Planning with PDDL3 Qualitative Constraints for Cost-Optimal Solutions Through Compilation

Luigi Bonassi[1], Enrico Scala[1] and Alfonso Emilio Gerevini[1]

[1]*Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy*

Abstract

We study the problem of finding cost-optimal solutions to planning problems that feature qualitative state trajectory constraints expressed in PDDL3. These constraints are properties that every plan must satisfy and can be seen as a fragment of LTL over finite traces. The state-of-the-art system for handling PDDL3 problems is a compilation-based approach. Such a compilation has been tested only using a satisficing planner, while the case where we require the planner to find optimal solutions is scarcely studied; with this paper, we want to fill this gap. We propose an experimental analysis that involves TCORE, the current state-of-the-art compilation approach to handle qualitative PDDL3 constraints, and two compilation approaches supporting arbitrary LTL formulas. We evaluate each system using two optimal planners, and we analyze the results using different metrics to explain the behavior of the considered approaches. Our analysis confirms the result previously obtained with a suboptimal planner; that is, in the optimal setting TCORE outperforms all other compilations over our benchmark domains.

Keywords

Automated Planning, PDDL3, Compilation, State-Trajectory Constraints

1. Introduction

The aim of this paper is to study the behavior of different state-of-the-art systems for handling trajectory constraints in the context of finding optimal solutions. In particular, we focus on planning problems that feature qualitative constraints defined by the PDDL3 language [1]. PDDL3 is a popular and standard planning formalism that provides primitives for the specification of such constraints through a subclass of LTL formulas [2]. State-of-the-art approaches deal with such problems either by directly modifying the search engines [3, 4, 5, 6] or by compiling temporal constraints away [7, 8, 9, 10, 11, 12, 13]. Compilation is a technique that works by reformulating a planning problem with temporal constraints into a new equivalent problem without them. Bonassi et al. [13] presents a novel compilation approach, named TCORE, for solving planning tasks with PDDL3 qualitative state-trajectory constraints. TCORE extends a planning task with atomic variables whose role is to maintain the truth of the temporal properties; then it exploits the concept of regression [14] to determine how actions would need to be modified to correctly evaluate the formula over time. Problems compiled through TCORE can be handled by any classical planner that supports conditional effects, making this approach highly modular. It has been shown that [13], when using a suboptimal planner, TCORE performs better than other state-of-the-art compilation approaches over the considered

10th Italian Workshop on Planning and Scheduling (IPS-2022)

l.bonassi005@unibs.it (L. Bonassi); enrico.scala@unibs.it (E. Scala); alfonso.gerevini@unibs.it (A. E. Gerevini)

benchmarks. Although the results obtained with a suboptimal planner provide a clear picture of the strengths and weaknesses of the tested compilations, the effectiveness of such approaches in the context of finding optimal solutions is still unknown. To understand how the optimal setting influences the performance of different compilations, we present additional experiments involving TCORE and other compilation approaches for handling arbitrary LTL formulas using optimal planners. The rest of the paper is structured as follows: Section 2 provides some background on planning problems with PDDL3 qualitative constraints; Section 3 summarizes the compilation schema of TCORE; Section 4 presents our experimental analysis; Section 5 gives the conclusions.

2. Background on Planning with PDDL3 constraints

We borrow standard notions and notations from propositional logic and the work by Gerevini et al. [1] on PDDL3; the reader is referred to this work for more details.

A classical planning problem is a tuple $\Pi = \langle F, A, I, G\rangle$ where F is a set of atoms, $I \subseteq F$ is the initial state, G is a propositional formula over F, and A is a set of actions. An action $a \in A$ is a pair $\langle Pre(a), Eff(a)\rangle$, where $Pre(a)$ is a formula over F expressing the preconditions of a, and $Eff(a)$ is a set of conditional effects, each of which is a pair $c \triangleright e$ where: c is a formula and e is a set of literals, both over F. With e^- and e^+ we indicate the partition of e featuring only negative and positive literals, respectively. A state s is a subset of F, with the meaning that if $p \in s$, then p is true in s, and if $p \notin s$, p is false in s. An action is applicable in s if $s \models Pre(a)$, and the application of an action a in s yields the state $s' = (s \setminus \bigcup_{\substack{c \triangleright e \in Eff(a) \\ \text{with } s \models c}} e^-) \cup \bigcup_{\substack{c \triangleright e \in Eff(a) \\ \text{with } s \models c}} e^+$. We indicate with $s' = s[a]$ the state resulting from applying action a in s, and assume conflicting effects (p and $\neg p$) are only yield by conditional effects having their conditions mutually exclusive in s.

