On Planning with Qualitative State-Trajectory Constraints in PDDL3 by Compiling them Away

Luigi Bonassi[1], Alfonso Emilio Gerevini[1], Francesco Percassi[2], Enrico Scala[1]
[1]Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy
[2] School of Computing and Engineering, University of Huddersfield, UK

Abstract

We tackle the problem of classical planning with qualitative state-trajectory constraints as those that can be expressed in PDDL3. These kinds of constraints allow a user to formally specify which temporal properties a plan has to conform with through a class of LTL formulae. We study a compilation-based approach that does not resort to automata for representing and dealing with such properties, as other approaches do, and generates a classical planning problem with conditional effects that is solvable iff the original PDDL3 problem is. Our compilation exploits a regression operator to revise the actions' preconditions and conditional effects in a way to (i) prohibit executions that irreversibly violate temporal constraints (ii) be sensitive to executions that traverse those necessary subgoals implied by the temporal specification. An experimental analysis shows that our approach performs better than other state-of-the-art approaches over the majority of the considered benchmark domains.

The problem: PDDL3.0 state trajectory constraints

PDDL3.0 qualitative state-trajectory constraints are a class of temporal logic formulae over state trajectories.

- Necessary conditions that the state trajectory of a valid plan **must satisfy**
- A subclass of Linear Temporal Logic (LTL) formulae
- They are **qualitative** if involving only non-numeric terms

State of the art

Previous works

- Native approaches supporting the trajectory constraints directly into the search engines (OPTIC, MIPS-XXL, SGPLAN, ...)
  - ➢ Can exploit the structure of the problem at a great extent, but can't exploit directly state of the art classical planners
- Compilation-based approaches for LTL constraints (LTL-EXP, LTL-POLY, ...)
  - ➢ Automata translation based
  - ➢ Planner independent
  - ➢ High overhead due to automaton translation into PDDL

Focus and research questions

- Can we achieve an efficient compilation tailored for PDDL3.0 constraints?
- How well our approach performs compared to the state of the art?

Background: PDDL3.0 planning problem

A PDDL3.0 planning problem is a tuple $\langle \Pi, C \rangle$ where:

- $\Pi = \langle F, A, I, G \rangle$ is a classical planning problem
- C is a set of qualitative state trajectory constraints

An action $a \in A$ is a pair $\langle Pre(a), Eff(a) \rangle$

- $Pre(a)$ is a formula over F
- $Eff(a)$ is a set of **conditional effects** $c \triangleright e$

A plan π for Π is a sequence of actions $\langle a_0, a_1, \dots, a_{n-1} \rangle$.
π is **valid** for $\langle \Pi, C \rangle$ iff there exists a state trajectory $\sigma = \langle s_0, \dots, s_n \rangle$ such that:

1. $\forall\, i \in [0, \dots, n-1],\ s_i \models Pre(a_i)$ and $s_{i+1} = s_i[a_i]$
2. $s_0 = I$ and $s_n \models G$
3. Every constraint in C is satisfied

PDDL3.0 qualitative state trajectory constraints

Let ϕ and ψ be formulae over F in negation normal form (NNF). The PDDL3.0 qualitative state trajectory constraints are: `always` ϕ (A_ϕ), `sometime` ϕ (ST_ϕ), `at-most-once` ϕ (AO_ϕ), `sometime-before` $\phi\ \psi$ ($\mathrm{SB}_{\phi,\psi}$), `sometime-after` $\phi\ \psi$ ($\mathrm{SA}_{\phi,\psi}$).

Trajectory constraints: semantics

A trajectory constraint c is satisfied in a state trajectory $\sigma = \langle s_0, \dots, s_n \rangle$ ($\sigma \models c$) if the following conditions hold:

