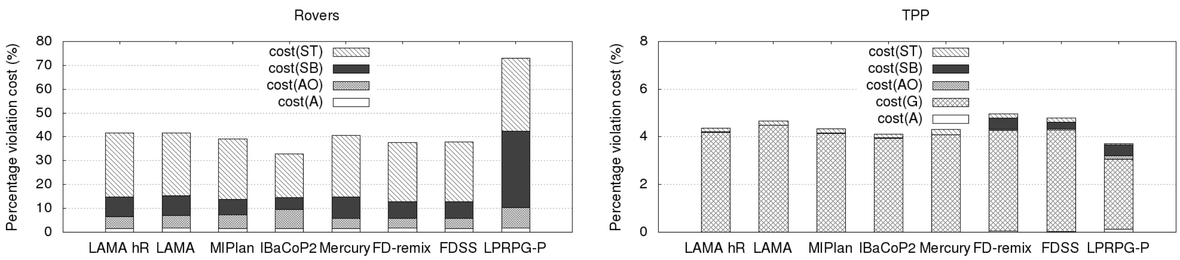

Figure 2: α_{cost} analysis for Rovers and TPP domain.

except Rovers, where there is no soft goal and h_{R} cannot be exploited by $\text{LAMA}_{\text{P}}(h_{\text{R}})$).

On the other hand, LPRPG-P performs similarly to $\text{LAMA}_{\text{P}}(h_{\text{R}})$ in Openstacks and much better than the other planners in TPP. The observed poor performance of the compilation approach in TPP is mainly due to presence of many soft goals. This is not very surprising since, as shown in (Percassi, Gerevini, and Geffner 2017), compiling soft goals through Keyder and Geffner's method can sometimes give poor planning performance. Indeed Table 2 shows that for soft goals LPRPG-P has higher IPC score than all others planners. We can also observe that, compared to LPRPG-P, the classical planners achieve better results for preferences of classes always, sometime-before and at-most-once; however, but this has not a crucial impact of the overall plan quality, because, according to the plan metrics specified in these problems, violating the soft-goals is more costly than violating the other preferences (or equivalently they are more useful to satisfy than the other preferences).

The comparison of the planners' performance using the α_{cost} helps to further understand the behaviour of the planners. For lack of space, we will focus this analysis on two selected domains: Rovers, one of the considered domains where the proposed compilation approach works better, and TPP, the only domain where we observed poor performance compare to LPRPG-P. Figure 2 shows, for each preference class, the planners' α_{cost} values obtained by adding the relative α_{cost} for every instance of the two considered benchmark domains. Each level of the stacked histograms represents the aggregated α_{cost} restricted to a specific class of preferences, which indicates how much each class of the (violated) preference contributes to the total preference violation cost.

For Rovers, the IPC score gap between the classical planners and LPRPG-P is mainly due to LPRPG-P's violation of the sometime-before preferences. Regarding other preferences classes, the violation costs in the generated plans are similar except for IBaCoP2. This planner satisfies more sometime-before and sometime preferences than the others planning, and violates more at-most-once preferences, generally obtaining better quality plans.

For TPP, indeed Figure 2 shows that the most important preferences are the soft goals, which are better satisfied by LPRPG-P. The search pruning technique in $\text{LAMA}_{\text{P}}(h_{\text{R}})$ exploiting h_{R} slightly helps LAMA to achieve more soft goals, but not enough to reach the performance of LPRPG-P.

Domain	h^{blind}		h^{max}		h^{cpdb}	
	WMN	Our	WMN	Our	WMN	Our
Storage	24.78	**57.0**	29.2	**45.0**	23.01	**57.0**
Rovers	17.14	**24.0**	21.43	**25.0**	15.17	**23.0**
Trucks	18.84	**24.0**	23.19	**25.0**	n/a	25
TPP	—	47.0	—	45.0	—	40.0

Table 3: Problem coverage of our compilation and WMN's compilation for the IPC-5 benchmarks set augmented with additional problems that have sampled soft-trajectory constraints. "n/a" means that the considered heuristic was not applicable; "—" means that no data are reported in (Wright, Mattmüller, and Nebel 2018).

Table 3 gives results about our compilation scheme for optimal planning. For each of the three considered admissible heuristics, the table indicates the percentage of solved problems. The results are compared with those reported in (Wright, Mattmüller, and Nebel 2018) for the "goal action penality compilation".[2] Domain Openstacks here is not considered because no one of the considered heuristics solved any instance. Also note that results for TPP are missing for WMN because they are not reported in WMN's paper.

