Iterative Oversubscription Planning with Goal-Conflict Explanations: Scaling Up Through Policy-Guidance Approximation

Rebecca Eifler[a,*], **Daniel Fišer**[b], **Aleena Siji**[b] **and Jörg Hoffmann**[b,c]

[a]LAAS-CNRS, Université de Toulouse, France
[b]Saarland University, Saarland Informatics Campus, Germany
[c]German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany

Abstract. In oversubscription planning (OSP), not all goals can be achieved. If a global optimization objective is difficult to fix, then an iterative planning process in which users refine their objective based on sample plans is suitable. Recent work has shown that, in such a process, explanations of plan trade-offs based on goal conflicts – minimal unsolvable goal subsets (MUGS) – are useful. A fundamental limitation of this approach is scalability. Computing MUGS is feasible only in relatively small planning instances; sometimes plan generation in iterative planning also is a limiting factor as users tend to be impatient. Here we address both these limitations by restricting the space of plans considered. We assume that an action policy π for the OSP task has been learned. We restrict both plan generation and MUGS analysis to the action sequences within a given radius r around π, so that r controls the tradeoff between scalability and the degree of approximation. We instantiate this idea with two different kinds of radii around a policy. We experimentally analyze performance as a function of r, for Action Schema Network policies. The results confirm that our approach can scale up further than prior work, and results on instances small enough to compute MUGS exactly indicate that we obtain informative MUGS even with limited runtime and memory.

1 Introduction

Oversubscription planning (OSP) models problems where due to budget constraints not all goals can be achieved, and the task is instead to achieve a maximum-utility subset of goals [e.g., 21, 3, 17]. As pointed out by Smith [20] the goal utilities in this setting may be difficult to elicitate, calling for an iterative planning process that suggests plan candidates for human inspection. Eifler et al. [4, 5, 6] show that contrastive explanations of plan trade-offs are useful in this context, answering user questions like “Why does the plan p you suggest not satisfy goal g_i?” with goal conflicts like “Because achieving g_i would necessitate to forego goal g_j”. To provide these answers, Eifler et al. compute minimal unsolvable goal subsets (MUGS).

A fundamental limitation of this approach is scalability. Computing MUGS is feasible only in relatively small planning instances (about the scale of optimal OSP planning [5]). Sometimes plan generation in iterative planning also is a limiting factor, as users tend to be impatient and ideally the next sample plan should become available instantaneously.

Given the underlying problem complexities, there is no way around these limitations in general other than by approximation. Here we explore a particular form of approximation leveraging learned domain knowledge. We assume that an action policy π for the OSP task has been learned (either a deterministic policy mapping states to actions, or a probabilistic policy mapping states to probability distributions over applicable actions). We use π to restrict the space of plans considered in plan generation and MUGS analysis. We consider only those action sequences P_r within a given “radius” r around π (concrete instances of this will be discussed below).

As P_r is a subset of action sequences, our approximation is pessimistic. Any plan in P_r is a plan in the task, and any goal subset unsolvable in the task is unsolvable in P_r. Due to the latter, any MUGS in P_r is a subset of a MUGS in the task, so that our approximate explanations err in one direction only.

The radius r is an algorithm parameter that controls the tradeoff between scalability and the degree of approximation. It permits to leverage progress on action-policy learning, in particular through neural networks [e.g., 15, 12, 10, 24, 22]. At the same time, with $r > 0$ we are not limited to the policy itself but have a stronger ability to find plans, and therewith a more informative view of which goal subsets can (or cannot) be achieved.

The user questions and answers in our setup remain the same as before, the only difference being that the answers are now subject to the restriction to P_r: “Why does the plan p you suggest not satisfy goal g_i?” “Because achieving g_i would necessitate to forego goal g_j, or to diverge from π by more than r”. A new option for the user here, based on the second part of the answer, is to increase the radius r to see whether that fixes the goal conflict (at the price of longer waiting times for plans and answers).

