Rolling in Classical Planning with Conditional Effects and Constraints

Rolling in Classical Planning with Conditional Effects and Constraints

Matteo Cardellini , Enrico Giunchiglia
DIBRIS, University of Genova, Genova, Italy
{matteo.cardellini, enrico.giunchiglia}@unige.it

Abstract

In classical planning, conditional effects (CEs) allow modelling non-idempotent actions, where the resulting state may depend on how many times each action is consecutively repeated. Though CEs have been widely studied in the literature, no one has ever studied how to exploit *rolling*, i.e., how to effectively model the consecutive repetition of an action. In this paper, we fill this void by (i) showing that planning with CEs remains PSPACE-complete even in the limit case of problems with a single action, (ii) presenting a correct and complete planning as satisfiability encoding exploiting rolling while effectively dealing with constraints imposed on the set of reachable states, and (iii) theoretically and empirically showing its substantial benefits.

1 Introduction

In *classical planning*, the environment in which agents operate is represented through Boolean variables, and actions are *idempotent*, i.e., applying the same action once or multiple times results in the same state. The idempotence property falls if we introduce *conditional effects* (CEs), i.e., actions with state-dependant effects. Dealing with *one* application of an action with CEs has been extensively studied and two main approaches exist: either (i) one deals with CEs in a *native* way, i.e., by encapsulating CEs directly in the procedure searching for a plan [Rintanen, 2011; Röger *et al.*, 2014; Katz, 2019], or (ii) one *compiles-away* actions with CEs [Gazen and Knoblock, 1997; Nebel, 2000; Gerevini *et al.*, 2024]. Logic-based native approaches (see, e.g. [Rintanen, 2011]), are all based on *planning as satisfiability* (PaS) [Kautz and Selman, 1992] where a planning task Π is solved by first encoding Π into a corresponding logic formula Π_n incorporating a *bound* (or *number of steps*) n, and then searching for a model of Π_n, increasing n upon failure. In all existing PaS approaches, an action a can be applied at most once per step, and thus, a plan with r consecutive repetitions of a will be found in a number of steps $n \geq r$. The *rolling* technique, introduced for numeric planning in [Scala *et al.*, 2016], allows modelling consecutive repetitions of a in a single step, thus reducing the bound n. Rolling has proved to be very effective in numeric planning [Scala *et al.*, 2016; Cardellini *et al.*, 2024]. Similarly to the numeric case, constraints on the state-space must be carefully handled while rolling to assure the plan's validity [Scala *et al.*, 2016].

In this paper, we ask if rolling can be effectively exploited also in classical planning with CEs and constraints. Firstly, we show that planning with CEs remains PSPACE-complete [Nebel, 2000] even in the limit case of problems with a single action, and thus, finding *how many times* the action has to be consecutively applied is a difficult problem on its own. Secondly, we present a PaS encoding in which an action can be consecutively repeated in a single step. Given an action a, we (i) compute its *transition relation* $\mathcal{T}_a$ representing all the states reachable after *one* application of a, (ii) compute its *transitive closure* (TC) relation $\mathcal{T}_a^+$ (see, e.g., [Matsunaga *et al.*, 1993]) representing the states reachable after *at least one* application of a, and (iii) define a PaS encoding Π^+, exploiting a propositional variable a^+ to denote if action a is applied at least once, and $\mathcal{T}_a^+$ to compute the states reachable with a. Given a model for the encoding Π^+ at bound n, we determine the number of times each action a is applied by exploiting the intermediate formulae used to compute $\mathcal{T}_a^+$.

Since deciding the reachability of a state by repetitions of an action with CEs is PSPACE-complete, the computation of the TC $\mathcal{T}_a^+$ can become impractical. To alleviate this burden, we (i) show how constraints can be leveraged to simplify $\mathcal{T}_a^+$, and (ii) construct $\mathcal{T}_a^+$ through *Reduced Ordered Binary Decision Diagrams* (simply BDDs) [Bryant, 1985], which have been widely employed for this purpose in *Model Checking* [Burch *et al.*, 1992; Clarke *et al.*, 1996] due to its often compact representation and canonicity. If the computation of $\mathcal{T}_a^+$ becomes unpractical even then, we can limit the time on the construction of $\mathcal{T}_a^+$ and exploit the intermediate formulae $\mathcal{T}_a^m$ with $m \geq 0$ produced while constructing $\mathcal{T}_a^+$ – in the best case being $\mathcal{T}_a^m = \mathcal{T}_a^+$ and in the worst case being $\mathcal{T}_a^0 = \mathcal{T}_a$ – modelling the states reachable in *up to* 2^m repetitions of a, reducing by a factor of 2^m the bound required to find a plan.

We prove that our approach is correct and complete, and run an experimental analysis on novel domains where a plan must have repetitions of an action. The analysis confirms our theoretical results and, by comparing to the case where rolling is disabled, confirms the benefits of rolling.

2 Preliminaries

2.1 Classical Planning With CEs and Constraints

A *classical planning task with* CEs *and constraints* (in the following, just *planning task*) is a tuple $\Pi = \langle V, A, I, G, C\rangle$. The set V contains *propositional variables* with domain $\{\top, \bot\}$, the symbols for truth and falsity. The set V induces the set of *literals* $\mathrm{lit}(V) = \{v \mid v \in V\} \cup \{\neg v \mid v \in V\}$. Let $L \subseteq \mathrm{lit}(V)$, we define $L^+ = \{v \mid v \in L\}$ and $L^- = \{v \mid \neg v \in L\}$. In the following, we consider only *consistent* sets L of literals, i.e., with $L^+ \cap L^- = \emptyset$. An *action* $a \in A$ is a pair $\langle \mathrm{pre}(a), \mathrm{post}(a)\rangle$ where the *precondition* $\mathrm{pre}(a)$ is a propositional formula over V and the *postcondition* $\mathrm{post}(a)$ is a set of CEs of a. A CE e is a pair $\langle \mathrm{cond}(e), \mathrm{eff}(e)\rangle$ where $\mathrm{cond}(e)$ is a propositional formula representing the *conditions* of applying the *effects* $\mathrm{eff}(e) \subseteq \mathrm{lit}(V)$. If $v \in \mathrm{eff}(e)$ we say that e *adds* v, if $\neg w \in \mathrm{eff}(e)$ we say that e *deletes* w. We denote by $\mathrm{add}(a)$ and $\mathrm{del}(a)$ the set of variables *added* or *deleted* by a, with $\Delta_a^+(v)$ and $\Delta_a^-(v)$ the set of CEs of an action a that adds or deletes a variable v If, for each $e \in \mathrm{post}(a)$, we have $\mathrm{cond}(e) = \top$, then a is *without* CEs, and *with* CEs otherwise. If, for each $a \in A$, a is without CEs (resp. with CEs), then also Π is. A *state* is a set $s \subseteq V$ containing the variables which are true in s, while $V \setminus s$ are false in s. We denote by $S = 2^V$ the set of all states. To conclude the description of Π, we have the *initial state* $I \in S$, and the *goal condition* G and the *constraints* C, both defined as propositional formulae over V. We take for granted the standard notions of satisfiability and write $s \models \phi$ to mean that state s satisfies the propositional formula ϕ. An action a is *applicable* in a state s if (i) $s \models \mathrm{pre}(a)$, (ii) there exists at least one CE $e \in \mathrm{post}(a)$ such that $s \models \mathrm{cond}(e)$, and (iii) there are no *conflicting* CEs, i.e., for each $v \in V$, $e^+ \in \Delta_a^+(v)$ and $e^- \in \Delta_a^-(v)$ then $s \not\models \mathrm{cond}(e^+) \wedge \mathrm{cond}(e^-)$. Applying an action a in a state s results in a state $s' = res(s, a)$, such that either

1. s' is undefined if either s is undefined or a is not applicable in s, or
2. for each $v \in V$ we have $v \in s'$ iff either (i) $v \in s$ and there is no CE $e \in \Delta_a^-(v)$ such that $s \models \mathrm{cond}(e)$, or (ii) there is a CE $e \in \Delta_a^+(v)$ such that $s \models \mathrm{cond}(e)$.

