An Approach to Quantify Plans Robustness in Real-world Applications

Francesco Percassi[1] , **Sandra Castellanos-Paez**[1] , **Romain Rombourg**[2] and **Mauro Vallati**[1]
[1]School of Computing and Engineering, University of Huddersfield, Huddersfield, United Kingdom
[2]G2ELab, Grenoble INP, CNRS, Université Grenoble Alpes, Grenoble, France
{f.percassi, s.castellanos}@hud.ac.uk, romain.rombourg@grenoble-inp.fr, m.vallati@hud.ac.uk

Abstract

Automated planning systems are increasingly deployed in real-world applications, often characterised by uncertainty and noise stemming from sensors, actuators, and environmental conditions. Under such circumstances, improving the deployability of generated plans requires assessing their robustness to varying conditions, thereby reducing the need for costly replanning. Replanning can be computationally intensive and may hinder the practical applicability of planning systems. In many domains, such as urban traffic control or underwater exploration, it is often sufficient for plans to reach an acceptable region rather than the exact goal.

A key distinction in this context lies between *valid* plans (which achieve the intended goal under ideal conditions) and *executable* plans (which remain feasible under uncertainty or perturbation). This paper formalises the notion of *execution-invariant planning tasks*, in which plans are robust to noise and uncertainty. To foster the adoption of automated planning in real-world settings, we propose a statistical framework for evaluating plan robustness, offering a quantifiable measure of a plan's ability to reach a goal within a specified tolerance under diverse perturbations or uncertainty. We validate our approach in two real-world domains, demonstrating its effectiveness.

1 Introduction

Automated planning is increasingly exploited to tackle complex problems in diverse real-world applications, including urban traffic control [Vallati *et al.*, 2016], digital forensics [Khan *et al.*, 2023], unmanned aerial vehicle control [Kiam *et al.*, 2020], and pharmacokinetic optimisation [Alaboud and Coles, 2019; Alon *et al.*, 2024]. However, the effective use of planning systems in such scenarios is often hindered by inherent uncertainty and noise.

Imperfect sensors, unreliable actuators, and unpredictable external factors, such as weather conditions or unexpected traffic demands, can render plans ineffective when deployed. When execution deviates from the expected trajectory, a traditional approach is to replan or to perform plan repair [Fox *et al.*, 2006a], but this can significantly diminish the utility of planning systems. Replanning and repairing are computationally expensive [Percassi *et al.*, 2023], especially when using expressive formalisms like PDDL+ [Fox and Long, 2006], and can disrupt ongoing operations. Consequently, the practical usability of plans in real-world scenarios heavily depends on their robustness to varying conditions, minimising the need for costly additional operations.

Several works have addressed plan robustness from various perspectives. Some focus on generating robust plans from the outset, explicitly considering exogenous events and dynamic environments [Fritz and McIlraith, 2009; Chrpa and Karpas, 2024]. Conformant probabilistic planning addresses uncertainty by synthesising plans that achieve a goal with a given probability under stochastic initial states and actions [Domshlak and Hoffmann, 2007; Taig and Brafman, 2015]. However, these approaches are limited to classical planning settings and rely on fully specified probability distributions. Others analyse the robustness of existing plans by investigating their sensitivity to action timing uncertainty [Fox *et al.*, 2006b]. Cashmore *et al.* (2020) propose an approach to identify decoupled robustness envelopes for temporal plans, characterising their validity concerning temporal constraints and contingencies. More recently, Percassi and Vallati (2024) introduced a "what-if" analysis for urban traffic control, assessing plan performance under manually injected perturbations. Although focusing on explainability rather than robustness, Sarwar *et al.* (2023) aimed to improve system reliability through contrastive plan comparisons in hybrid models.

However, these analyses often neglect the true underlying distribution of variables, limiting their effectiveness when historical data is available. This motivates the need for approaches leveraging such data for a more accurate assessment of robustness. Furthermore, in many real-world domains operating under noisy and dynamic conditions, goals often represent desired outcomes rather than strict targets. Achieving the precisely stated goal is frequently unnecessary; it suffices for plans to guide execution towards an acceptable region around the goal. This is evident in domains like underwater exploration [Chrpa *et al.*, 2017], where the objective is often to provide a general survey of an area rather than reaching a precise set of waypoints. Another example arises in autonomous driving, where arriving within a reasonable proximity of a destination is typically sufficient. This shift from

strict goal satisfaction to achieving acceptable outcomes necessitates new approaches to robustness evaluation, focusing on plan executability rather than strict validity.

To advance the adoption of automated planning in real-world applications, this paper defines the concept of execution-invariant planning tasks, scenarios in which plans are guaranteed to remain executable despite noise and uncertainty. We propose a novel statistical framework for evaluating the robustness of such plans, offering a quantifiable measure of their resilience in reaching a goal within defined bounds. Our approach can explicitly incorporate statistical information about perturbations, allowing for more accurate and informative robustness assessments when historical data or variable distributions are available. We demonstrate the applicability of our framework on realistic benchmarks, showcasing its effectiveness in evaluating plan robustness.

2 Background

In this section, we briefly introduce the syntax and semantics of discrete-time PDDL+, which serves as the planning formalism used in our experiments. This choice is not a requirement of the proposed framework, but reflects the planning formalism used for the experimental evaluation.