A plan π for a problem $\Pi = \langle F, A, I, G\rangle$ is a sequence of actions $\langle a_0, a_1, ..., a_{n-1}\rangle$ in Π; plan π is valid for Π iff there exists a sequence of states (state trajectory) $\langle s_0, s_1, ..., s_n\rangle$ such that $s_0 = I$, $\forall\, i \in [0, \dots, n-1]$ we have that $s_i \models Pre(a_i)$ and $s_{i+1} = s_i[a_i]$, and $s_n \models G$. The cost of a plan $c(\pi)$, is given by the sum of the cost $c(a)$ of each action a in π. A plan π is said to be optimal iff no plan π' with $c(\pi') < c(\pi)$ exists. PDDL3 state-trajectory constraints are a class of temporal formulae over trajectory of states, and they involve necessary conditions that the state trajectory of a valid plan must satisfy. In this work, we consider planning tasks with constraints that in PDDL3 are called "qualitative" [1], because involving only non-numeric terms. In addition to the standard problem goals, a trajectory constraint can be of the following types: `always` ϕ (A_ϕ), which requires that every state traversed by the plan satisfies formula ϕ; `at-most-once` ϕ (AO_ϕ), which requires that formula ϕ is true in at most one continuous subsequence of traversed states; `sometime-before` ϕ ψ ($\mathsf{SB}_{\phi,\psi}$), which requires that if ϕ is true in a state traversed by the plan, then also ψ is true in a previously traversed state; `sometime` ϕ (ST_ϕ), which requires that there is at least one state traversed by the plan where ϕ is true; `sometime-after` ϕ ψ ($\mathsf{SA}_{\phi,\psi}$), which requires that if ϕ is true in a traversed state, then also ψ is true in that state or in a later traversed state.

A *PDDL3 planning problem* is a tuple $\langle \Pi, C\rangle$ where Π is a classical planning problem and C is a set of trajectory constraints; the valid plans of $\langle \Pi, C\rangle$ are all valid plans of Π whose state

trajectories satisfy all constraints in C.

3. TCORE: Trajectory Constraints COmpilation via Regression

This section briefly describes how TCORE translates a PDDL3 problem into an equivalent planning problem without constraints. TCORE identifies two classes of constraints: *invariant trajectory constraints* (*ITCs*) and *landmark trajectory constraints* (*LTCs*). Intuitively, ITCs can be checked along any plan prefix and if they are violated, there is no way the planner can ever re-establish them; they are invariant conditions that must be maintained over the state trajectory of the plan. LTCs are constraints that require certain conditions true at *some* state over the state trajectory of the plan. TCORE works by extending the action preconditions and conditional effects in order to (i) block the violation of ITCs during planning, and (ii) keep track of the truth of relevant (w.r.t. trajectory constraints) formulae in the states generated by the plan prefix. This is achieved by making use of the effect regression operator R, and by introducing a set of *monitoring atoms*. The effect regression is a formula manipulation technique: given a propositional formula ϕ and an action a, $R(\phi, a)$ is a propositional formula such that, for any state s, $s \vDash R(\phi, a)$ iff $s[a] \vDash \phi$. The regression makes it possible to identify what actual influence the action has over the trajectory constraint of interest. Monitoring atoms serve the purpose of collecting relevant facts on the plan state trajectory and asserting their truth/falsity. The ITCs are: A_ϕ, AO_ϕ, $\mathsf{SB}_{\phi,\psi}$. For each AO_ϕ and $\mathsf{SB}_{\phi,\psi}$, TCORE adds the fresh predicates $seen_\phi$ and $seen_\psi$ to record whether ϕ and ψ have ever held. The LTCs are: ST_ϕ and $\mathsf{SA}_{\phi,\psi}$. For each LTC, the compilation adds a predicate $hold_c$. Such a predicate is meant to record whether the constraint is already satisfied or not according to the current plan prefix.

