Action Policy Explanations in Oversubscription Planning

Aleena Siji,[1] Rebecca Eifler,[1] Daniel Fišer,[1] Jörg Hoffmann[1,2]

[1] Saarland University, Saarland Informatics Campus, Germany
[2] German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
alsi00002@stud.uni-saarland.de, eifler@cs.uni-saarland.de, danfis@cs.uni-saarland.de, hoffmann@cs.uni-saarland.de

Abstract

Oversubscription planning (OSP) is a planning formalism that can deal with planning scenarios where not all goals can be achieved. If the global optimization goal is not fixed, an iterative process in which users refine their preferences based on the sample plans is suitable. To help the users, the planning systems should be able to provide answers to questions such as "Why is goal g not satisfied by the sample plan?"

In this paper, we focus on explaining the behavior of a given black-box policy under the aforementioned planning scenario. That is, we assume that sample plans are provided by a state-dependent deterministic policy ϕ (in our case a neural network policy), and we try to automatically answer the question "Why is the goal g not satisfied by ϕ?" by providing information how much the policy ϕ would need to diverge from its decision in order to satisfy the goal g. Moreover, we also provide information in which state and how the policy should diverge. To this end, we extended action policies based on action schema networks to support OSP tasks, and design an algorithm that is able to provide the desired explanations. We evaluate the performance of the proposed algorithm and provide a case study with sample explanations.

1 Introduction

Action policies represented by (deep) neural networks (NN) are receiving increasing attention in recent years (Issakkimuthu, Fern, and Tadepalli 2018; Groshev et al. 2018; Toyer et al. 2018; Garg, Bajpai et al. 2019; Ståhlberg, Bonet, and Geffner 2022a,b). Those approaches target a planning setting where the objective is to reach a state which satisfies all goals while optimizing the number of actions needed. However, in real-life settings, we often have Oversubscription planning (OSP), this means not all goals can be achieved because of, for example, strict resource or timing constraints. Moreover, a global optimization objective is often not fully specified. As pointed out by Smith (2012), preferences for individual goals are not all given or fixed from the beginning. Thus an iterative process where users can refine their preferences based on the given sample plans is more suitable. In this setup, explanations elucidating the conflicts between the goals are beneficial to refine preferences and find better trade-offs.

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Eifler et al. (2020a) introduced a framework that provides such explanations based on minimal unsolvable goal subsets (MUGS). They answer contrastive questions like "Why is goal g_i not satisfied in plan π?" with "Because then you have to forgo goal g_j?". This provides the user with the information that g_i and g_i can not both be satisfied because of some resource or timing constraints.

Now, if the plan π is generated by a successive application of an action policy ϕ, then a new reason why g_i and g_j are not satisfied arises. It may not be possible, if the policy is strictly followed, to reach a state where both goals are satisfied. Thus the new question is "How much do we need to diverge from ϕ in order to satisfy both g_i and g_j?" Answering such questions clearly depends on how we decide to quantify divergence. Moreover, we want to minimize the divergence metric, because we consider diverging from the policy ϕ unfavorable. This is based on the assumption that the policy is tested (Eniser et al. 2022; Eisenhut et al. 2023) or even verified (Vinzent and Hoffmann 2022; Vinzent, Sharma, and Hoffmann 2023), and by diverging from ϕ you can not necessarily guarantee all properties.

We propose two distinct divergence metrics. The first metric counts the number of steps in which we need to diverge from the policy decision in order to satisfy the goals, i.e., we count how many times do we need to take a different action than the policy. The second metric takes a more local view. In each state, it considers a subset of applicable actions most preferable by the policy (assuming the policy is internally ranking the applicable actions somehow).

Explanations are then based on the MUGS induces by plans less than a certain threshold distance from the policy. Those capture the goal trade-offs we need to make unless we are willing to diverge more.

So far, there is no approach for training action policies targeting OSP. Thus, as another contribution, we focus on action schema networks (ASNet) (Toyer et al. 2018, 2020) previously used for probabilistic and deterministic action policies generalizing over domains, and extend them for the OSP setting.

Eifler et al. (2020b) introduced an algorithm based on an exhaustive state space search to compute all MUGS for a given planning task. We extend this approach to compute all MUGS within a given radius. Thereby local and global distinct divergence metrics are handled as path-independent

and path dependent pruning function respectively.

A preliminary performance analysis is performed for ASNet policies and the distance metric based on the number of deviating actions. Sample explanations are illustrated with a case study of 3 different resource constraint domains

2 Background

We consider a variant of the oversubscription planning (OSP) (Smith 2004; Domshlak and Mirkis 2015) with binary-domain variables (Bäckström and Nebel 1995; Helmert 2009). An **OSP task** is a tuple $\tau = (F, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ where F is a set of **facts**, A is a set of **actions**, and $c : A \to \mathbb{R}_0^+$ is an action **cost** function. A **state** $s \subseteq F$ is a set of facts, and S denotes the set of all states. $I \subseteq F$ is the **initial state**, $G^{\text{hard}} \subseteq F$ and $G^{\text{soft}} \subseteq F$ denote **hard** and **soft** goal, respectively, and it holds that $G^{\text{hard}} \cap G^{\text{soft}} = \emptyset$. $b \in \mathbb{R}_0^+$ is the **cost bound**.