A *discrete-time* PDDL+ *planning task*, denoted as Π, is defined as a tuple $\langle F, X, I, G, A, E, P\rangle$, together with a *time discretisation step* $\delta \in \mathbb{Q}^+$, which determines the temporal granularity of the model. F is a finite set of Boolean variables, each taking values in $\{\bot, \top\}$. X is a finite set of numeric variables, each ranging over $\mathbb{Q}$. A *state* s is a complete assignment of values to all variables in F and X. For a state s and variable $v \in F \cup X$, $s[v]$ denotes the value of v in s. The initial state I is such an assignment, describing the system's starting configuration. The goal G describes the conditions that must hold in the final state, and is defined as a set of Boolean and numeric conditions. Boolean conditions take the form $\langle v = b\rangle$, where $v \in F$ and $b \in \{\bot, \top\}$. Numeric conditions take the form $\langle \varphi \bowtie 0\rangle$, where φ is an arithmetic expression over X, and $\bowtie \in \{>, \geq, =, \leq, <\}$. A, E, and P are the finite sets of actions, events and processes, respectively. Each element $z \in A \cup E \cup P$ is described as a pair $\langle \mathit{pre}(z), \mathit{eff}(z)\rangle$, where $\mathit{pre}(z)$ is the precondition and $\mathit{eff}(z)$ is the effect associated with z. Preconditions $\mathit{pre}(z)$ are expressed as propositional formulae composed of Boolean and numeric conditions. For actions and events, the effects $\mathit{eff}(z)$ consist of a set of assignments to Boolean and numeric variables, i.e., $\langle v := b\rangle$ and $\langle v := \varphi\rangle$, respectively. Processes behave differently from actions and events. Like the other elements, processes have preconditions, but their effects are described as a set of discrete-time numeric effects of the form $\langle v, \varphi\rangle$. This means that, as time progresses by steps of size δ, each such expression contributes incrementally to the value of v. With a slight abuse of notation, this can be expressed as $v(t + \delta) = v(t) + \varphi(t) \cdot \delta$.

A PDDL+ plan π_t is represented as a pair $\langle \pi, t_e\rangle$, where π is a sequence of timestamped actions $\langle\langle a_1, t_1\rangle, \ldots, \langle a_n, t_n\rangle\rangle$, and $t_e \in \mathbb{Q}_{\geq 0}$ denotes the plan's makespan or total duration.

The validity of π_t is determined by simulating its execution over discrete timestamps multiples of δ. This simulation tracks all changes to the state over time by considering the effects of actions, any events triggered during execution, and processes active at each time step. Each action a applied at its timestamp t_i induces an instantaneous state transition, applying its effects and producing a new state. After the application of each action and after the time progresses by δ, events are evaluated and applied if their preconditions are satisfied. At every discrete timestamp within the interval $\{0, \delta, 2 \cdot \delta, \ldots, t_e\}$, processes also contribute to temporal state transitions. This iterative procedure continues throughout the entire makespan. A plan π_t is valid for Π if and only if every action is applicable and the final state satisfies the goal condition G. For further details about the semantics we are considering, refer to [Fox *et al.*, 2012; Percassi *et al.*, 2025].

3 Case Studies

This section presents the three case studies that will be considered in the paper. The first domain, Non-Linear-Car, is a toy problem introduced for explanatory purposes and will be used as a running example. Urban Traffic Control and Baxter are drawn from real-world applications and will be considered for the experimental analysis.

Non-Linear-Car In this well-known domain, the aim is to guide a car along a straight road. The domain involves three numeric variables, i.e., $X = \{x, v, a\}$, where x, v, and a represent the car's position, velocity, and acceleration, respectively. c_{drag}, i_a, and $v_{threshold}$ are constants that characterise drag, acceleration intensity, and the velocity above which drag begins to apply. Two actions control acceleration, i.e., $A = \{acc, dec\}$, where $acc = \langle \top, \{\langle a := a + i_a\rangle\}\rangle$ and $dec = \langle \top, \{\langle a := a - i_a\rangle\}\rangle$. In the initial state, the car is stationary, and c_{drag} and i_a are set according to estimated values. The physics of the car is modelled through the processes $P = \{\rho_{move}, \rho_{drag}\}$. $\rho_{move} = \langle \top, \{\langle x, v\rangle, \langle v, a\rangle\}\rangle$ accounts for the position and velocity of the car. $\rho_{drag} = \langle\langle v \geq v_{threshold}\rangle, \{\langle v, -c_{drag} \cdot v\rangle\}\rangle$ models the drag force, which is linearly dependent on the velocity of the car, and acts as a resistive force opposing the motion whenever $v_{threshold}$ is reached. The objective is to control the car through a sequence of timed accelerations and decelerations, bringing it to a specific distance while returning it to a stationary state.

Urban Traffic Control The urban traffic control (UTC) PDDL+ domain addresses the problem of optimising traffic signals in an urban road network, to maximise throughput and minimise congestion [Kouaiti *et al.*, 2024]. The network is modelled as a directed graph, where nodes correspond to junctions and edges denote road links. Each junction operates through predefined traffic signal configurations, which regulate vehicle flows between incoming and outgoing links. Traffic flows are modelled as continuous processes, while transitions between signal stages are represented as discrete events. The turn rates, defined as constants of the problem, determine the proportion of vehicles that move from an incoming link to an outgoing one when a green signal is active between them. The planner is responsible for selecting which configuration to apply at each junction and when. In the formulation under consideration, the goal is to ensure that

a specified number of vehicles traverse a corridor within the network while avoiding saturation at the corridor's exit.

Baxter This domain model encodes in PDDL+ a robotic manipulation scenario of an articulated object in a three-dimensional space [Bertolucci *et al.*, 2020]. More specifically, it considers a Baxter robot provided with two arms, tasked to manipulate an object into a desired final configuration. The manipulation involves a sequence of actions that allows the Baxter to grasp two links and modify the angle of the joint connecting these links together. Joints are not explicitly represented as separate entities in the model. Instead, their presence is implicitly defined by the connection between two links and by the value of a numeric function that captures their relative orientation on the vertical or horizontal plane. Notably, the effects of the manipulation of two connected links are propagated via corresponding predicates to the upstream part of the object.

4 Assumptions and Notation

Before introducing the concepts of execution-invariant tasks and the framework for assessing plans' robustness, it is necessary to outline notations and assumptions.

Usage of Invariant Variables We require that arithmetic expressions follow a specific syntactic form. In particular, we define expressions as $\varphi := \varphi + \varphi \mid \varphi - \varphi \mid \varphi \cdot \varphi \mid \varphi / \varphi \mid v$, where v denotes a numeric variable. This formulation explicitly excludes the use of rational values, thereby enforcing that all constant parameters of the domain must be modelled as variables. For example, in the Non-Linear-Car domain, the parameters c_{drag}, i_a and $v_{threshold}$ must be included in X.

Numeric Variables Domains Although our framework is general, it is particularly suitable for planning tasks involving numeric variables with naturally bounded domains. In many application contexts, numeric variables are constrained by physical or operational limits derived from the domain. For example, in the UTC domain, the occupancy of a link l, denoted as $occupancy_l$, varies within $[0, capacity_l]$, where $capacity_l$ is the link's maximum capacity. This is a well-known observation in the literature and has important implications for the decidability of numeric planning tasks [Helmert, 2002; Gigante and Scala, 2023].