Applying a sequence of n actions $\alpha = a_1; \ldots; a_n$ from a state s_0 induces a sequence of $n+1$ states $s_0; \ldots; s_n$ where $s_{i+1} = res(s_i, a_{i+1})$ with $i \in [0, n)$ and we say that $s_n = res(s_0, \alpha)$. The sequence α is *valid* if $s_i \models C$ for each $i \in [0, n]$. A *valid plan* π for a planning task Π is a valid sequence of actions $\pi = a_1; \ldots; a_n$ such that $res(I, \pi) \models G$. We denote by $|\pi| = n$ the length of $\pi = a_1; \ldots; a_n$, and with a^k, with $k \geq 0$, the sequence of k repetitions of a.

2.2 Planning as Satisfiability

Let $\Pi = \langle V, A, I, G, C\rangle$ be a planning task. In *planning as satisfiability* (PAS) [Kautz and Selman, 1992], an *encoding* E *of* Π is a tuple $\Pi^E = \langle \mathcal{X}, \mathcal{A}, \mathcal{I}(\mathcal{X}), \mathcal{T}^E(\mathcal{X}, \mathcal{A}, \mathcal{X}'), \mathcal{G}(\mathcal{X})\rangle$ where $\mathcal{X}$ is a finite set of *state variables*, in our case equal to V, $\mathcal{A}$ is a finite set of *action variables*, $\mathcal{I}(\mathcal{X})$ is the *initial state formula*, i.e., the conjunction of $\{v \mid v \in I\} \cup \{\neg w \mid w \in V \setminus I\}$, and, $\mathcal{G}(\mathcal{X})$ is the *goal formula*, i.e., the formula G.

Each encoding E is characterized by the *symbolic transition relation* $\mathcal{T}^E(\mathcal{X}, \mathcal{A}, \mathcal{X}')$, where $\mathcal{X}' = \{v' \mid v \in \mathcal{X}\}$ is the set of *next state variables*. A *model* μ for $\mathcal{T}^E$ is an assignment $\mu\colon \mathcal{X} \cup \mathcal{A} \cup \mathcal{X}' \mapsto \{\top, \bot\}$. As standard for PAS encodings, Π^E has to guarantee *correctness* and *completeness*.[1]

1. Correctness: each model μ of $\mathcal{T}^E$ corresponds to at least one *valid* sequence of actions α such that if $s = \{v \mid \mu(v)\}$ and $s' = \{v \mid \mu(v')\}$ then $s' = res(s, \alpha)$.
2. Completeness: for each state s and action a applicable in s, if $s' = res(s, a)$, $s, s' \models C$, then there is a model μ of $\mathcal{T}^E$ s.t. $s = \{v \mid \mu(v)\}$, $s' = \{v \mid \mu(v')\}$.

Given Π^E, in the PAS approach, an integer $n \geq 0$ called *bound* or *number of steps* is fixed, $n + 1$ disjoint copies $\mathcal{X}_0, \ldots, \mathcal{X}_n$ of the set $\mathcal{X}$ of state variables, and n disjoint copies $\mathcal{A}_1, \ldots, \mathcal{A}_n$ of the set $\mathcal{A}$ of action variables are made, and the E *encoding of* Π *with bound* n is the formula

$$\Pi_n^E = \mathcal{I}(\mathcal{X}_0) \wedge \bigwedge_{i=0}^{n-1} \mathcal{T}^E(\mathcal{X}_i, \mathcal{A}_{i+1}, \mathcal{X}_{i+1}) \wedge \mathcal{G}(\mathcal{X}_n), \quad (1)$$

in which $\mathcal{I}(\mathcal{X}_0)$ is obtained by substituting in $\mathcal{I}(\mathcal{X})$ each variable $v \in \mathcal{X}$ with $v_0 \in \mathcal{X}_0$, and similarly for $\mathcal{T}^E$ and $\mathcal{G}$. The satisfiability of Π_n^E is checked by calling a SAT solver starting from $n = 0$ and incrementing n until a model (and thus a valid plan) is found. The correctness of $\mathcal{T}^E$ ensures the *correctness of* Π^E: each model of Π_n^E corresponds to a valid plan. The completeness of $\mathcal{T}^E$ ensures the *completeness of* Π^E: if there exists a plan for Π of length n, Π_n^E is satisfiable.

3 Complexity of Rolling

In this section, we present our main complexity result for classical planning with CEs.

Theorem 1. *Deciding whether a valid plan exists for a classical planning task with* CEs *and with only one action is* PSPACE*-complete.*

Proof. Let $\Pi = \langle V, \{a\}, I, G, \top\rangle$. *(Membership)* The problem is in PSPACE because the size of a state is bounded by $|V|$ and we iteratively apply a from I until we reach $s_n \models G$. Since there are at most $2^{|V|}$ possible states, no more than $2^{|V|}$ repetitions of a are required to reach s_n. *(Hardness)* Let M be a *Deterministic Turing Machine* (DTM) with bounded tape, i.e., $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_a, n\rangle$ (see, e.g., [Sipser, 1997]). Q is the set of *states*, Σ the *input alphabet*, Γ the *tape alphabet* including Σ and the symbol $\sqcup$ for *"blank"*, δ is a partial function $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$, the states q_0, $q_a \in Q$ are the *initial* and *accepting states*, respectively, $n \in \mathbb{N}$ is the bounded length of the tape. Let the input of the DTM be $y_1; \ldots; y_m$ with $y_i \in \Sigma$ and $m \leq n$. We reduce the DTM M to the planning task Π such that

$$\begin{aligned} V &= \{tp_i \mid i \in [0, n+1]\} \cup \{at_{i,q} \mid i \in [1, n], q \in Q\} \\ &\quad \cup \{in_{i,x} \mid i \in [1, n], x \in \Gamma\} \cup \{accept\}, \\ I &= \{tp_i \mid i \in [1, n]\} \cup \{at_{1,q_0}\} \\ &\quad \cup \{in_{i,y_i} \mid i \in [1, m]\} \cup \{in_{i,\sqcup} \mid i \in (m, n]\}, \\ G &= accept. \end{aligned}$$

[1] In the rest of the paper $\mu(v)$ corresponds to the equation $\mu(v) = \top$ and $s, s' \models \phi$ to $s \models \phi \wedge s' \models \phi$.