Each action $a \in A$ has a **precondition** $pre_a \subseteq F$, an **add list** $add_a \subseteq F$ and a **delete list** $del_a \subseteq F$ such that $add_a \cap del_a = \emptyset$ and $pre_a \cap add_a = \emptyset$. An action a is **applicable** in a state s if $pre_a \subseteq s$. The set of all applicable actions in s is denoted by $A(s)$. Applying a to s, denoted as $s[\![a]\!]$, results in the state $s[\![a]\!] = (s \cup add_a) \setminus del_a$. A sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$ is **applicable** in a state s_0 if there are states $s_1, \ldots, s_n$ such that a_i is applicable in s_{i-1} and $s_i = s_{i-1}[\![a_i]\!]$ for $1 \leq i \leq n$. The resulting state of this application is $s_0[\![\pi]\!] = s_n$. The sequence $\langle s_0, s_1, \ldots, s_n \rangle$ of the aforementioned states is called **intermediate state sequence** of π. The cost of the sequence of actions π is defined as $c(\pi) = \sum_{i=1}^{n} c(a_i)$, and the length of π is denoted as $|\pi| = n$. Moreover, given $i \in \{1, \ldots, n\}$, $\pi[i] = a_i$ denotes the i-th action of the action sequence π. Given i, j, s.t. $1 \leq i \leq j \leq n$, $\pi[i, j] = \langle a_i, \ldots a_j \rangle$ denotes the specified sub-sequence of actions.

A sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$ is called **plan** if $c(\pi) \leq b$ and $G^{\text{hard}} \subseteq I[\![\pi]\!]$, i.e., the costs of plans in OSP tasks are upper-bounded by b and they must reach hard goals. Given the plan π, $G^{\text{true}}(\pi) = \{g \mid g \in G^{\text{soft}}, g \in I[\![\pi]\!]\}$ denotes the soft goals satisfied by π, and $G^{\text{false}}(\pi) = G^{\text{soft}} \setminus G^{\text{true}}(\pi)$ denotes the soft goals not satisfied by π. Π denotes the set of all plans.

Following Eifler et al. (2020a), we do not define a plan utility over G^{soft}. Instead, we focus on the analysis of the conflicts between different goals from G^{soft}. Given a task τ and a set of plans $\Pi' \subseteq \Pi$, the conflicts between soft goals within the plans Π' are represented by the **minimal unsolvable goal subsets** (MUGS). A set of soft goals $C \subseteq G^{\text{soft}}$ is called a MUGS if there is no plan $\pi \in \Pi'$ such that $C \in I[\![\pi]\!]$, but for all $C' \subsetneq C$ there exists such a $\pi \in \Pi'$. The set of all MUGS within the set of plans Π' is denoted by $\text{MUGS}(\Pi')$.

We consider deterministic state-dependent policies. A **policy** $\phi \colon S \mapsto A \cup \{\emptyset\}$ is a function mapping states to applicable actions or null ($\emptyset$) if there is no applicable action, i.e., for every state $s \in S$ and policy ϕ it holds that $\phi(s) = \emptyset$ if $A(s) = \emptyset$ and $\phi(s) \in A(s)$ otherwise. Given a policy ϕ and a state s, the **execution** of ϕ in s, denoted by $\sigma^\phi(s)$, is a sequence of actions $\sigma^\phi(s) = \langle a_1, \ldots, a_n \rangle$ applicable in s (with its intermediate state sequence $\langle s_0, \ldots, s_n \rangle$) such that

(i) for every $i \in \{1, \ldots, n\}$ it holds that $a_i = \phi(s_{i-1})$, and

(ii) for every $i \in \{0, \ldots, n-1\}$ it holds that if s_i is a goal state, then s_n is also a goal state and $|G^{\text{soft}} \cap s_i| < |G^{\text{soft}} \cap s_n|$, and

(iii) for every $i, j \in \{0, \ldots, n-1\}$ s.t. $i \neq j$ it holds that $s_i \neq s_j$.

In other words, the execution of ϕ in s is the action sequence resulting from iteratively applying the policy ϕ starting in the state s in such a way that the execution terminates either when there is no applicable action, or when the first goal state maximazing the number of soft goals is reached, or when the policy loops, i.e., it reaches some state for the second time. Clearly, $\sigma^\phi(s)$ is finite and unique for every s and ϕ. We also, w.l.o.g., consider executions of policies limited to a certain number of actions.

3 Action Policy Explanations

In practice, a global optimization function for user preferences is often not available. Therefore, an iterative planning approach where users refine their objectives based on example plans is more suitable (Smith 2012).

In this paper, we follow the framework implementing such an iterative process for OSP tasks proposed by Eifler et al. (2020a): Given an OSP task τ with hard goals G^{hard} and soft goals G^{soft}, each iteration is conducted as follows. First, the user selects a subset $G^{\text{enf}} \subseteq G^{\text{soft}}$ of soft goals that is enforced. Then, the system finds a plan π for τ such that (at least) G^{enf} is satisfied, i.e., $G^{\text{enf}} \subseteq G^{\text{true}}(\pi)$. At this point, users can ask questions of the form "Why is Q not satisfied in π?", where $Q \subseteq G^{\text{false}}(\pi)$ can be any set of soft goals not satisfied by π. The answer provided by the system is then of the form "Because then you have to forgo $A(Q)$", where $A(Q)$ is the set of all possible minimal subsets of soft goals satisfied by π that cannot be satisfied anymore by plans satisfying Q. These questions and answers help users find better trade-offs between soft goals that are conflicting with each other as shown by a user study (Eifler et al. 2022).

