# Complete and Optimal Robust Planning Against Nature

**Lukáš Chrpa**[1], **Erez Karpas**[2,3]

[1]Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, Prague, Czechia
[2]Faculty of Data and Decision Sciences, Technion - Israel Institute of Technology, Haifa, Israel
[3]Department of Computer Science, The University of Texas at Austin, Austin, Texas, USA
chrpaluk@cvut.cz, karpase@technion.ac.il

**Abstract**

Planning and acting in environments in which "nature" can modify the environment by exogenous events presents a challenge for intelligent agents that have to make sure their plans can still be executed. In principle, events of nature might disrupt the agent's plan, or worse, cause damage to the agent.

In this paper, we build upon a recent concept of *robust plans* that are guaranteed to be successfully executed even if nature acts adversarially. The advantages of robust plans involve better understandability and not needing to sense the state of the environment during plan execution. We address the gap in prior work by presenting two methods for generating robust plans that are complete and guarantee optimality if an admissible heuristic is used with the A$^*$ algorithm. The methods are evaluated on five domains.

## Introduction

Automated planning provides a crucial mechanism for long-term decision making of autonomous agents (Ingrand and Ghallab 2017) such as planetary rovers (Ai-Chang et al. 2004) or autonomous underwater vehicles (Cashmore et al. 2018; Carreno et al. 2020; Chrpa et al. 2015). Environments in which these agents operate are rarely static, because exogenous events beyond the control of the agent might change their state. These exogenous events can be triggered by an actor which we refer to as *nature*, to signify that it does not necessarily have a goal of its own. A rational agent thus has to take nature's events into consideration while planning and acting. This class of problems is, in the literature, called *planning against nature* (Chrpa and Karpas 2024b, 2025).

**Example 1.** *In AUV operations, we deploy AUVs to sample given objects of interest in the sea. However, the area might contain corridors for ships that might pass by (we can assume that ships are controlled by nature). If a ship collides with an AUV, then the AUV can be destroyed. A rational AUV would hence avoid passing through ship corridors if a ship is nearby. For illustration, see Figure 1. In that case, the bottom left AUV samples the objects on the left-hand side of the ship corridor, while the top-right AUV samples the object in the bottom row (on the right-hand side of the ship corridor).*

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

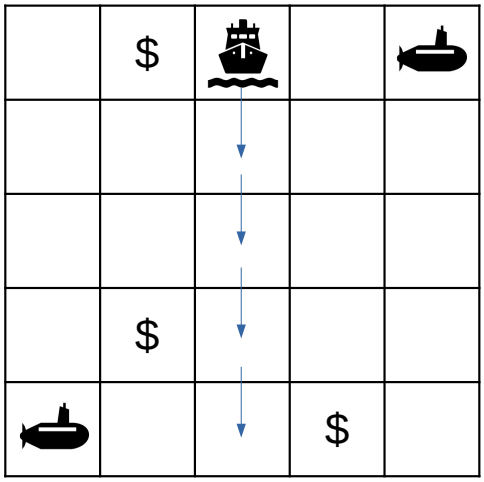


Figure 1: An example problem from the AUV domain. Objects of interest are marked as "$". The ship corridor is highlighted by arrows.

The concept of planning in the presence of exogenous events has been studied for some time (Dean and Wellman 1990; Musliner, Durfee, and Shin 1993; Iocchi, Nardi, and Rosati 2000). However, addressing such planning tasks usually requires reasoning with a large portion of the state space. Methods that reason with Markov Decision Process (MDP) models (Mausam and Kolobov 2012) are a good example. These methods generate policies, assigning the most promising action in each state. Policies, however, might not be easy to interpret and explain to human operators, who oversee the acting agent.

*Fully Observable Non-deterministic (FOND) planning* concerns fully observable environment and non-deterministic actions (Cimatti et al. 2003). While planning against nature can be tackled in a similar way to generating strong plans in FOND planning (Muise, McIlraith, and Beck 2012; Muise et al. 2016), this is not really a tractable approach when we consider that nature might apply an arbitrarily finitely long sequence of exogenous events after every action of the agent.

*Robust Plans* are sequences of actions that cannot be disrupted by nature, and the goal is achieved after the plan is

executed (Chrpa and Karpas 2024a). Robust plans are easier to explain (to human users), interpret, and execute by the agent. On top of that, during the execution of a robust plan, we do not need to observe the state of the environment, which can be very practical, for example, for AUV operations. Robust plans are, in a spirit, similar to *conformant plans* in partially-observable or unobservable environments (Cimatti and Roveri 2000; Bonet 2010). The fundamental difference is that we initially assume full state observability in planning against nature, and nature can gradually increase the uncertainty of the environment. Our concept of robust plans against nature differs from the recent work of Percassi et al. (2025) that studies robustness of plans with respect to uncertain values of (initial state) variables and the likelihood of successful execution of the plan.

In this paper, we address a major gap in the recent work concerning robust plan generation (Chrpa and Karpas 2024a, 2025). The existing approach leverages delete-relaxation to over-approximate the effects of nature's events during robust plan generation. Although such an approach is sound and can handle the problem mentioned in Example 1, it is incomplete, which has several implications. Firstly, we cannot guarantee that the "no solution" outcome actually means that no robust plan exists. Secondly, it might fail to find a solution even if it exists, and, thirdly, it does not guarantee solution optimality.

**Example 2.** *We extend Example 1 by considering that a ship has limited fuel and its single move consumes one unit of its fuel. Absence of the top-right AUV from the example of Figure 1 would mean that no robust plan exists under normal circumstances (as the bottom-left AUV cannot safely cross the ship corridor). However, if the ship's fuel level is at most* 3 *units, then the ship cannot reach the bottom row (as it runs out of fuel on the way). Hence, the bottom-left AUV can safely reach the bottom-row object.*

The delete-relaxed over-approximation overlooks the ship's fuel consumption and would incorrectly assume that the ship can reach the bottom row even with limited fuel, as described in the above example (Example 3 of Chrpa and Karpas (2024a) provides a similar argument). In this paper, we address this gap by introducing two complete methods for robust plan generation in planning against nature. Both methods are based on identifying *affected variables*, i.e., those that nature can modify. We show that we can compile the problem of deciding whether a variable is affected into a classical planning problem. Also, we show that if an action does not "enable" any event, the set of affected variables will not grow. The second method, on top of that, leverages the aforementioned delete relaxation for providing a *pessimistic* estimate on which variables are affected, and we propose a technique for identifying which events can certainly occur, which provides us with an *optimistic* estimate on which variables are affected. These estimates are leveraged to narrow the set of variables for which we need the (full) check. Lastly, we show how we can use delete-relaxed heuristics such as $h_{max}$ (Bonet and Geffner 2001) to generate optimal robust plans. We evaluate our (complete) methods empirically, comparing them with the existing delete-relaxed methods (Chrpa and Karpas 2024a, 2025) while also highlighting how our methods address the (theoretical) limitations of the delete-relaxed ones.

