Explaining the Space of SSP Policies via Policy-Property Dependencies: Complexity, Algorithms, and Relation to Multi-Objective Planning

Marcel Steinmetz[1], Sylvie Thiébaux[1,2], Daniel Höller[3], Florent Teichteil-Königsbuch[4]

[1]LAAS-CNRS, Université de Toulouse, France
[2]Australian National University, Australia
[3]Saarland University, Saarland Informatics Campus, Germany
[4]Airbus AI Research, France
marcel.steinmetz@laas.fr, sylvie.thiebaux@laas.fr, hoeller@cs.uni-saarland.de, florent.teichteil-koenigsbuch@airbus.com

Abstract

Stochastic shortest path (SSP) problems are a common framework for planning under uncertainty. However, the reactive structure of their solution policies is typically not easily comprehensible by an end-user, nor do planners justify the reasons behind their choice of a particular policy over others. To strengthen confidence in the planner's decision-making, recent work in classical planning has introduced a framework for explaining to the user the possible solution space in terms of necessary trade-offs between user-provided plan properties. Here, we extend this framework to SSPs. We introduce a notion of policy properties taking into account action-outcome uncertainty. We analyze formally the computational problem of identifying the exclusion relationships between policy properties, showing that this problem is in fact harder than SSP planning in a complexity theoretical sense. We show that all the relationships can be identified through a series of heuristic searches, which, if ordered in a clever way, yields an anytime algorithm. Further, we introduce an alternative method, which leverages a connection to multi-objective probabilistic planning to move all the computational burden to a preprocessing step. Finally, we explore empirically the feasibility of the proposed explanation methodology on a range of adapted IPPC benchmarks.

1 Introduction

Stochastic shortest path (SSP) problems (Bertsekas and Tsitsiklis 1991) are a common framework for planning under action-outcome uncertainty. SSP solutions take into account arising contingencies by means of policies deciding what action to take next as a function of what happened in the past. This complex structure, however, leads to an astronomically large space Π of different possible solutions, a direct overview of which is beyond the capability of actual end-users of planning technology. At the same time, current SSP solution methods lack functionality for providing reasons behind the choice of a particular policy over others.

Krarup et al. (2021) introduced a generic iterative planning framework aiming at helping a user to interactively explore and comprehend the solution space Π. The goal of the user-planner interactions is to arrive at a solution that is in line with the user's preferences. Eifler et al. (2020a,b) proposed an instantiation of this framework for classical planning. The user-planner interactions are based on a set of user-provided **plan properties** Ω, i.e. Boolean functions on plans. The user iteratively navigates through the solution space by selecting different property subsets $\Phi \subseteq \Omega$. In response, she receives a summary of the part of the solution space satisfying Φ in terms of implied *exclusions*, i.e., the property subsets Ψ impossible to achieve simultaneously with Φ. Such *contrastive* explanations (Miller 2019) have proved particularly beneficial in strengthening the user's understanding of the solution space (Eifler et al. 2022).

Explaining solutions to probabilistic planning models has received significant attention in recent years, yet most prior work deals with the explanation of *individual* solutions. In explainable reinforcement learning, there is a particular focus on pointing out the importance of specific input features to a policy's decision (Milani et al. 2023; Selvey, Grastien, and Thiébaux 2023). Khan, Poupart, and Black (2009) and Juozapaitis et al. (2019) explain single action choices of a given policy by contrasting them to all available options by exposing trade-offs between different reward or cost objectives. Topin and Veloso (2019) summarize an entire RL policy through the use of state abstraction.

Building on Eifler et al.'s (2020a) explanation methodology, we propose a novel approach to explaining the **global** space of all SSP policies, or a **local** user-chosen subset. We introduce a notion of **policy properties**, taking into account action-outcome uncertainty, and formally study the computational problem of identifying their exclusion relationships. In order to compute the explanations, we devise two approaches that relate to constrained SSPs (Altman 1999) and multi-objective SSPs (Chen, Trevizan, and Thiébaux 2023), respectively. We show that both approaches can be efficiently realized by adapting the heuristic search algorithm I-DUAL (Trevizan et al. 2016), which embeds occupation-measure heuristics for effective guidance (Trevizan, Thiébaux, and Haslum 2017). Finally, we explore the feasibility of the proposed explanation methodology empirically on a range of adapted IPPC and new benchmarks.

2 Background

For a finite set X, we refer with $\Delta(X)$ to the set of all probability distributions over X. For $\delta \in \Delta(X)$, $supp(\delta) := \{x \in X \mid \delta(x) > 0\}$ gives the **support** of δ.

2.1 Probabilistic Planning

We consider probabilistic planning tasks in a SAS$^+$-like notation (Bäckström and Nebel 1995). A **planning task** is given by a tuple $\tau = \langle \mathcal{V}, \mathcal{A}, s_0, \mathcal{G}, c \rangle$, where $\mathcal{V}$ is a finite set of **state variables**, each $v \in \mathcal{V}$ having a finite **domain** $\mathcal{D}_v$; $\mathcal{A}$ is a finite set of **actions**; s_0 is the **initial state** (a complete variable assignment); $\mathcal{G}$ is a conjunctive **goal** (a variable assignment); and $c \colon \mathcal{A} \mapsto \mathbb{R}_0^+$ is the **cost function**. Pairs $\langle v, d \rangle$ of variables $v \in \mathcal{V}$ and values $d \in \mathcal{D}_v$ are called **facts**. The set of all facts is denoted with $\mathcal{F}$. The **states** of τ are the complete variable assignments. The set of all states is denoted with $\mathcal{S}$. For a (partial) variable assignment G, we denote with $G[v] \in \mathcal{D}_v$ the value assigned to v, if defined, and with $\mathcal{S}_G := \{s \in \mathcal{S} \mid G \subseteq s\}$ the subset of states that *satisfy* G. Each action a is associated with a **precondition** pre_a (a variable assignment) and a probability distribution Out_a over **probabilistic outcomes** (variable assignments). An action a is **applicable** in a state s if $s \in \mathcal{S}_{pre_a}$. The set of all actions applicable in s is denoted with $\mathcal{A}(s)$. The state resulting from an action outcome $o \in supp(Out_a)$ is $s[\![o]\!][v] = o[v]$ if $o[v]$ is defined, else $s[\![o]\!][v] = s[v]$.

A task τ induces the **stochastic shortest path (SSP) problem** $\Theta_\tau = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, s_0, \mathcal{S}_\mathcal{G}, c \rangle$ with same states, actions, initial state, goal states, and cost function, and the **transition probability function** $\mathcal{T} \colon \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ makes the outcome of action applications explicit. Unless important, we treat τ and Θ_τ interchangeably.

We consider **deterministic policies** (policies for short), which are (partial) functions $\pi \colon \mathcal{S} \rightharpoonup \mathcal{A}$ such that $\pi(s) \in \mathcal{A}(s)$ if $\pi(s)$ is defined. We write $\pi(s) = \bot$ if $\pi(s)$ is not defined. The **terminal states** $\mathcal{S}_\bot^\pi \subseteq \mathcal{S}$ under π are those s where $\pi(s) = \bot$. A state s' is **reachable** from a state s via π if there are states $s_1, \dots, s_n$ such that $s_1 = s$, $s_n = s'$, and, for all $1 \leq i < n$, $\pi(s_i) \neq \bot$ and $\mathcal{T}(s_i, \pi(s_i), s_{i+1}) > 0$. The set of states reachable from s via π is denoted with $\mathcal{S}^\pi(s)$; the reachable terminal states with $\mathcal{S}_\bot^\pi(s)$.

Let G be a variable assignment. The G**-reachability probabilities** of π are given by the piecewise smallest function $P^{G,\pi} \colon \mathcal{S} \mapsto [0,1]$ where $P^{G,\pi}(s) = 1$ for $s \in \mathcal{S}_\bot^\pi \cap \mathcal{S}_G$, $P^{G,\pi}(s) = 0$ for $s \in \mathcal{S}_\bot^\pi \setminus \mathcal{S}_G$, and otherwise $P^{G,\pi}(s) = \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') P^{G,\pi}(s')$. The $\mathcal{G}$-reachability probabilities are also called **goal probabilities**. We say that π is s**-proper** if $P^{\mathcal{G},\pi}(s) = 1$. We denote the set of all s-proper policies with $\Pi(s)$. The **expected cost** of π under the cost function c is given by $J^{c,\pi} : \mathcal{S} \mapsto \mathbb{R}_0^+ \cup \{\infty\}$; defined similarly to $P^{\mathcal{G},\pi}$, where $J^{c,\pi}(s) = \infty$ iff π is not s-proper. $J^{c,*}(s) := \inf_{\pi \in \Pi(s)} J^{c,\pi}(s)$ is the **optimal value** for s under c. π is **optimal** for s under c if $J^{c,\pi}(s) = J^{c,*}(s)$. When c is omitted, we refer to the task's cost function.