In what follows, we restrict our attention to numeric variables with known bounded domains. Specifically, for each numeric variable v, we assume a domain $\mathbb{D}_v = [\underline{v}, \overline{v}]$ with $\underline{v}, \overline{v} \in \mathbb{Q}$ and $\underline{v} \leq \overline{v}$, as introduced above. Moreover, regarding notation, given a set of variables V, we denote by $S(V)$ the set of all complete assignments to V over their respective domains. While inferring such bounds automatically is not always trivial [Kuroiwa *et al.*, 2023], in practice, meaningful ranges can often be identified from domain knowledge or defined based on application requirements.

Executable and Valid Plans In this work, we are interested in distinguishing between the notions of executability and validity of a plan. Given a plan π for a planning task Π, we define π to be *executable* if every action in π is applicable in the state in which it is executed, assuming actions are applied sequentially starting from the initial state, and following the semantics of the chosen planning formalism. A plan is valid if it is executable and, in addition, it reaches a state satisfying the goal. The sets of all executable and valid plans for Π are denoted by $\textsc{Exec}(\Pi)$ and $\textsc{Plans}(\Pi)$, respectively.

In terms of notation, since we will investigate the validity of a plan under unknown variations of the initial state, we denote by Π the original planning task and by $\Pi[I']$ the planning task where the nominal initial state I is replaced by I'. The initial states are assumed to be realisations of a random variable $\mathcal{I}$ with a distribution $f_{\mathcal{I}}$ representative of its possible outcomes. In general, the distribution $f_{\mathcal{I}}$ is unknown, but we assume access to a set of its realisations in the form of historical data.

5 Execution-Invariant Planning Tasks

In many real-world scenarios, certain numeric variables may vary without compromising the ability to execute a plan. To capture this intuition, we first define how such variations can be represented through perturbation state spaces. Then, we introduce the notion of execution-invariant planning tasks, where all plans remain executable despite changes in specific initial values. As we will discuss, this class of tasks is especially well-suited to the robustness framework we propose.

Definition 1 (Perturbation State Space)**.** *Let Π be a planning task with variables V where $\tilde{V} \subseteq V$ and $\tilde{V} \neq \emptyset$, and let $s \in S(\Pi)$. The* perturbation state space induced by s over $\tilde{V}$ *is defined as:*

$$\tilde{S}(s, \tilde{V}) = \{\tilde{s} = s_{fix} \cup s_{noise} \mid s_{noise} \in S(\tilde{V})\}$$
$$s_{fix} = \{x := s[v] \mid v \in V \setminus \tilde{V}\}.$$

Intuitively, the state space $\tilde{S}(s, \tilde{V})$ is obtained by keeping the state s fixed for the variables that are not perturbed, i.e., s_{fix}, while allowing all the variables in $\tilde{V}$ to vary within their domain, i.e., s_{noise}. Henceforth, $\tilde{S}$ denotes a generic perturbation state space induced by a set of variables $\tilde{V}$ and an initial state of a planning task Π, with $\tilde{s}$ representing any state in this space.

The perturbation state space can be partitioned based on a given plan π for Π, according to how states within it impact the validity of π. In particular, given a perturbed state $\tilde{I}$, we can test the validity of π for the task $\Pi[\tilde{I}]$. This validation evaluation can result in three possible outcomes: (i) the plan is valid, meaning it is executable and achieves the goal, (ii) the plan is not valid but is executable, and (iii) the plan is not valid and is not executable. The states in a perturbation state space can be grouped based on these three outcomes.

Definition 2 (Validity, Executability and Failure Region)**.** *Let Π be a planning task and $\tilde{S}$ a perturbation state space for Π. A plan π for Π induces a partition of $\tilde{S} = VAL(\tilde{S}, \pi) \cup EX(\tilde{S}, \pi) \cup FAIL(\tilde{S}, \pi)$ such that:*

$$VAL(\tilde{S}, \pi) = \{\tilde{I} \in \tilde{S} \mid \pi \in \textsc{Plans}(\Pi[\tilde{I}])\}$$
$$EX(\tilde{S}, \pi) = \{\tilde{I} \in \tilde{S} \mid \pi \notin \textsc{Plans}(\Pi[\tilde{I}]),\ \pi \in \textsc{Exec}(\Pi[\tilde{I}])\}$$
$$FAIL(\tilde{S}, \pi) = \{\tilde{I} \in \tilde{S} \mid \pi \notin \textsc{Exec}(\Pi[\tilde{I}])\},$$

where VAL, EX and FAIL are referred to as plan validity, executability *and* failure region*, respectively.*

With this definition in place, we can introduce the subclass of planning tasks we are interested in studying in this work. Specifically, we focus on planning tasks where the plan remains executable even if certain (numeric) variables in the initial state are perturbed in a given space. First, given a plan, we characterise this property.

Definition 3 (Execution-Invariant Plan)**.** *Let Π be a planning task and $\tilde{S}$ a perturbation state space for Π and some variables $\tilde{V}$. A plan $\pi \in \textsc{Exec}(\Pi)$ is said to be* execution invariant *to $\tilde{V}$ if and only if $FAIL(\tilde{S}, \pi) = \emptyset$.*

In other words, an execution-invariant plan remains executable regardless of variations in the initial state. This condition is enforced by requiring the failure region to be empty.

Definition 4 (Execution-Invariant Planning Tasks)**.** *A planning task is* execution-invariant *if and only if there exists a non-empty subset of its variables $\tilde{V}$ such that every plan $\pi \in \textsc{Exec}(\Pi)$ is execution-invariant for $\tilde{V}$.*

In other words, a planning task is execution-invariant if it admits a group of numeric variables whose variation does not affect the applicability of any executable plan. (To avoid trivial cases, we require that all variables in $\tilde{V}$ are relevant to the task, i.e., they appear in at least one precondition, effect, or goal condition.)