Here, we extend this framework to plans provided by a given deterministic policy ϕ. That is, we assume we are given not only an OSP task τ with soft goals G^{soft}, but also a deterministic policy ϕ for τ that can solve τ, i.e., we assume $\sigma^\phi(I)$ is a plan. Since the policy is deterministic and therefore able to produce only a single plan, users cannot enforce any soft goals. However, they can still ask questions of a similar form as before: "Why is Q not satisfied by the policy ϕ?", where $Q \subseteq G^{\text{false}}(\sigma^\phi(I))$ can be any subset of soft goals not satisfied by the policy's execution. Now, we cannot answer this question in the same way as before, because users cannot enforce any soft goals or choose a different plan. We can, however, provide users with information on how much would the policy need to change in order to reach the soft goals Q. Such an answer, obviously, depends on how we decide to measure the amount of change.

We propose to deal with this problem by defining a *policy distance function* that maps each plan to a numerical value

expressing its distance from the given deterministic policy, i.e., it quantifies how much the given plan *diverges* from the policy. With such a distance function at hand, users can ask questions of the form "Is it possible to satisfy Q within the radius k of the policy ϕ?", where $Q \subseteq G^{\text{false}}(\sigma^\phi(I))$ is as before, and $k \in \mathbb{R}^+$ is a threshold on the policy distance function, i.e. the question is asking whether there exists a plan such that its distance from the policy ϕ is at most k. We can answer this question by either "No, it is not possible", or "Yes, but you have to forego $A(Q,k)$" where $A(Q,k)$ is, similarly to before, a set of all minimal subsets of soft goals satisfied by the policy but not satisfied by plans within the radius k satisfying Q. We could even provide information in which states it is necessary to diverge from the policy and how, but we will touch on this question only partially here.

Moreover, we think this approach establishes a solid basis for different sets of questions and answers that we plan to investigate in the future. For example, instead of selecting a specific radius k, users could simply ask what is the minimum radius to achieve Q. Or we could use more policy distance functions at once and users could ask for different radii for each distance function. Also note that, for this setting, it would make sense to train a policy that can take both a state and a set of target soft goals instead of just a state. Such policy would be able to adapt its decisions based on the set of soft goals that ought to be satisfied. This would allow users to enforce soft goals as was done in the framework of Eifler et al. (2020a). It would also make sense from the application point of view, but it would require a different type of explanations as we would need to reason more carefully about what a divergence in this setting means—we would not have a single state-dependent deterministic policy anymore.

In the following, we formally introduce the general framework for our approach, and then we instantiate it with two different policy distance functions providing reasonable ways to quantify divergence from a policy.

3.1 General Framework for Policy Divergence Explanations

Our explanation framework requires measuring the distance between a plan and a policy in order to be able to quantify how much the plan diverges from the policy. So, we start by the definition of a *policy distance function* as any function mapping plans to non-negative numbers such that it returns zero for the plan resulting from the execution of the given deterministic policy. The policy distance function then naturally induces a set of plans whose distance from the policy is within the given threshold value which we call *radius*.

Definition 1 (Policy Distance Function)**.** *Given an OSP task τ with the initial state I and plans Π, and a policy ϕ for τ such that $\sigma^\phi(I) \in \Pi$ is a plan, a function $\delta : \Pi \mapsto \mathbb{R}^+$ mapping plans to non-negative numbers is called a **policy distance function for** ϕ if $\delta(\sigma^\phi(I)) = 0$.*

*Given a number $k \in \mathbb{R}^+$, a policy ϕ for τ such that $\sigma^\phi(I) \in \Pi$, and a policy distance function δ for ϕ, $\Pi(\delta, k)$ denotes the **plans within the radius** k defined as $\Pi(\delta,k) = \{\pi \in \Pi \mid \delta(\pi) \leq k\}$.*

Note that our policy distance function is related to the distance functions used in the context of diverse planning (e.g., Boddy et al. 2005; Goldman and Kuter 2015; Katz and Sohrabi 2020). Diverse planning requires measuring the distance between two plans and the goal is usually to produce a set of plans whose pairwise distance is maximal (possibly within some other constraints). The difference here is that we compute the distance from a plan to the fixed policy. Investigation of whether the distance metrics used in diverse planning can be adopted (and how) to our setting remains for future research.

Now, we can define explanations based on analyzing plans diverging from the given policy. We focus on questions of the form "Is it possible to satisfy Q within the radius k of the policy ϕ?" and answers of the form either "No, it is not possible." or "Yes, but you have to forego $A(Q,k)$" where $A(Q,k)$ concisely lists which soft goals reached by the policy cannot be reached together with Q.

Definition 2 (Policy Divergence Explanation)**.** *Let τ denote an OSP task with plans Π, let ϕ denote a policy for τ such that $\sigma^\phi(I) \in \Pi$, and let δ denote a policy distance function for ϕ. Given a **question** (Q,k) where $Q \subseteq G^{\text{false}}(\sigma^\phi(I))$ and $k \in \mathbb{R}^+$, the **answer** is defined as $A(Q,k) = \min_\subseteq\{C \setminus Q \mid C \in \text{MUGS}(\Pi(\delta,k)), C \subseteq Q \cup G^{\text{true}}(\sigma^\phi(I))\}$ where $\min_\subseteq \mathcal{S}$ denotes the inclusion-wise minimal set of elements from $\mathcal{S}$ such that for every $s \in \mathcal{S}$ there exists $s' \in \min_\subseteq \mathcal{S}$ such that $s' \subseteq s$.*

Next, we instantiate the policy distance function with two specific distance functions. The first one focuses on the local behavior of the policy and the second one on global behavior.