## Preliminaries

Planning against nature can be understood as a special case of multi-agent planning (Brafman and Domshlak 2008) in which an intelligent agent that plans towards its goal acts against an agent (or nature) that might act randomly without a specific purpose (or goal) (Chrpa and Karpas 2024b). To represent the environment, we use Finite Domain Representation (FDR) (Helmert 2009). This is captured in the following definition.

**Definition 1.** *A* **planning task against nature** *(or* **planning task***, for short) is a tuple $\mathcal{P} = (V, A, E, I, G)$, where $V$ is a set of finite-domain variables, $A$ is a set of actions of the agent, $E$ is a set of actions of nature (or,* **events***), $I$ is a complete variable assignment representing the* **initial state** *and $G$ is a partial variable assignment representing the* **goal***.*

Let $V$ be a set of ***variables*** where each variable $v \in V$ is associated with its domain $D(v)$. An ***assignment*** of a variable $v \in V$ is a pair $(v, val)$, where its value $val \in D(v)$. Hereinafter, an assignment of a variable is also denoted as a ***fact***. A (partial) ***variable assignment*** $p$ over $V$ is a set of assignments of individual variables from $V$, where $vars(p)$ is the set of all variables in $p$ and $p[v]$ represents the value of $v$ in $p$. A ***state*** is a complete variable assignment (over $V$). We say that a (partial) variable assignment $q$ ***holds*** in a (partial) variable assignment $p$, denoted as $p \models q$, iff $vars(q) \subseteq vars(p)$ and for each $v \in vars(q)$ it is the case that $q[v] = p[v]$.

An ***action*** is a pair $a = (pre(a), eff(a))$, where $pre(a)$ is a partial variable assignment representing $a$'s precondition and $eff(a)$ is a partial variable assignment representing $a$'s effects. We say that an action $a$ is ***applicable*** in state $s$ if and only if $s \models pre(a)$. The ***result*** of applying $a$ in $s$, denoted as $\gamma(s, a)$, is a state $s'$ such that for each variable $v \in V$, $s'[v] = eff(a)[v]$ if $v \in vars(eff(a))$ while $s'[v] = s[v]$ otherwise. If $a$ is not applicable in $s$, $\gamma(s, a)$ is undefined. The notion of action application can be extended to sequences of actions, i.e., $\gamma(s, \langle a_1, \ldots, a_n \rangle) = \gamma(\ldots \gamma(s, a_1) \ldots, a_n)$. ***Events*** are defined analogously as actions (but have different semantics).

The next definition of *event reachability* defines $\delta_E$, which maps a set of states to the set of states nature can reach from any of them by applying any sequence of events.

**Definition 2.** *Let $\delta_E : 2^S \rightarrow 2^S$ be defined as $\delta_E(S) = \{s' \mid s' = \gamma(s, \langle e_1 \ldots, e_k \rangle), s \in S,\ k \geq 0,\ e_1, \ldots, e_k \in E\}$. For convenience, we write $\delta_E(s)$ instead of $\delta_E(\{s\})$ for a single state $s$.*

Inspired by Chapman (1987) who studied relations between actions such as being an "achiever", i.e., one action sets a value of a variable for another action, we define the notion for both actions and events.

**Definition 3.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task. We say that an action/event $x \in A \cup E$ is an* **achiever** *for an action/event $y \in A \cup E$ if there exists a variable $v \in V$ such that $eff(x)[v] = pre(y)[v]$.*

The notion of *robust plans* has been defined by Chrpa, Gemrot, and Pilát (2020) and Chrpa and Karpas (2024a) as a sequence of actions that always achieves the goal even in the worst-case scenario of adversarial nature, which cannot invalidate the precondition of any action, nor the goal. In this sense, a robust plan is similar to a conformant plan (Cimatti and Roveri 2000) or to a strong (acyclic) plan in FOND planning (Cimatti et al. 2003).

At first, we define the notions of *robust action applicability* and the *result of robust action application*. The latter notion reflects the model in which an agent's action can be followed by a finite sequence of nature's events (Chrpa and Karpas 2024a; Karpas, Shleyfman, and Tennenholtz 2017).

**Definition 4.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task. We say that an action $a \in A$ is* **robustly applicable** *in a set of states $S$ (over $V$) if and only if $\forall s \in S : s \models pre(a)$. We say that a set of states $S'$ is the* **result of robust application** *of $a$ in $S$ if and only if $a$ is robustly applicable in $S$ and $S' = \delta_E(\{\gamma(s, a) \mid s \in S\})$. We denote the result of robust application of $a$ in $S$ as $\rho(S, a)$ (which is undefined if $a$ is not robustly applicable in $S$). The notion of robust action application can be extended to action sequences, i.e., $\rho(S, \langle a_1, \ldots, a_n\rangle) = \rho(\ldots \rho(S, a_1) \ldots, a_n)$ (note that $\rho(S, \langle\rangle) = S$).*

Now we can define the notion of a *robust plan* that coincides with the same notion from (Chrpa and Karpas 2024a) alongside the notion of *robust state trajectory*.

**Definition 5.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task. Let $\pi = \langle a_1, \ldots, a_n\rangle$ ($a_1, \ldots, a_n \in A$) be a sequence of actions. We say that $\pi$ is a* **robust plan** *for $\mathcal{P}$ if and only if $\rho(\delta_E(I), \pi)$ is defined and for all $s \in \rho(\delta_E(I), \pi) : s \models G$. We also define the* **robust state trajectory** *for $\mathcal{P}$ and a robust plan $\pi$ as sets of states $S^0, S^1, \ldots, S^n$ such as $S^0 = \delta_E(I)$ and $S^i = \rho(S^{i-1}, a_i)$ ($1 \leq i \leq n$).*

In analogy to classical planning, we can determine *optimality* of robust plans such that we minimise the total cost of actions in the robust plan. Note that, unlike in contingent planning, where plan optimality is much harder to define (Shmaryahu, Shani, and Hoffmann 2019), in robust planning, optimality makes much more sense – we only care about the costs of the agent's actions, not about the cost of nature's events.

**Definition 6.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task, and $cost : A \to \mathbb{R}_0^+$ be a function. We say that $\pi = \langle a_1, \ldots, a_n\rangle$, a robust plan for $\mathcal{P}$, is* **optimal** *if and only if for each $\pi' = \langle a'_1, \ldots, a'_k\rangle$ being a robust plan for $\mathcal{P}$, it holds that $\sum_{i=1}^{n} a_i \leq \sum_{j=1}^{k} a'_j$.*

## Affected Variables

Definitions 4 and 5 indicate that it is problematic if nature can modify the values of variables that the later actions or the goal might need. Following the terminology of Chrpa and Karpas (2024a), we will call these variables *affected*. For practical reasons, we provide a more general definition that depends only on sets of states and is not directly dependent on action sequences. We adopt the lemma stating that none of the variables of the precondition of the next action in the robust plan (or the goal at the end) can be affected (Chrpa and Karpas 2024a).

**Definition 7.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task and $S$ be a set of states (over $V$). We say that a variable $v \in V$ is* **affected** *in $S$ if and only if there exist states $s, s' \in S$ such that $s[v] \neq s'[v]$.*

The following lemma is paraphrased from Lemma 2 of (Chrpa and Karpas 2024a).