$$\sigma \models (\texttt{always}\ \phi) \text{ iff } \forall i : 0 \le i \le n \cdot s_i \models \phi$$
$$\sigma \models (\texttt{sometime}\ \phi) \text{ iff } \exists i : 0 \le i \le n \cdot s_i \models \phi$$
$$\sigma \models (\texttt{at-most-once}\ \phi) \text{ iff } \forall i : 0 \le i \le n \cdot \text{if } s_i \models \phi \text{ then}$$
$$\exists j : j \ge i \cdot \forall k : i \le k \le j \cdot s_k \models \phi \text{ and } \forall k : k > j \cdot s_k \models \neg\phi$$
$$\sigma \models (\texttt{sometime-after}\ \phi\ \psi)$$
$$\text{iff } \forall i : 0 \le i \le n \cdot \text{if } s_i \models \phi \text{ then } \exists j : i \le j \le n \cdot s_j \models \psi$$
$$\sigma \models (\texttt{sometime-before}\ \phi\ \psi)$$
$$\text{iff } \forall i : 0 \le i \le n \cdot \text{if } s_i \models \phi \text{ then } \exists j : 0 \le j < i \cdot s_j \models \psi$$

Invariant and landmark temporal constraints

We classify PDDL3.0 qualitative constraints into two types:

- **ITC** (Invariant Temporal Constraints): once violated, they cannot be re-established
  - ➢ `always`, `sometime-before`, `at-most-once`
- **LTC** (Landmark Temporal Constraints): they require certain conditions true at *some* state
  - ➢ `sometime`, `sometime-after`

Our approach prevents the violation of the ITC and ensures the satisfaction of the LTC at the end of the plan

TCORE: Trajectory constraints COmpilation via REgression

- We deal with trajectory constraints by compiling them away and then use a classical planner to solve the problem
- The compilation is based on the use of **regression** and the notion of **monitoring atoms**

Definition (Regression operator)

The regression $R(\phi, a)$ of a NNF formula ϕ through the effects $Eff(a)$ of action a is the formula obtained from ϕ by replacing every atom f in ϕ with $\Gamma_f(a) \vee (f \wedge \neg\Gamma_{\neg f}(a))$, where $\Gamma_l(a)$ for a literal l is defined as

$$\Gamma_l(a) = \bigvee_{\substack{c \triangleright e \in Eff(a) \\ \text{with } l \in e}} c.$$

- This definition implies that $s[a] \models \phi$ iff $s \models R(\phi, a)$
- We can predict the truth of a formula before executing any action

Monitoring atoms

- A set of fresh atoms used in the compilation
- **seen** atoms to record the satisfaction of a formula
  - ➢ Example in `at-most-once` ϕ: atom $seen_\phi$ records whether or not ϕ became true in the past, so we can prevent ϕ from becoming true a second time.
- **hold** atoms to record the satisfaction of a LTC

Constraint c	monitoring atom
`sometime-before` $\phi\ \psi$	$seen_\psi$
`at-most-once` ϕ	$seen_\phi$
`sometime` ϕ	$hold_c$
`sometime-after` $\phi\ \psi$	$hold_c$

Table: kinds of monitoring atoms added by TCORE

Compilation overview

- Let $\Pi = \langle \langle F, A, I, G \rangle, C \rangle$ be a PDDL3.0 planning problem
- TCORE compiles Π into a classical planning problem $\Pi' = \langle F', A', I', G' \rangle$
- $F' = F \cup monitoringAtoms(C)$
- $G' = G \wedge \bigwedge_{LTC\ c\ \in C} hold_c$
- I' is obtained by extending I so that it reflects the initial status of the trajectory constraints
- A' is the set featuring the compiled version of every action in A

Action compilation

```
for action a in A:
    E = {}
    for ITC c in C:
        $Pre(a) = Pre(a) \wedge \neg\rho$
        $E = E \cup \epsilon$
    for LTC c in C:
        $E = E \cup \epsilon$
    $Eff(a) = Eff(a) \cup E$
```

Constraint	ρ	ϵ
A_ϕ	$R(\neg\phi, a)$	$\{\}$
$\mathrm{SB}_{\phi,\psi}$	$R(\phi, a) \wedge \neg seen_\psi$	$\{R(\psi, a) \triangleright \{seen_\psi\}\}$
AO_ϕ	$R(\phi, a) \wedge seen_\phi \wedge \neg\phi$	$\{R(\phi, a) \triangleright \{seen_\phi\}\}$
ST_ϕ	/	$\{R(\phi, a) \triangleright \{hold_c\}\}$
$\mathrm{SA}_{\phi,\psi}$	/	$\{R(\phi, a) \wedge \neg R(\psi, a) \triangleright \{\neg hold_c\}, R(\psi, a) \triangleright \{hold_c\}\}$