The case studies introduced earlier align with the definition of execution-invariance. In particular, the Non-Linear-Car domain is execution-invariant, as the actions *accelerate* and *decelerate* are always executable, regardless of the state. This domain was intentionally selected as a simple and illustrative example of our framework. In contrast, UTC and Baxter are domains that naturally exhibit execution-invariance. In UTC, this arises because initial occupancies do not appear in action preconditions, making their values irrelevant to action applicability. The same applies to the turn rates. For Baxter, the configuration of two connected links can be changed regardless of their initial positions.

More in general, we argue that execution-invariance naturally emerges in planning tasks with numeric structure, especially those modelled in PDDL+. This stems from PDDL+'s explicit distinction between agent and environment, which gives rise to scenarios where variables, such as link occupancies in UTC, affect only environmental dynamics without impacting action applicability.

6 Plan Robustness

The notions presented above, such as the perturbed state space, are only relevant for the concept of execution-invariant planning tasks. Regarding plan robustness, it is computed using either historical data or a model of uncertainty in the initial states of the planning tasks. Thus, the initial state distribution is either implicitly or explicitly modelled. In the following, plan robustness is cast in the execution invariance setting to limit its scope to numerical variables and to ensure that all expected values are well defined.

Intuitively, robustness is the probability that a given plan π achieves the goal G under the distribution over the initial state. The formal definition is as follows:

Definition 5 (Plan Robustness)**.** *Let Π be a planning task, let $\mathcal{I}$ be a random variable representing the possible initial states and $f_{\mathcal{I}}$ its distribution. The robustness of a plan π for Π with respect to $\mathcal{I}$ is defined as:*

$$R_{\mathcal{I}}(\pi) = \mathbb{E}_{\mathcal{I} \sim f_{\mathcal{I}}} \left[[\![\pi \in \textsc{Plans}(\Pi[\mathcal{I}])]\!] \right],$$

where $[\![P]\!]$ is the Iverson bracket which returns 1 if P is true and 0 otherwise.

The nature of the probability distribution $f_{\mathcal{I}}$ depends on the application. It may be unknown, as in the UTC case, or known when the underlying sources of uncertainty can be accurately modelled.

To compute an approximate value of $R_{\mathcal{I}}$, we calculate a confidence interval using Bayesian estimation. Since the random variable $[\![\pi \in \textsc{Plans}(\Pi[\mathcal{I}])]\!]$ takes binary outcomes, it follows a Bernoulli distribution with probability of success $R_{\mathcal{I}}$. As we estimate the robustness using a Bayesian method, we must specify a prior. The prior represents the beliefs on the robustness value before any observations. Since we have no reason to assume that any robustness value is more probable *a priori*, a uniform prior represents exactly this belief. This prior leads to the *a posteriori* robustness (the distribution of possible robustness values given the observations) being distributed as a Beta distribution. Using this approach, we can derive confidence bounds for $R_{\mathcal{I}}$ based on the observed number of successes S across N trials. The confidence interval $[\underline{R_{\mathcal{I}}}, \overline{R_{\mathcal{I}}}]$ for $R_{\mathcal{I}}$ is given by:

$$[\underline{R_{\mathcal{I}}}, \overline{R_{\mathcal{I}}}] = \begin{cases} [0, b_{1-\alpha, S+1, N-S+1}] & \text{if } S = 0 \\ [b_{\alpha, S+1, N-S+1}, 1] & \text{if } S = N \\ [b_{\frac{\alpha}{2}, S+1, N-S+1}, b_{1-\frac{\alpha}{2}, S+1, N-S+1}] & \text{otherwise,} \end{cases} \quad (1)$$

where $b_{q, S+1, N-S+1}$ is the q-quantile of the Beta distribution with shape parameters $S+1$ (number of success plus 1) and $N-S+1$ (number of failure plus one), and $1-\alpha$ is the desired confidence level.

In the following example, we apply this definition to the Non-Linear-Car domain.

Example 1 (Non-Linear-Car Robustness)**.** *In this domain, c_{drag} and i_a are invariant variables used to model domain parameters. Assume the initial state is $I = \{x : 0, v : 0, a : 0, c_{drag} : 0.1, i_a = 1\}$ and the goal is $G = \{\langle x \geq 99\rangle, \langle x \leq 101\rangle, \langle v \leq 0.1\rangle\}$. Consider two valid plans: $\pi_t^{\mathsf{s}} = \langle \pi^{\mathsf{s}}, 25\rangle$ (slow) and $\pi_t^{\mathsf{f}} = \langle \pi^{\mathsf{f}}, 17\rangle$ (fast). π_t^{s} has a longer makespan with fewer actions, whereas π_t^{f} reaches the goal faster via more frequent accelerations and decelerations.*

Suppose that c_{drag} and i_a are subject to uncertainty; we aim to study plan robustness with respect to these variables, i.e., $\tilde{V} = \{c_{drag}, i_a\}$. We assume, for this example, that their initial values follow uniform distributions centred on the nominal values, with bounds defined by an error factor $\epsilon \in \{0.01, 0.05, 0.1\}$. $\mathcal{U}^{\epsilon}$ denotes the uniform distribution $\mathcal{U}(-\epsilon, \epsilon)$ over the interval $[-\epsilon, \epsilon]$. Applying the approximation of plan robustness, using a sample population of size 1000 and a significance level of $\alpha = 0.05$, we obtain robustness estimates for different error rates shown in Table 1.

The results indicate that, for these uncertainty distributions, the slow plan is generally more robust than the fast plan, regardless of the magnitude of the error considered.

ϵ	π_t^{s}	π_t^{f}
0.01	$[0.99, 1.00]$	$[0.54, 0.60]$
0.05	$[0.18, 0.22]$	$[0.09, 0.12]$
0.1	$[0.08, 0.12]$	$[0.04, 0.07]$

Table 1: Plan robustness for different ϵ values.