Example 1. *We use the flight planning instance from Fig. 1 as our running example. The state variables are* $\mathcal{V} = \{X, Y\}$ *and describe the aircraft's current position on the grid. There are four types of actions:* `left`(x,y), `right`(x,y), `up`(x,y) *and* `down`(x,y), *moving from the grid cell* (x,y) *in one of the four directions. Their effect is however subject to weather conditions. Specifically, we assume that each action has a* $\frac{1}{5}$ *chance of drifting to the cell left to the target cell (unless* $x = 0$*), e.g.,* $pre_{\mathtt{up}(2,3)} = \{\langle X, 2\rangle, \langle Y, 3\rangle\}$ *and* $Out_{\mathtt{up}(2,3)} = \{o_1 \mapsto \frac{4}{5}, o_2 \mapsto \frac{1}{5}\}$ *where* $o_1 = \{\langle X, 2\rangle, \langle Y, 2\rangle\}$ *and* $o_2 = \{\langle X, 1\rangle, \langle Y, 2\rangle\}$. *Each action* a *takes one time unit,* $c(a) = 1$. *The task is to find a policy that brings the aircraft from its initial position* $s_0 = \{\langle X, 2\rangle, \langle Y, 3\rangle\}$ *to the airport* $\mathcal{G} = \{\langle X, 2\rangle, \langle Y, 1\rangle\}$ *with minimal expected travel time. The arrows in Fig. 1 provide the action choices for three different policies, distinguished by the arrow color. All three policies are proper. The blue policy* π_b *is optimal with* $J^{c,\pi_b}(s_0) = \frac{49}{20} = J^{c,*}(s_0)$.

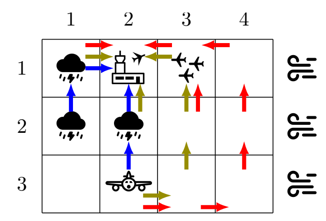

Figure 1: Illustration of the flight planning running example.

2.2 Classical Planning & Plan-Space Explanations

Classical planning is a special case of probabilistic planning where all actions have a single outcome. For brevity, we denote the unique resulting state of applying an action sequence π in a state s, if applicable, by $s[\![\pi]\!]$. π is called an s**-plan** if it is applicable in s and $\mathcal{G} \subseteq s[\![\pi]\!]$. The set of all s_0-plans, also called **plan space**, is denoted $\Pi(s_0)$.

A **plan property** (Eifler et al. 2020a) ϕ can be any predicate on plans representing some abstract behavior interpretable by the user. Eifler et al.'s (2020a) methods consider specifically properties ϕ expressible as facts $\langle v_\phi, t_\phi \rangle$ such that π satisfies ϕ iff $s_0[\![\pi]\!][v_\phi] = t_\phi$. Richer properties, such as ones formalized via LTL$_\text{f}$ (Giacomo and Vardi 2013), can be compiled into this format (Eifler et al. 2020b).

Let $\Omega \subseteq \mathcal{F}$ be a set of user-provided plan properties. Let $\Phi \subseteq \Omega$ be a subset of properties. Φ is **solvable** if there is a plan $\pi \in \Pi(s_0)$ such that $\Phi \subseteq s_0[\![\pi]\!]$. In this case, we also say π **satisfies** Φ. Φ is **unsolvable** if it is not satisfied by any plan. Φ is a **minimal unsolvable subset (MUS)** if Φ is unsolvable but every $\Phi' \subset \Phi$ is solvable. Φ **excludes** another $\Psi \subseteq \Omega$, written $\Phi \Rightarrow_\Pi \neg\Psi$, if all plans $\pi \in \Pi(s_0)$ that satisfy Φ violate Ψ. An exclusion $\Phi \Rightarrow_\Pi \neg\Psi$ is **non-dominated** if there is no exclusion $\Phi' \Rightarrow_\Pi \neg\Psi'$ such that $\Phi' \subseteq \Phi$ and $\Psi' \subseteq \Psi$ while one of the subset relations is proper. $\Phi \Rightarrow_\Pi \neg\Psi$ is **non-rhs-dominated** (non right-hand-side dominated) if no $\Psi' \subset \Psi$ is excluded by Φ. Observe that $\Phi \Rightarrow_\Pi \neg\Psi$ iff $\Phi \cup \Psi$ is unsolvable, and it is non-dominated iff $\Phi \cup \Psi$ is a MUS.

Example 2. *Desired plan properties in Fig. 1 could be avoiding thunderstorms and high traffic areas. The satisfaction of these properties can be tracked via additional Boolean state variables thund and traf, initially true, and set to false when entering the corresponding grid cells. Ignoring the weather influences, the shortest plan is moving the aircraft straight up to the airport. This plan satisfies the property* $\{\langle \mathit{traf}, 1\rangle\}$, *yet violates* $\{\langle \mathit{thund}, 1\rangle\}$. *In fact,* $\{\langle \mathit{traf}, 1\rangle, \langle \mathit{thund}, 1\rangle\}$ *is a MUS, yielding the exclusion* $\langle \mathit{thund}, 1\rangle \Rightarrow_\Pi \neg\langle \mathit{traf}, 1\rangle$.

Eifler et al. (2020a) distinguish *local* and *global* explanations. The former assume a given subset of properties $\Phi \subseteq \Omega$, and aim at explaining the subspace of plans which satisfy Φ. The **Φ-explanation** gives the consequences of this plan-space restriction in terms of the set $\{\Psi \subseteq \Omega \mid \Phi \Rightarrow_{\Pi} \neg\Psi$ is a non-rhs-dominated exclusion$\}$. In contrast, the **global explanation** provides a view on the entire plan space by means of a directed graph over property sets with an arc from Φ to Ψ if $\Phi \Rightarrow_{\Pi} \neg\Psi$ is a non-dominated exclusion relation. Computing either type of explanations boils down to solving **AllMUSes**, i.e., computing the set of all MUSes. For global explanations, this is inherent from the definition. For local explanations, this follows from the fact that the non-rhs-dominated exclusions of any Φ are given by the MUSes in the task with goal $\mathcal{G}' := \mathcal{G} \cup \Phi$.

3 Exclusion-Based Explanations for SSPs

3.1 Policy Properties

We generalize Eifler et al.'s (2020a) concept of plan properties to SSPs. The exclusion-based explanation framework requires properties to be Boolean predicates. In classical planning, each plan π uniquely determines an execution trace along with the resulting outcome state $s_0[\![\pi]\!]$. This makes the evaluation of every condition on π's execution inherently Boolean: either the condition is satisfied by π (e.g., $s_0[\![\pi]\!][v_\phi] = t_\phi$ for the plan property ϕ) or it is violated ($s_0[\![\pi]\!][v_\phi] \neq t_\phi$). One of the two cases definitely applies. In the probabilistic setting, this is more complicated given that policies no longer induce a unique outcome but instead give rise to probability distributions over many possible executions, while conditions on individual executions may be satisfied in some but not all the executions. Properties expressing characteristics of policies need to take into account this uncertainty, and therewith need to reason over the *expectation* of the properties' objectives. We consider two classes of such *policy properties*.

Reachability policy properties extend Eifler et al.'s (2020a) plan properties as introduced above.

Definition 1 (Policy Reachability Property)**.** *A* ***policy reachability property*** *ϕ consists of a variable assignment G_ϕ and a lower bound $\alpha_\phi \in (0, 1]$. A policy π satisfies ϕ iff $P^{G_\phi,\pi}(s_0) \geq \alpha_\phi$.*

Cost properties allow the user to explore trade-offs between different cost functions and reachability properties:

Definition 2 (Policy Cost Properties)**.** *A* ***policy cost property*** *ϕ consists of a cost function c_ϕ and an upper bound $\beta_\phi \in \mathbb{R}^+_0$. A policy π satisfies ϕ iff $J^{c_\phi,\pi}(s_0) \leq \beta_\phi$.*

As in Eifler et al.'s (2020a) framework, we assume the user specifies the relevant policy properties Ω, where $\Omega = \Omega^R \uplus \Omega^C$ is partitioned into reachability Ω^R and cost properties Ω^C. For a policy-property subset $\Phi \subseteq \Omega$, we similarly use $\Phi = \Phi^R \uplus \Phi^C$. We explain $\Pi(s_0)$ via the induced property-subset exclusion relationships. To this end, we extend the concepts of *solvable* and *unsolvable* property subsets, property-subset *exclusions*, and *local* and *global explanations* to policy properties in the obvious manner.