**Lemma 1.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task, $\pi = \langle a_1, \ldots, a_n\rangle$ be a sequence of actions such that $G \subseteq \gamma(I, \pi)$ and $V^0, \ldots, V^n$ be sets of affected variables such that $V^i = \{v \mid v$ is affected in $\rho(\delta_E(I), \langle a_1, \ldots, a_i\rangle)\}$. Then, $\pi$ is a robust plan if and only if $V^{i-1} \cap vars(pre(a_i)) = \emptyset$ ($1 \leq i \leq n$) and $V^n \cap vars(G) = \emptyset$.*

The above lemmas indicate that knowing which variables are affected (by events) in a given step provides sufficient information to determine whether an action is robustly applicable. In other words, we do not necessarily need to enumerate the set of states that might occur in a given step. Thus, instead of dealing with a possibly exponentially large number of states, we need only information about affected variables to determine the robust applicability of actions. For example, the variable representing the availability of the center cell (see Figure 1) is affected as the ship might enter the cell and make it unavailable.

In particular, a robust plan generation method nondeterministically selects robustly applicable actions until it reaches the goal (or finds that no such action can be selected). To determine the robust applicability of an action via Lemma 1, we can leverage the following idea of how to determine affected variables, given the planning task description and the sequence of previously selected robustly applicable actions.

The idea of how we can determine whether a given variable is affected in a given step of the process (i.e., after robustly applying a sequence of actions) is inspired by the construction of *invalidating tasks* presented by Chrpa and Karpas (2024a). In a nutshell, the idea behind invalidating tasks is to find whether nature can invalidate an agent's plan by its events by invalidating a precondition of some action in the plan. Here, the task is slightly different as the task is to determine whether nature can modify a variable by its events in a given step (after the agent executes a sequence of robustly applicable actions). In the following definition, we formally present how *$v$-affecting task* is constructed.

**Definition 8.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task, $\langle a_1, \ldots, a_k\rangle$ be a sequence of actions from $A$ (can be empty), and $v \in V$ be a variable.*

*Let "goal" and "applied" be variables (without loss of generality not being present in $V$), such that $D(goal) = \{\top, \bot\}$ and $D(applied) = \{0, \ldots, k\}$.*

*Let $a_i^*$ with $1 \leq j \leq k$ be actions such that*

$$pre(a_i^*) = pre(a_i) \cup \{(applied, i-1)\}$$
$$eff(a_i^*) = eff(a_i) \cup \{(applied, i)\}$$

*Then, we define sets of actions $A_{goal}$ as*

$$A_{goal} = \bigcup_{val \neq \gamma(I,\langle a_1,\ldots,a_k\rangle)[v]} a^{val}_{goal}$$
$$\textit{pre}(a^{val}_{goal}) = \{(\textit{applied}, k), (\textit{goal}, \bot), (v, val)\},$$
$$\textit{eff}(a^{val}_{goal}) = \{(\textit{goal}, \top)\}$$

*Now, we create a classical planning task $\mathcal{P}^v_k = (V^v_k, A^v_k, I^v_k, G^k_v)$, called $v$-**affecting task** for $\langle a_1, \ldots, a_k\rangle$ and $\mathcal{P}$, as follows.*

$$V^v_k = V \cup \{\textit{applied}, \textit{goal}\}$$
$$A^v_k = E \cup \{a^*_1, \ldots, a^*_k\} \cup A_{goal}$$
$$I^\pi = I \cup \{(\textit{goal}, \bot), (\textit{applied}, 0)\}$$
$$G^\pi = \{(\textit{goal}, \top)\}$$

The following theorem formally proves that a variable $v$ is affected in a given step if and only if a corresponding $v$-affecting task is solvable (i.e., there exists a plan for the task).

**Theorem 1** (Completeness of affected variable determination)**.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task, $v \in V$ be a variable and $\pi = \langle a_1, \ldots, a_k\rangle$ be a sequence of actions from $A$ such that $S = \rho(\delta_E(I)), \pi)$ is defined. It holds that $v \in V$ is affected in $S$ if and only if there exists a solution for $v$-affecting task $\mathcal{P}^v_k = (V^v_k, A^v_k, I^v_k, G^k_v)$ for $\pi$ and $\mathcal{P}$.*

*Proof.* From the premise that $\pi$ is (sequentially) robustly applicable in $\delta_E(I)$, we can derive that all actions can be applied in the given order regardless of any (valid) sequences of events that can be applied in between.

We have to show that $s \in S$ ($S = \rho(\delta_E(I), \pi)$) if and only if there exists a sequence of actions $\nu$ from $A^v_k$ such that $s' = \gamma(I^v_k, \nu)$ and $s' \models s \cup \{(\textit{applied}, k)\}$. The *applied* variable controls the application of the $a^*_i$ actions such that, in $\nu$, they have to be in the same order that their corresponding counterparts in $\pi$. In between the $a^*_i$ actions, $\nu$ might contain any valid sequence of actions from $E$ (that are the same as events in $\mathcal{P}$). That coincides with the definition of $\rho$ (see Definition 4) and hence $s \in \rho(\delta_E(I), \pi)$ if and only if there exists $\nu$ such that $s' = \gamma(I^v_k, \nu)$ and $s' \models s \cup \{(\textit{applied}, k)\}$.

The goal of $\mathcal{P}^v_k$ can be achieved if and only if there exists $\nu$ such that $\gamma(I^v_k, \nu)[v] \neq \gamma(I, \pi)[v]$. That can happen only if some event changes the value of $v$ after the last action in $\pi$ that has $v$ in its effects (which is implied from Definition 4).

Hence, $v$ is affected in $\rho(\delta_E(I), \pi)$ if and only if $\mathcal{P}^v_k$ is solvable. □

The following theorem proves that deciding whether a variable is affected in a given step of the robust plan generation process is PSPACE-complete.

**Theorem 2.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task, $v \in V$ be a variable, and $\pi = \langle a_1, \ldots, a_k\rangle$ be a sequence of actions from $A$ that $\rho(\delta_E(i), \pi)$ is defined. Deciding whether $v$ is affected in $\rho(\delta_E(I), \pi)$ is PSPACE-complete.*

Algorithm 1: A complete algorithm for robust plan generation

**Require:** Planning task $\mathcal{P} = (V, A, E, I, G)$
**Ensure:** $\pi$ being a robust plan for $\mathcal{P}$.

$s \leftarrow I$; $\pi \leftarrow \langle\rangle$
*aff* $\leftarrow$ Affected-Vars$(s, \pi, \emptyset, V)$
**while** $s \not\models G$ or $vars(G) \cap$ *aff* $\neq \emptyset$ **do**
  non-deterministically select $a \in A$ s.t. $s \models \textit{pre}(a)$ and $vars(\textit{pre}(a)) \cap$ *aff* $= \emptyset$
  **if** no $a$ is selected or $|\pi| \geq 2^{\prod_{v \in V} D(v)}$ **then**
    **return** "No solution"
  **end if**
  $\pi$.append($a$)
  $s \leftarrow \gamma(s, a)$
  *aff* $\leftarrow$ *aff* $\setminus (vars(\textit{eff}(a)) \setminus vars(\textit{pre}(a)))$
  **if** $a$ is an achiever for some $e \in E$ **then**
    *aff* $\leftarrow$ Affected-Vars$(s, \pi,$ *aff*$, V)$
  **else**
    *aff* $\leftarrow$ Affected-Vars$(s, \pi,$ *aff*$,$ *aff* $\cup (vars(\textit{eff}(a)) \setminus vars(\textit{pre}(a)))$
  **end if**
**end while**
**return** $\pi$