Where tp_i is used to control the allowed cells of the tape, $at_{i,q}$ signals that the DTM is in tape's cell i and state q, and $in_{i,x}$ signals that cell i contains symbol x. The only action in Π is $a = \langle \top, \text{post}(a) \rangle$ where, for each $i \in [1, n]$ and $(q, x) \in Q \times \Gamma$, if $\delta(q, x) = (q', x', L)$ then $\text{post}(a)$ contains

$$\langle at_{i,q} \wedge in_{i,x} \wedge tp_{i-1}, \{\neg at_{i,q}, at_{i-1,q'}, \neg in_{i,x}, in_{i,x'}\} \rangle,$$

i.e., if the DTM is in cell i and state q, reading x, and the cell on the left is allowed, then the DTM moves left to cell $i-1$ and to state q' and the symbol in cell i is replaced with x'. Similarly, if $\delta(q, x) = (q', x', R)$ then $\text{post}(a)$ contains

$$\langle at_{i,q} \wedge in_{i,x} \wedge tp_{i+1}, \{\neg at_{i,q}, at_{i+1,q'}, \neg in_{i,x}, in_{i,x'}\} \rangle.$$

Finally, for each $i \in [1, n]$, $\text{post}(a)$ contains

$$\langle at_{i,q_a} \wedge \bigwedge \{\neg in_{i,x} \mid x \in \Gamma, q' = \sigma(q_a, x)\}, \{accept\} \rangle,$$

i.e., the DTM *accepts* if it is in the accepting state, and it is not reading any symbol that would change the DTM state.

Since the CEs encode the transition δ, a valid plan can be found iff M accepts. Since $|Q| \times |\Gamma| \times n$ is polynomial w.r.t. the size of M, we conclude that a DTM with polynomially bounded tape and its input can be polynomially reduced to a planning task with only one action with CEs. □

4 Rolling Actions with Conditional Effects

In this section, we show how to exploit rolling of actions with CEs. Firstly, we describe the concept of *transition relation*, i.e., a logic formula denoting the states reachable with one application of an action. Then, we show how to compute its *transitive closure* (TC), obtaining a logic formula denoting all the states reachable by at least one repetition of an action.

4.1 Transition Functions and Transition Relations

Let $\Pi = \langle V, A, I, G, C \rangle$ be a planning task. The *transition function* of $a \in A$ is a function $T_a : S \times S \mapsto \{\top, \bot\}$ such that, for each $s, s' \in S$, we have $T_a(s, s') = \top$ iff (i) a is applicable in s and (ii) $s' = res(s, a)$. Notice how, for now, we allow $s, s' \not\models C$, which we will disallow later on.

As standard for PaS encodings (see, e.g., [Rintanen, 2011]), we model an action's transition function through a *transition relation*. A transition relation of an action a is a propositional formula $\mathcal{T}_a(\mathcal{X}, \mathcal{X}')$, where $\mathcal{X} = V$ and $\mathcal{X}' = \{v' \mid v \in \mathcal{X}\}$ is a copy of $\mathcal{X}$. The models of $\mathcal{T}_a$ represent the states $s, s' \in S$ such that $T_a(s, s') = \top$. For the case of classical planning with CEs and constraints, the transition relation $\mathcal{T}_a(\mathcal{X}, \mathcal{X}')$ is the conjunction of the union of the following sets of formulae:

1. pre_a which contains
$$\text{pre}(a) \wedge \bigvee_{e \in \text{post}(a)} \text{cond}(e),$$
ensuring the preconditions of a and the conditions of at least one CE in $\text{post}(a)$ are satisfied,
2. $\text{eff}_a(V)$ which contains, for each $v \in V$,
$$v' \leftrightarrow \Big(v \wedge \bigwedge_{e \in \Delta_a^-(v)} \neg\text{cond}(e)\Big) \vee \bigvee_{e \in \Delta_a^+(v)} \text{cond}(e),$$
ensuring $s' = res(s, a)$, i.e., v' is true if v is true and no condition of a CE deleting v ($\Delta_a^-(v)$) is satisfied, or at least one condition of a CE adding v ($\Delta_a^+(v)$) is satisfied,
3. $\text{conflict}_a(V)$, containing for each $v \in V$ and for each $e_1 \in \Delta_a^+(v)$ and $e_2 \in \Delta_a^-(v)$
$$\neg\big(\text{cond}(e_1) \wedge \text{cond}(e_2)\big),$$
ensuring no conflicting CEs' conditions are satisfied.

4.2 Computing the Transitive Closure

Let $\Pi = \langle V, A, I, G, C \rangle$ be a planning task and a be an action of Π with transition function T_a. The TC *function* of a is $T_a^+ \colon S \times S \mapsto \{\top, \bot\}$ such that for each state $s, s' \in S$, $T_a^+(s, s') = \top$ iff (i) a is applicable in s, and (ii) there exists a valid sequence a^k with $k \geq 1$ such that $s' = res(s, a^k)$.

The TC function T_a^+ can be computed iteratively (see, e.g. [Matsunaga *et al.*, 1993]) using the TC *relation* $\mathcal{T}_a^+$, starting from the transition relation $\mathcal{T}_a$. Let $\mathcal{T}_a^i(\mathcal{X}, \mathcal{X}')$ be the *i-th step transition relation of* a, such that $\mathcal{T}_a^0(\mathcal{X}, \mathcal{X}') = \mathcal{T}_a(\mathcal{X}, \mathcal{X}')$ and for each $i \geq 0$,

$$\mathcal{T}_a^{i+1}(\mathcal{X}, \mathcal{X}'') = \mathcal{C}(\mathcal{X}) \wedge \mathcal{C}(\mathcal{X}'') \rightarrow \mathcal{T}_a^i(\mathcal{X}, \mathcal{X}'') \vee \exists \mathcal{X}' \colon \mathcal{T}_a^i(\mathcal{X}, \mathcal{X}') \wedge \mathcal{C}(\mathcal{X}') \wedge \mathcal{T}_a^i(\mathcal{X}', \mathcal{X}''), \quad (2)$$

where $\mathcal{C}(\mathcal{X})$ is obtained by replacing the variables in C with the ones in $\mathcal{X}$ (and similarly for $\mathcal{X}'$ and $\mathcal{X}''$). The models of $\mathcal{T}_a^i$ represent the states $s, s'' \in S$ such that the *i-th step transition function* $T_a^i(s, s'') = \top$. Notice how $T_a^{i+1}(s, s'')$ is true if either $s \not\models C$ or $s'' \not\models C$. This relaxation allows, in some cases, a more compact formula for $\mathcal{T}_a^+$, and will be better explained in Section 4.3. The requirement that $s, s'' \models C$ will be enforced at a later stage.

Continuing the computation, by the finiteness of the states, there exists a $p \geq 0$ – called the *fix-point index* of $\mathcal{T}_a$– such that $\mathcal{T}_a^p(\mathcal{X}, \mathcal{X}')$ is logically equivalent to $\mathcal{T}_a^{p+1}(\mathcal{X}, \mathcal{X}')$. By representing each $\mathcal{T}_a^i$ with BDDs, which are canonical representations of propositional formulae, we check logical equivalence of $\mathcal{T}^p$ and $\mathcal{T}^{p+1}$ by assessing if they have the same BDD. The TC relation $\mathcal{T}_a^+$ is thus the *fix-point relation* $\mathcal{T}_a^p$.