Example 3. *Reconsider the plan properties from Ex. 2. Notice that every proper policy has a chance, even just slight, to dissatisfy $G_\phi = \{\langle thund, 1\rangle\}$. The maximal possible probability of any proper policy keeping G_ϕ satisfied is ≈ 0.78, and is achieved by the red policy π_r from Fig. 1. That policy however trades this for achieving $G_\psi = \{\langle traf, 1\rangle\}$ with a probability of only $\vec{P}^{G_\psi,\pi_r}(s_0) = 0.168$. The green policy π_g satisfies G_ϕ with a slightly lower probability, $\vec{P}^{G_\phi,\pi_g}(s_0) = 0.752$, but, in exchange, has $\vec{P}^{G_\psi,\pi_g}(s_0) = 0.36$. The trade-off between ϕ and ψ is summarized by exclusion $\phi \Rightarrow_{\Pi} \neg\psi$, for any $\alpha_\phi > 0$ and $\alpha_\psi > 0.36$.*

It can be desirable to have in Ω multiple policy properties $\phi_1, \ldots, \phi_k$ with related base objectives (i.e., same cost function $c_{\phi_i} = c_{\phi_j}$, or variable assignments such that $G_{\phi_i} \subseteq G_{\phi_j}$) and different threshold values. This is useful, in particular, for exploring the effect of different property *relaxations* on the solvability of other properties. For that reason we slightly adapt the conditions of *minimal unsolvable subset* and *exclusion dominance*, taking into account the policy properties' flexibility in the threshold choice. We say that a cost property $\phi \in \Omega^C$ is a **relaxation** of another cost property $\psi \in \Omega^C$, written $\phi \sqsubseteq \psi$, if $c_\phi = c_\psi$ and $\beta_\phi \geq \beta_\psi$; and a reachability property $\phi \in \Omega^R$ is a **relaxation** of another reachability property $\psi \in \Omega^R$ ($\phi \sqsubseteq \psi$) if $G_\phi \subseteq G_\psi$ and $\alpha_\phi \leq \alpha_\psi$. A relaxation $\phi \sqsubseteq \psi$ where $\phi \neq \psi$ is called **strict** (denoted $\phi \sqsubset \psi$). A property subset Φ is a relaxation of another property subset Ψ, written $\Phi \sqsubseteq \Psi$ if every $\phi \in \Phi$ relaxes some $\psi \in \Psi$. Φ is a strict relaxation of Ψ ($\Phi \sqsubset \Psi$), if $\Phi \sqsubseteq \Psi$ and $\Phi \neq \Psi$. To define MUSes and policy-property subset exclusion dominance, we substitute the subset relation by relaxation:

Definition 3 (Minimal unsolvable policy-property subset)**.** *$\Phi \subseteq \Omega$ is a* ***minimal unsolvable property subset (MUS)*** *if it is unsolvable but each of its strict relaxations $\Phi' \subseteq \Omega$, i.e., $\Phi' \sqsubset \Phi$, is solvable.* **AllMUSes** *names the problem of computing the set of all MUSes.*

Definition 4 (Policy-property exclusion dominance)**.** *A policy-property subset exclusion $\Phi \Rightarrow_{\Pi} \neg\Psi$ is* ***non-dominated*** *if there is no other exclusion $\Phi' \Rightarrow_{\Pi} \neg\Psi'$ such that $\Phi' \sqsubseteq \Phi$ and $\Psi' \sqsubseteq \Psi$ and at least one of the relaxations is strict. Similarly, $\Phi \Rightarrow_{\Pi} \neg\Psi$ is* ***non-rhs-dominated*** *if there is no $\Psi' \subseteq \Omega$ such that $\Psi' \sqsubset \Psi$ and $\Phi \Rightarrow_{\Pi} \neg\Psi'$.*

Example 4. *Reconsider the reachability properties ϕ and ψ from Ex. 3, and let $\varphi_1, \varphi_2, \varphi_3$ be three cost properties, all on the travel time cost function, with $\beta_{\varphi_1} = 3$, $\beta_{\varphi_2} = 5$, and $\beta_{\varphi_3} = 8$, i.e., $\varphi_3 \sqsubset \varphi_2 \sqsubset \varphi_1$. Then, $\varphi_1 \Rightarrow_{\Pi} \neg\phi$ is an exclusion for any $\alpha_\phi > 0$; $\varphi_2 \Rightarrow_{\Pi} \neg\phi$ for any $\alpha_\phi > 0.752$; and $\varphi_3 \Rightarrow_{\Pi} \neg\{\phi, \psi\}$ for any $\alpha_\phi > 0$ and $\alpha_\psi > 0.36$. Moreover, as per the latter, $\varphi_1 \Rightarrow_{\Pi} \neg\{\phi, \psi\}$ and $\varphi_2 \Rightarrow_{\Pi} \neg\{\phi, \psi\}$ are valid exclusions as well, but both are dominated according to the relaxation relationships.*

Note that both definitions subsume their original counterparts. This follows directly from the observation that if $\Phi' \subseteq \Phi$ for any two policy-property subsets, then it also holds that $\Phi' \sqsubseteq \Phi$. Vice versa, however, there can be non-minimal unsolvable policy-property subsets Φ (respectively, dominated exclusions $\Phi' \Rightarrow_\Pi \neg\Psi'$), although every subset of Φ is solvable (respectively, no exclusion satisfies the subset-based dominance criterion). Also note that it is still the case that an exclusion $\Phi \Rightarrow_\Pi \neg\Psi$ is non-dominated iff the combined property set $\Phi \cup \Psi$ is a MUS, and that $\Phi \Rightarrow_\Pi \neg\Psi$ is non-rhs-dominated iff, for every strict relaxation $\Psi' \subseteq \Omega$ of Ψ, $\Phi \cup \Psi'$ is solvable. Finally, observe that for every non-dominated exclusion $\Phi \Rightarrow_\Pi \neg\Psi$, both Φ and Ψ must be **unambiguous** in that neither of them can contain two properties ϕ and ϕ' such that $\phi \sqsubset \phi'$.

3.2 Exclusion-Explanation Computation

With the ability to express the satisfaction of plan properties as properties on states, AllMUSes in classical planning degenerates to standard goal reachability questions for different goal sets. Eifler et al. (2020b) showed that these different questions can be solved with a single state-space search, resulting in a method offering similar scalability than that of optimal classical planners. Unfortunately, conditions on the expected behavior of policies can no longer be expressed as properties on states. This has two consequences. First, AllMUSes can no longer be solved via methods similar to the algorithm just sketched. Second, whereas in classical planning, a local Φ-explanation could be computed by solving AllMUSes for a goal-extended task, this is no longer possible for SSPs.

Property-Subset Lattice Exploration Alg. 1 depicts our general algorithm for computing local Φ-explanations, and closely resembles previous algorithms for computing minimal unsolvable goal/constraint subsets (Eifler et al. 2020a; Liffiton et al. 2016) with the difference of using property-subset relaxation in place of the subset relation. At its core, Alg. 1 conducts an exhaustive exploration of the space of all property subsets, testing the solvability of the combined set $\Phi \cup \Psi$ for each visited Ψ. As $\Phi \Rightarrow_\Pi \neg\Psi$ holds iff $\Phi \cup \Psi$ is unsolvable, Φ's non-rhs-dominated exclusions are given exactly by the non-relaxable Ψ for which the combined set was found unsolvable (line 6). To avoid the complete enumeration of all the property subsets, Alg. 1 utilizes the relaxation relationships with already tested property subsets to remove from consideration (line 4) (i) known to be solvable as well as (ii) known to be unsolvable property subsets. Note that (ii) only discards property subsets that are dominated by an already found one, hence not affecting the correctness of the overall algorithm. Additionally, (iii) discards ambiguous property subsets, which as previously noted cannot be members of any non-dominated exclusion. The specific choice of Ψ is not important for the correctness. In our implementation, we use MARCO (Liffiton et al. 2016), which chooses Ψ in a way guaranteeing that each Ψ added to *Unsolvable* yields a non-rhs-dominated exclusion, resulting in an anytime explanation enumeration procedure. Finally, note that by setting $\Phi = \emptyset$, the final result $\min_{\sqsubseteq}$ *Unsolvable* will be exactly the set of all MUSes.

Theorem 1. *Alg. 1 terminates with the Φ-explanation. If $\Phi = \emptyset$ it returns all MUSes.*

The Property-Subset Solvability Problem Alg. 1 relies on a sub-procedure deciding the property subsets' solvability. Before discussing possible implementations, here we define and analyze the solvability problem formally.

Algorithm 1: Generic property-subset lattice exploration for computing the Φ-explanation for a given property subset Φ.

```
Input: Probabilistic planning task $\tau$, set of policy properties $\Omega$,
    property subset $\Phi \subseteq \Omega$
Output: All non-rhs-dominated exclusions of $\Phi$
 1: $Solvable \leftarrow \emptyset$;
 2: $Unsolvable \leftarrow \emptyset$;
 3: while true do
 4:     Pick some $\Psi \subseteq \Omega \setminus \Phi$ such that
            (i) $\forall \Psi' \in Solvable\colon \Psi \not\sqsubseteq \Psi'$, and
            (ii) $\forall \Psi' \in Unsolvable\colon \Psi' \not\sqsubseteq \Psi$, and
            (iii) $\Psi$ is unambiguous
 5:     if such a $\Psi$ does not exist then
 6:         return $\min_{\sqsubseteq} Unsolvable$
 7:     end if
 8:     if $\Phi \cup \Psi$ is solvable then
 9:         $Solvable \leftarrow Solvable \cup \{\Psi\}$;
10:     else
11:         $Unsolvable \leftarrow Unsolvable \cup \{\Psi\}$;
12:     end if
13: end while
```

Definition 5 (PSS). ***The property-subset solvability problem* (PSS)** *is that of deciding, given a probabilistic planning task τ and a non-empty set of policy properties Φ, whether there exists an s_0-proper policy π that satisfies Φ.*

We distinguish decision problems along three lines: (1) classical versus probabilistic planning, (2) the input being the *factored* planning task description versus the *flat* state space, and (3) in the probabilistic case, allowing *stochastic* policies versus imposing the restriction to *deterministic* policies. It is well-known that with flat representations, the standard plan/policy existence problems can be decided in polynomial time (e.g., Puterman 1994). The policy type does not make a difference. For factored representations, classical planning is PSPACE-COMPLETE (Bylander 1994) and probabilistic-planning EXP-COMPLETE (Littman, Goldsmith, and Mundhenk 1998).