3.2 Confidence Distance Function

Although we assume a deterministic state-dependent policy ϕ, here we also assume that the policy is internally assigning a numerical confidence value to each action in every state, and we have access to these confidence values. Formally, we assume we have a *confidence function* $\phi^\star : S \times A \mapsto [0,1]$ mapping each state and action to a value between zero and one, and for every state $s \in S$ it holds that $\sum_{a \in A} \phi^\star(s,a) = \sum_{a \in A(s)} \phi^\star(s,a) = 1$, i.e., it returns zero for every inapplicable action and the sum of confidence over applicable actions is always one in any given state. Moreover, we also assume the confidence function $\phi^\star$ relates to the policy ϕ so that the policy ϕ deterministically selects the action with the highest confidence, i.e., $\phi(s) = \arg\max_{a \in A(s)} \phi^\star(s,a)$ for every $s \in S$ (ties can be broken in any way, but deterministically).

Now we can define the *confidence distance function* as the maximum difference between the policy's confidence in the action selected by the policy (i.e., the highest confidence value) and the policy's confidence in the action actually used in the given plan.

Definition 3 (Confidence Distance Function)**.** *Given an OSP task τ and a policy ϕ for τ with the confidence function $\phi^\star$, the **confidence distance function** δ_C **for** ϕ is defined as*

$$\delta_C(\pi) = \max_{i \in \{1,\dots,n\}} \{\phi^*(s_{i-1}, \phi(s_{i-1})) - \phi^*(s_{i-1}, a_i)\}$$

for every plan $\pi = \langle a_1, \ldots, a_n \rangle$ *with its intermediate state sequence* $\langle s_0, s_1, \ldots, s_n \rangle$.

Clearly, the confidence distance function for ϕ is a policy distance function for ϕ because its value for the plan induced by the policy is zero. It is easy to see that meaningful radii for the confidence distance function lie between zero and one. Since we want to use this distance function in explanations, it is important for us that the radius value has an intuitive interpretation that users can understand and utilize. The radius k for the confidence distance function expresses that the system is allowed to consider only the plans containing actions that differ by at most k in terms of the policy's confidence in the action chosen by the policy.

In Figure 1 an example of the reachable states within $\Pi(\delta_C, 0.1)$ is given. The policy provides a plan that satisfies g_2 and g_3. If the user asks: "Can g_1 be satisfied with a radius of 0.1?" the answer would be: ""Yes, but you have to give up g_3 to satisfy g_1." Here the policy is relatively confident except for the alternative leading to $\{g_1, g_2\}$. If the user would be willing to diverge further, there would open a possibility to forgo g_2 to satisfy g_1.

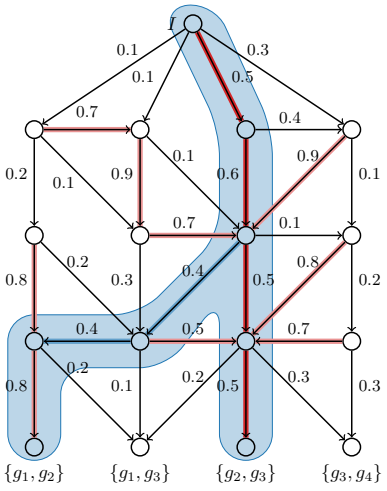


Figure 1: Example action probability radius; $G^{\text{soft}} = \{g_1, g_2, g_3, g_4\}$; top: Initial state; bottom: goal states; $MUGS(\Pi) = \{\{g_1, g_2, g_3\}, \{g_1, g_4\}, \{g_2, g_4\}\}$; red: action selected by the policy ϕ; number at actions: confidence assigned by ϕ^*; bright red: π_ϕ; blue: states reachable within $\Pi(\delta_C, 0.1)$; $MUGS(\Pi(\delta_C, 0.1)) = \{\{g_1, g_3\}, \{g_1, g_4\}, \{g_2, g_4\}\}$

Diverging based on the confidence of the policy makes sense if a lack of confidence is reasonable with respect to the task. For example in tasks with a lot of symmetries i.e. tasks with many ways to achieve a goal. Furthermore, in OSP tasks there are states in which one commits to a certain set of soft goals depending on the action. If all achievable subsets have the same utility, it makes sense that this is reflected in the confidence of the policy. However, depending on the strategy and the training data, the policy may not include these decision points. Instead, the policy commits to solving the task in a particular way for a particular set of soft goals. In this case, it makes more sense to consider not the confidence, but the absolute number of applied actions not following the policy.

3.3 Diverging Actions Distance Function

For measuring the overall distance between a plan and the policy from the "global" perspective, we propose to simply count the number of actions that differ from what the policy would choose.

Definition 4 (Diverging Actions Distance Function)**.** *Given an OSP task* τ *and a policy* ϕ *for* τ*, the* ***diverging actions distance function*** $\delta_\#$ ***for*** ϕ *is defined as*

$$\delta_\#(\pi) = \sum_{i \in \{1,\ldots,n\}} [\phi(s_{i-1}) = a_i]$$

for every plan $\pi = \langle a_1, \ldots, a_n \rangle$ *with its intermediate state sequence* $\langle s_0, s_1, \ldots, s_n \rangle$*, where* $[\phi(s) = a]$ *denotes the characteristic function of equality predicate, i.e.,* $[\phi(s) = a]$ *is* 1 *if* $\phi(s) = a$ *and it is* 0 *otherwise.*

Reasonable values of radius k for the diverging actions distance are natural numbers. Setting k to zero only considers the single plan returned by the policy. Any k higher than zero expresses that the system is allowed to consider plans that differ in at most k actions from the policy.