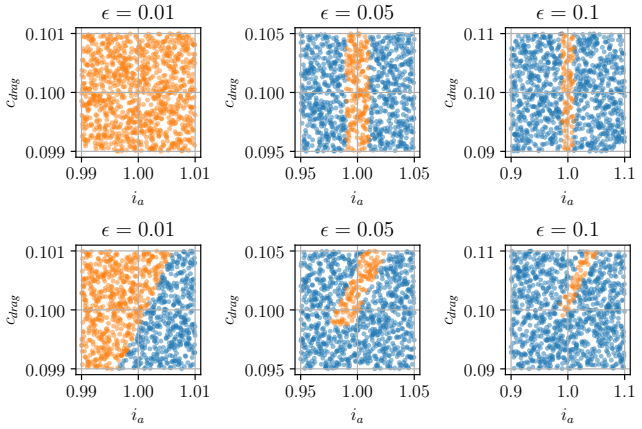


Figure 1: Outcome of $[\![\pi \in \textsc{Plans}(\Pi[\mathcal{I}])]\!]$ for plan π_t^{s} (upper) and π_t^{f} (lower), with $\mathcal{I} \sim I + \mathcal{U}^{\epsilon}$ and $\epsilon \in \{0.01, 0.05, 0.1\}$. An orange dot corresponds to an initial state realisation that ensures the plan's validity, while a blue dot corresponds to a realisation that guarantees executability but not validity.

Figure 1 provides a graphical illustration of the robustness calculation, where the outcome of $[\![\pi_t \in \textsc{Plans}(\Pi[\mathcal{I}])]\!]$ is displayed for each perturbation. This behaviour occurs because the slower plan induces lower velocities overall, due to fewer acceleration and deceleration actions. As a consequence, the system avoids reaching the velocity thresholds that would trigger drag, making the plan less sensitive to variations in c_{drag}. This effect is also visible in Figure 1 (upper), where plan π_t^{s} remains valid across a broader range of c_{drag} perturbations, as long as i_a stays within a bounded interval.

In many application domains, goals should be treated as indications of expected outcomes rather than strict requirements. Achieving the exact goal is often unnecessary; it suffices for plans to reach an acceptable region around the goal. This perspective is especially relevant in domains such as urban traffic control or underwater exploration, where uncertainty makes the given robustness definition, which relies on exactly achieving the original goal, too restrictive.

To address this, we extend the notion of plan robustness to measure the plan's ability to reach a neighbourhood of states that satisfy the goal, controlled by an acceptance tolerance. We define B-robustness as a measure of a plan's ability to achieve the goal within a tolerance factor B. When $B = 0$, the extended definition collapses to Definition 5, meaning we accept a plan only if it strictly achieves the goal.

To characterise B-robustness, we need a notion of distance between the state reached by executing a plan from a realisation of the initial state random variable and the set of states described by the goal.

Distance Let Π be a planning task with a goal description G, and let π be an executable plan for Π. Consider a realisation I of the initial state random variable $\mathcal{I}$, which induces a final state s when the plan π is applied starting from I. This final state is computed as $s = \gamma(I, \pi)$, where γ represents the state transition function that transforms I according to the plan π and the semantics of the given planning task.

The notion of distance is specific to each planning task; however, to be meaningful, it must satisfy the following properties:

- $\forall s,\ d_G(s) \geq 0$;
- $d_G(s) = 0 \iff s \models G$.

This means that the distance to the goal G from a state s, denoted as $d_G(s)$, should always be non-negative, and equal to zero if and only if s satisfies the goal.

Definition 6 (B-Robustness)**.** *Given a tolerance factor B, the B-robustness of a plan π is defined as follows:*

$$R_{\mathcal{I}}(\pi, B) = \mathbb{E}_{\mathcal{I} \sim f_{\mathcal{I}}}\left[[\![d_G(\gamma(\mathcal{I}, \pi)) \leq B]\!]\right].$$

To estimate plan B-robustness, we use Equation 1, which yields a confidence interval from sampled executions.

While B-robustness quantifies the probability that a plan reaches a relaxed version of the goal, it depends on the choice of the tolerance factor B. In practice, one may wish to determine the smallest B that ensures a desired robustness level. Let $R^{\star} \in [0, 1]$ denote this target level. We then introduce $B_{\min}$, the minimum tolerance required for the plan to succeed with probability at least $R^{\star}$. Formally:

$$B_{\min}(\pi, R^{\star}) = \inf_{\mathbb{R}_+}\{B \mid R(\pi, B) \geq R^{\star}\},$$

where $\inf_D U$ is the infimum of U with $U \subset D$, i.e., the greatest element in D that is less than or equal to all elements of U. Note that if $U = \emptyset$ and $D = \mathbb{R}_+$, then $\inf_D U = +\infty$.

The value $B_{\min}$ can be efficiently approximated using a dichotomic search, provided that the domain-specific distance function d_G satisfies the properties described earlier. If this holds, we can iteratively narrow the search interval for B. Starting with an initial interval $[B_{\text{low}}, B_{\text{high}}]$, we compute the midpoint B_{mid}, which splits the current interval in half. Next, we evaluate the robustness $R(\pi, B_{\text{mid}})$. Depending on whether $R(\pi, B_{\text{mid}})$ meets the target robustness $R^{\star}$, we update the interval by focusing on either the lower or the upper half. This process is repeated until the interval is sufficiently small, providing an approximation of $B_{\min}$. Note that in the first iteration, $B_{\text{low}} = 0$, while B_{high} may be set based on prior knowledge of relevant ranges in the domain.

Example 2 (Non-Linear-Car Distance)**.** *To illustrate the concept of distance in the Non-Linear-Car domain, we focus on the subgoals related to distance, specifically $G' = \{\langle x \geq 99\rangle, \langle x \leq 101\rangle\}$. Let s be the state resulting from the execution of a plan for the task under consideration from an uncertain initial state. The distance $d_G(s)$ is defined as the Euclidean distance measuring how much the state s violates the numeric goal conditions in G'. Particularly, each condition $g \in G'$ has the form $\langle x \bowtie K\rangle$, where $\bowtie \in \{\geq, \leq\}$. For each condition g such that $s \not\models g$, the term $(s'[x] - K)^2$ is*

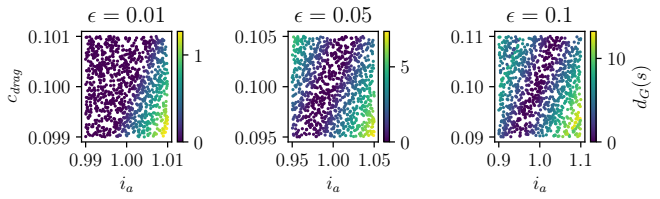


Figure 2: Outcome of $d_G(s)$ for Non-Linear-Car. Each point's gradient colour represents the car's deviation from the required distance after applying the plan, starting from a realisation of the initial state.

added to the sum. If $s \models g$, its contribution to the distance is zero. Finally, the square root of this sum is taken. Thus, the distance reflects the extent of goal violation and is zero when all conditions are satisfied. Figure 2 illustrates the distance $d_G(s)$ for π^{f}_t under different realisations of $\mathcal{I}$.