**function** AFFECTED-VARS$(s, \pi,$ *aff*$, V)$
  **for all** $v \in V \setminus$ *aff* **do**
    **if** $v$-affecting task for $\pi$ is solvable **then**
      *aff* $\leftarrow$ *aff* $\cup \{v\}$
    **end if**
  **end for**
  **return** *aff*
**end function**

*Proof.* Since deciding plan existence in classical planning is in PSPACE (Bylander 1994), we can immediately derive PSPACE membership of our decision problem from Theorem 1 and the fact that constructing the $v$-affecting task from $\pi$ and $\mathcal{P}$ can be done in polynomial time.

PSPACE-hardness of our decision problem can be proven by reducing the problem of plan existence in classical planning, which is PSPACE-hard (Bylander 1994), to our decision problem. Let $Q = (V', A', I', G')$ be a classical planning task. Without loss of generality, we introduce a variable $g$ such that $g \notin V'$ and $D(G) = \{\bot, \top\}$, and an action $a_g$ such that $\textit{pre}(a_g) = G' \cup \{(g, \bot)\}$ and $\textit{eff}(a_g) = \{(g, \top)\}$. We can observe that $Q$ has a solution if and only if $g$ is affected in $\delta_{A' \cup \{a_g\}}(I \cup \{(g, \bot)\})$, because $g$ can be flipped only by $a_g$ that becomes applicable only if $G'$ holds and thus there exists a sequence of actions from $A'$ to achieve $G'$, which is a solution of $Q$. Note that we refer to the special case of our decision problem in which $k = 0$, i.e., the agent did not take any action, and thus $\rho(\delta_{A \cup \{a_g\}}(I \cup \{(g, \bot)\}), \langle\rangle) = \delta_{A \cup \{a_g\}}(I \cup \{(g, \bot)\})$. □

## Robust Plan Generation

Lemma 4 of Chrpa and Karpas (2024a), which is stated below, claims that the set of affected variables monotonically

grows if the actions have all their effect variables "covered" by their preconditions. An affected variable might become unaffected only if an action that has that variable in its effects but not in its precondition is applied.

**Lemma 2.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task, $\pi = \langle a_1, \ldots, a_n \rangle$ be a robust plan for $\mathcal{P}$, and $S^0, S^1, \ldots, S^n$ be the robust state trajectory for $\mathcal{P}$ and $\pi$. Then, let $V^0, \ldots, V^n$ be the sets of affected variables for $S^0, S^1, \ldots, S^n$, respectively. The following claims hold for all ($1 \leq i \leq n$):*

*(a) If $vars(\mathit{eff}(a_i)) \subseteq vars(\mathit{pre}(a_i))$, then $V^{i-1} \subseteq V^i$*

*(b) $(V^{i-1} \setminus V^i) \subseteq (vars(\mathit{eff}(a_i)) \setminus vars(\mathit{pre}(a_i)))$*

We extend the above result by another claim concerning the fact that if an action is not an achiever for any event, then the set of affected variables will not grow after the action is applied.

**Lemma 3.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task, $\pi = \langle a_1, \ldots, a_n \rangle$ be a robust plan for $\mathcal{P}$, and $S^0, S^1, \ldots, S^n$ be the robust state trajectory for $\mathcal{P}$ and $\pi$. Then, let $V^0, \ldots, V^n$ be the sets of affected variables for $S^0, S^1, \ldots, S^n$, respectively. For all $1 \leq i \leq n$, it holds that if $a_i$ is not an achiever for any $e \in E$, then $V^{i-1} \setminus \big(vars(\mathit{eff}(a_i)) \setminus vars(\mathit{pre}(a_i))\big) \subseteq V^i \subseteq V^{i-1}$.*

*Proof.* Claim $V^{i-1} \setminus \big(vars(\mathit{eff}(a_i)) \setminus vars(\mathit{pre}(a_i))\big) \subseteq V^i$ follows immediately from Lemma 2.

Since there does not exist any event for which $a_i$ is an achiever, it can be immediately observed that for each event applicable in some state from $S^i$, it is the case that there exists a state in $S^{i-1}$ in which the event is also applicable. Hence, $V^i \subseteq V^{i-1}$. □

Algorithm 1 describes a method that generates robust plans. The algorithm extends the classical planning routine that keeps non-deterministically selecting (robustly) applicable actions until the goal is achieved or no action can be selected. The extension is based on Lemma 1 and concerns the identification of a set of affected variables in each step and making sure that the selected action does not contain any variable in its precondition that is affected. According to Lemma 2, we can determine what variables remain certainly affected after selecting an action (Line 10). On top of that, we leverage Lemma 3 such that we do not need to try to determine any new affected variables if the selected action is not an achiever for any event (Lines 11–15).

**Theorem 3** (Completeness of Algorithm 1)**.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task. Algorithm 1 is complete, i.e., it returns $\pi$, a robust plan for $\mathcal{P}$, if there exists a robust plan for $\mathcal{P}$, or "no solution", otherwise.*

*Proof.* It has been shown that if there exists a robust plan for $\mathcal{P}$, then there exists a robust plan for $\mathcal{P}$ that contains at most $2^{\prod_{v \in V} D(v)}$ actions (Chrpa and Karpas 2024a). Hence Algorithm 1 terminates.

Completeness of the Affected-Vars function, i.e., that it returns the set of affected variables for a sequence of robustly applicable actions $\pi$, is straightforwardly implied from Theorem 1 as each variables not being already affected in the given step (the *aff* set) is checked whether is is affected by trying to solve a corresponding $v$-affecting task.

According to Lemma 2, the set of affected variables is updated such that variables being present only in the effects of selected action (and not in its precondition) are removed from the set of affected variables (as in Line 10). If the selected action is not an achiever for some event, then according to Lemma 3, only variables that were removed from the set of affected variables (in Line 10) need to be rechecked. Hence, in each step, Algorithm 1 maintains the correct set of affected variables.

Then, according to Lemma 1, the action selection (Line 4) is made from all actions that are robustly applicable in the given step. Hence, Algorithm 1 is complete. □

## Estimating Affected Variables

We can improve Algorithm 1 by providing *optimistic* and *pessimistic* estimates on which variables are certainly affected and which can be affected, respectively. If such estimates can be provided in polynomial time, we can reduce the burden by solving only $v$-affecting tasks for variables that are pessimistically assumed to be affected but not optimistically assumed to be affected.