Lemma 1. *Let p be the fix-point index of the transition function T_a of an action a of a planning task with constraints C. For each $i \in [0, p]$ and $s, s'' \in S$ s.t. $s, s'' \models C$*

$$T_a^i(s, s'') \leftrightarrow \exists r \in [1, 2^i] \colon s'' = res(s, a^r).$$

Sketch Proof. For $i = 0$, $T_a^0(s, s'') \leftrightarrow T_a(s, s'')$ which by definition models $s'' = res(s, a)$, i.e., $2^0 = 1$ repetitions of a. For $i > 0$, by Eq. 2, $T_a^1(s, s'')$ models the states reachable after *up to* 2 repetitions of a: we can model (i) $s'' = res(s, a^2)$ since $T_a^0(s, s')$ models $s' = res(s, a)$ and $T_a^0(s', s'')$ models $s'' = res(s', a)$, and (ii) we can model $s'' = res(s, a^1)$ because $\mathcal{T}_a^i(\mathcal{X}, \mathcal{X}'') \wedge \mathcal{C}(\mathcal{X}) \wedge \mathcal{C}(\mathcal{X}'')$ entails $\mathcal{T}_a^{i+1}(\mathcal{X}, \mathcal{X}'')$ in Eq. 2. Since $s, s'' \models C$, Eq. 2 guarantees that $s' \models C$. By induction, we prove that $T_a^i(s, s'')$ can model up to 2^i repetition of a , since we have $T_a^{i-1}(s, s')$ to reach s' from s in at most 2^{i-1} steps, and $T_a^{i-1}(s', s'')$ to reach s'' from s' in at most 2^{i-1} steps. □

The function $T_a^i(s, s')$ thus specifies if s' is reachable from s in *at most* 2^i repetitions of a. In the following sections we will also need to know if the state is reachable in *exactly* 2^i repetitions of a. For this reason, we introduce the *i-th*

step exponential reachability relation of a as $\mathcal{R}_a^i$ such that $\mathcal{R}_a^0(\mathcal{X}, \mathcal{X}'') = \mathcal{T}_a(\mathcal{X}, \mathcal{X}'')$ and for $i > 0$

$$\mathcal{R}_a^{i+1}(\mathcal{X}, \mathcal{X}'') = \exists \mathcal{X}' \colon \mathcal{R}_a^i(\mathcal{X}, \mathcal{X}') \wedge \mathcal{C}(\mathcal{X}') \wedge \mathcal{R}_a^i(\mathcal{X}', \mathcal{X}''). \quad (3)$$

The *i-th step exponential reachability function* $R_a^i : S \times S \mapsto \{\top, \bot\}$ can be retrieved from $\mathcal{R}_a^i$ as for T_a^i.

Lemma 2. *Let a be an action and let R_a^i be its i-th step exponential reachability function.*

$$R_a^i(s, s') \;\leftrightarrow\; s' = res(s, a^{2^i}).$$

Sketch Proof. As for Lem. 1, by induction on Eq. 3. □

As stated in the introduction, the TC computation can become intractable, due to formulae becoming exponentially long, even when employing BDDs. However, the computation of the TC can be stopped at any time, before its fix-point index p, returning the last computed relation $\mathcal{T}_a^m$ with $m \in [0, p]$, which models the states reachable with up to 2^m repetitions of a (Lem. 1). We name m the *timeout index*.

4.3 Transitive Closure and Constraints

Let $\Pi = \langle V, A, I, G, C\rangle$ be a planning task. In this section, we show how constraints can, in some cases, simplify TCs.

Example 1. *A robot can move in one direction on a ring tape with k cells, where the last and first cell are connected. The problem can be modelled as a planning task where $V = \{c_1, \ldots, c_k\}$ models the position on the tape, $I = \{c_1\}$, and there is only one action* $\texttt{mv}$ *with* $\mathrm{pre}(\texttt{mv}) = \top$ *and*

$$\mathrm{eff}(\texttt{mv}) = \{\langle c_1, \{\neg c_1, c_2\}\rangle, \ldots, \langle c_k, \{\neg c_k, c_1\}\rangle\}.$$

Let's suppose that $C = \top$, i.e., there are no constraints. One would imagine that, since the robot can reach any cell in the ring tape by applying $\texttt{mv}$ at most k times, then the TC function $T^+_{\texttt{mv}}(s, s')$ would be $\top$ for each $s, s' \in S$. However, the TC has to consider also the cases where $|s|$ or $|s'|$ is not 1, e.g., $T^+_{\texttt{mv}}(\{c_1\}, \{c_2, c_4\}) = \bot$. To avoid this problem, we can specify in C that the robot must be exactly in one cell, i.e.,

$$C = \textstyle\bigvee_{i=1}^{k} c_i \wedge \bigwedge_{i=1}^{k} \bigwedge_{j=i+1}^{k} \neg(c_i \wedge c_j).$$

Employing this constraint we have that $T^+_{\texttt{mv}}(s, s') = \top$ for each $s, s' \in S$, since, if either $s \not\models C$ or $s' \not\models C$ then, in Eq. 2, the l.h.s. of the implication is $\bot$. Intuitively, if the robot starts from any state $s_0 \models C$ it can reach any other state $s_n \models C$ by passing only in states $s_i \models C$, $i \in (0, n)$. If the robot starts from a state $s \not\models C$ then it can reach any state. By guaranteeing that the robot always start moving from a state $s \models C$ (later in the encoding), we will guarantee correctness.

The use of constraints thus can, in some cases, be beneficial in producing smaller $\mathcal{T}_a^+$ by not considering states which do not align with our knowledge of the problem.

5 The Closure-Encoding

Let $\Pi = \langle V, A, I, G, C\rangle$ be a planning task. For each action $a \in A$, let $\mathcal{T}_a(\mathcal{X}, \mathcal{X}')$ be the transition relation of a and let $\mathcal{T}_a^+(\mathcal{X}, \mathcal{X}')$ be its TC relation. We now present the closure encoding Π^+ for Π. As standard for PaS approaches, the sets $\mathcal{X}$, $\mathcal{A}$, $\mathcal{X}'$ represent the *current state*, *action*, and *next state* variables. In $\mathcal{A}$ there is a propositional variable a^+ for each action a in A, denoting if a is repeated at least one time. The current state variables $\mathcal{X}$ are equal to V and the set $\mathcal{X}'$ is a copy of $\mathcal{X}$. The closure-encoding for Π is thus $\Pi^+ = \langle \mathcal{X}, \mathcal{A}, \mathcal{I}(\mathcal{X}), \mathcal{T}^+(\mathcal{X}, \mathcal{A}, \mathcal{X}'), \mathcal{G}(\mathcal{X})\rangle$ in which $\mathcal{I}(\mathcal{X})$ and $\mathcal{G}(\mathcal{X})$ are defined as in Sec. 2.2 and $\mathcal{T}^+(\mathcal{X}, \mathcal{A}, \mathcal{X}')$ is a *closure symbolic transition relation*, i.e., the conjunction of the union of the following sets:

1. $\mathrm{closure}^+(A, V)$, which contains, for each action $a \in A$,
$$a^+ \rightarrow \mathcal{T}_a^+(\mathcal{X}, \mathcal{X}'),$$
i.e., if a^+ is executed, $\mathcal{X}'$ must be reachable from $\mathcal{X}$ in at least one repetition of a,
2. $\mathrm{frame}^+(V)$, which contains, for each $v \in V$,
$$v' \neq v \rightarrow \Big(\bigvee_{a : v \in \mathrm{add}(a)} a^+ \vee \bigvee_{a : v \in \mathrm{del}(a)} a^+ \Big)$$
i.e, an action must have triggered the change of v,
3. $\mathrm{amo}^+(A)$, where for each $a_1 \neq a_2 \in A$ we have,
$$\neg(a_1^+ \wedge a_2^+)$$
i.e., at each step, at most one action is applied,
4. $\mathrm{constraints}^+(V, C)$, where we have,
$$\mathcal{C}(\mathcal{X}) \wedge \mathcal{C}(\mathcal{X}'),$$
ensuring the respect of C, skipped in Eq. 2.