Algorithm 1 describes the full compilation. As a very first step, TCORE creates the necessary atoms (line 3) and sets up the initial state so as to reflect the current status of the trajectory constraints; in particular, the algorithm captures if a LTC is already achieved in I, or if a formula (ϕ or ψ) that is necessary for the evaluation of an ITC is already true in I. Then TCORE checks whether any ITC is already unsatisfied; if so, the problem is unsolvable.

After the initialization phase, the algorithm iterates over all actions and constraints to modify each original action model (preconditions and effects) by considering the interactions between the constraints and the action model. If the constraint is an ITC, the algorithm determines, by regression, a condition ρ such that, if ρ holds in the state where the action is applied, the execution of such an action will violate the constraint. For example, in the case of A_ϕ, ρ models whether the action makes formula ϕ false (line 10). The regressed condition ρ is negated and then conjoined with the precondition of the action. In this way, if the action will violate the constraint in a given state, such an action is deemed inapplicable by the planner. In the case of ITCs, conditional effects are added to keep track of whether relevant formulae have ever held in the state trajectory of the plan prefix. For instance, $\mathsf{SB}_{\phi,\psi}$ requires to deal with the truth of $seen_\psi$: if an action makes ψ true, then the action must make $seen_\psi$ true too. In this way, the compilation prevents applying an action a when it makes ϕ true and ψ has not held before in current plan state trajectory (lines 14–17).

For each LTC $c \in C$, the algorithm yields a formula ρ that is true only in those states where the action achieves the targeted formula expressed in c. Note here the slightly different treatment

Algorithm 1: TCORE

Input : A PDDL3 Planning Problem $\langle\langle F, A, I, G\rangle, C\rangle$

1 $A' = \{\}$
2 $G' = \top$
3 $F' = monitoringAtoms(C)$
4 $I' = \bigcup_{c:\mathsf{ST}_\phi \in C} \{hold_c \mid I \models \phi\} \cup \bigcup_{c:\mathsf{SA}_{\phi,\psi} \in C} \{hold_c \mid I \models \psi \vee \neg\phi\} \cup \bigcup_{\mathsf{SB}_{\phi,\psi} \in C} \{seen_\psi \mid I \models \psi\} \cup \bigcup_{\mathsf{AO}_\phi \in C} \{seen_\phi \mid I \models \phi\}$
5 **if** $\exists \mathsf{SB}_{\phi,\psi} \in C.I \models \phi \vee \exists \mathsf{A}_\phi \in C.I \models \neg\phi$ **then**
6 | **return** *Unsolvable Problem*
7 **foreach** $a \in A$ **do**
8 | $E = \{\}$
9 | **foreach** *ITC* $c \in C$ **do**
10 | | **if** *c is* A_ϕ **then** $\rho = R(\neg\phi, a)$;
11 | | **else if** *c is* AO_ϕ **then**
12 | | | $\rho = R(\phi, a) \wedge seen_\phi \wedge \neg\phi$
13 | | | $E = E \cup \{R(\phi, a) \triangleright \{seen_\phi\}\}$
14 | | **else if** *c is* $\mathsf{SB}_{\phi,\psi}$ **then**
15 | | | $\rho = R(\phi, a) \wedge \neg seen_\psi$
16 | | | $E = E \cup \{R(\psi, a) \triangleright \{seen_\psi\}\}$
17 | | $Pre(a) = Pre(a) \wedge \neg\rho$
18 | **foreach** *LTC* $c \in C$ **do**
19 | | **if** *c is* ST_ϕ **then** $\rho = R(\phi, a)$;
20 | | **else if** *c is* $\mathsf{SA}_{\phi,\psi}$ **then**
21 | | | $E = E \cup \{R(\phi, a) \wedge \neg R(\psi, a) \triangleright \{\neg hold_c\}\}$
22 | | | $\rho = R(\psi, a)$
23 | | $E = E \cup \{\rho \triangleright \{hold_c\}\}$
24 | $Eff(a) = Eff(a) \cup E$
25 | $A' = A' \cup \{\langle Pre(a), Eff(a)\rangle\}$
26 **foreach** *LTC* $c \in C$ **do**
27 | $G' = G' \wedge hold_c$
28 **return** *Classical Planning Problem* $\langle F \cup F', A', I \cup I', G \wedge G'\rangle$

for the two types of LTCs. While ST_ϕ only requires ϕ to be true, for $\mathsf{SA}_{\phi,\psi}$ the compilation needs to signal the necessity of ψ only when ϕ becomes satisfied; this is done by introducing two conditional effects (lines 21 and 22) affecting the additional goal $hold_c$ of G' (line 27). Also observe that ϕ can become true multiple times, and each state satisfying ϕ needs to be followed by a state such that ψ is true again; this state can also be the same state in which ϕ holds, as prescribed by the semantics of PDDL3.