Moving to PSS, the additional constraints imposed by the properties have an impact on the complexity in some but not all cases. Consider classical planning. PSS for the flat case becomes NP-COMPLETE, being a generalization of the weight-limited shortest-path problem (Garey and Johnson 1979). For the factored representation, PSS stays in the same complexity class as plan existence thanks to the ability to compile the properties into facts. For SSPs, PSS in the flat and stochastic-policy case can still be solved in time polynomial in the size of the SSP via an extension of the standard SSP linear program (as we will see in the next section). For deterministic policies, however, PSS becomes NP-COMPLETE, given the classical planning result. PSS for factored SSPs τ remains EXP-complete if stochastic policies are allowed; membership follows from applying the "flat" polynomial-time algorithm to Θ_τ. However, as in the flat case, the restriction to deterministic policies again raises the complexity by the "non-deterministic" factor. Therefore, and this is in contrast to classical planning, generating explanations for SSPs becomes computationally more demanding than solving the planning task itself (unless NEXP = EXP).

Theorem 2. *For singleton property sets* Φ, $|\Phi| = 1$, PSS *is* EXP-COMPLETE. *For two properties or more,* $|\Phi| \geq 2$, PSS *is* NEXP-COMPLETE.

Proof. A single property $|\Phi| = 1$ can be directly mapped into the SSP's optimization function. Φ is solvable iff the corresponding optimal solution satisfies the threshold. This shows the first part of the claim. For $|\Phi| = 2$, one can adapt the proof by (Feinberg 2000) showing NP-hardness in the flat case based on a reduction from the Hamiltonian cycle problem. We can apply the same proof idea, however considering a succinct circuit representation of the graph, i.e., yielding a reduction from the succinct Hamiltonian cycle problem which is known to be NEXP-complete (Galperin and Wigderson 1983). The circuit can be encoded as a probabilistic planning task, whose size relates polynomially to the size of the circuit, using binary variables to track the succinct representation of graph nodes, and implementing the circuit's transition test via actions. This shows that PSS is NEXP-hard. Membership follows via guess and check, both can be done in exponential time. $\square$

Despite that difference between stochastic and deterministic policies, the general preference of the latter in practical applications raises the need for methods tackling PSS.

4 Solving PSS by Search in the Dual Space

Finding a policy that satisfies a given property subset Φ boils down to solving the SSP under additional *policy constraints*. This relates to the class of **constrained SSPs (CSSPs)** (Altman 1999). This section develops an extension of I-DUAL (Trevizan et al. 2016) – the so far only known heuristic-search algorithm for solving SSPs supporting both expected cost and reachability-probability constraints[1] – to solving PSS. To this end, we start with a brief recap of CSSPs and I-DUAL, then introduce a mixed-integer linear program (MIP) characterization of PSS, and finally show how to adapt I-DUAL to solve this integer program efficiently.

4.1 Background: CSSPs and I-DUAL

CSSPs extend SSPs with a list of additional cost functions $c_1, \dots, c_k$ and accompanying bounds $\beta_1, \dots, \beta_k$ on the policies' corresponding expected costs. Unlike in the unconstrained version, optimally solving CSSPs in general requires *stochastic* policies, i.e., ones mapping states to probability distributions over actions. The optimal solutions are characterized exactly by the linear program (LP) depicted in Fig. 2 (cf., e.g., Altman 1999).

The basis of the encoding are the *occupation-measure (OM) variables* $\mathsf{X}_{s,a}$ which represent the expected number of times action a is to be executed in state s. The expected cost of the policy represented by the OM variables is given by the linear combination with the cost function (cf. def.

$$
\begin{array}{llr}
\text{Minimize } \mathit{cost}[c] + \mathit{cost}^{\hat{v}}[c] & & \text{(C1)} \\
\text{Subject to} & & \\
\mathsf{X}_{s,a} \geq 0, & s \in \mathcal{S} \setminus \mathcal{S}_\mathcal{G}, a \in \mathcal{A}(s) & \text{(C2)} \\
\mathit{cost}[c_i] + \mathit{cost}^{\hat{v}}[c_i] \leq \beta_i, & 1 \leq i \leq k & \text{(C3)} \\
\mathit{out}[s] - \mathit{in}[s] = [s = s_0], & s \in \mathcal{S} \setminus \mathcal{S}_\mathcal{G} & \text{(C4)} \\
\sum_{s \in \mathcal{S}_\mathcal{G}} \mathit{in}[s] = 1 & & \text{(C5)} \\
\mathit{cost}[c'] := \sum_{s \in \mathcal{S}, a \in \mathcal{A}(s)} \mathsf{X}_{s,a} c'(a), & c' \in \{c, c_1, \dots, c_n\} & \\
\mathit{out}[s] := \sum_{a \in \mathcal{A}(s)} \mathsf{X}_{s,a}, & s \in \mathcal{S} & \\
\mathit{in}[s] := \sum_{s' \in \mathcal{S}, a \in \mathcal{A}(s')} \mathsf{X}_{s',a} \mathcal{T}(s', a, s), & s \in \mathcal{S} & \\
\mathsf{X}^{v}_{d,a} \geq 0, & v \in \mathcal{V}, d \in \mathcal{D}_v, a \in \mathcal{A} & \text{(C6)} \\
v\text{-projection OM constraints}, & v \in \mathcal{V} & \text{(C7)} \\
\text{constraints tying } \mathsf{X}^{v}_{\cdot,a} \text{ with } \mathsf{X}^{v'}_{\cdot,a}, & v, v' \in \mathcal{V}, v \neq v', a \in \mathcal{A} & \text{(C8)}
\end{array}
$$

Figure 2: LP encoding of a CSSP. $[\varphi]$ denotes the *Iverson bracket* and evaluates to 1 iff φ is satisfied and 0 otherwise. $\mathit{cost}[c']$ is a shorthand for the OM variables' induced expected cost under c'; $\mathit{out}[s]$ and $\mathit{in}[s]$ for the flow leaving respectively entering state s. It is assumed that $s_0 \notin \mathcal{S}_\mathcal{G}$. Parts in blue are I^2-DUAL extensions; $\hat{v} \in \mathcal{V}$ is arbitrary.

of $\mathit{cost}[c']$). The objective function (C1) asks for minimizing the expected cost under the SSP's primary cost function. The bounds on secondary cost functions map into additional constraints (C3). (C4) bounds the states' *flow*, asserting the expected numbers of times a non-goal state is entered (the *in-flow*) and exited (the *out-flow*) to be the same. A flow surplus of 1 is inserted at the initial state. (C5) implements the s_0-proper policy requirement by asserting that the entire inserted flow sinks at the SSP's goal states eventually.

The LP necessitates building the entire state space Θ_τ. I-DUAL (Trevizan et al. 2016) aims to avoid this by iteratively solving progressively larger sub-SSPs $\hat{\Theta} = \langle \hat{\mathcal{S}}, \mathcal{A}, \hat{\mathcal{T}}, s_0, \hat{\mathcal{S}_\mathcal{G}} \rangle$ of Θ_τ. One iteration of I-DUAL consists of solving the CSSP LP for the current sub-SSP and subsequently expanding $\hat{\Theta}$ at the *fringe states* visited by the found solution, where $\hat{\Theta}$'s **fringe** $F \subseteq \hat{\mathcal{S}}$ is the set of non-goal states whose transitions from Θ_τ are not present in $\hat{\Theta}$ yet. An optimal CSSP solution is found when all flow is sinking at goal states, which may happen before the entire state space was built. To guide the exploration, I-DUAL leverages heuristics to estimate the expected remaining cost-to-goal, for each cost function, depending on the reached fringe states. The estimates become additional summands in the objective and the cost constraints. Solution optimality is preserved if the heuristics are admissible.