Algorithm 1 Computation of the rolling of a needed to reach s'' from s.

```
1: global T_a^0, ..., T_a^p, R_a^0, ..., R_a^p   //previously computed
2: function REPETITIONS(a, s, s'', p)
3:     if T_a^0(s, s'') then
4:         return 1
5:     for j ∈ [1, p] do
6:         if T_a^j(s, s'') then
7:             s' ← s' : R_a^{j-1}(s, s')
8:             return 2^{j-1} + REPETITIONS(a, s', s'', j − 1)
9:     return -1
```

5.1 Valid Plan with the Closure-Encoding

Let $\Pi = \langle V, A, I, G, C\rangle$ be a planning task, and let Π^+ be the closure-encoding of Π with closure symbolic transition relation $\mathcal{T}^+(\mathcal{X}, \mathcal{A}, \mathcal{X}')$. For any model μ of $\mathcal{T}^+(\mathcal{X}, \mathcal{A}, \mathcal{X}')$ we associate an action a such that $\mu(a^+) = \top$. As stated in Sec. 2.2, to find a plan, we employ Eq. 1, starting from $n = 0$ and increasing n until Π_n^+ is satisfiable. Let μ be the model satisfying Π_n^+. We denote by

$$\textsc{Plan}(\Pi_n^+, \mu) = a_1; \ldots; a_n \quad (4)$$

the sequence of actions where each a_i with $i \in [1, n]$ is the action obtained by the model μ of $\mathcal{T}^+(\mathcal{X}_{i-1}, \mathcal{A}_i, \mathcal{X}_i)$. It is clear that $\textsc{Plan}(\Pi^+, n, \mu)$ may not be a valid plan for Π, since each action a_i in the plan only indicates that it is applied *at least* one time. To produce a valid plan for Π, we

need thus to compute how many times each action must be actually repeated.

For each action a, let $T_a^0, \ldots, T_a^+$ be the transition functions computed when constructing the TC of a, and let $R_a^0, \ldots, R_a^+$ be the correspondent exponential reachability function. Alg. 1 shows the structure of the REPETITIONS algorithm, taking as input an action a, the state before (s) and after (s'') the repetition of a and, initially, the index p is either the fix-point index or the timeout index of $\mathcal{T}_a^+$. The algorithm, using T_a^j and starting from $j = 0$, checks whether s'' is reachable from s in at most 2^j repetitions of a (Lem. 1) increasing j upon failure. When such a j is found, it means that the rolling of a lies in $(2^{j-1}, 2^j]$. Line 7 employs the following Lemma:

Lemma 3. *Let s be a state in which a is applicable. For any $i \geq 1$, there exists at most one state s' such that $R_a^i(s, s')$.*

Sketch Proof. The proof follows from the determinism of $res(s, a)$, since, applying a in any state s leads to only one state s'. □

Thus, using the exponential reachability function R_a^{j-1}, Line 7 finds the intermediate state s' reached after 2^{j-1} repetitions of a from s and calls again the function REPETITIONS, computing the repetitions of a needed to reach s'' from s'. If state s'' is not reachable from s in up to 2^p repetitions of a, then REPETITIONS returns -1, signalling unreachability.

Theorem 2. *Let s, s'' be states, a be an action applicable in s, and $p \geq 0$. If $r = \text{REPETITIONS}(a, s, s'', p) > 0$ then $s'' = res(s, a^r)$.*

Proof. By induction. If $s'' = res(s, a)$, then $T_a^0(s, s'') = \top$ by construction and $\text{REPETITIONS}(a, s, s'', p) = 1$. Suppose that the thesis holds for all $r \in [1, 2^{j-1}]$, for some $j \in [1, m]$. If $r \in (2^{j-1}, 2^j]$, then, by Lem. 1, we know $T_a^0(s, s'') = \cdots = T_a^{j-1}(s, s'') = \bot$ and $T_a^j(s, s'') = \top$. Thus by Lem. 3, we know there exists only one state s' such that $R_a^{j-1}(s, s') = \top$ and, by Lem. 2 we know that $s' = res(s, a^{(2^{j-1})})$ thus, to reach s'' from s', we still need $r' = r - 2^{j-1}$ repetitions. Since $r \in (2^{j-1}, 2^j]$, then $r' \in [1, 2^{j-1}]$ and following the inductive hypothesis, r' can be computed as $\text{REPETITIONS}(a, s', s'', j - 1)$. □

Finally, let μ be a model of Π_n^+. We denote by

$$\text{REPEATEDPLAN}(\Pi_n^+, \mu) = a_1^{r_1}; \ldots; a_n^{r_n}$$

the sequence of actions where for each $i \in [1, n]$, each a_i is the action as of Eq. 4 , $r_i = \text{REPETITIONS}(a_i, s_{i-1}, s_i, p_i)$, s_{i-1} and s_i are the states before and after the repetition of a_i, obtained from μ, i.e., $s_i = \{v \mid v_i \in \mathcal{X}_i, \mu(v_i)\}$ (and similarly for s_{i-1}) and p_i is the fix point or timeout index found when computing each TC $\mathcal{T}_{a_i}^+(\mathcal{X}, \mathcal{X}')$. In the next section, we prove the correctness and completeness of the approach, i.e., $\text{REPEATEDPLAN}(\Pi^+, \mu)$ is a valid plan for Π.

5.2 Correctness, Completeness and Domination

Let Π^0, with symbolic transition relation $\mathcal{T}^0(\mathcal{X}, \mathcal{A}, \mathcal{X}')$ be the closure-encoding constructed as Π^+ but where, for each action a the TC $\mathcal{T}_a^+(\mathcal{X}, \mathcal{X}')$ is substituted with the transitive relation $\mathcal{T}_a^0(\mathcal{X}, \mathcal{X}')$ i.e., a can be executed at most once.