Table: Precondition and effects added by TCORE for every kind of constraint

Compilation example

- Let a be an action with $Pre(a) = \alpha$ and $Eff(a) = \{\top \triangleright \{\neg\alpha, \beta, \gamma\}\}$
- let c be a `sometime-before` $\phi\ \psi$ with $\phi = \beta \wedge \delta$ and $\psi = \gamma$
- The compiled version of a will be defined as follows:

$$Pre(a) = \alpha \wedge\ (\neg\delta \vee seen_\psi) \qquad (R(\beta \wedge \delta, a) = \delta)$$
$$Eff(a) = \{\top \triangleright \{\neg\alpha, \beta, \gamma\}, \top \triangleright \{seen_\psi\}\} \qquad (R(\gamma, a) = \top)$$

- If $I \models \psi$ then $seen_\psi$ will be added to I'. If $I \models \phi$ then the problem is unsolvable and TCORE terminates

Theorem

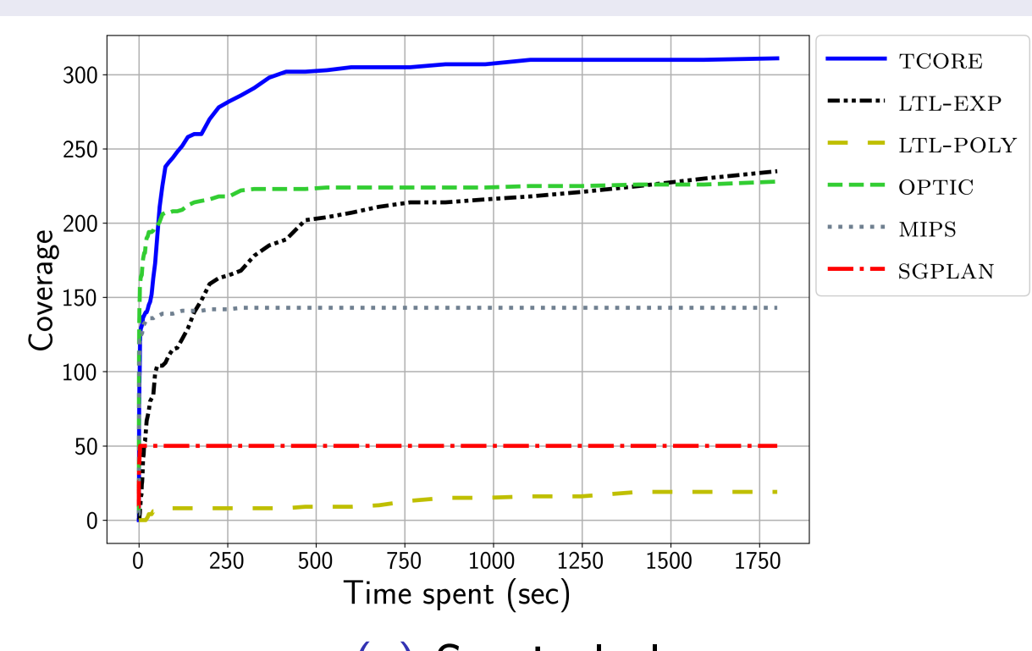
Let $\Pi = \langle \langle F, I, A, G \rangle, C \rangle$ be a PDDL3.0 planning problem, and let $\Pi' = \langle F', I', A', G' \rangle$ be the classical planning problem generated by TCORE. Π admits a solution iff so does Π'.