An example for the reachable states within $\Pi(\delta_\#, 1)$ is given in Figure 2. The policy again provides a plan that satisfies g_2 and g_3, but if the user asks: "Can g_1 be satisfied with a radius of 1?" the answer is, "Yes, but you have to give up g_2 to satisfy g_1". This time the alternative to satisfy $\{g_1, g_3\}$ is within the radius, although the policy is less confident about this path. However, you only have to diverge once while for reaching $\{g_1, g_2\}$ you have to diverge twice.

Diverging without considering the policy's confidence can be considered dangerous, because one might choose very bad actions intentionally assigned very low confidence. Also, because the training data is based on states sampled from the policies ranking, following actions with low confidence, can lead to parts of the state space not well covered during training.

3.4 MUGS Computation for a given Radius

To compute $MUGS(\Pi(\delta, k))$ we adapt the algorithm to compute $MUGS(\Pi)$ introduced by (Eifler et al. 2020b). They perform an exhaustive state space exploration while keeping track of the *maximal solvable goal subsets* (MSGS) reachable. Those are then used to compute the MUGS. To only explore the states reachable within radius k for a given distance function δ we use δ as a *pruning function*.

Thereby we differentiate between local and global distance functions. For a local distance function, like δ_C, it is possible to decide whether applying action a state s exceeds the thresholds just based on a and s. Thus it can be implemented as a state-dependent pruning function. For a global distance function, like $\delta_\#$, you also need the information how s was reached, hence the implementation is a path-dependent pruning function.

To decrease the search space size Eifler at al. (2020b) use an underapproximation of the reachable soft goals to

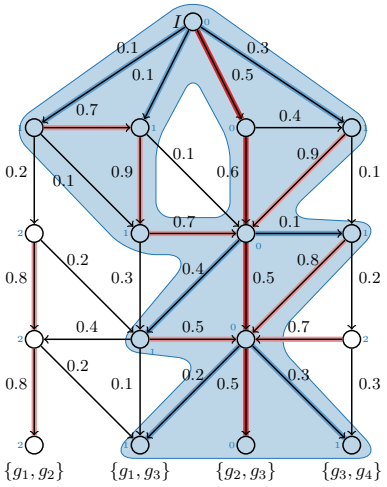


Figure 2: Example diverging action radius; $G^{\text{soft}} = \{g_1, g_2, g_3, g_4\}$; top: initial state; bottom: goal states; $MUGS(\Pi) = \{\{g_1, g_2, g_3\}, \{g_1, g_4\}, \{g_2, g_4\}\}$; red: action selected by the policy ϕ; number at actions: confidence assigned by ϕ^*; bright red: π_ϕ; blue: states reachable within $\Pi(\delta_\#, 1)$; $MUGS(\Pi(\delta_\#, 1)) = \{\{g_1, g_2\}, \{g_1, g_4\}, \{g_2, g_4\}\}$

prune states from which no superset of any set of the current MSGS is reachable. This approach is also applicable here and can be used alongside the pruning based on the policy distance function.

4 Action Policies for OSP

To the best of our knowledge, there is no published work on training action policies for OSP. So, in order to be able to evaluate our explanation framework, we decided to train new action policies for OSP using Action Schema Networks (ASNets) (Toyer et al. 2018, 2020). ASNets policies can generalize over a domain by training on a subset of relatively small tasks from the domain that can be solved quickly by some reference solver. This solver, called *trainer*, is repeatedly called on different states during the training process and the ASNets policy is trained by imitating the trainer. The details of the internal structure of ASNets or the training procedure are not important here (we refer interested readers to the works of Toyer et al. (2018, 2020)), because the only part that we changed in order to obtain some workable policy for our evaluation was that we used an OSP solver as a trainer. Other than that we used the simplest configuration of ASNets for deterministic policies (trained without a heuristic function) referred to as "ASNets (no h.)" by Toyer et al. (2020).

As the trainer, we used an optimal OSP planner (Eifler et al. 2020b) which, for each state, finds a plan maximizing the number of satisfied soft goals. We trained the policy on three OSP domains with resource constraints: NoMystery, TPP (Nakhost, Hoffmann, and Müller 2012) and a version of Blocksworld with two hands and limited resources for using them. To measure the quality of the resulting policy ϕ, we limit the policy execution to thousand steps, and, for each task, we select the goal state s_g reached by the policy execution such that the number satisfied soft goals in s_g is maximal among all reached goal states.

For each domain, Blocksworld, NoMystery, and TPP, we obtained a policy that was able to reach a goal state with the maximum possible number of satisfied soft goals in 30%, 10%, and 30% of tasks, respectively. Moreover, the policy satisfied almost 80% of the maximum possible number of satisfiable soft goals over all tasks in Blocksworld, 67% in NoMystery, and 56% in TPP. We think that such quality of OSP action policies is sufficient to conduct an experimental evaluation of our explanation framework.

5 Evaluation of Action Policy Explanations

We implemented the MUGS computation for both distance functions in Fast Downward (Helmert 2006) using a re-implementation of ASNet in C as a library to evaluate the policies. We are using the same benchmark set as for the policy evaluation containing the domains: Blocksworld, NoMystery, and TPP. with no hard goals and $6-9$, $4-9$ and $4-6$ soft goals respectively. The experiments are run on a cluster of Intel Xeon CPU E5-2660 with 2.20GHz with a memory limit of 4GB and a timeout of 30min.