7 Experimental Evaluation

In this section, we present the application of the proposed framework to two case studies, i.e., UTC and Baxter. For each case study, we evaluate the following aspects across a set of plans: (i) their robustness for variations in the initial state concerning the numeric component, (ii) the minimum tolerance factor $B_{\min}$ required to achieve a desired robustness level $R^\star = 0.9$ considering two approaches: a conservative $\underline{R}$ (the confidence lower bound of R) and a most probable R (the number of successes S across N trials), and (iii) the trade-off between $B_{\min}$ and $R^\star \in [0.6, 0.95]$. For both domains, we set the significance level $\alpha = 0.05$, while the probability distribution about $\mathcal{I}$ and the number of trials N will depend on the case study. Furthermore, for points (ii) and (iii), we will define two domain-specific distances. An example of how to set up these computations is available at https://gitlab.com/EdmondDantes/robustnessijcai.

All plans are generated using ENHSP, an expressive numeric planner supporting PDDL+ [Scala *et al.*, 2020].

7.1 Urban Traffic Control

We consider the PDDL+ formulation *Fixed Repetition* (FIRE), [Kouaiti *et al.*, 2024], as it has proven to be the most effective among deployable UTC domains.

The uncertain variables for which we study the robustness of UTC plans are link occupancies and turn rates. Link occupancies reflect network congestion in the initial state and evolve during planning depending on the control strategy applied. These variables are bounded between 0 and the maximum capacity of each link. Turn rates can be safely assumed invariant over the predefined time window of 15 minutes [Bhatnagar *et al.*, 2022]. They are also bounded, with upper limits estimated from legal speed constraints and the structure of the road network. Therefore, given the set of link and movements LINKS and MOVS, we set $\tilde{V} = \{occ_l \mid l \in \text{LINKS}\} \cup \{tr_{l,l',s} \mid \langle l, l', s\rangle \in \text{MOVS}\}$, where occ_l is the occupancy of the link l and $tr_{l,l',s}$ is the turnrate associated with the vehicle movement from link l to l' during a stage s.

We analysed a statistical population of scenarios, from a major corridor in the UK county of Yorkshire, consisting of 90 instances derived from historical data corresponding to the morning peak hour on weekdays, at 8:00 AM. We considered weekdays between the 26th of January and the 15th of February, 2022, for which we have reliable sensor data. For the robustness calculations, we collected 10 plans generated across the first 10 days. Plans were generated using ENHSP with Greedy Best-First Search and either $h^{\max}$ or h^{add} heuristics [Scala *et al.*, 2016].

Figure 3a shows the robustness confidence intervals computed for the plans considered. It is interesting to note that, while the robustness confidence can vary quite significantly between plans for different days, the heuristic used for generating such plans does not usually have a major impact on the corresponding robustness.

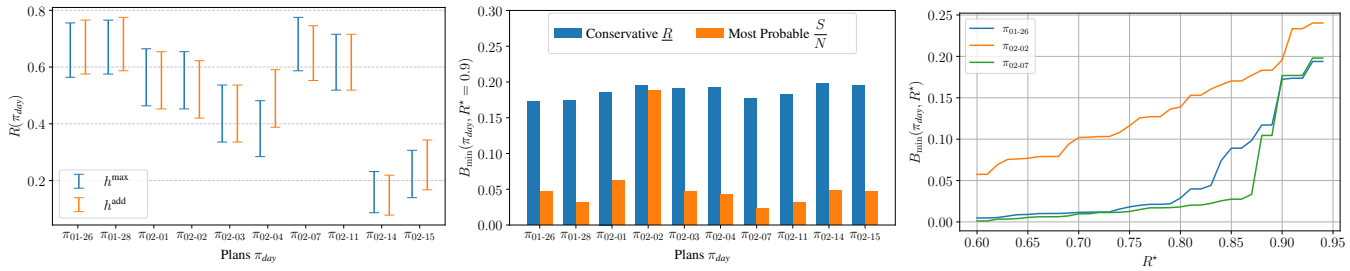
To quantify the minimum tolerance factor that can guarantee a robustness $R^\star = 0.9$, we need to define $d_G(s)$. In this UTC formulation, we denote by CORRIDOR the subset of network links that lie on the main corridor of the network. Each of them is associated with a numeric variable $count_l$, which tracks the number of vehicles that have traversed l during the temporal progression of the plan. We also denote l_{last} as the final link corresponding to the corridor's exit. The goal of a UTC planning is expressed as $G = \{\langle count_l \geq L_l\rangle \mid l \in \text{CORRIDOR}\} \cup \{\langle count_{l_{last}} \leq U\rangle\}$. In other words, we require that for each link in the main corridor, a minimum number of vehicles equal to L_l must flow through, while on the final link, this number must not exceed U. Specifically, the latter condition aims to prevent the network from discharging too many vehicles at the exit, thus avoiding congestion in the adjacent region. Given such a goal, we specialise d_G for UTC as follows:

$$d_G(s) = \sqrt{\sum_{\substack{g=\langle count_l \bowtie K\rangle \in G:\\ s \not\models g}} (s[count_l] - K)^2}.$$

This distance function penalises violations of goal conditions by measuring the squared deviation from the required threshold for each unsatisfied numeric condition. This distance function satisfies the previously discussed properties. We can therefore calculate $B_{\min}(\pi_{day}, 0.9)$ for each of the computed plans π_{day}. The results of this evaluation are shown in Figure 3b, where the distance has been normalised. These results indicate that the conservative approach requires a tolerance factor of approximately 20%, while the most probable case is usually around 5%. Finally, Figure 3c illustrates the trade-off between $B_{\min}$ and the required robustness $R^\star \in [0.6, 0.95]$ for three plans computed for distinct days.