Trevizan, Thiébaux, and Haslum (2017) presented an extension of I-DUAL, baptized I^2-DUAL, which integrates the computation of a particular heuristic, the **projection occupation-measure (POM) heuristic** h^{pom}, directly in I-DUAL's LP. This has the advantage over using cost-function individual heuristics in being able to *jointly* consider all CSSP's cost constraints inside the heuristic computation. The blue parts in Fig. 2 summarize the integration. In brief, h^{pom} is based on the *projections* of τ onto the individual state variables $\mathcal{V}$, where the v-projection is the simplified task obtained from τ by discarding all other variables. h^{pom}

[1]Hong and Williams (2023) introduced a heuristic search based method solving CSSPs for deterministic policies, similar to PSS, yet it does not support constraints on reachability probabilities.

$$
\begin{array}{lrr}
\text{Minimize } \mathit{cost}[c] + \mathit{cost}^{\hat{v}}[c] & & \text{(C9)} \\
\text{Subject to} & & \\
\mathsf{X}_{s,a} \geq 0, & \text{for } \mathit{all}\ s \in \mathcal{S}, a \in \mathcal{A}(s) & \text{(C10)} \\
\mathit{cost}[c_\phi] + \mathit{cost}^{\hat{v}}[c_\phi] \leq \beta_\phi & \text{for each } \phi \in \Phi^C & \text{(C11)} \\
\mathit{out}[s] - \mathit{in}[s] \leq [s = s_0], & s \in \mathcal{S} & \text{(C12)} \\
\mathit{res}[\mathcal{S}_\mathcal{G}] + \mathsf{P}_\mathcal{G} = 1 - [s_0 \in \mathcal{S}_\mathcal{G}] & & \text{(C13)} \\
\mathit{res}[\mathcal{S}_{G_\phi}] + \mathsf{P}_{G_\phi} \geq \alpha_\phi - [s_0 \in \mathcal{S}_{G_\phi}], & \phi \in \Phi^R & \text{(C14)} \\
\mathsf{D}_{s,a}K - \mathsf{X}_{s,a} \geq 0, & s \in \mathcal{S}, a \in \mathcal{A}(s) & \text{(C15)} \\
\sum_{a \in \mathcal{A}(s)} \mathsf{D}_{a,s} + \mathsf{T}_s \leq 1, & s \in \mathcal{S} & \text{(C16)} \\
\mathit{out}[s] - \mathit{in}[s] + \mathsf{T}_s \geq [s = s_0], & s \in \mathcal{S} & \text{(C17)} \\
\mathsf{D}_{s,a} \in \{0,1\}, & s \in \mathcal{S}, a \in \mathcal{A}(s) & \text{(C18)} \\
\mathsf{T}_s \in \{0,1\}, & s \in \mathcal{S} & \text{(C19)} \\
\mathit{res}[S] := \sum_{s \in S} (\mathit{in}[s] - \mathit{out}[s]), & S \subseteq \mathcal{S} & \\
\mathsf{X}^v_{d,a} \geq 0, & v \in \mathcal{V}, d \in \mathcal{D}_v, a \in \mathcal{A} & \text{(C20)} \\
\mathsf{P}_G \geq 0, & G \in \{G_\phi \mid \phi \in \Phi^R\} \cup \{\mathcal{G}\} & \text{(C21)} \\
\textit{v-projection OM constraints, excluding the proper-policy} & & \text{(C22)} \\
\textit{constraint,} & v \in \mathcal{V} & \\
\text{constraints } \textit{tying } \mathsf{X}^v_{\cdot,a} \text{ with } \mathsf{X}^{v'}_{\cdot,a}, & v, v' \in \mathcal{V}, v \neq v', a \in \mathcal{A} & \text{(C23)} \\
\mathsf{P}_G \leq \mathit{res}^v[G[v]], & v \in \mathcal{V} \text{ s.t. } G[v] \text{ is defined} & \text{(C24)}
\end{array}
$$

Figure 3: MIP encoding of PSS for the property subset $\Phi = \Phi^R \uplus \Phi^C$. Builds on Fig. 2; objective and constraints are extended to also take into account occupation-measure variables of goal states. $\mathit{res}[S]$ is a shorthand for the flow residing in the state set S. $K \in \mathbb{R}^+_0$ is a large constant. Parts in blue are I^2-DUAL extensions; $\hat{v}$ is arbitrary.

combines in a single LP (represented by (C6) – (C8)) the OM-based representation of the projections' SSPs, *tying* together the OM variables between the projections by enforcing that each projection executes every action overall exactly the same number of times. By inserting flow into the projections' states according to the probabilities sinking in the fringe states, the projections' OM variables yield the heuristic summands for (C1) and (C3). Given (C8), it does not matter whose projection's OM variables are used to this end exactly. Fig. 2 chooses those of an arbitrary but fixed $\hat{v} \in \mathcal{V}$.

4.2 Characterizing PSS as a MIP

Fig. 3 builds on the LP from Fig. 2, modeling PSS as a MIP. CSSPs differ from PSS in two regards: CSSPs (1) consider solely constraints on secondary cost functions; and (2) assume stochastic policies. In the presence of a single goal-reachability objective, i.e., assumption (1), one can assume w.l.o.g. that goal states are *absorbing*, forcing policies to immediately terminate once reaching those states. This assumption allows to exclude in the LP (a) the trivial special case $s_0 \in \mathcal{S}_\mathcal{G}$, (b) omit goal-state leaving transitions, setting $\mathit{out}[s_\mathcal{G}] = 0$ implicitly for all goal states $s_\mathcal{G}$, and (c) ignore flow constraints for the goal states.

In contrast, in the presence of multiple reachability constraints, continuing the execution in the SSP's goal states might be necessary for achieving the other reachability objectives. For illustration, consider the simple SSP with two states s_0 and s_{G_ϕ} and a single transition $\mathcal{T}(s_0, a, s_{G_\phi}) = 1$. s_0 is the initial state, both states are goal states of the SSP, but only s_{G_ϕ} satisfies the secondary reachability objective G_ϕ. While the empty policy (not assigning an action to any state) is s_0-proper, $\pi(s_0) := a$ is necessary to obtain $P^{G_\phi,\pi}(s_0) = 1$. Fig. 3 accounts for additional reachability objectives by introducing occupation-measure variables also for goal-leaving transitions. Thus, the flow constraints (C12) need to be defined for all states, while to allow flow sinking at some states, the equality needs to be changed to an upper-bound constraint. As flow may now escape from goal states, (C5) is no longer a sufficient proper policy condition. (C13) in Fig. 3 instead ensures that all the flow enters and *resides* in the SSP's goal states eventually by additionally taking into account their out-flow. Furthermore, (C13) handles the case $s_0 \in \mathcal{S}_\mathcal{G}$ by taking into account the initial flow inserted at s_0 if s_0 is a goal state. Reachability properties can now be encoded as flow-residual constraints (C14) in the same manner as the SSP's goal constraint (C13).

The second source of complication arises from the fact that PSS talks about deterministic policies, which in the presence of additional policy constraints are no longer equally expressive as stochastic policies, as previously mentioned. We implement that requirement by binary integer *decision* variables $\mathsf{D}_{s,a}$ for each state-action pair. (C15) ensures that $\mathsf{D}_{s,a}$ is set if the corresponding transition is assigned a positive occupation measure, while (C16) insists on setting at most one $\mathsf{D}_{s,a}$ per state. The additional binary integer variables T_s are explicit indicators of the policy terminating in s. These are necessary because with the policy's termination being no longer determined by the satisfaction of the SSP's goal, termination becomes a deliberative policy choice. (C16) makes sure that this choice remains deterministic. That this is indeed necessary can be seen in the previous SSP example. Suppose that $\alpha_\phi = \frac{1}{2}$ and that there is a second reachability property ψ with $\alpha_\psi = \frac{1}{2}$ and G_ψ is satisfied in s_0. Clearly, there is no s_0-proper policy that satisfies both ϕ and ψ. However, without the additional termination variables, $\mathsf{X}_{s_0,a} = \frac{1}{2}$ and $\mathsf{D}_{s_0,a} = 1$ would yield a solution to the MIP. Finally, (C17) ensures that the termination indicator T_s is set whenever some flow is sinking in state s. Putting everything together, we conclude:

Theorem 3. *Let τ be an SSP and Φ be some set of policy properties. For a suitably large constant K, the MIP from Fig. 2 has an optimal solution iff Φ is solvable.*

4.3 Solving PSS with Heuristic Search

The principles of I-DUAL apply directly to PSS. The main change required is, obviously, the substitution of the CSSP LP by the MIP from Fig. 3. Some care must be taken to correctly handle the subtle differences between PSS and CSSPs. First, given that goal states can no longer be assumed to be absorbing, goal states must now be included in I-DUAL's fringe (and thus possibly expanded). Second, given that goal states may now be contained in the fringe, we must change I-DUAL's termination condition testing explicitly that no flow is sinking in the fringe. Lastly, the fringe states must be handled appropriately in PSS's reachability constraints. That is, we must make sure to *optimistically* (hence upper) bound the expected achievement of the individual reachability properties given the fringe states reached. This can be done, in the same vein as for the cost constraints, with the help of admissible goal-probability heuristics. After these changes, I-DUAL's original correctness arguments can be carried over to show that our adapted variant correctly solves PSS.

To complete the adaptation of I^2-DUAL, it only remains to extend the h^{pom} part to deliver the necessary optimistic reachability-probability bounds. The blue parts in Fig. 3 highlight the main changes. In summary, we discard h^{pom}'s proper-policy constraint. h^{pom}'s reachability-probability bounds are represented through additional variables P_G for each relevant G, cf. (C21), which become additional summands in the proper-policy (C13) and the reachability-property constraints (C14) of the overall MIP. The constraints (C24) synchronize the values of those reachability variables with the actual probability of residing in the projections' G-achieving states. That this indeed yields an upper bound on the probability of achieving G from the fringe states follows similarly to the cost bounds from the fact that projections are solution preserving.

Theorem 4. *Run* I^2-DUAL *or* I-DUAL *with admissible heuristics. If at any point in time, the* MIP *for one of the sub-*SSP*s* $\hat{\Theta}$ *becomes infeasible, then* Φ *is unsolvable. If they return* π*, then* π *is* s_0*-proper and* π *satisfies* Φ.