Note that Algorithm 1 can add irrelevant preconditions and conditional effects that can easily be omitted by looking at whether regression leaves a formula unaltered. E.g., for A_ϕ, if $\rho = R(\neg\phi, a) = \neg\phi$, there is no need to extend $Pre(a)$ with $\neg\rho = \phi$ at line 17. Such optimizations are implemented but omitted here for clarity and compactness.

As trajectory constraints are monitored along the entire plan, and regression through effects provides sufficient conditions for ensuring that no ITC is violated by an action and no LTC remains unsatisfied at the plan end, it is easy to see that the compiled problem always finds a solution that conforms with the trajectory constraints of the problem. Moreover, since the exploited regression establishes necessary conditions too, the existence of a solution in the compiled problem implies that the original problem is solvable.

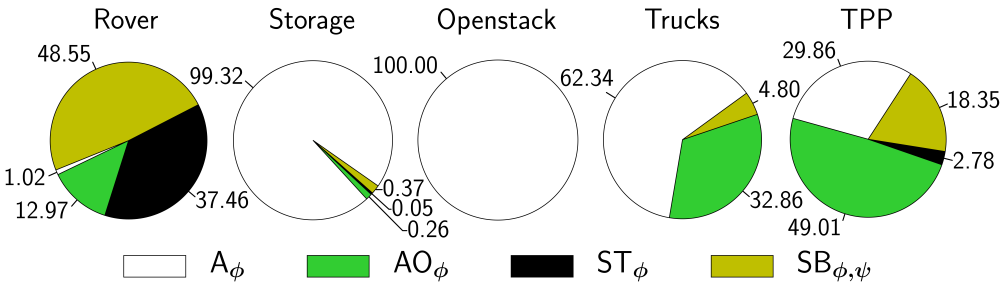

Figure 1: Constraints grouped by type across all domains.

4. Experimental Results

Our experimental analysis studies the behavior of TCORE and two state-of-the-art compilation approaches dealing with LTL constraints in the context of finding optimal solutions. Specifically, we considered the exponential compilation by Baier and McIlraith [8], in short EXP, and the polynomial compilation by Torres and Baier [11], in short POLY[1]. To evaluate the compilations, we considered two optimal planners: A^* with the h^{max} heuristic (implemented in FastDownward [15]) and SYM-K [16]. The former performs a classical A^* search guided by the admissible h^{max} heuristic, while the latter is a top-k planner[2]. Such planners are cost-optimal and support axioms, a feature that is necessary to solve problems compiled through EXP. In total, we tested six different compiler-planner combinations: the three compilations paired with $A^*(h^{max})$ ({TCORE, EXP, POLY}^{max} in short) and with SYM-K ({TCORE, EXP, POLY}^{symk} in short).

The performance of each system is evaluated in terms of the number of solved instances (coverage), the time spent to find a solution (computed as the compilation time plus planning time), the number of nodes expanded by $A^*(h^{max})$, and dimension of the compiled problems. All experiments were performed on a Xeon Gold 6140M 2.3 GHz, with time and memory limits of 1800s and 8GB, respectively. We tested the systems using the benchmark by Bonassi et al. [13]. The suite involves domains from the fifth IPC (https://lpg.unibs.it/ipc-5/), and has a total of 416 instances: 79 for `Trucks`, 90 for `Openstack`, 55 for `Storage`, 94 for `Rover`, and 98 for `TPP`. In such domains, each action a has cost $c(a) = 1$, i.e., the optimal plan is the shortest. Figure 1 gives a general overview on how each type of constraint is partitioned between the considered domains. Overall, the predominant constraint type is `always`, followed by `at-most-once`, `sometime-before` and `sometime`.