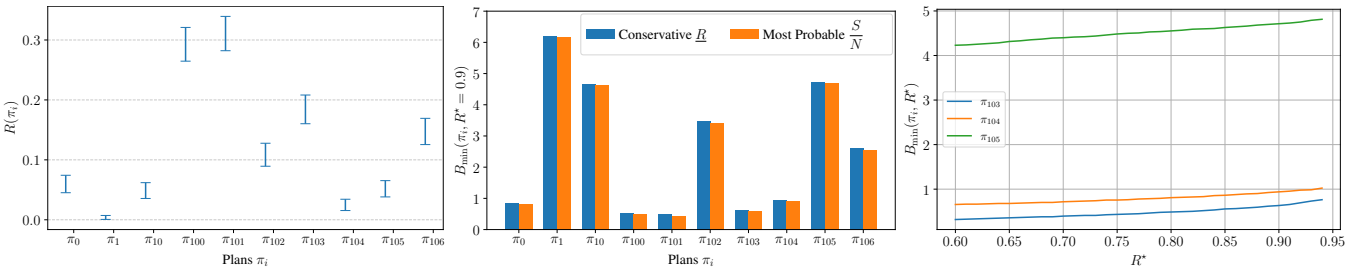
7.2 Baxter

For this experiment, we consider the formulation presented in the work by Bertolucci *et al.* (2020), and we only consider uncertainty over the initial object pose. The initial object pose is characterised by the orientation of each link, denoted by l, which is described by two angles: θ^{xy}_l for the horizontal plane and θ^{z}_l for the z-axis. All angular variables are bounded in the interval $[0, 2\pi]$, corresponding to a full rotation of $360°$. With five links, this results in a total of ten angles that completely define the object pose. Finally, we consider a goal characterised by five numeric conditions imposed on the orientation



(a) $[\underline{R}, \overline{R}]$ per plan day. (b) $B_{\min}$ per plan day. (c) $B_{\min}$ vs $R^\star$ in UTC.

Figure 3: UTC results: (a) robustness confidence intervals for plans from the two heuristics across weekdays; (b) $B_{\min}$ per day; and (c) trade-off between $B_{\min}$ and required robustness $R^\star \in [0.6, 0.95]$ for three plans from different days.



(a) $[\underline{R}, \overline{R}]$ per plan. (b) $B_{\min}$ per plan. (c) $B_{\min}$ vs $R^\star$ in Baxter.

Figure 4: Baxter results: (a) robustness confidence intervals across weekdays; (b) $B_{\min}$ per plan; and (c) trade-off between $B_{\min}$ and required robustness $R^\star \in [0.6, 0.95]$ for three plans computed for different variations of the initial object pose.

variables of links l_3, l_4, and l_5. These conditions define orientation thresholds for the links: l_3: $\theta^{xy}_{l_3} > 40.1$, $\theta^{z}_{l_3} > 107.2$; l_5: $\theta^{xy}_{l_5} > 92.3$, $\theta^{z}_{l_5} > 249.7$; l_4: $\theta^{xy}_{l_4} < 157.4$.

Since this planning task is not linked to a setup where historical data could be collected or object pose distribution could be inferred, we chose a distribution that tries to emulate a noisy pose estimation for the object. More precisely, we generate a population of 1000 instances by adding an error to each of the ten angles of the same initial object pose. For the angle θ in a given initial state, we assume a random variable θ_ϵ that follows $\theta_\epsilon \sim \mathcal{N}(0, \sigma^2)$ with $\sigma = \frac{\pi}{12}$. We use the following function $f(x) = x - 2\pi\lfloor(\frac{x}{2\pi})\rfloor$ to guarantee that $\theta + \theta_\epsilon$ lies within the interval $[0, 2\pi]$. Note that the chosen distribution does not represent any physical system.

In this domain, computing the distance d_G involves identifying angles in the final state that did not reach their target values and measuring their deviation from the goal configuration. We specialise the distance d_G for Baxter as:

$$d_G(s) = \sum_{\substack{g=\langle\theta \bowtie \beta\rangle \in G: \\ s \not\models g}} \arccos(cos(s[\theta] - \beta))$$

More intuitively, d_G is the cumulated angle the links need to turn in any direction to satisfy the goal.

Figure 4a presents the confidence intervals of R for 10 plans for the generated population of 1000 instances. The plans were computed for 10 random instances representing different variations of the studied initial object pose. Figure 4b shows the computation for each plan of $B_{\min}(\pi_i)$ for a target robustness $R = 0.9$. In this domain, differently from UTC, there is no significant difference between the conservative and the most probable case, also given the randomised distribution that has been used. However, in general, plans tend to be robust. Finally, Figure 4c presents the trade-off between $B_{\min}$ and the desired robustness for $R^\star \in [0.6, 0.95]$ for three plans computed from distinct variations of the initial object pose.

8 Conclusion

In summary, this work introduces execution-invariant planning tasks and a statistical framework for assessing plan robustness in the presence of initial state uncertainty, capturing both exact and approximate goal achievement. Our contributions are as follows. We formalised the notion of execution invariance and identified its relevance for planning tasks in which variability in specific variables does not compromise executability. We developed a statistical framework for quantifying robustness based on the distribution of the initial state. Third, we extended the robustness analysis through the concept of B-robustness, which captures approximate goal satisfaction under controlled tolerances. Fourth, we introduced a method for estimating the minimum tolerance $B_{\min}$ required to achieve a desired robustness target. We validated our framework in two realistic domains, urban traffic control and Baxter robotic manipulation, demonstrating its effectiveness in evaluating and comparing the robustness of different plans. Future work includes estimating numeric bounds from data, repairing executable plans that are not B-robust, and integrating robustness metrics into the plan generation process.

Acknowledgement

Francesco Percassi and Mauro Vallati were supported by a UKRI Future Leaders Fellowship [grant number MR/Z00005X/1].

References

[Alaboud and Coles, 2019] Fares K. Alaboud and Andrew Coles. Personalized Medication and Activity Planning in PDDL+. In *ICAPS*, pages 492–500. AAAI Press, 2019.

[Alon *et al.*, 2024] Lee-or Alon, Hana Weitman, Alexander Shleyfman, and Gal A. Kaminka. Planning to be healthy: Towards personalized medication planning. In *ECAI*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pages 4232–4239. IOS Press, 2024.