Since ASNets provide very confident policies, where choosing a reasonable radius k was not possible, the preliminary results are limited to the diverging actions distance function. The radius is scaled from $k = 0$ up to $k = 10$ diverging actions.

In Figure 3 the fraction of instances where the MUGS could be be computed is depicted.

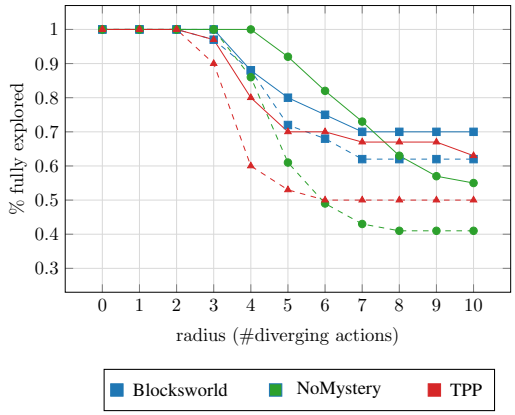


Figure 3: x-axis: radius k i.e maximal number of diverging actions; y-axis: fraction of instances where the MUGS could be computed; dashed pruning based on an underapproximation of reachable soft goals, solid without pruning.

For NoMystery and TPP using pruning based on an underapproximation of reachable soft goals increased the fraction of terminating instances on average by 20%, for Blocksworld only by 10%. While for small distances, up to 3, it is feasible to explore all states within the radius, for larger distances the policy evaluation is the bottleneck. For

a radius larger than 3 on average more than 90% of the run time is used for the policy evaluation.

The number of states contained in each radius for the instances commonly solved for all radius sizes is depicted in Figure 4. In the beginning, the number of states increases exponentially. After radius size 4-6 depending on the domain the increase levels off. However, for Blocksworld and No-mystery, the search space size for $k = \infty$ is far from being reached. This shows that for the policy a considerable part of the search space is unpromising and can only be reached by considerable deviations.

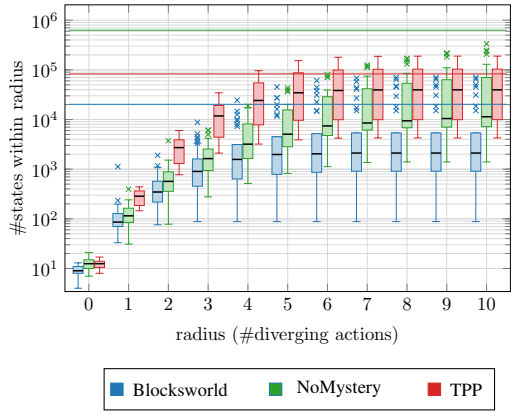


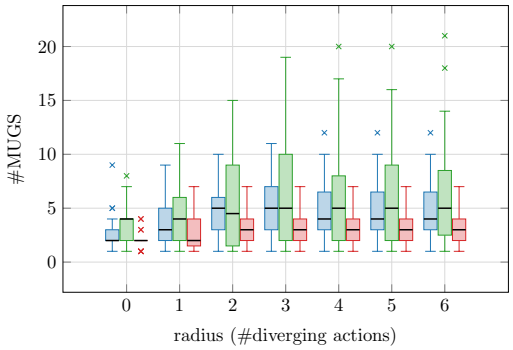
Figure 4: x-axis: threshold k i.e maximal number of diverging actions; y-axis: number of states reachable within $\Pi(\delta, k)$; horizontal lines: average number of states for $k = \infty$.

The number and average size of MUGS depending on the radius are depicted in Figure 5. For Blocksworld the number of MUGS and their size increases up to radius 3 and then the number decreases again. This nicely shows that with a larger radius, more trade-off options involving more soft goals are available. For even larger radii more soft goals are reachable decreasing the number of MUGS again. For NoMystery there is a similar picture with a less prominent decrease in the number of MUGS. For TPP the number of MUGS only increases. This is because the MUGS are linked to which buying order of goods are possible. There is an example in the case study in the next section which illustrates this behavior.

5.1 Case Study

In the following, we take a closer look at one example from each domain.

Blocksworld In Blocksworld one has to move blocks from an initial to a goal configuration. Here we are using a version with two hands where each action costs 1 energy unit. Each hand starts with 6 units of energy. The initial and goal configuration are given in Figure 6. The policy first puts all blocks on the table. Then it decides to pick up block 5 to satisfy the soft goal $on(5, 3)$. Within a distance of size 1 there are two alternative soft goals $on(3, 2)$ and $on(0, 5)$ reachable, depending on which block is picked up next.



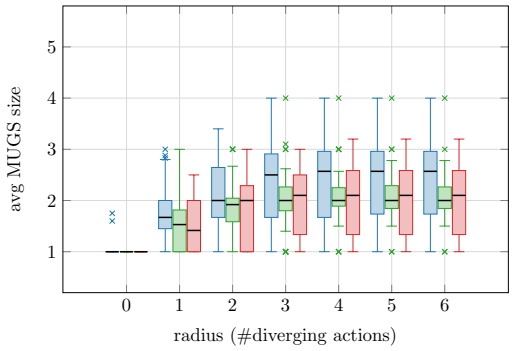


Figure 5: x-axis: threshold k i.e maximal number of diverging actions; y-axis: top number and bottom average size of MUGS.

This leads to : $MUGS(\Pi(\delta_{\#}, 1)) = \{\{on(0, 5), on(3, 2)\}, \{on(0, 5), on(5, 3)\}, \{on(3, 2), on(5, 3)\}\}$. So the answer to the question "Can I also stack block 0 onto block 5?" is "Within a distance of 1 from the policy you have to forgo $on(5, 3)$ if you want $on(0, 5)$."