According to these results, for the considered benchmarks, our compilation approach seems quite preferable, because an higher coverage is obtained in all three considered domains. This is the case even though the machine that we used for our experiments if less powerful (CPU and memory wise) than the one used by WMN, and moreover our CPU-time limit was half of that used by WMN.

Conclusions

We have proposed a new compilation schema for solving propositional planning augmented with PDDL3 soft state-trajectory constraints. Our work significantly extends the original approach of Keyder and Geffner that deals only soft goals. The results of an experimental analysis show that, despite the compilation of only soft goals may be less effective than other approaches such as the LPRPG-P planner, for the considered class of soft state-trajectory constraints our compilation is quite competitive with the state-of-the-art. Moreover, since the planning language of the compiled problem is very simple, many available planners can use it.

[2]A more detailed experimental comparison with WMN's approach is very difficult because, at the time of writing, WMN's compiler, compiled files and solution plans are not available.

References

Baier, J., and McIlraith, S. 2006. Planning with first-order temporally extended goals using heuristic search. In *Proceedings of the AAAI Twenty-First National Conference on Artificial Intelligence*, 788–795.

Baier, J.; Bacchus, F.; and McIlraith, S. 2009. A heuristic search approach to planning with temporally extended preferecens. *Artificial Intelligence* 173:593–618.

Benton, J.; Coles, A.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, (ICAPS 2012)*, 2–10.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1-2):5–33.

Cenamor, I.; De La Rosa, T.; and Fernández, F. 2014. IBaCoP and IBaCoP2 planner. In *Eighth International Planning Competition Booklet (IPC 2014)*, 35–38.

Ceriani, L., and Gerevini, A. 2015. Planning with always preferences by compilation into strips with action costs. In *Proceedings of the Eighth International Symposium on Combinatorial Search (SoCS 2015)*, 161–165.

Coles, A. J., and Coles, A. 2011. LPRPG-P: Relaxed plan heuristics for planning with preferences. In *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011)*, 26–33.

Coles, A. J., and Coles, A. 2013. Searching for good solutions in goal-dense search spaces. In *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 37–45.

Cresswell, S., and Coddington, A. 2004. Compilation of LTL goal formulas into PDDL. In *Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECAI 2004)*, volume 16, 985–986.

De Giacomo, G.; De Masellis, R.; and Montali, M. 2014. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 1027–1033.

Edelkamp, S.; Jabbar, S.; and Naizih, M. 2006. Large-scale optimal PDDL3 planning with MIPS-XXL. In *Eighth International Planning Competition Booklet (IPC 2006)*, 28–30.

Edelkamp, S. 2006. On the compilation of plan constraints and preferences. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*, 374–377.

Gazen, B., and Knoblock, C. 1997. Combining the expressivity of UCPOP with the efficiency of Graphplan. In *Proceedings of the 4th European Conference on Planning (ECP 1997)*, 221–233.

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

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; Koenig, S.; et al. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proceedings of the AAAI Twenty-second Conference on Artificial Intelligence*, 1007–1012.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research (JAIR)* 26:191–246.

Katz, M., and Hoffmann, J. 2014. Mercury planner: Pushing the limits of partial delete relaxation. In *Eighth International Planning Competition Booklet (IPC 2014)*, 43–47.

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

Nebel, B. 2000. On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research (JAIR)* 12:271–315.

Núnez, S.; Borrajo, D.; and Linares López, C. 2014. MIPlan and DPMPlan. In *Eighth International Planning Competition Booklet (IPC 2014)*, 13–16.

Percassi, F.; Gerevini, A.; and Geffner, H. 2017. 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)*, 144–148.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research (JAIR)* 39(1):127–177.

Rintanen, J. 2000. Incorporation of temporal logic control into plan operators. In *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI 2000)*, 526–530.

Seipp, J., and Röger, G. 2018. Fast downward stone soup 2018. In *Ninth International Planning Competition Booklet (IPC 2018)*, 72–74.

Seipp, J. 2018. Fast downward remix. In *Ninth International Planning Competition Booklet (IPC 2018)*, 67–69.

Wright, B.; Mattmüller, R.; and Nebel, B. 2018. Compiling away soft trajectory constraints in planning. In *Proceedings of the Sixteenth Conference on Principles of Knowledge Representation and Reasoning (KR 2018)*, 474–482.