[Bertolucci *et al.*, 2020] Riccardo Bertolucci, Alessio Capitanelli, Marco Maratea, Fulvio Mastrogiovanni, and Mauro Vallati. Collaborative Robotic Manipulation: A Use Case of Articulated Objects in Three-dimensions with Gravity. In *ICTAI*, pages 1167–1174. IEEE, 2020.

[Bhatnagar *et al.*, 2022] Saumya Bhatnagar, Sumit Mund, Enrico Scala, Keith McCabe, Thomas Leo McCluskey, Mauro Vallati, et al. On-the-Fly Knowledge Acquisition for Automated Planning Applications: Challenges and Lessons Learnt. In *ICAART (2)*, pages 387–397, 2022.

[Cashmore *et al.*, 2020] Michael Cashmore, Daniele Magazzeni, and Parisa Zehtabi. Planning for Hybrid Systems via Satisfiability Modulo Theories. *J. Artif. Intell. Res.*, 67:235–283, 2020.

[Chrpa and Karpas, 2024] Lukás Chrpa and Erez Karpas. On Verifying and Generating Robust Plans for Planning Tasks with Exogenous Events. In *KR*, volume 21, pages 273–283, 2024.

[Chrpa *et al.*, 2017] Lukás Chrpa, José Pinto, Tiago Sa Marques, Manuel A. Ribeiro, and João Borges de Sousa. Mixed-initiative planning, replanning and execution: From concept to field testing using AUV fleets. In *IROS*, pages 6825–6830. IEEE, 2017.

[Domshlak and Hoffmann, 2007] Carmel Domshlak and Jörg Hoffmann. Probabilistic Planning via Heuristic Forward Search and Weighted Model Counting. *J. Artif. Intell. Res.*, 30:565–620, 2007.

[Fox and Long, 2006] Maria Fox and Derek Long. Modelling Mixed Discrete-Continuous Domains for Planning. *J. Artif. Intell. Res.*, 27:235–297, 2006.

[Fox *et al.*, 2006a] Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan Stability: Replanning versus Plan Repair. In *ICAPS*, pages 212–221. AAAI, 2006.

[Fox *et al.*, 2006b] Maria Fox, Richard Howey, and Derek Long. Exploration of the Robustness of Plans. In *AAAI*, pages 834–839. AAAI Press, 2006.

[Fox *et al.*, 2012] Maria Fox, Derek Long, and Daniele Magazzeni. Plan-based Policies for Efficient Multiple Battery Load Management. *J. Artif. Intell. Res.*, 44:335–382, 2012.

[Fritz and McIlraith, 2009] Christian Fritz and Sheila A. McIlraith. Computing Robust Plans in Continuous Domains. In *ICAPS*. AAAI, 2009.

[Gigante and Scala, 2023] Nicola Gigante and Enrico Scala. On the Compilability of Bounded Numeric Planning. In *IJCAI*, pages 5341–5349, 2023.

[Helmert, 2002] Malte Helmert. Decidability and Undecidability Results for Planning with Numerical State Variables. In *AIPS*, pages 44–53. AAAI, 2002.

[Khan *et al.*, 2023] Saad Khan, Simon Parkinson, Monika Roopak, Rachel Armitage, and Andrew Barlow. Automated Planning to Prioritise Digital Forensics Investigation Cases Containing Indecent Images of Children. In *ICAPS*, pages 500–508. AAAI Press, 2023.

[Kiam *et al.*, 2020] Jane Jean Kiam, Enrico Scala, Miquel Ramírez Jávega, and Axel Schulte. An AI-Based Planning Framework for HAPS in a Time-Varying Environment. In *ICAPS*, pages 412–420, 2020.

[Kouaiti *et al.*, 2024] Anas El Kouaiti, Francesco Percassi, Alessandro Saetti, Thomas Leo McCluskey, and Mauro Vallati. PDDL+ Models for Deployable yet Effective Traffic Signal Optimisation. In *ICAPS*, pages 168–177. AAAI Press, 2024.

[Kuroiwa *et al.*, 2023] Ryo Kuroiwa, Alexander Shleyfman, and J. Christopher Beck. Extracting and Exploiting Bounds of Numeric Variables for Optimal Linear Numeric Planning. In *ECAI*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 1332–1339. IOS Press, 2023.

[Percassi and Vallati, 2024] Francesco Percassi and Mauro Vallati. Leveraging AI Planning in a What-If Analysis Framework for Assessing Traffic Signal Strategies. In *ITSC*, pages 1330–1335. IEEE, 2024.

[Percassi *et al.*, 2023] Francesco Percassi, Enrico Scala, and Mauro Vallati. Fixing Plans for PDDL+ Problems: Theoretical and Practical Implications. In *ICAPS*, pages 324–333. AAAI Press, 2023.

[Percassi *et al.*, 2025] Francesco Percassi, Enrico Scala, and Mauro Vallati. On the Notion of Plan Quality for PDDL+. In *ICAPS*. AAAI press, 2025.

[Sarwar *et al.*, 2023] Mir Md Sajid Sarwar, Rajarshi Ray, and Ansuman Banerjee. A Contrastive Plan Explanation Framework for Hybrid System Models. *ACM Trans. Embed. Comput. Syst.*, 22(2):22:1–22:51, 2023.

[Scala *et al.*, 2016] Enrico Scala, Patrik Haslum, Sylvie Thiébaux, and Miquel Ramírez. Interval-Based Relaxation for General Numeric Planning. In *Proc. of ECAI*, pages 655–663, 2016.

[Scala *et al.*, 2020] Enrico Scala, Patrik Haslum, Sylvie Thiébaux, and Miquel Ramírez. Subgoaling Techniques for Satisficing and Optimal Numeric Planning. *J. Artif. Intell. Res.*, 68:691–752, 2020.

[Taig and Brafman, 2015] Ran Taig and Ronen I. Brafman. A Compilation Based Approach to Conformant Probabilistic Planning with Stochastic Actions. In *ICAPS*, pages 220–224. AAAI Press, 2015.

[Vallati *et al.*, 2016] Mauro Vallati, Daniele Magazzeni, Bart De Schutter, Lukás Chrpa, and Thomas Leo McCluskey. Efficient Macroscopic Urban Traffic Models for Reducing Congestion: A PDDL+ Planning Approach. In *AAAI*, pages 3188–3194. AAAI Press, 2016.