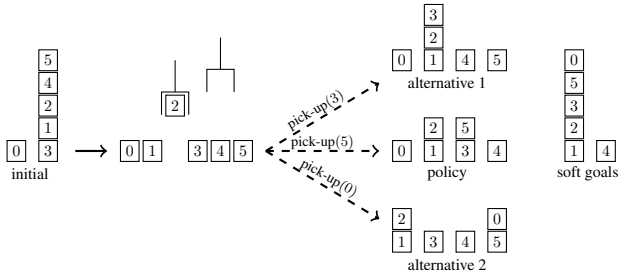


Figure 6: Blocksworld example: Policy execution and possible diverging actions.

NoMystery In NoMystery you have to deliver packages by transporting them with trucks between locations. Here we have two trucks each with 6 units of fuel. The road map is given in Figure 7. The policy provides the plan given in Figure 8 on the left, delivering packages P_1 and P_3.

Now if the user asks "Can I deliver package P_2 if I diverge once from the policy?" Unfortunately, the answer

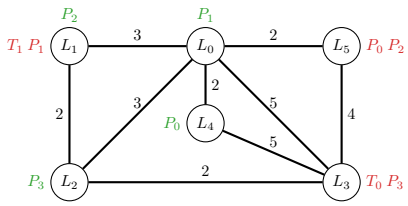


Figure 7: Road map of NoMystery example: initial locations of packages and trucks in red; goal location in green; numbers at edges represent the fuel consumption.

$load(P_1, T_1, L_1)$	$drive(T_0, L_3, L_5, 6 \to 2)$
$drive(T_1, L_1, L_2, 6 \to 4)$	$load(P_0, T_0, L_5)$
$drive(T_1, L_2, L_0, 4 \to 1)$	$load(P_2, T_0, L_5)$
$load(P_3, T_0, L_3)$	$drive(T_0, L_5, L_0, 2 \to 0)$
$drive(T_0, L_3, L_2, 6 \to 4)$	$load(P_1, T_1, L_1)$
$unload(P_3, T_0, L_2)$	$unload(P_2, T_0, L_0)$
$drive(T_0, L_2, L_1, 4 \to 2)$	$drive(T_1, L_1, L_0, 6 \to 3)$
$drive(T_0, L_1, L_2, 2 \to 0)$	$load(P_2, T_1, L_0)$
$unload(P_1, T_1, L_0)$	$unload(P_1, T_1, L_0)$
	$drive(T_1, L_0, L_1, 2 \to 0)$
	$unload(P_2, T_1, L_1)$

Figure 8: NoMystery Example plans: left: plan provided by the policy; right: plan with 3 diverging actions (red) to deliver P_2 instead of P_3.

is "No". You have to diverge at least 3 times to allow for P_2 to be delivered. Then you have to forgo P_3, based on $MUGS(\Pi(\delta_{\#}, 1)) = \{\{at(P_0, L_4), at(P_2, L_1)\}, \{at(P_0, L_4), at(P_3, L_2)\}, \{at(P_2, L_1), at(P_3, L_2)\}\}$. The corresponding plan is depicted in Figure 8 on the right. To deliver P_2 the trucks have to work together. This does not seem to be represented by the policy, hence the large distance.

TPP In TPP you have to buy goods at a market and then use trucks to transport the goods to the depot. Here we have 2 trucks and 4 goods and the map depicted in Figure 9. Buying goods and driving consumes the same resource, initially at 6 for both trucks. The plan the policy provides, given in Figure 10 on the left, delivers goods G_1 and G_3 to the depot.

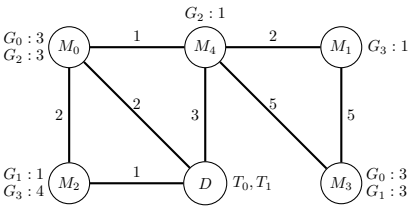


Figure 9: Road map of TPP example: each market is labeled with the available goods and their price; numbers at edges represent the fuel consumption.

For a distance of 1, we get $MUGS(\Pi(\delta_{\#}, 1)) = \{\{stored(G_0), stored(G_3)\}, \{stored(G_1), stored(G_2)\}, \{stored(G_2), stored(G_3)\}\}$. So the question "Why is the good G_2 not stored?" is answered with "Because then you have to forgo G_1 and G_3". The plan for G_2 is given in Figure 10 on the right. Here you only need to diverge once, because after buying G_2 the policy adds the only other possible good G_0 and then delivers both to the depot, without needing to diverge further.

$buy(T_0, G_3, M_2, 6 \to 2)$	$buy(T_1, G_2, M_4, 6 \to 5)$
$buy(T_1, G_2, M_4, 6 \to 5)$	$buy(T_0, G_2, M_0, 6 \to 3)$
$buy(T_1, G_1, M_2, 5 \to 4)$	$buy(T_0, G_0, M_0, 3 \to 0)$
$buy(T_1, G_3, M_1, 4 \to 3)$	$drive(T_1, D, M_2, 5 \to 4)$
$buy(T_1, G_2, M_0, 3 \to 0)$	$drive(T_1, M_2, M_0, 4 \to 2)$
$drive(T_0, D, M_2, 2 \to 1)$	$load(G_2, T_1, M_0)$
$load(G_3, T_0, M_2)$	$load(G_0, T_1, M_0)$
$load(G_1, T_0, M_2)$	$drive(T_1, M_0, D, 2 \to 0)$
$drive(T_0, M_2, D, 1 \to 0)$	$unload(G_2, T_1, D)$
$unload(G_3, T_0, D)$	$unload(G_0, T_1, D)$
$unload(G_1, T_0, D)$	

Figure 10: TPP Example plans: left: plan provided by the policy; right: plan with 1 diverging action (red) to store G_2 (and G_0) instead of G_1 and G_3.