Note that the h^{pom} modifications sketched so far do not ensure that the projection's OM variables resemble a deterministic policy. One might be tempted to enforce determinism within every projection via additional integer variables as in Fig. 3. This however breaks h^{pom}'s admissibility property, because due to the tying of the projections, different actions might need to be applied in a single projection state.

5 AllMUSes via MO Optimization

Ignoring the properties' thresholds, one is left with a set of policy metrics whose simultaneous optimization becomes a variant of **multi-objective (MO) SSPs** (Chen, Trevizan, and Thiébaux 2023). In the following, we leverage this connection, showing that the solution to this MOSSP variant contains all the relevant information to decide any PSS.

5.1 Background: MOSSPs

Multi-objective SSPs (Roijers and Whiteson 2017; Chen, Trevizan, and Thiébaux 2023) differ from regular SSPs in optimizing a **cost-function vector** $\vec{c} = [\, c_1 \cdots c_n \,]$ rather than a single cost function. Optimality in the MO setting is defined via a dominance order between real vectors, where for two real vectors $\vec{x} = [\, x_1 \cdots x_n \,]$ and $\vec{y} = [\, y_1 \cdots y_n \,]$, $\vec{x}$ **weakly dominates** $\vec{y}$ ($\vec{x} \preceq \vec{y}$) if $x_i \leq y_i$ holds for all i. $\vec{x}$ **dominates** $\vec{y}$ ($\vec{x} \prec \vec{y}$) if $\vec{x} \preceq \vec{y}$ and $\vec{x} \neq \vec{y}$. Associating each policy π with the vector of expected-cost functions $\vec{J}^{\vec{c},\pi} := [\, J^{c_1,\pi} \ldots J^{c_n,\pi} \,]$, the optimal MOSSP policies are those s_0-proper policies π where $\vec{J}^{\vec{c},\pi}(s_0)$ is not dominated by $\vec{J}^{\vec{c},\pi'}(s_0)$ for any other π'. Since $\preceq$ is not a total order, optimal policies can have different value vectors. The **optimal MO-value function** $\mathbb{J}^*$ assigns every state to the set of all these optimal cost vectors. Like constrained SSPs, MOSSPs assume stochastic policies, and there can exist $\vec{J}^* \in \mathbb{J}^*(s)$ that is not achievable by any deterministic policy. $|\mathbb{J}^*(s)|$ is in general infinite but can be represented as the convex hull of a finite coverage set (Roijers and Whiteson 2017).

5.2 MOSSPs with Deterministic Policies, and Connection to PSS

To be able to translate the policy property set Ω into a multi-objective optimization problem, we introduce a slight variation of MOSSPs that supports multiple cost as well as goal objectives. Concretely, let $\vec{c} = [\, c_1 \cdots c_n \,]$ be a cost-function vector, as before, and let $\vec{G} = [\, G_1 \ldots G_m \,]$ be a vector of variable assignments. For simplicity's sake, we denote for every policy π the combined policy value vector with $\vec{V}^{\pi} = [\, J^{c_1,\pi} \ldots J^{c_n,\pi} \; -P^{G_1,\pi} \ldots -P^{G_m,\pi} \,]$ (note the negation of the reachability probabilities). A deterministic policy $\pi \in \Pi(s_0)$ is called **deterministic optimal** if $\vec{V}^{\pi}$ is not dominated by $\vec{V}^{\pi'}$ of any $\pi' \in \Pi(s_0)$. The deterministic-optimal MO-value function is given by $\mathbb{V}^*_{\mathsf{d}}(s) := \{\vec{V}^{\pi}(s) \mid \pi \in \Pi(s) \text{ s.t. } \nexists \pi' \in \Pi(s) : \vec{V}^{\pi'}(s) \prec \vec{V}^{\pi}(s)\}$. Note that $|\mathbb{V}^*_{\mathsf{d}}(s)|$ is always guaranteed to be finite.

Let $\vec{c} = [\, c_0 \cdots c_n \,]$ and $\vec{G} = [\, G_1 \ldots G_m \,]$ where c_0 denotes the SSP's main cost function, $c_1, \ldots, c_n$ are the cost functions underlying the cost properties Ω^C, and $G_1, \ldots, G_m$ are the goal sets underlying Ω^R. Let $\Phi \subseteq \Omega$ be an unambiguous property subset. We associate with $\beta_\Phi(c_i) := \beta_\phi$ the cost threshold assigned by the property $\phi \in \Phi^C$ such that $c_\phi = c_i$, if it exists, and define $\beta_\Phi(c_i) := \infty$ otherwise. $\alpha_\Phi(G_i)$ is defined similarly. Suppose we are given $\mathbb{V}^*_{\mathsf{d}}(s_0)$:

Theorem 5. *Let* $\Phi \subseteq \Omega$ *be an unambiguous property subset. Let* $\vec{v}_\Phi := [\, \infty \; \beta_\Phi(c_1) \ldots \beta_\Phi(c_n) \; -\alpha_\Phi(G_1) \ldots -\alpha_\Phi(G_m) \,]$. $\vec{v}_\Phi$ *is dominated by one of the vectors in* $\mathbb{V}^*_{\mathsf{d}}(s_0)$ *iff* Φ *is solvable.*

Plugged into Alg. 1, this yields an alternative approach to computing the exclusions, where all property-subset solvability tests boil down to lookups; in particular no additional MIPs need to be solved. For any Φ, the domination condition on $\vec{v}_\Phi$ can obviously be checked in time linear in the number of entries in $\mathbb{V}^*_{\mathsf{d}}(s_0)$. Therefore, if the size of $\mathbb{V}^*_{\mathsf{d}}(s_0)$ is small compared to the number of possible property subsets, then this approach can be expected to be more effective than solving the requested property subsets via individual planner calls – provided that the overhead of the $\mathbb{V}^*_{\mathsf{d}}(s_0)$ precomputation does not outweigh this benefit.

The decision-problem formulation asking whether $\vec{V}^{\pi} \in \mathbb{V}^*_{\mathsf{d}}(s_0)$, for a given $\vec{V}^{\pi}$, is equivalent to the definition of PSS (Def. 5). Hence, as a corollary from Thm. 2, this decision problem is NEXP-COMPLETE. Existing MOSSP techniques cannot be used to solve our multi-objective SSP variant because they are limited to stochastic policies.

5.3 Enumerating Non-Dominated Solutions

Let $\vec{c} = [\, c_1 \cdots c_n \,]$ be a cost-function and $\vec{G} = [\, G_1 \ldots G_m \,]$ be a goal vector. The non-dominated solution vectors $\mathbb{V}^*_{\mathsf{d}}(s_0)$ can be enumerated by solving $|\mathbb{V}^*_{\mathsf{d}}(s_0)|$ MIPs. Fig. 4 depicts the encoding for finding a single new non-dominated solution. The MIP is iteratively refined taking into account the set *Incumbent* of policies computed so far. During the course

Minimize

$$\sum_{i=1}^{n} \omega_{c_i}\, cost[c_i] - \sum_{j=1}^{m} \omega_{G_j}\, res[\mathcal{S}_{G_j}] + \sum_{i=1}^{n} \omega_{c_i}\, cost^{\hat{v}}[c_i] - \sum_{j=1}^{m} \omega_{G_j} \mathsf{P}_{G_j} \quad \text{(C25)}$$

Subject to

MIP *from Fig. 3 without* (C*11*) *and* (C*14*) (C26)

$$cost[c_i] + cost^{\hat{v}}[c_i] - \mathsf{W}^{\pi}_{c_i} K \leq J^{c_i,\pi}(s_0) - \varepsilon, \quad \pi \in \mathit{Incumbent}, i \in \{1,\ldots,n\}$$
$$res[\mathcal{S}_{G_j}] + \mathsf{P}_{G_j} + \mathsf{W}^{\pi}_{G_j} K \geq P^{G_j,\pi}(s_0) + \varepsilon, \quad \pi \in \mathit{Incumbent}, j \in \{1,\ldots,m\}, \quad \text{(C27)}$$
$$\sum_{i=1}^{n} \mathsf{W}^{\pi}_{c_i} + \sum_{i=1}^{m} \mathsf{W}^{\pi}_{G_i} \leq n + m - 1, \quad \pi \in \mathit{Incumbent} \quad \text{(C28)}$$
$$\mathsf{W}^{\pi}_{c_i} \in \{0,1\}, \quad \pi \in \mathit{Incumbent}, i \in \{1,\ldots,n\}$$
$$\mathsf{W}^{\pi}_{G_j} \in \{0,1\}, \quad \pi \in \mathit{Incumbent}, j \in \{1,\ldots,m\} \quad \text{(C29)}$$

Figure 4: MIP for enumerating the non-dominated solution vectors $\mathbb{V}^*_{\mathsf{d}}(s_0)$. *Incumbent* denotes the set of policies extracted thus far. ω is a predefined convex combination of all objectives. $\varepsilon \in (0, \infty)$ is the precision parameter. Blue parts show I^2-DUAL modifications.

of all iterations, it is guaranteed that (1) only deterministic-optimal policies are ever added to *Incumbent*, and (2) that for each value vector $\vec{V}^* \in \mathbb{V}^*_{\mathsf{d}}(s_0)$, there is at most one policy $\pi \in \mathit{Incumbent}$ such that $\vec{V}^{\pi}(s_0) = \vec{V}^*$. $\mathbb{V}^*_{\mathsf{d}}(s_0)$ is found when the MIP becomes infeasible. Since the number of policies satisfying (1) and (2) is exactly $|\mathbb{V}^*_{\mathsf{d}}(s_0)|$, termination must happen after the claimed number of iterations.