The *pessimistic* estimate can be provided by leveraging the delete-relaxed planning graph (Bonet and Geffner 2001) by which we can identify what facts the nature can possibly reach by its events (Chrpa and Karpas 2024a, 2025). Delete relaxation assumes that no fact (variable assignment) can be invalidated by an event (in this case), and hence, variables can have multiple values in the process. If a variable has only one fact reachable by events, then we can claim that that variable is certainly not affected.

The *optimistic* estimate, which we introduce in this paper, identifies (sub)sets of events that certainly occur in the given step in polynomial (linear) time (in analogy to the pessimistic variant). The idea, in a nutshell, that an event $e$ can certainly occur is based on (i) whether non-affected variables in a given state satisfy (part) of the $e$'s precondition, or (ii) whether there exists another event which can certainly occur before whose effects satisfy (the remaining part) of the $e$'s precondition. The rationale behind (i) is the same as for determining robust applicability of actions. Regarding (ii), we know that if a (single) event occurred just before $e$, it can set the required value of (some) variables in $e$'s precondition. Events that can certainly occur in a given step determine a subset of variables that are certainly affected (in that step). Note that this is an *under-approximation* of what events can occur since, for instance, multiple events that do not interfere with each other can be achievers for $e$.

The following lemma states that the values of variables from a set of states appearing earlier in the robust plan trajectory are carried forward unless they are modified by actions. In consequence, the effects of some event occurring earlier stay valid unless some action changes the value of some of the variables in the event's effects. The claim is important for the above condition (ii).

**Lemma 4.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task, $\pi = \langle a_1, \ldots, a_n \rangle$ be a robust plan for $\mathcal{P}$, and $S^0, \ldots, S^n$ be its*

Algorithm 2: An improved complete algorithm for robust plan generation

**Require:** Planning task $\mathcal{P} = (V, A, E, I, G)$
**Ensure:** $\pi$ being a robust plan for $\mathcal{P}$.
1: $s \leftarrow I$; $\pi \leftarrow \langle\rangle$
2: $f \leftarrow$Expand-Relaxed($s$)
3: $E_{opt} \leftarrow$Trigger-events($\emptyset$, $s$,noop, $\emptyset$)
4: $(\mathit{aff}, E_{cur}, f) \leftarrow$ Affected-Vars($s, \pi, \emptyset, E_{opt}, f$)
5: **while** $s \not\models G$ or $vars(G) \cap \mathit{aff} \neq \emptyset$ **do**
6: non-deterministically select $a \in A$ s.t. $s \models \mathit{pre}(a)$ and $vars(\mathit{pre}(a)) \cap \mathit{aff} = \emptyset$
7: **if** no $a$ is selected or $|\pi| \geq 2^{\prod_{v \in V} D(v)}$ **then**
8: **return** "no solution"
9: **end if**
10: $\pi$.append($a$)
11: $s \leftarrow \gamma(s, a)$
12: $f \leftarrow f \setminus \{(v, val) \mid v \in vars(\mathit{eff}(a))\} \cup \mathit{eff}(a)$
13: $\mathit{aff} \leftarrow \mathit{aff} \setminus (vars(\mathit{eff}(a)) \setminus vars(\mathit{pre}(a)))$
14: **if** $a$ is an achiever for some $e \in E$ or $vars(\mathit{eff}(a)) \not\subseteq vars(\mathit{pre}(a))$ **then**
15: $f \leftarrow$Expand-Relaxed($f$)
16: $E_{opt} \leftarrow$Trigger-events($E_{cur}, s, a, \mathit{aff}$)
17: $(\mathit{aff}, E_{cur}, f) \leftarrow$ Affected-Vars($s, \pi, \mathit{aff}, E_{opt}, f$)
18: **end if**
19: **end while**
20: **return** $\pi$

21: **function** AFFECTED-VARS($s, \pi, \mathit{aff}, E_{opt}, f$)
22: $\mathit{aff} \leftarrow \mathit{aff} \cup \{v \mid e \in E_{opt}, \mathit{eff}(e)[v] \neq s[v]\}$
23: $\mathit{aff}_{pes} \leftarrow \{v \mid (v, val) \in f, (v, val') \in f, val \neq val'\}$
24: **for all** $v \in (\mathit{aff}_{pes} \setminus \mathit{aff})$ **do**
25: **if** $v$-affecting task for $\pi$ yields $\nu$ as a solution **then**
26: $\mathit{aff} \leftarrow \mathit{aff} \cup \{v\}$
27: $E_{opt} \leftarrow \{e \mid e \in E \cap \nu\}$
28: **else**
29: $f \leftarrow f \setminus \{(v, val) \mid s[v] \neq val\}$
30: **end if**
31: **end for**
32: **return** $\mathit{aff}, E_{opt}, f$
33: **end function**

34: **function** EXPAND-RELAXED($f$)
35: **repeat**
36: $E_{last} \leftarrow E_{sel}$
37: $E_{sel} \leftarrow \{e \mid e \in E, f \models \mathit{pre}(e)\}$
38: $f \leftarrow f \cup \bigcup_{e \in E_{sel}} \mathit{eff}(e)$
39: **until** $E_{sel} = E_{last}$
40: **return** $f$
41: **end function**

42: **function** TRIGGER-EVENTS($(E_{prec}, s, a, \mathit{aff})$)
43: $E_{cur} \leftarrow E_{prec} \setminus \{e \mid e \in E_{prec}, vars(\mathit{eff}(e)) \cap vars(\mathit{eff}(a)) \neq \emptyset\}$
44: **for all** $e \in E \setminus (E_{cur})$ **do**
45: **if** $\exists e' \in E_{cur} \forall v \in vars(\mathit{pre}(e)) : (\mathit{pre}(e)[v] = s[v] \implies v \notin \mathit{aff}) \vee \mathit{pre}(e)[v] = \mathit{eff}(e')[v]$ **then**
46: $E_{cur} \leftarrow E_{cur} \cup \{e\}$
47: **end if**
48: **end for**
49: **return** $E_{cur}$
50: **end function**

*robust state trajectory. For all $0 \leq j < i \leq n$, and each $s_j \in S^j$, there exists $s_i \in S^i$ such that $s_j[v] = s_i[v]$ for all $v \in V \setminus (\bigcup_{l=j+1}^{i} vars(\mathit{eff}(a_l)))$*

*Proof.* The claim is implied by the fact that $S \subseteq \delta_E(S)$, which can be immediately derived from Definition 2 by considering an empty event sequence, and the fact that values of variables from $V \setminus (\bigcup_{l=j+1}^{i} vars(\mathit{eff}(a_l)))$ cannot be changed by actions $a_{j+1}, \ldots, a_i$. □

**Corollary 1.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task, $e \in E$ be an event, $\pi = \langle a_1, \ldots, a_n \rangle$ be a robust plan for $\mathcal{P}$, and $S^0, \ldots, S^n$ be its robust state trajectory. For all $0 \leq j < i \leq n$ such that $vars(\mathit{eff}(e)) \cap \bigcup_{l=j+1}^{i} vars(\mathit{eff}(a_l)) = \emptyset$, it is the case that if $\exists s_j \in S^j : s_j \models \mathit{eff}(e)$, then $\exists s_i \in S^i : s_i \models \mathit{eff}(e)$*

Algorithm 2 extends Algorithm 1 by *optimistic* and *pessimistic* estimates on what variables are affected in a given step. The pessimistic estimate is inspired by the method for robust plan generation introduced by Chrpa and Karpas (2024a), where we leverage delete-relaxation to determine what facts are possibly reachable by events (see the Expand-Relaxed procedure). If two (or more) different values of a variable can be reached (by events), we pessimistically assume that the variable is affected (see Line 23 where the set of variables pessimistically assumed to be affected is computed). The optimistic estimate is handled by the Trigger-Events procedure that identifies a subset of events that can certainly occur in a given step by leveraging Corollary 1 (see Line 45). For variables that are only pessimistically identified as affected (but not optimistically), we still need to try to solve $v$-affecting tasks to decide whether variables are actually affected or not.