4.1. Results analysis

Problem dimensions. Table 1 presents the dimensions of the compiled instances in terms of average number of fluents and average number of effects. Such metrics are computed only

[1]POLY extends a planning problem with many additional synchronization actions that are necessary to update the automaton representing a LTL formula. In this experimental analysis, we set the cost of synchronization actions to zero, as this guarantees that optimal plans correspond to optimal plans of the original problem.

[2]The objective of top-k planning is to determine a set of k different plans with the lowest cost for a problem [16]. In our experimental analysis we used a symbolic bidirectional search obtained from the top-k planner with $k = 1$.

Domain	Fluents			Effects		
	TCORE	EXP	POLY	TCORE	EXP	POLY
Openstack	**251.5**	571.2	883.6	**3906.4**	452635.8	12815.8
Rover	**116.8**	346.5	614.1	**875.8**	155595.9	3238.0
Storage	**87.6**	184.9	315.2	**961.8**	18000.6	1666.7
TPP	**27.7**	98.7	268.3	**226.0**	3181.3	1014.7
Trucks	**374.6**	522.5	754.9	**8667.6**	497196.0	15225.9

Table 1
Average number of fluents and effects in the instances resulting from each compilation. For the sake of a fair comparison, the reported statics refer to the ground instances. For EXP, the number of fluents include the number of derived fluents. Averages are computed on instances compiled by all systems.

Domain	A^* with h^{max}			SYM-K		
	EXP	POLY	TCORE	EXP	POLY	TCORE
Openstack (90)	10	3	10	1	0	**88**
Rover (94)	19	0	34	12	0	**89**
Storage (55)	19	9	**33**	15	3	31
TPP (98)	18	1	24	14	0	**25**
Trucks (79)	17	0	39	13	0	**44**
Total (416)	83	13	140	55	3	**277**

Table 2
Coverage achieved by all systems across all domains. In parenthesis, the number of instances for a given domain.

for instances compiled by each system. EXP and POLY work on the lifted representation of a planning task, while TCORE has to instantiate the planning problem before performing the actual compilation. For the sake of a fair comparison, the averages reported in Table 1 were calculated by grounding the instances resulting from the two LTL systems using the FastDownward translator.

We can observe that the tasks compiled by TCORE contain on average less effects and less fluents compared to the instances compiled with EXP/POLY. This is due to the fact that EXP and POLY are capable of handling arbitrary nested LTL formulas (unsupported by PDDL3); to deal with such expressive power, these two compilations rely on automata theory to compile the LTL formulas away. Integrating the resulting automatons into the domain model requires many additional fluents and effects as shown by Table 1. On the other hand, TCORE is tailored at handling PDDL3 qualitative constraints and this allows for a more efficient compilation. Indeed, TCORE introduces up to one fluent for each trajectory constraint (zero in the case of an always) and takes advantage of the regression computation to minimize the number of additional effects.

Coverage. Table 2 shows an overall picture of the coverage achieved by each system across all domains. TCORE^{symk} achieves the highest coverage in four out of five domains, solving a total of 277 instances. Remarkable are the performances in Rover and Openstack; TCORE^{symk} solves 95% and 98% of the instances in these domains, respectively. The performances of TCORE