The encoding is based on Fig. 3, dropping the policy-property constraints. The optimization function is a linear scalarization of all objectives, i.e., any choice of weights $\omega > 0$ such that $\sum_{i=1}^{n} \omega_{c_i} + \sum_{j=1}^{m} \omega_{G_j} = 1$. This suffices to guarantee (1), which can be shown straightforwardly by contraposition. The bulk of the MIP deals with ensuring progress in the sense of forbidding finding the same solutions again. This is accomplished through constraints (C27), which require improving over the previously computed policy in at least one objective. The selection of this improving objective is implemented via binary integer *wildcard* variables W^{π}_{O}, for each policy $\pi \in \mathit{Incumbent}$ and objective O. When set, $\mathsf{W}^{\pi}_{O} = 1$, the MIP solution may perform worse than π wrt. O. However, due to (C28), it is not possible to wildcard all objectives. In other words, each iteration finds a non-dominated $\hat{\pi}$ whose value vector $\vec{V}^{\hat{\pi}}(s_0)$ differs from all the previous ones; yielding property (2). There is a little caveat, however. To model strict inequality constraints, enforcing strict improvement in at least one objective, (C27) has to include a non-zero ε summand. Nevertheless:

Theorem 6. *There always exists* $\varepsilon \in (0, \infty)$ *such that the sketched algorithm terminates with* $\mathbb{V}^*_{\mathsf{d}}(s_0)$.

In practice, the exact value of ε is task specific and may be difficult to find. The ε parameter can be used to control the precision, and therewith size of the computed solution set, at the cost of losing the formal correctness guarantee.

As before, it is possible to leverage the principles of I-DUAL, solving each MIP in multiple iterations, while expanding the state space incrementally. Along the lines, and in addition to our adaptations from the previous section, this requires taking into account the fringe states' optimistic heuristic estimates in the ensure-progress constraints (C27); and, to foster finding non-dominated solutions, in the optimization function (C25). Fig. 4 illustrates the changes for I^2-DUAL's h^{pom} representation. Correctness follows with similar arguments as before.

6 Experimental Evaluation

The focus of our experiments is evaluating the feasibility of the proposed explanation architecture. Our implementation is based on Probabilistic Fast Downward (Helmert 2006; Steinmetz, Hoffmann, and Buffet 2016). The code is publicly available (Steinmetz et al. 2024). All experiments were run on a cluster with Intel Xeon E5-2695v4 CPUs, using runtime and memory limits of 30 minutes and 4 GB.

Setup We implemented Alg. 1 via MARCO (Liffiton et al. 2016), using MiniSAT 2.2 (Eén and Sörensson 2003) as the SAT solver. The LPs/MIPs were solved using CPLEX 22.11. We compare 6 PSS methods: *PSS-MIP* via the MIP encoding of PSS, either *(F)* building and solving the MIP over the full (reachable) state space directly, or *(I)* solving the MIP incrementally via our I-DUAL variant using as heuristic the state-of-the-art canonical PPDB heuristic over all patterns of size 2 (Klößner and Hoffmann 2021; Klößner et al. 2021), or *(I^2)* solving the MIP with our I^2-DUAL variant. The other three configurations *(MO)* compute up front the optimal MO value function, using either F, I, or I^2 for enumerating the non-dominated solution vectors. We set $\varepsilon = 0.05$. As a base reference, we also experimented with LP relaxations of the PSS-MIP methods, which compute the non-dominated exclusions for the space of stochastic policies. Applying LP relaxations to the MO methods is not possible as it would enumerate the optimal value vectors of stochastic policies, whose number is in general not bounded. Unfortunately, we cannot compare to other MOSSP solvers (e.g., Chen, Trevizan, and Thiébaux 2023) either, which due to the presence of 0-cost/reward cycles, do not support goal reachability objectives (cf. Kolobov et al. 2011; Steinmetz, Hoffmann, and Buffet 2016). As an additional reference, we include the results of a state-of-the-art MaxProb planner, based on heuristic search with iLAO* (Hansen and Zilberstein 2001) and FRET-π (Steinmetz, Hoffmann, and Buffet 2016) using a PPDB heuristic (Klößner et al. 2021).

Benchmarks Our benchmark set is based on existing and new PPDDL benchmarks. A benchmark instance here consists of a PPDDL task τ, a set of properties Ω, and a property subset $\Phi \subset \Omega$ for computing local explanations. The benchmark set is composed of three parts. We leave Φ empty, i.e., computing all MUSes, except in the first part:

- **OSP** Following Eifler et al. (2020a), we generate "OSP" variants of existing benchmarks from the IPPCs and other sources (Steinmetz, Hoffmann, and Buffet 2016; Klößner et al. 2021) that have more than one goal fact. We use Ω as the representation of "soft goals" and Φ to enforce the cost bound. Specifically, for each PPDDL base task τ, we generate for each combination of $\mathtt{P} \in \{0.6, 0.75, 0.9\}$ and $\mathtt{C} \in \{0.25, 0.5, 0.75\}$ a benchmark instance (1) making the goal of the task empty, (2) each original goal fact g becomes a reachability property ϕ_g with $G_{\phi_g} = \{g\}$ and $\alpha_{\phi_g} = \mathtt{P} \cdot P^{\mathcal{G}*}$ where $P^{\mathcal{G}*}$ is the MaxProb of τ; (3) $\Phi = \{\phi\}$ where ϕ is the cost property enforcing as bound $\beta_\phi = \mathtt{C} \cdot h^*$ on τ's cost function, where h^* is the minimal path cost required to reach τ's goal (J^* cannot be used as some benchmarks have no proper policy).

- **Search and Rescue** An adaptation of Trevizan, Thiébaux, and Haslum's (2017) CSSP benchmark. m cells of an n-by-n grid can possibly hold a victim. The agent must navigate through the grid in order to find, and as necessary, rescue the victims, while there are bounds on the total time and fuel consumption. We created 75 instances, randomly generating 5 instances for each combination of $n \in \{6, \dots, 10\}$ and $m \in \{2, 3, 4\}$. Ω includes the reachability properties ϕ_x requiring the victim from cell x being rescued with probability of 1; and for the time and fuel cost functions, the cost properties $\phi_{c,\mathtt{C}}$ with $\beta_{\phi_{c,\mathtt{C}}} = \mathtt{C} \cdot J^{c,*}$, where $J^{c,*}$ is the minimal expected cost under c of rescuing all victims, and $\mathtt{C} \in \{0.5, 0.75, 1.0, 1.25, 1.5\}$. Note that $\mathtt{C} > 1$ makes sense, because being optimal wrt. one cost function does not mean being optimal wrt. the other cost function.
- **Beluga** A use-case of an actual company. Incoming product components must be stashed in first-in-first-out storage racks, and taken out of the racks as requested by the production lines. There is uncertainty about the component requested next. Components can be moved between racks. Ideally, all production lines should be served while using as few racks as possible and moving as few components between the racks as possible. We generated 90 random instances with $\{3, 4, 5, 6\}$ incoming components, $\{2, 3\}$ available racks, and $\{2, 3\}$ production lines. We encode the three different preferences as properties: the probability $\mathtt{P}$ of successfully serving all production lines, considering $\mathtt{P} \in \{0.5, 0.6, \dots, 1.0\}$; a reachability property ϕ_n where G_{ϕ_n} encodes that n racks have not been used and $\alpha_{G_{\phi n}} = 1$; and bounds on component movements between racks, modeled as the cost property ϕ_m where $\beta_{\phi_m} = m$ and $m \in \{1, \dots, 5\}$.

Results The left-hand side of Table 1 displays the coverage results. Comparing the three MIP solving strategies, I^2 has a clear advantage in both PSS-MIP and MO algorithm variants. Although constructing the full MIP was actually often possible, F can solve only the very smallest instances, and typically timed out already during the first PSS call. The comparison between I and I^2 shows the advantage of I^2-DUAL's heuristic being able to simultaneously reason across all the property objectives. I^2 considered on average only half as many states as I. Exceptions are Elevators and Beluga where I^2-DUAL's heuristic caused a runtime overhead overshadowing that advantage. The PSS-MIP and MO methods perform overall similarly. However, in some domains (notably Se&Re and Beluga) the enumeration of $\vec{V}^*$ turned out infeasible due to too many non-dominated but almost indistinguishable value vectors. Comparing the runtime of PSS-MIP and MO using I^2-DUAL, MO is considerably slower almost throughout. It should be noted, however, that MO spends all this time on the computation of $\vec{V}^*$. Once computed, the generation of the explanation only takes a split second. This can become an advantage for computing multiple local explanations in a row, which we did not evaluate here. Compared to the references, the theoretical complexity results are partially reflected in the data. Overall, explanations could be computed in only a fraction of the instances feasible for MaxProb. However, PSS-MIP I^2's close performance to its LP-relaxed counterpart suggests that the reason of this discrepancy already lies in the secondary constraints and objectives rather than the original source of the complexity increase (deterministic policies).