Experimental results

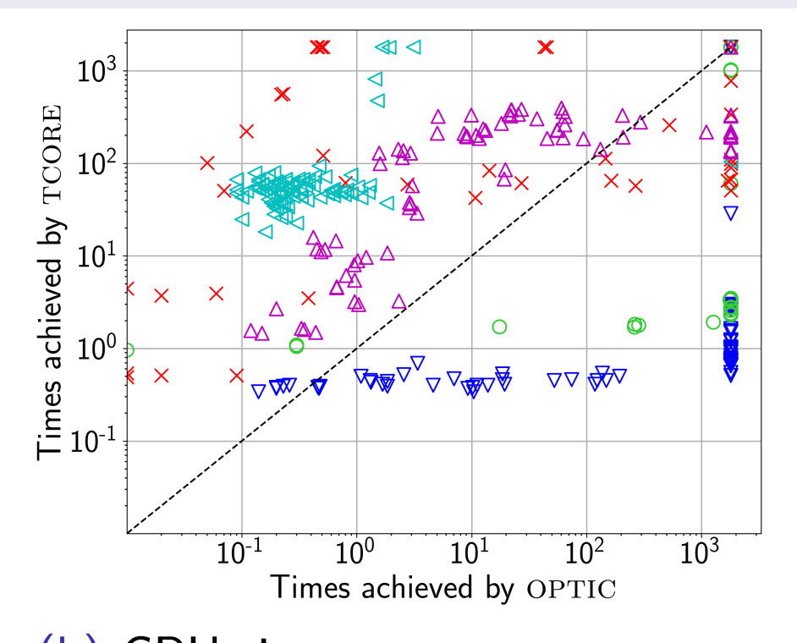
- We compared TCORE with five state of the art approaches handling trajectory constraints
  - ➢ Three native PDDL3.0 planners: SGPLAN, OPTIC and MIPS-XXL
  - ➢ Two compilation-based approaches supporting LTL constraints: LTL-EXP and LTL-POLY
- We generated our benchmarks starting from the domains of the 5th IPC
- Experiments were performed with a time and memory limits of 1800s and 8GB
- Compiled problems generated by TCORE, LTL-EXP and LTL-POLY were solved using LAMA

Domain	TCORE	LTL-EXP	LTL-POLY	OPTIC	MIPS-XXL	SGPLAN
`Rover` (94)	**92**	35	0	33	11	0
`TPP` (98)	22	**58**	1	9	1	1
`Trucks` (79)	**78**	41	0	67	29	35
`Openstack` (90)	88	78	10	**89**	**89**	0
`Storage` (55)	**31**	23	8	30	13	14
Total (416)	**311**	235	19	228	143	50

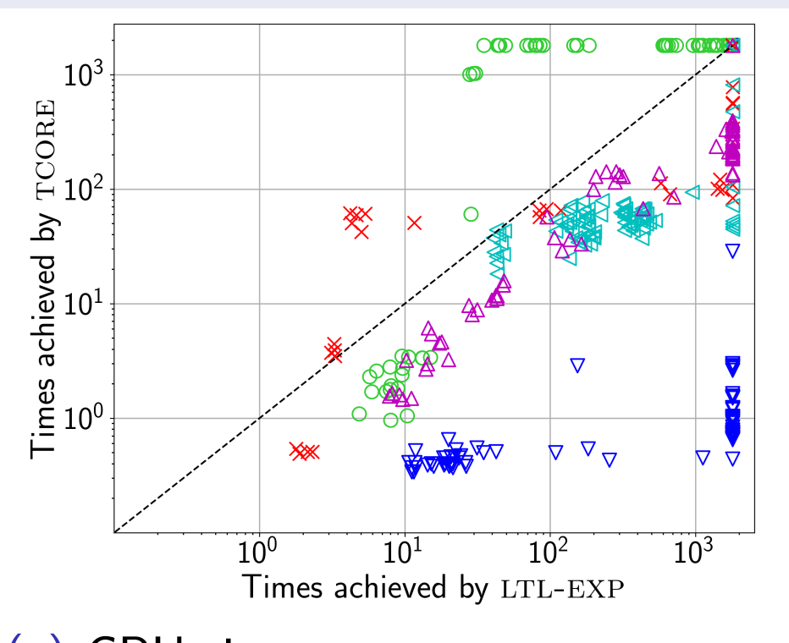
Table: Coverage of all systems across all domains



(a) Survival plot



(b) CPU time: TCORE vs OPTIC



(c) CPU time: TCORE vs LTL-EXP

Conclusions

Planning through our approach can achieve better performances over the tested domains

- TCORE achieves higher coverage compared to the state of the art
- TCORE is faster compared to other compilation-based systems