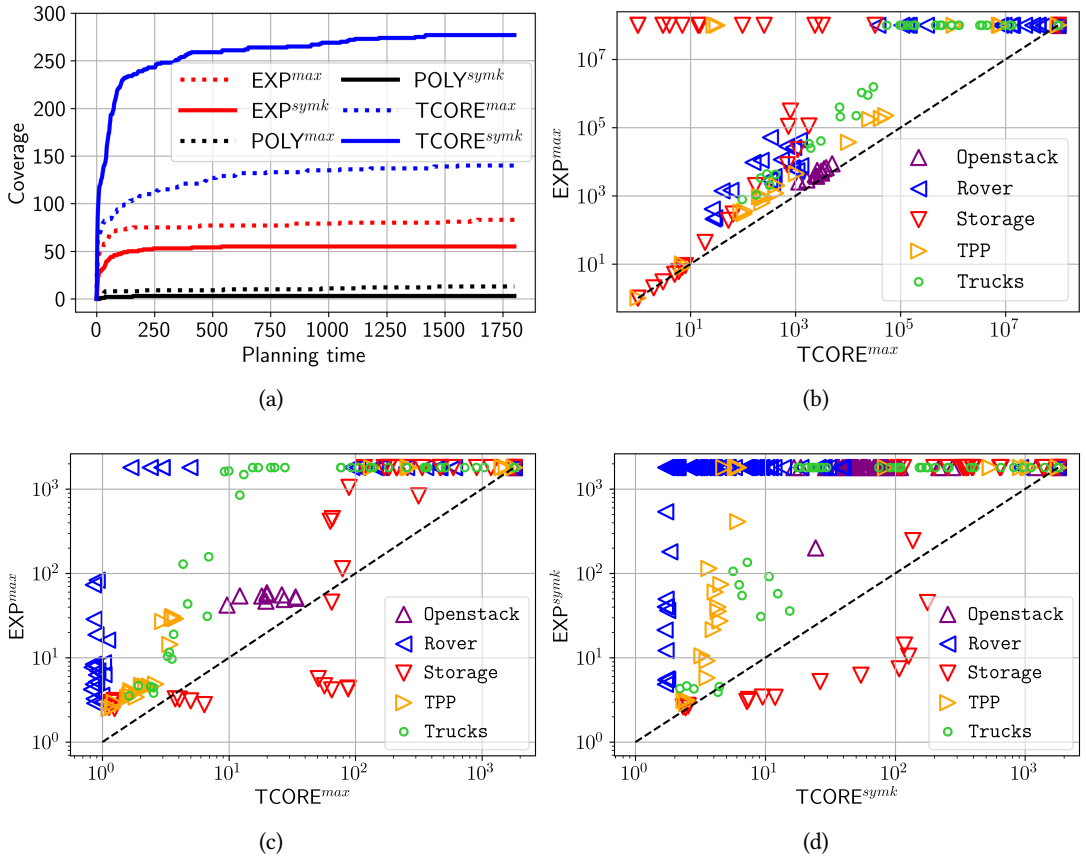

Figure 2: Coverage versus planning time (a). Comparison between TCORE^{max} and EXP^{max} in terms of expanded nodes (b). CPU-time comparison of TCORE with EXP when planning with $A^*(h^{max})$ (c) and with SYM-K (d).

with an optimal planner are close to those of TCORE with a satisficing planner: in the same instances, TCORE with LAMA [17] solves only 34 more instances [13]. This may indicate that, in many instances, the challenge is to effectively deal with the given set of constraints, rather than finding the optimal solution.

As expected, TCORE^{max} cannot solve many problems, achieving roughly half the coverage of TCORE^{symk}. We attribute this behavior to the fact that A^* with h^{max} is less sophisticated compared to SYM-K. Interestingly, though, A^* with h^{max} is more effective at solving instances compiled with EXP and POLY; with both compilations, such a system outperforms SYM-K in all domains. We believe that this is due to the fact that SYM-K does not scale well when problems have many atoms and effects, and this is the case for the instances compiled through EXP and POLY, as shown in Table 1.

Coverage-wise, Bonassi et al. [13] observed that with a suboptimal planner EXP outperforms TCORE in TPP. This is caused by the fact that TCORE fails to ground most instances in this domain, while EXP directly works on the first-order representation. Our experimental results

show that the two optimal planners cannot exploit this advantage of EXP; this indicates that finding optimal solutions to problems compiled with EXP in TPP is a challenging task for the considered planners.

Analogously to the results presented by Bonassi et al. [13], EXP dominates POLY across all domains. One could expect that the polynomial compilation performs better than the exponential compilation. This is not the case, as the LTL formulas deriving from PDDL3 constrains do not lead to an exponential blow-up of the compilation performed by EXP.

CPU time analysis. Figure 2a reports how systems increase their coverage over time. TCORE^{symk} surpasses EXP^{symk} right from the start, and achieves 90% of its maximum coverage after 333 seconds. TCORE^{max} increases its coverage at a slower rate, achieving 90% of its maximum coverage in about 474 seconds. Differently, A^* with h^{max} is faster than SYM-K with EXP, as EXP^{max} dominates EXP^{symk}.

Figures 2c and 2d show a pairwise comparison of the runtimes of TCORE with the runtimes of EXP, while Figure 2b compares TCORE^{max} with EXP^{max} in terms of expanded nodes. Most of the PDDL3 instances are solved generally faster than the instances compiled with EXP. From Figure 2b, we can observe that TCORE^{max} expands less nodes than EXP^{max}, and this indicates that the h^{max} heuristic is more informed when tasks are compiled with TCORE; the EXP compilation introduces axioms, a feature that seems to be not fully supported by the heuristic. There is an exception for a small set of Storage problems; in such instances EXP performs more efficiently than TCORE regardless of the planner. We attribute this to the fact that TCORE has to spend time to ground the instances before performing the actual compilation, while EXP directly works on the first-order representation.