**Example 3.** *Let us recall Example 2 (see Figure 1). Let us assume that the ship's fuel level is 2. The pessimistic estimate identifies the variables representing the ship's position and the availability of all cells in the center column as affected, since delete-relaxation does not take fuel consumption into account. The optimistic estimate identifies (besides the ship position) only the availability variables of the top three center-column cells as affected. The ship's move to the second row can certainly occur as the corresponding event is applicable in the current state. The ship's move to the third row can certainly occur because the center cell is available and the previous move event achieves the ship's position and ship's fuel level 1 for the current event. The ship might not move further down since the last event consumed the fuel, and hence, there is no single event that achieves both the ship's position and a non-empty fuel level. The availability variables of the bottom two center-column cells need to be checked by solving the corresponding $v$-affecting tasks.*

**Theorem 4** (Completeness of Algorithm 2)**.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task. Algorithm 2 is complete, i.e., it returns $\pi$, a robust plan for $\mathcal{P}$, if there exists a robust plan for $\mathcal{P}$, or "no solution", otherwise.*

*Proof.* Completeness of Algorithm 2 is implied from (i) Algorithm 1 it extends, (ii) soundness of the delete-relaxation

method for the pessimistic estimate of sets of affected variables, i.e., variables that are not deemed affected are indeed not affected (for the proof, see (Chrpa and Karpas 2024a)), and (iii) Corollary 1 that ensures the events selected in the Trigger-Events procedure can certainly occur and thus we can correctly identify a subset of certainly affected variables (Line 22). □

## Heuristic Search and Optimality

Heuristic search is prominent in classical planning (see, e.g. (Bonet and Geffner 2001; Hoffmann and Nebel 2001; Richter and Westphal 2010)), and recent work demonstrated that delete-relaxation-based heuristics can be adopted for robust plan generation (Chrpa and Karpas 2025).

We start with the definition of the perfect heuristic estimate for robust plan generation.

**Definition 9.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task, and $\mathcal{S}$ be the set of all states over $V$. We define a function $h^*_{\mathcal{P}} : \mathcal{S} \rightarrow \mathbb{R}_0$ representing, for each $S \in \mathcal{S}$, the minimum cost of actions robustly applicable in $S$ such that $G$ is satisfied in all the in the resulting states (of the consecutive robust application of the actions). We call $h^*_{\mathcal{P}}$ a* **perfect heuristic** *estimate for $\mathcal{P}$.*

Inspired by Chrpa and Karpas (2025), we leverage delete-relaxation-based heuristics widely used in classical planning (Bonet and Geffner 2001; Hoffmann and Nebel 2001) to provide an (admissible) heuristic estimate for the (remaining) number of actions to form a robust plan. In classical planning, $h^*_{\mathcal{P}'}(s)$ (resp. $h^+_{\mathcal{P}'}(s)$) denotes the minimum cost of actions (resp. delete-relaxed actions) to achieve the goal from $s$ in $\mathcal{P}'$. For a planning task $\mathcal{P} = (V, A, E, I, G)$, we construct a classical planning task $\mathcal{P}' = (V', A, I, G)$ such that we extend the domain of each variable from $V$ by the "undef" value, i.e., $V' = \{v' \mid D(v') = D(v) \cup \{\mathit{undef}\}, v \in V\}$. Then for a state $s$ and a set of affected variables *aff*, we determine the delete-relaxed heuristic estimate as:

$$h^+_{\mathcal{P}}(s, \mathit{aff}) = h^+_{\mathcal{P}'}(s')$$

where $s'[v] = s[v]$ for all $v \in V \setminus \mathit{aff}$ and $s'[v] = \mathit{undef}$, otherwise.

**Theorem 5.** *Let $\mathcal{P} = (V, A, E, I, G)$ be a planning task, $s$ be a state over $V$, and aff be a set of affected variables. For each set of states $S$ such that aff is the set of its affected states and for all $s' \in S$ and all $v \in V \setminus \mathit{aff}$ it holds that $s'[v] = s[v]$, it is the case that $h^+_{\mathcal{P}}(s, \mathit{aff}) \leq h^*_{\mathcal{P}}(S)$.*

*Proof.* We can observe that a robust plan for $\mathcal{P}$ is also a plan for the underlying classical planning task (Chrpa and Karpas 2024a). Restricting the classical planning task as described above (to $\mathcal{P}'$) does not invalidate the above claim, since the restriction forbids applying actions whose precondition contains some affected variable that prohibits their robust applicability (see Lemma 1). Variables with the *undef* value can be modified only by actions that have that variable only in their effects, which coincides with Lemma 2's claim (b). Hence, we can derive that $h^+_{\mathcal{P}'}(s') \leq h^*_{\mathcal{P}}(S)$ ($s'$ determined from $s$ as above). Since $h^+_{\mathcal{P}'}(s') \leq h^*_{\mathcal{P}'}(s')$ (Bonet and Geffner 2001), we get $h^+_{\mathcal{P}}(s, \mathit{aff}) \leq h^*_{\mathcal{P}}(S)$. □