6 Conclusion & Future Work

We presented explanations for the question "Why is the goal g not met by policy ϕ?" by pointing to the trade-offs that must be made if one is not to deviate further than a certain distance from ϕ. We introduced two distance measures based on a threshold on the confidence of the policy and the number of diverging actions. Our preliminary results show that the computation is feasible for small distances. The case study shows that the resulting explanations can give interesting insights into the policy's behavior.

To further evaluate our explanations we want to address different types of action policies next considering the framework of (Ståhlberg, Bonet, and Geffner 2022b). They provide policies where the confidence distance function is applicable.

Currently, the underapproximation of reachable soft goals in the state space exploration is only based on the model. It does not incorporate reachability within the policy. However, precisely such approximations can be obtained by linking to policy verification (Vinzent and Hoffmann 2022), analyzing the reachability of goal conditions. We will explore policy abstractions accounting for the radius to be used as a pruning function.

So far we are only computing the MUGS for one given distance function δ and threshold k. However, a natural next question would be "How much do you have to diverge from policy ϕ for a specific MUGS C to disappear?". To answer this questions the minimal threshold k_{min} where $C \notin MUGS(\Pi_{|\delta^{\phi} k_{min}})$, needs to be computed. The trivial approach is to iteratively increase k and to use the algorithm introduced in Section 3.4 to compute the MUGS. This has the overhead of generating the same search space multiple times. Instead iteratively increasing one search space based

on increasing radius and tracking the MUGS until C disappears would be more efficient.

Once the MUGS for a given policy are identified and the user's preferences over the soft goals became clearer, retraining of the policy might be necessary to reflect those preferences. In this case, the data computed for the explanations could be used to guide the retraining.

Finally, to determine how useful the introduced policy explanations are, we plan to conduct a user study.

Acknowledgments

This material is based upon work supported by the German Research Foundation (DFG) under grant 389792660 as part of TRR 248 (see https://perspicuous-computing.science) and by the European Union's Horizon Europe Research and Innovation program under the grant agreement TUPLES No 101070149 (see https://tuples.ai/).

References

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS^+ Planning. *Computational Intelligence*, 11(4): 625–655.

Boddy, M.; Gohde, J.; Haigh, T.; and Harp, S. 2005. Course of Action Generation for Cyber Security Using Classical Planning. In *ICAPS*.

Domshlak, C.; and Mirkis, V. 2015. Deterministic Oversubscription Planning as Heuristic Search: Abstractions and Reformulations. *JAIR*, 52: 97–169.

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

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 *AAAI*.

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

Eisenhut, J.; Alvaro, T.; Maria, C.; and Hoffmann, J. 2023. Automatic Metamorphic Test Oracles for Action-Policy Testing. In *ICAPS*.

Eniser, H. F.; Gros, T. P.; Wüstholz, V.; Hoffmann, J.; and Christakis, M. 2022. Metamorphic relations via relaxations: An approach to obtain oracles for action-policy testing. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, 52–63.

Garg, S.; Bajpai, A.; et al. 2019. Size independent neural transfer for rddl planning. In *ICAPS*.

Goldman, R. P.; and Kuter, U. 2015. Measuring Plan Diversity: Pathologies in Existing Approaches and a New Plan Distance Metric. In *AAAI*, 3275–3282.

Groshev, E.; Goldstein, M.; Tamar, A.; Srivastava, S.; and Abbeel, P. 2018. Learning generalized reactive policies using deep neural networks. In *ICAPS*.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M. 2009. Concise Finite-Domain Representations for PDDL Planning Tasks. *AI*, 173: 503–535.

Issakkimuthu, M.; Fern, A.; and Tadepalli, P. 2018. Training deep reactive policies for probabilistic planning problems. In *ICAPS*.

Katz, M.; and Sohrabi, S. 2020. Reshaping Diverse Planning. In *AAAI*, 9892–9899.

Nakhost, H.; Hoffmann, J.; and Müller, M. 2012. Resource-Constrained Planning: A Monte Carlo Random Walk Approach. In *Proc. ICAPS*, 181–189.

Smith, D. 2012. Planning as an Iterative Process. In *AAAI*, 2180–2185.

Smith, D. E. 2004. Choosing Objectives in Over-Subscription Planning. In *ICAPS*, 393–401.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022a. Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits. In *ICAPS*.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022b. Learning generalized policies without supervision using gnns. *arXiv preprint arXiv:2205.06002*.

Toyer, S.; Thiébaux, S.; Trevizan, F. W.; and Xie, L. 2020. ASNets: Deep Learning for Generalised Planning. *Journal of Artificial Intelligence Research*, 68: 1–68.

Toyer, S.; Trevizan, F.; Thiebaux, S.; and Xie, L. 2018. Action Schema Networks: Generalised Policies with Deep Learning. In *AAAI*.

Vinzent, M.; and Hoffmann, J. 2022. Neural Policy Verification via Predicate Abstraction: CEGAR.

Vinzent, M.; Sharma, S.; and Hoffmann, J. 2023. Neural Policy Safety Verification via Predicate Abstraction: CEGAR.