Theorem 3. *Let Π be a classical planning task with* CE*s. The encoding Π^0 is correct and complete.*[2]

Proof. (Correctness) Each model μ of $\mathcal{T}^0(\mathcal{X}, \mathcal{A}, \mathcal{X}')$ corresponds to a sequence of actions $\alpha = a$ with only one action, due to the $\text{amo}^+(A)$ axioms. The action a is applicable in the state $s = \{v \mid v \in \mathcal{X}, \mu(v)\}$ due to the axioms pre_a and $\text{conflict}_a(V)$ inherited from $\mathcal{T}_a(\mathcal{X}, \mathcal{X}')$ in $\text{closure}^+(A, V)$ and the last state induced by a is $s' = \{v \mid v' \in \mathcal{X}, \mu(v')\}$ due to $\text{eff}_a(V)$ and $\text{frame}^+(A, V)$. Both s and $s' \models C$ due to the $\text{constraints}^+(V, C)$ axioms. *(Completeness)* Let a be an action applicable in a state s and let $s' = res(s, a)$. The formula $\mathcal{T}^0(\mathcal{X}, \mathcal{A}, \mathcal{X}')$ joined with the conjunction of the set $\{a^+\} \cup \{\neg a_1^+ \mid a_1 \in A, a_1 \neq a\} \cup s \cup \{\neg v \mid v \notin s\} \cup s' \cup \{\neg v' \mid v \notin s'\}$ is equivalent to $\top$. □

Let $M \colon A \mapsto \mathbb{N}$. We denote by Π^M the encoding constructed in the same way as Π^+ but where, for each action a, the TC $\mathcal{T}_a^+(\mathcal{X}, \mathcal{X}')$ is substituted with the transitive relation $\mathcal{T}_a^{M(a)}(\mathcal{X}, \mathcal{X}')$, obtained in the construction of $\mathcal{T}_a^+(\mathcal{X}, \mathcal{X}')$

Theorem 4. *Let Π be a classical planning task with* CE*s. The encoding Π^M is correct and complete.*

Proof. If, for each $a \in A$, $M(a) = 0$, we fall back to Thm. 3. *(Correctness)* Each model μ of $\mathcal{T}^M(\mathcal{X}, \mathcal{A}, \mathcal{X}')$ corresponds to a sequence of actions $\alpha = a^r$, with $r \geq 1$. Let p be the fix point index at which the TC $\mathcal{T}_a^+(\mathcal{X}, \mathcal{X}')$ is found. If $M(a) \leq p$ then Thm. 2 guarantees that $r = \text{REPETITIONS}(a, s, s', p) > 0$ with $s = \{v \mid v \in \mathcal{X}, \mu(v)\}$ and $s' = \{v \mid v' \in \mathcal{X}', \mu(v')\}$. If $M(a) > p$ the fix point computation ensures that $\mathcal{T}_a^{M(a)}(\mathcal{X}, \mathcal{X}') \equiv \mathcal{T}_a^p(\mathcal{X}, \mathcal{X}')$. *(Completeness)* The completeness follows directly from Thm. 3 since, the sequence $\alpha = a^1$ is also a solution of Π^M. □

Theorem 5. *Let Π be a classical planning task with* CE*s. The closure-encoding Π^+ is correct and complete.*

Proof. Special case of Thm. 4 where $M(a)$ corresponds to the fix point of $\mathcal{T}_a^+(\mathcal{X}, \mathcal{X}')$ for each action $a \in A$. □

Given two correct and complete PAS encoding E_1 and E_2 we say that E_1 *dominates* E_2 ($E_1 \leq E_2$) if, for each bound n, $\Pi_n^{E_2}$ satisfiability implies satisfiability of $\Pi_n^{E_1}$. Thus let $n_1, n_2 \in \mathbb{N}$ be the smallest bounds such that $\Pi_{n_1}^{E_1}$ and $\Pi_{n_2}^{E_2}$ are satisfiable, if $E_1 \leq E_2$ then $n_1 \leq n_2$.

Theorem 6. $\Pi^+ \leq \Pi^M \leq \Pi^0$.

Proof. We have to prove that, for any bound n, if Π_n^M is satisfiable then also Π_n^0 is satisfiable, which follows from the fact that any model μ of $\mathcal{T}^0(\mathcal{X}, \mathcal{A}, \mathcal{X}')$ is also a model of $\mathcal{T}^M(\mathcal{X}, \mathcal{A}, \mathcal{X}')$. The same applies for $\Pi^+ \leq \Pi^M$. □

Thm. 6 provides the practical approach described in the introduction: if computing the TC $\mathcal{T}_a^+(\mathcal{X}, \mathcal{X}')$ at fix point p for each action a takes too long, we can stop at some intermediate step $m \in [0, p]$ and still have benefits from rolling using $\mathcal{T}_a^m(\mathcal{X}, \mathcal{X}')$.

[2] The definitions of correctness and completeness are in Sec. 2.2.

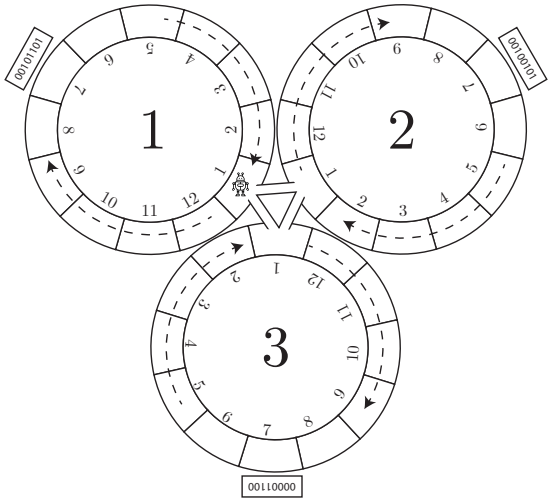

Figure 1: Example of the `Loops` domain with 3 loops of length 12.

6 Experimental Analysis

We now present an experimental analysis run on three novel domains where any valid plan must contain consecutive repetitions of an action. We run the analysis by comparing Π^+ with Π^0, i.e., the approach where rolling is disabled.

6.1 Domains

`Counter`. The `Counter` domain models a counter of x, a number represented with B bits. An action can increase x by one, except if $x = 2^{B-1}$ when x is reset back to 0. Starting from $k > 0$, x has to reach $k - 1$. The number x is represented by the variables $V = \{x_B, \dots, x_1\}$ with x_B being the most significant bit. There is only one action `inc` with $\text{pre}(\texttt{inc}) = \top$ and with $\text{post}(\texttt{inc}) = \{\texttt{ix}_B, \dots, \texttt{ix}_1, \texttt{ow}\}$ where (i) $\texttt{ix}_1$ models the transition from 0 to 1 of x_1, (ii) $\texttt{ix}_i$, for $i \in (1, B]$, models the transition of the substring of bits $x_i; x_{i-1}; \dots; x_1$ from $01 \dots 1$ to $10 \dots 0$, and (iii) `ow` models the transition of x from $1 \dots 1$ to $0 \dots 0$. There are no constraints (i.e., $C = \top$). Any valid plan must contain $2^B - 1$ repetitions of `inc`.

`Cube`. The `Cube` domain models three robots which move on the three axis x, y, z in a cubic space with side L. The three robots start at the vertices $(L, 0, 0)$, $(0, L, 0)$ and $(0, 0, L)$ and have to reach the vertex (L, L, L). The problem is a more complicated version of Ex. 1. There are $9 \times L$ variables modelling the integer position in $[1, L]$ of the three robots in x, y, z. For each robot, there are three actions to move forward, right or up and increase by one either x,y or z. As for Ex. 1 the constraints in C state that each robot must always be in only one position. The smallest valid plan must contain at least $3 \times 2 \times (L - 1)$ actions, i.e., the actions to move through two sides of the cube of the three robots.