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| AUV | | | | | | |
| REL (B) | 0.07 | 0.04 | 6.92 | 6.65 | 447.71 | 437.58 |
| REL (H) | 0.04 | 0.04 | 5.32 | 5.23 | 628.14 | 649.74 |
| COMP (B) | 0.49 | 0.46 | 3.56 | 3.52 | 135.77 | 137.86 |
| COMP (H) | 0.44 | 0.45 | 3.47 | 3.37 | 130.09 | 133.78 |
| ECOMP (B) | 0.06 | 0.06 | 3.27 | 3.26 | 154.24 | 151.84 |
| ECOMP (H) | 0.06 | 0.06 | 3.07 | 3.01 | 156.59 | 154.66 |
| AUV-nosol | | | | | | |
| REL (B) | 0.03 | 0.03 | 0.05 | 0.05 | 0.08 | 0.08 |
| REL (H) | 0.03 | 0.03 | 0.04 | 0.04 | 0.05 | 0.05 |
| COMP (B) | 0.38 | 0.37 | 0.87 | 0.86 | 1.25 | 1.24 |
| COMP (H) | 0.39 | 0.40 | 0.86 | 0.86 | 1.24 | 1.23 |
| ECOMP (B) | 0.06 | 0.06 | 0.13 | 0.13 | 0.30 | 0.28 |
| ECOMP (H) | 0.06 | 0.06 | 0.12 | 0.12 | 0.26 | 0.27 |
| AUV-deep | | | | | | |
| REL (B) | 0.04 | 0.08 | 0.42 | 0.47 | 7.02 | 6.21 |
| REL (H) | 0.04 | 0.08 | 0.26 | 0.42 | 6.64 | 5.92 |
| COMP (B) | 2.38 | 11.25 | 51.39 | 52.63 | 336.00 | 439.48 |
| COMP (H) | 2.36 | 13.84 | 15.95 | 52.49 | 366.48 | 424.93 |
| ECOMP (B) | 0.06 | 0.11 | 0.37 | 0.37 | 2.81 | 2.60 |
| ECOMP (H) | 0.07 | 0.13 | 0.39 | 0.46 | 3.15 | 2.86 |
| ServiceRobot | | | | | | |
| REL (B) | 0.03 | 0.03 | 0.07 | 0.06 | 0.90 | 19.82 |
| REL (H) | 0.03 | 0.03 | 0.08 | 0.05 | 0.51 | 9.08 |
| COMP (B) | 0.15 | 8.00 | 40.13 | 38.46 | 214.69 | 1092.06 |
| COMP (H) | 0.16 | 6.37 | 35.74 | 20.11 | 152.25 | 805.17 |
| ECOMP (B) | 0.03 | 0.04 | 0.05 | 0.04 | 0.40 | 10.15 |
| ECOMP (H) | 0.03 | 0.03 | 0.06 | 0.04 | 0.32 | 5.21 |
| AUV-fuel | | | | | | |
| REL (B) | *0.03* | 0.03 | *0.05* | *0.05* | 0.09 | *0.08* |
| REL (H) | *0.03* | 0.03 | *0.05* | *0.05* | 0.07 | *0.07* |
| COMP (B) | 2.63 | 0.42 | 39.89 | 23.45 | 17.05 | 557.04 |
| COMP (H) | 2.39 | 0.41 | 31.98 | 17.00 | 1.31 | 454.95 |
| ECOMP (B) | 0.37 | 0.06 | 4.00 | 1.51 | 2.77 | 119.23 |
| ECOMP (H) | 0.34 | 0.06 | 3.30 | 1.12 | 0.27 | 96.90 |
| dog-leash | | | | | | |
| REL (B) | *0.04* | 0.04 | *0.06* | *0.17* | *0.18* | *0.26* |
| REL (H) | *0.03* | 0.04 | *0.06* | *0.18* | *0.17* | *0.26* |
| COMP (B) | 0.39 | 0.38 | 4.24 | 34.16 | 79.69 | 183.56 |
| COMP (H) | 0.24 | 0.37 | 2.91 | 15.29 | 66.66 | 177.25 |
| ECOMP (B) | 0.16 | 0.27 | 3.92 | 33.84 | 79.06 | 193.10 |
| ECOMP (H) | 0.11 | 0.28 | 2.66 | 15.01 | 66.09 | 186.68 |

Table 1: Results for the robust plan generation. (B) denotes the BFS variant and (H) denotes the $h_{max}$ variant. Runtimes are in seconds. Problem instances correctly identified as unsolvable are highlighted in teal, problem instances incorrectly identified as unsolvable are highlighted in *orange*.

The above theorem provides guarantees that using *admissible* delete-relaxation-based heuristics such as $h_{max}$ (Bonet and Geffner 2001), we can guarantee optimality of the generated robust plan (by either Algorithm 1 or 2) if we use the $A^*$ search algorithm.

## Experimental Evaluation

The experiments evaluate our proposed complete methods for robust plan generation – COMP (Algorithm 1) and ECOMP (Algorithm 2) – and compare them against REL, the incomplete delete-relaxed method for robust plan gener-

ation (Chrpa and Karpas 2024a). We consider two variants of REL, using Breadth-First Search (BFS) with early dead-end detection (when a goal variable becomes affected and no action can make it unaffected again) (Chrpa and Karpas 2024a) and using A* with the $h_{max}$ heuristic (Chrpa and Karpas 2025). Analogously, COMP and ECOMP come with the BFS variant (with an early dead-end detection) and the A* with $h_{max}$ variant.

To solve $v$-affecting tasks, we used LAMA (Richter and Westphal 2010). The time limit for each problem was 1800 seconds, and the memory limit was 4GB. Experiments were run on an AMD Ryzen 9 7950X 16-Core cluster.[1]

For the evaluation, we use five benchmark domains, where three of those – AUV, Deep AUV, and Service Robot – are taken from (Chrpa and Karpas 2024a). In the AUV domain, we have two AUVs that have to collect resources in a grid-like environment, while there are ships that can move within their corridors in a given direction (each ship has a single column in the grid). If a ship collides with an AUV, the AUV is destroyed. We also specified a set of unsolvable problem instances (where a robust plan does not exist) in the AUV domain (denoted as "AUV nosol"). The Deep AUV domain allows AUVs to go into depth, so they can avoid a potential collision with ships. The problem instances consider a single AUV. The ServiceRobot domain, in a nutshell, describes a task in which two-handed robots have to move objects between different rooms. However, some objects are fragile and can be damaged if the robot carries the fragile object and some other object (in its other hand), or the robot encounters another robot in a (narrow) corridor at the same time. The additional domain "AUV-fuel" is an extension of the AUV domain in which ships have a limited amount of fuel, which limits the number of their moves. We also introduce a new domain "dogs on a leash", in which an agent moves on a grid, where several dogs, controlled by nature, are tied on a leash. Each dog is on a leash of a fixed length, and a dog can pull the leash or go back to its previous position (adding slack back to the leash). If the dog reaches the same position as the agent, it can bite. The agent must reach its goal location without being bitten.

Table 1 shows the results of our experiments. In the original benchmark set (see (Chrpa and Karpas 2024a)), we can see that REL solves all problems and generates optimal robust plans (despite being incomplete), yet, as expected, REL failed to solve all AUV-fuel and dog-leash instances. Interestingly, REL is outperformed by ECOMP in almost all cases, particularly on more challenging problems. We observed two reasons for that. Firstly, leveraging Lemma 3 prunes a large number of unnecessary checks for affected variables (that REL blindly does) and, secondly, although solving or proving unsolvability of $v$-affecting tasks is trivial in all benchmark problems, limiting the number of $v$-affecting tasks that need to be solved can considerably improve the performance as can be seen while comparing the runtimes for ECOMP and COMP. Performance of COMP is slightly better than ECOMP for problems 5 and 6 in the AUV domain. The reason is that the number of $v$-affecting tasks that need to be solved is low for COMP and the difference with ECOMP is not that significant (e.g., COMP (H) has to solve $48$ $v$-affecting tasks while ECOMP (H) $8$ for problem 6). In ServiceRobot, COMP (H) needed to solve more than $30000$ $v$-affecting tasks for problem 6, while ECOMP did not have to solve any (as the optimistic and pessimistic assumptions decided all the cases). Hence, COMP was considerably slower in ServiceRobot than ECOMP.