5. Conclusions

We have presented an experimental analysis that compares three different compilation-based systems to handle PDDL3 qualitative constraints with optimal planners. Results shows that TCORE remains the state-of-the-art approach to handle the considered class of problems. Recently, Bonassi et al. [18] has shown that, by expressing control knowledge in PDDL3, TCORE can be used as a tool to improve the coverage of a satisficing planner. In the future, we intend to test TCORE with new benchmarks featuring control knowledge in PDDL3 to improve the coverage of an optimal planner. Finally, we plan to extend TCORE for handling quantitative state-trajectory constraints and to study a compilation that works on the lifted representation of a planning task.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work has been partially supported by EU-H2020 projects AIPlan4EU (No. 101016442) and TAILOR (No. 952215), and by MUR PRIN-2020 project RIPER (No. 20203FFYLK).

References

[1] A. Gerevini, P. Haslum, D. Long, A. Saetti, Y. Dimopoulos, Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners, Artif. Intell. 173 (2009) 619–668.

[2] A. Pnueli, The temporal logic of programs, in: FOCS, IEEE Computer Society, 1977, pp. 46–57.

[3] A. J. Coles, A. Coles, LPRPG-P: relaxed plan heuristics for planning with preferences, in: ICAPS, AAAI, 2011.

[4] J. Benton, A. J. Coles, A. Coles, Temporal planning with preferences and time-dependent continuous costs, in: ICAPS, AAAI, 2012.

[5] Y. Chen, B. W. Wah, C. Hsu, Temporal planning using subgoal partitioning and resolution in sgplan, J. Artif. Intell. Res. 26 (2006) 323–369.

[6] C. Hsu, B. W. Wah, R. Huang, Y. Chen, Constraint partitioning for solving planning problems with trajectory constraints and goal preferences, in: IJCAI, 2007, pp. 1924–1929.

[7] J. A. Baier, F. Bacchus, S. A. McIlraith, A heuristic search approach to planning with temporally extended preferences, Artif. Intell. 173 (2009) 593–618.

[8] J. A. Baier, S. A. McIlraith, Planning with first-order temporally extended goals using heuristic search, in: AAAI, AAAI Press, 2006, pp. 788–795.

[9] S. Edelkamp, S. Jabbar, M. Nazih, Large-scale optimal pddl3 planning with mips-xxl, 5th International Planning Competition Booklet (IPC-2006) (2006) 28–30.

[10] S. Edelkamp, On the compilation of plan constraints and preferences, in: ICAPS, AAAI, 2006, pp. 374–377.

[11] J. Torres, J. A. Baier, Polynomial-time reformulations of LTL temporally extended goals into final-state goals, in: IJCAI, AAAI Press, 2015, pp. 1696–1703.

[12] B. Wright, R. Mattmüller, B. Nebel, Compiling away soft trajectory constraints in planning, in: KR, AAAI Press, 2018, pp. 474–483.

[13] L. Bonassi, A. E. Gerevini, F. Percassi, E. Scala, On planning with qualitative state-trajectory constraints in PDDL3 by compiling them away, in: ICAPS, AAAI Press, 2021, pp. 46–50.

[14] J. Rintanen, Regression for classical and nondeterministic planning, in: ECAI, volume 178 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2008, pp. 568–572.

[15] M. Helmert, The fast downward planning system, J. Artif. Intell. Res. 26 (2006) 191–246.

[16] D. Speck, R. Mattmüller, B. Nebel, Symbolic top-k planning, in: AAAI, AAAI Press, 2020, pp. 9967–9974.

[17] S. Richter, M. Westphal, The LAMA planner: Guiding cost-based anytime planning with landmarks, J. Artif. Intell. Res. 39 (2010) 127–177.

[18] L. Bonassi, A. E. Gerevini, E. Scala, Planning with qualitative action-trajectory constraints in PDDL, in: IJCAI, ijcai.org, 2022, pp. 4606–4613.