`Loops`. The `Loops` domain puts together all the previous examples. A robot can move on K loops of length L, in each loop there is a counter of B bits initialized at a random number. Fig. 1 shows an example with $K = 3$ loops

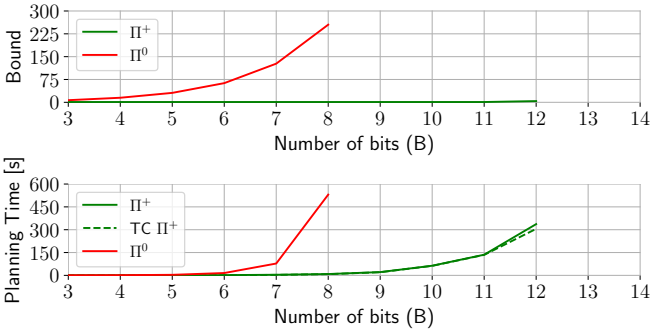

Figure 2: Experimental analysis on the `Counter` domain.

of length $L = 12$ and with counters of $B = 8$ bits. Each cell in the loop is indexed from 1 to L. Each loop's counter is at cell $\lceil L/2 \rceil + 1$. A robot can only move clockwise in the loop. When the robot is in cell 1 it can move in cell 1 of another loop. When in counter's cell, the robot can increase the counter by one. The goal is to have all the counters with the same value.

6.2 Analysis

For each domain presented and for each problem of increasing size, we compare two encodings: Π^+, i.e., the encoding presented in this paper employing the TC to execute (possibly) more than one action in one step, and Π^0, i.e., the approach presented in Thm. 3, where at most one action can be executed at each step. For both encodings, the PDDL domain and problem files (i.e., the planning task Π) are first encoded into Π^+ and Π^0, and then translated into propositional formulae as shown in Eq. 1, starting from $n = 0$. We implemented the approach on the PATTY solver[3] [Cardellini *et al.*, 2024] which makes use of `Z3` v4.12.2 [de Moura and Bjørner, 2008] to compute the model (if any) satisfying Π^+_n and Π^0_n. Each problem and encoding has a time limit of 10 minutes on an Intel Xeon Platinum 8000 3.1GHz with 8GB of RAM. The results for the `Counter`, `Cube` and `Loops` domains are shown in Figs. 2, 3 and 4, respectively. For each domain, we show how the bound and planning time change while varying the size of the problem. The red line represents Π^0 and the green line Π^+. For the planning time, we consider the time elapsed from the parsing of the PDDL problem to the output of the plan. We indicate with a dashed green line the time it takes to compute the TC of all the actions in the planning task. The transitive closure is computed via BDDs, employing the `pyeda` tool[4]. In the BDD the variables in $\mathcal{X} \cup \mathcal{X}'$ have been ordered alphabetically by name. We limit the computation of all actions' TCs to 3 minutes, equally divided among all actions, and for each action we return the last computed transition relation before the time limit.

In the `Counter` domain, we recall that a valid plan must contain at least $2^B - 1$ `incr` actions. In Fig. 2 we see how, in fact, the bound required by Π^0 to find a plan increases exponentially with B. For Π^+ instead, the bound is 1 for $B \in [3, 11]$ where the TC can be computed under the

[3]https://pattyplan.com

[4]https://pyeda.readthedocs.io/

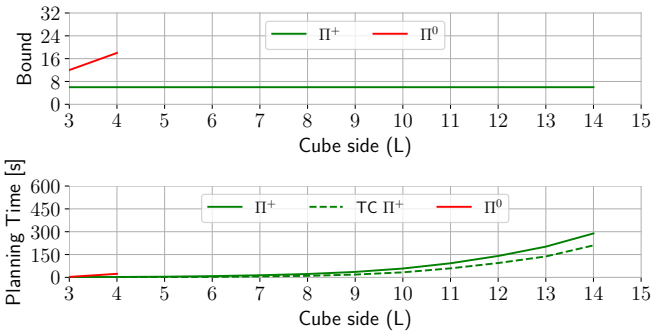

Figure 3: Experimental analysis on the Cube domain.

time limit. When B is 12 the TC computation reaches the time-limit and $\mathcal{T}^{10}_{\texttt{incr}}$ is returned, modelling 2^{10} repetitions of `incr` (Lem. 1), and requiring a bound of 4 to find the valid plan of $2^{12}-1$ repetitions. Notably, when the fix point is reached, the total time is mainly the time to compute the TC.

In the Cube domain, the shortest valid plan has $6\times(L-1)$ actions which are reflected in the bound required by Π^0 in Fig. 3. When $L=5$ the bound required by Π^0 would be 24, but the system timeouts before being able to find a solution. For Π^+ instead, the TC can be computed under the time-limit until $L=15$. Due to the $\text{amo}^+(A)$ axioms of the encoding, the bound for Π^+ is equal to 6, i.e., the different actions in the plan. As discussed in Sec. 4.3, the use of constraints can allow, in some cases, the transitive closure to be smaller. Indeed, comparing the transitive closure size (in terms of BDD nodes) of the planning task with and without constraints, we obtain a 40% reduction in the BDDs nodes of the TC.

In the more complicated Loops domain, for each dimension, i.e., K, L and B, we plot how the bound and planning time change by varying one dimension and keeping the other one fixed to their minimum value (i.e., 3). By increasing the number of loops and counters (i.e., K), we see in the first and second chart from the top of Fig. 4, how the bounds for both Π^+ and Π^0 increase linearly with K. However, the bound of Π^0 diverges linearly from the one of Π^+, since, for each loop, a valid plan has $L=3$ actions to move through the loop and at most 2^3-1 increases of the associated counter. By only increasing the length of the loops L in the third and fourth chart from the top, we see how again the bound of Π^0 is greater than the bound of Π^+. However, in this case, the planning time of Π^0 is shorter than Π^+, since, for smaller bounds, the computation of the TC is actually pricier than iteratively increasing the bound. The fifth and sixth charts present the same behaviour of the Counter domain.

In all the domains we have experimentally confirmed Thm. 6, showing that $\Pi^+ \leq \Pi^0$.

7 Conclusions and Future Work

We presented a technique to perform rolling in classical planning with CEs and constraints, which was previously unexplored. We theoretically and experimentally demonstrated that the approach is interesting and beneficial. Several works have been published introducing new approaches to reduce the bound required to find a solution, being one of the weaknesses of the PaS approach [Wehrle and Rintanen, 2007; Balyo, 2013; Bofill *et al.*, 2016; Cardellini *et al.*, 2024; Cardellini and Giunchiglia, 2025]. In the future, we plan to (i) minimize the bound even further by integrating our TC approach in the latest SoTA PaS approach [Cardellini *et al.*, 2024] where a finite sequence of actions is employed to apply different actions in the same step and (ii) study how to compute the TC of multiple actions, effectively combining the possible effects of different actions in one transition relation.