	Coverage									Time (s)	
	Reference		PSS-MIP			MO				MIP	MO
	$P^{\mathcal{G}*}$	LP	F	I	I^2	F	I	I^2		I^2	I^2
Blocksw (135)	108	66	26	46	**50**	25	31	49		14.7	86.8
Elevators (135)	135	82	48	**72**	69	43	46	47		2.9	46.5
ExpBlock (126)	126	63	2	40	**50**	4	36	41		17.6	11.4
NoMyst (45)	36	19	0	**18**	**18**	0	13	13		2.1	12.5
Random (108)	72	68	0	46	56	3	52	**60**		33.5	20.7
Rovers (90)	90	74	22	**63**	**63**	26	40	44		30.0	46.0
Schedule (90)	45	53	18	27	51	18	27	**54**		8.6	2.6
TPP (90)	90	35	0	**26**	**26**	0	8	12		13.0	58.0
Zenotrav (63)	54	28	0	27	**28**	0	25	25		3.2	16.0
Se&Re (75)	75	67	30	33	**36**	0	0	0			
Beluga (90)	90	54	24	**55**	50	26	16	10		42.1	773.4

Table 1: Number of instances where the explanation was computed within the limits. Abbreviations as described in the text; reference values for solving MaxProb $P^{\mathcal{G}*}$ for the original goal $\mathcal{G}$, and the LP relaxation of PSS-MIP via I^2-DUAL. Time averaged over commonly solved instances.

7 Conclusion

In the presence of action-outcome uncertainty, characteristics of solution policies are defined by the expectation over the policies' executions. We introduced accordingly policy reachability and cost variants of Eifler et al.'s (2020a) plan properties. The analysis of mutual relationships between user-provided properties can comprehensibly summarize trade-offs in the infeasibly large space of all (global) or selected (local) policies. We showed that under the restriction to deterministic policies, this analysis is however computationally more difficult than the computation of a single solution policy. Our empirical results reflected to an extent this complexity, but also showed that proposed explanation methodology can be feasible. We introduced two algorithm variants identifying the properties' exclusion relationships. While analyzing property combinations individually tends to be more efficient in computing a single explanation, taking the detour via multi-objective optimization can have strengths in an interactive setting where a user might request a series of local explanations. The adaptation of multi-objective heuristics (Geißer et al. 2022) is a promising direction to improve scalability of the proposed methods.

Acknowledgments

This work was funded by the European Union's Horizon Europe Research and Innovation program under the grant agreement TUPLES No 101070149. Marcel Steinmetz, Florent Teichteil-Königsbuch and Sylvie Thiébaux would like to acknowledge the support of the Artificial and Natural Intelligence Toulouse Institute (ANITI), funded by the French Investing for the Future PIA3 program under the Grant agreement ANR-19-PI3A-000. Sylvie Thiébaux's work was also supported by Australian Research Council grant DP220103815.

References

Altman, E. 1999. *Constrained Markov Decision Processes*.

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS+ Planning. *Comput. Intell.*, 11: 625–656.

Bertsekas, D. P.; and Tsitsiklis, J. N. 1991. An Analysis of Stochastic Shortest Path Problems. *Math. Oper. Res.*, 16(3): 580–595.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artif. Intell.*, 69(1-2): 165–204.

Chen, D. Z.; Trevizan, F. W.; and Thiébaux, S. 2023. Heuristic Search for Multi-Objective Probabilistic Planning. In *Proc. of AAAI 2023*, 11945–11954.

Eén, N.; and Sörensson, N. 2003. An Extensible SAT-solver. In *Proc. of SAT 2003*, volume 2919, 502–518.

Eifler, R.; Brandao, M.; Coles, A. J.; Frank, J.; and Hoffmann, J. 2022. Evaluating Plan-Property Dependencies: A Web-Based Platform and User Study. In *Proc. of ICAPS 2022*, 687–691.

Eifler, R.; Cashmore, M.; Hoffmann, J.; Magazzeni, D.; and Steinmetz, M. 2020a. A New Approach to Plan-Space Explanation: Analyzing Plan-Property Dependencies in Oversubscription Planning. In *Proc. of AAAI 2020*, 9818–9826.

Eifler, R.; Steinmetz, M.; Torralba, Á.; and Hoffmann, J. 2020b. Plan-Space Explanation via Plan-Property Dependencies: Faster Algorithms & More Powerful Properties. In *Proc. of IJCAI 2020*, 4091–4097.

Feinberg, E. A. 2000. Constrained Discounted Markov Decision Processes and Hamiltonian Cycles. *Math. Oper. Res.*, 25(1): 130–140.

Galperin, H.; and Wigderson, A. 1983. Succinct Representations of Graphs. *Inf. Control.*, 56(3): 183–198.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman. ISBN 0-7167-1044-7.

Geißer, F.; Haslum, P.; Thiébaux, S.; and Trevizan, F. W. 2022. Admissible Heuristics for Multi-Objective Planning. In *Proc. of ICAPS 2022*, 100–109.

Giacomo, G. D.; and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *Proc. of IJCAI 2013*, 854–860.

Hansen, E. A.; and Zilberstein, S. 2001. LAO*: A heuristic search algorithm that finds solutions with loops. *Artif. Intell.*, 129(1-2): 35–62.

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.*, 26: 191–246.

Hong, S.; and Williams, B. C. 2023. An anytime algorithm for constrained stochastic shortest path problems with deterministic policies. *Artif. Intell.*, 316: 103846.

Juozapaitis, Z.; Koul, A.; Fern, A.; Erwig, M.; and Doshi-Velez, F. 2019. Explainable Reinforcement Learning via Reward Decomposition. In *Proceedings of the IJCAI/ECAI Workshop on Explainable Artificial Intelligence*.

Khan, O. Z.; Poupart, P.; and Black, J. P. 2009. Minimal Sufficient Explanations for Factored Markov Decision Processes. In *Proc. of ICAPS 2009*, 194 – 200.

Klößner, T.; and Hoffmann, J. 2021. Pattern Databases for Stochastic Shortest Path Problems. In *Proc. of SOCS 2021*, 131–135.

Klößner, T.; Hoffmann, J.; Steinmetz, M.; and Torralba, Á. 2021. Pattern Databases for Goal-Probability Maximization in Probabilistic Planning. In *Proc. of ICAPS 2021*, 201–209.

Kolobov, A.; Mausam; Weld, D. S.; and Geffner, H. 2011. Heuristic Search for Generalized Stochastic Shortest Path MDPs. In *Proc. of ICAPS 2011*, 130–137.

Krarup, B.; Krivic, S.; Magazzeni, D.; Long, D.; Cashmore, M.; and Smith, D. E. 2021. Contrastive Explanations of Plans through Model Restrictions. *J. Artif. Intell. Res.*, 72: 533–612.

Liffiton, M. H.; Previti, A.; Malik, A.; and Marques-Silva, J. 2016. Fast, flexible MUS enumeration. *Constraints An Int. J.*, 21(2): 223–250.

Littman, M. L.; Goldsmith, J.; and Mundhenk, M. 1998. The Computational Complexity of Probabilistic Planning. *J. Artif. Intell. Res.*, 9: 1–36.

Milani, S.; Topin, N.; Veloso, M.; and Fang, F. 2023. Explainable Reinforcement Learning: A Survey and Comparative Review. *ACM Comput. Surv.*

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267: 1–38.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley. ISBN 978-0-47161977-2.

Roijers, D. M.; and Whiteson, S. 2017. *Multi-Objective Decision Making*. Synthesis Lectures on Artificial Intelligence and Machine Learning.

Selvey, R.; Grastien, A.; and Thiébaux, S. 2023. Formal Explanations of Neural Network Policies for Planning. In *Proc. of IJCAI 2023*, 5446–5456. ijcai.org.

Steinmetz, M.; Hoffmann, J.; and Buffet, O. 2016. Goal Probability Analysis in Probabilistic Planning: Exploring and Enhancing the State of the Art. *J. Artif. Intell. Res.*, 57: 229–271.

Steinmetz, M.; Thiébaux, S.; Höller, D.; and Teichteil-Königsbuch, F. 2024. Code and Benchmarks for Explaining the Space of SSP Policies via Policy-Property Dependencies: Complexity, Algorithms, and Relation to Multi-Objective Planning. https://doi.org/10.5281/zenodo.10793823.

Topin, N.; and Veloso, M. 2019. Generation of Policy-Level Explanations for Reinforcement Learning. In *Proc. of AAAI 2019*, 2514–2521.

Trevizan, F. W.; Thiébaux, S.; and Haslum, P. 2017. Occupation Measure Heuristics for Probabilistic Planning. In *Proc. of ICAPS 2017*, 306–315.

Trevizan, F. W.; Thiébaux, S.; Santana, P. H.; and Williams, B. C. 2016. Heuristic Search in Dual Space for Constrained Stochastic Shortest Path Problems. In *Proc. of ICAPS 2016*, 326–334.