For the unsolvable problem instances (AUV-nosol, instances 2 and 5 of AUV-fuel, and instance 2 of dog-leash), we were able to identify unsolvability in all cases, often in under a second. Since REL is sound, it correctly identified unsolvability, however, only COMP and ECOMP can guarantee that. The AUV-fuel and dog-leash domains manifest the main weakness of the incomplete REL algorithm as it incorrectly identified all problems as unsolvable, while only problems 2 and 5 of AUV-fuel and problem 2 of dog-leash were actually unsolvable. COMP and ECOMP can correctly identify solvable and unsolvable problems and return an optimal solution (if the problem is solvable), albeit at the cost of considerably higher runtime.

In summary, the results demonstrate that for simpler problems that REL can handle, our methods (especially ECOMP) are competitive and even better. On top of that, our methods can deal with problems in which REL fails, such as AUV-fuel and dog-leash, and can correctly identify problem unsolvability (that a robust plan does not exist for a given task).

## Conclusion

Concerning *planning against nature*, this paper investigated robust plans, sequences of actions that are guaranteed to succeed regardless of the acts of nature. In particular, we addressed a knowledge gap concerning the completeness and optimality of generating robust plans. In the existing works, the effects of nature's events were over-approximated by delete-relaxation in robust plan generation that, albeit being sound, does not guarantee both completeness and optimality as the approach might prune valid solutions (Chrpa and Karpas 2024a, 2025). We introduced two complete algorithms for generating robust plans. In a nutshell, they are based on identifying what variables can be *affected* by events, i.e., nature can modify their values. We have also shown that we can adopt admissible delete-relaxation-based heuristics for classical planning for optimal robust plan generation. We have experimentally evaluated our complete methods with the existing delete-relaxed ones, showing that our methods can be competitive on simpler problems (that the delete-relaxed methods can handle) while also dealing with more complex problems and correctly identifying unsolvability of problems where a robust plan does not exist.

In the future, we plan to investigate how we can leverage abstractions for restricting the size of $v$-affecting tasks and for improving optimistic estimates of affected variables. Also, we plan to investigate whether and under what conditions we might identify inadmissible subsequences of actions and how we can leverage such information in the robust plan generation process.

[1] Source code and benchmark data can be found at: https://github.com/lchrpa/Robust-Plans-ICAPS-26

## Acknowledgements

This research is supported by the Czech Science Foundation (project no. 24-13337L) and by the European Union under the project ROBOPROX (reg. no. CZ.02.01.01/00/22_008/0004590), and by the Israeli Ministry of Science and Technology. Partial support for this research came from the Donald D. Harrington Fellows Program at the University of Texas at Austin.

## References

Ai-Chang, M.; Bresina, J. L.; Charest, L.; Chase, A.; Hsu, J. C.; Jónsson, A. K.; Kanefsky, B.; Morris, P. H.; Rajan, K.; Yglesias, J.; Chafin, B. G.; Dias, W. C.; and Maldague, P. F. 2004. MAPGEN: Mixed-Initiative Planning and Scheduling for the Mars Exploration Rover Mission. *IEEE Intelligent Systems*, 19(1): 8–12.

Bonet, B. 2010. Conformant plans and beyond: Principles and complexity. *Artificial Intelligence*, 174(3-4): 245–269.

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

Brafman, R. I.; and Domshlak, C. 2008. From One to Many: Planning for Loosely Coupled Multi-Agent Systems. In *ICAPS*, 28–35. AAAI Press.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69: 165–204.

Carreno, Y.; Pairet, È.; Pétillot, Y. R.; and Petrick, R. P. A. 2020. A Decentralised Strategy for Heterogeneous AUV Missions via Goal Distribution and Temporal Planning. In *ICAPS*, 431–439. AAAI Press.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; and Ridder, B. 2018. Opportunistic Planning in Autonomous Underwater Missions. *IEEE Transactions on Automation Science and Engineering*, 15(2): 519–530.

Chapman, D. 1987. Planning for Conjunctive Goals. *Artificial Intelligence*, 32(3): 333–377.

Chrpa, L.; Gemrot, J.; and Pilát, M. 2020. Planning and Acting with Non-Deterministic Events: Navigating between Safe States. In *AAAI*, 9802–9809. AAAI Press.

Chrpa, L.; and Karpas, E. 2024a. On Verifying and Generating Robust Plans for Planning Tasks with Exogenous Events. In *KR*, 273–283.

Chrpa, L.; and Karpas, E. 2024b. On Verifying Linear Execution Strategies in Planning Against Nature. In *ICAPS*, 86–94. AAAI Press.

Chrpa, L.; and Karpas, E. 2025. On Generating Robust Plans and Linear Execution Strategies in Planning Against Nature. In *ICAPS*, 169–177. AAAI Press.

Chrpa, L.; Pinto, J.; Ribeiro, M. A.; Py, F.; de Sousa, J. B.; and Rajan, K. 2015. On mixed-initiative planning and control for Autonomous underwater vehicles. In *IROS*, 1685–1690.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1-2): 35–84.

Cimatti, A.; and Roveri, M. 2000. Conformant Planning via Symbolic Model Checking. *Journal of Artificial Intelligence Research*, 13: 305–338.

Dean, T.; and Wellman, M. 1990. *Planning and Control*. Morgan Kaufmann Publishers.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173(5-6): 503–535.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research (JAIR)*, 14: 253–302.

Ingrand, F.; and Ghallab, M. 2017. Deliberation for autonomous robots: A survey. *Artificial Intelligence*, 247: 10–44.

Iocchi, L.; Nardi, D.; and Rosati, R. 2000. Planning with sensing, concurrency, and exogenous events: logical framework and implementation. In *KR*, 678–689.

Karpas, E.; Shleyfman, A.; and Tennenholtz, M. 2017. Automated Verification of Social Law Robustness in STRIPS. In *ICAPS*, 163–171.

Mausam; and Kolobov, A. 2012. *Planning with Markov Decision Processes: An AI Perspective*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Muise, C. J.; Felli, P.; Miller, T.; Pearce, A. R.; and Sonenberg, L. 2016. Planning for a Single Agent in a Multi-Agent Environment Using FOND. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, 3206–3212.

Muise, C. J.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-Deterministic Planning by Exploiting State Relevance. In *ICAPS*.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: a cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6): 1561–1574.

Percassi, F.; Castellanos-Paez, S.; Rombourg, R.; and Vallati, M. 2025. An Approach to Quantify Plans Robustness in Real-world Applications. In *IJCAI*, 8600–8607. ijcai.org.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research (JAIR)*, 39: 127–177.

Shmaryahu, D.; Shani, G.; and Hoffmann, J. 2019. Comparative criteria for partially observable contingent planning. *Autonomous Agents and Multi-Agent Systems*, 33(5): 481–517.