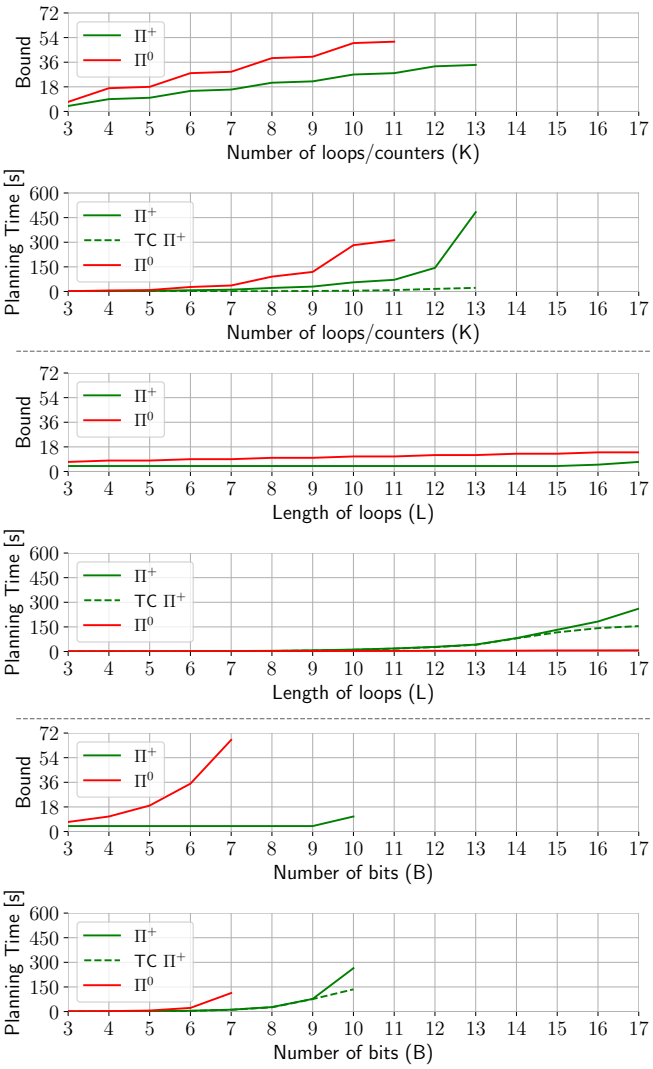

Figure 4: Experimental analysis on the Loops domain. For each dimension, i.e., K, L and B, we plot how the bound and planning-time change by varying one dimension and keeping the other fixed to their minimum value.

Acknowledgments

Matteo Cardellini and Enrico Giunchiglia were each partially supported by the projects FAIR (PE00000013) and SERICS (PE00000014), respectively, under the NRRP MUR program funded by the EU - NGEU.

References

[Balyo, 2013] Tomás Balyo. Relaxing the Relaxed Exist-Step Parallel Planning Semantics. In *25th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2013, Herndon, VA, USA, November 4-6, 2013*, pages 865–871. IEEE Computer Society, 2013.

[Bofill *et al.*, 2016] Miquel Bofill, Joan Espasa, and Mateu Villaret. The RANTANPLAN planner: system description. *Knowl. Eng. Rev.*, 31(5):452–464, 2016.

[Bryant, 1985] Randal E. Bryant. Symbolic manipulation of Boolean functions using a graphical representation. In Hillel Ofek and Lawrence A. O'Neill, editors, *Proceedings of the 22nd ACM/IEEE conference on Design automation, DAC 1985, Las Vegas, Nevada, USA, 1985*, pages 688–694. ACM, 1985.

[Burch *et al.*, 1992] Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. Symbolic Model Checking: 10ˆ20 States and Beyond. *Inf. Comput.*, 98(2):142–170, 1992.

[Cardellini and Giunchiglia, 2025] Matteo Cardellini and Enrico Giunchiglia. Temporal numeric planning with patterns. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(25):26481–26489, Apr. 2025.

[Cardellini *et al.*, 2024] Matteo Cardellini, Enrico Giunchiglia, and Marco Maratea. Symbolic Numeric Planning with Patterns. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, February 20-27, 2024, Vancouver, Canada*, pages 20070–20077. AAAI Press, 2024.

[Clarke *et al.*, 1996] Edmund M. Clarke, Kenneth L. McMillan, Sérgio Vale Aguiar Campos, and Vasiliki Hartonas-Garmhausen. Symbolic Model Checking. In Rajeev Alur and Thomas A. Henzinger, editors, *Computer Aided Verification, 8th International Conference, CAV '96, New Brunswick, NJ, USA, July 31 - August 3, 1996, Proceedings*, volume 1102 of *Lecture Notes in Computer Science*, pages 419–427. Springer, 1996.

[de Moura and Bjørner, 2008] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: An Efficient SMT Solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[Gazen and Knoblock, 1997] B. Cenk Gazen and Craig A. Knoblock. Combining the Expressivity of UCPOP with the Efficiency of Graphplan. In Sam Steel and Rachid Alami, editors, *Recent Advances in AI Planning, 4th European Conference on Planning, ECP'97, Toulouse, France, September 24-26, 1997, Proceedings*, volume 1348 of *Lecture Notes in Computer Science*, pages 221–233. Springer, 1997.

[Gerevini *et al.*, 2024] Alfonso Emilio Gerevini, Francesco Percassi, and Enrico Scala. An Effective Polynomial Technique for Compiling Conditional Effects Away. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, February 20-27, 2024, Vancouver, Canada*, pages 20104–20112. AAAI Press, 2024.

[Katz, 2019] Michael Katz. Red-Black Heuristics for Planning Tasks with Conditional Effects. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7619–7626. AAAI Press, 2019.

[Kautz and Selman, 1992] Henry A. Kautz and Bart Selman. Planning as Satisfiability. In Bernd Neumann, editor, *10th European Conference on Artificial Intelligence, ECAI 92, Vienna, Austria, August 3-7, 1992. Proceedings*, pages 359–363. John Wiley and Sons, 1992.

[Matsunaga *et al.*, 1993] Yusuke Matsunaga, Patrick C. McGeer, and Robert K. Brayton. On Computing the Transitive Closure of a State Transition Relation. In Alfred E. Dunlop, editor, *Proceedings of the 30th Design Automation Conference. Dallas, Texas, USA, June 14-18, 1993*, pages 260–265. ACM Press, 1993.

[Nebel, 2000] Bernhard Nebel. On the Compilability and Expressive Power of Propositional Planning Formalisms. *J. Artif. Intell. Res.*, 12:271–315, 2000.

[Rintanen, 2011] Jussi Rintanen. Heuristics for Planning with SAT and Expressive Action Definitions. In Fahiem Bacchus, Carmel Domshlak, Stefan Edelkamp, and Malte Helmert, editors, *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011*. AAAI, 2011.

[Röger *et al.*, 2014] Gabriele Röger, Florian Pommerening, and Malte Helmert. Optimal Planning in the Presence of Conditional Effects: Extending LM-Cut with Context Splitting. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 765–770. IOS Press, 2014.

[Scala *et al.*, 2016] Enrico Scala, Miquel Ramírez, Patrik Haslum, and Sylvie Thiébaux. Numeric Planning with Disjunctive Global Constraints via SMT. In Amanda Jane Coles, Andrew Coles, Stefan Edelkamp, Daniele Magazzeni, and Scott Sanner, editors, *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2016, London, UK, June 12-17, 2016*, pages 276–284. AAAI Press, 2016.

[Sipser, 1997] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[Wehrle and Rintanen, 2007] Martin Wehrle and Jussi Rintanen. Planning as Satisfiability with Relaxed ∃-Step Plans. In Mehmet A. Orgun and John Thornton, editors, *AI 2007: Advances in Artificial Intelligence, 20th Australian Joint*

Conference on Artificial Intelligence, Gold Coast, Australia, December 2-6, 2007, Proceedings, volume 4830 of *Lecture Notes in Computer Science*, pages 244–253. Springer, 2007.