We instantiate our approach with two different kinds of radii around a policy. The first type of radius, *diverging actions distance*, restricts the number of times in which we diverge from the deterministic policy decision, to at most r steps. The second type of radius, *action-probability distance*, takes a more local view. Assuming we have a probabilistic policy π assigning some probability to applicable actions in each state, we consider within each step the subset of applicable actions whose probability differs at most r from the maximum probability assigned by π. To compute MUGS in P_r, we augment

* Corresponding Author. Email: rebecca.eifler@laas.fr

the most competitive algorithm by Eifler et al. [5] (an explicit-state search) with additional state-pruning functions.

As the empirical basis for our experiments, since there has been no dedicated work on policy learning for OSP as yet, we use classical-planning policies trained on domains without cost bounds. We specifically experiment with ASNets [23, 24], a competitive method for learning action policies that generalize across instances within a PDDL domain. We run experiments on OSP benchmarks from 11 domains used in prior works evaluating ASNets policies [8, 9] and one new domain. We experiment with a) plan generation and b) MUGS computation as a function of the radius r. The results show that our approach can improve a), and scales up b) far beyond the range of exact MUGS computation. Results on instances small enough to compute MUGS exactly indicate that our approximation yields informative MUGS even for relatively small radii and hence with limited runtime.

2 Background

We consider a STRIPS variant of the oversubscription planning (OSP) [21, 3]. An **OSP task** is a tuple $\tau = \langle \mathcal{F}, \mathcal{A}, c, I, G^{\text{hard}}, G^{\text{soft}}, b\rangle$ where $\mathcal{F}$ is a set of **facts**, $\mathcal{A}$ is a set of **actions**, and $c : \mathcal{A} \to \mathbb{R}_0^+$ is an action **cost** function. A **state** $s \subseteq \mathcal{F}$ is a set of facts, and $\mathcal{S}$ denotes the set of all states. $I \subseteq \mathcal{F}$ is the **initial state**, $G^{\text{hard}} \subseteq \mathcal{F}$ and $G^{\text{soft}} \subseteq \mathcal{F}$ 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 $a \in \mathcal{A}$ has a **precondition** $pre_a \subseteq \mathcal{F}$, an **add list** $add_a \subseteq \mathcal{F}$ and a **delete list** $del_a \subseteq \mathcal{F}$ 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 $\mathcal{A}(s)$. Applying applicable action a to state s, denoted as $s[\![a]\!]$, results in the state $s[\![a]\!] = (s \cup add_a) \setminus del_a$. A sequence of actions $p = \langle a_1, \dots, a_n\rangle$ is **applicable** in a state s_0 if there are states $s_1, \dots, s_n$ such that a_i is applicable in s_{i-1} and $s_i = s_{i-1}[\![a_i]\!]$ for $i \in \{1, \dots, n\}$. The resulting state of this application is $s_0[\![p]\!] = s_n$, and the cost of p is defined as $c(p) = \sum_{i=1}^{n} c(a_i)$. The sequence $\langle s_0, s_1, \dots, s_n\rangle$ of the aforementioned states is called **intermediate state sequence** of p. Given a sequence of actions $p = \langle a_1, \dots, a_n\rangle$, a sub-sequence $\langle a_1, \dots, a_i\rangle$ for any $i \leq n$ is called a **prefix** of p.

A sequence of actions $p = \langle a_1, \dots, a_n\rangle$ is called **plan** if p is applicable in the initial state, $c(p) \leq b$ and $G^{\text{hard}} \subseteq I[\![p]\!]$, i.e., the costs of plans in OSP tasks are upper-bounded by b and they must reach all hard goals. Given a plan p, $G^{\text{true}}(p) = G^{\text{soft}} \cap I[\![p]\!]$ denotes the soft goals satisfied by p, and $G^{\text{false}}(p) = G^{\text{soft}} \setminus G^{\text{true}}(p)$ denotes the soft goals not satisfied by p. $\mathcal{Q}$ denotes the set of all sequences of actions, and $\mathcal{P} \subseteq \mathcal{Q}$ denotes the set of all plans. We also call a set of facts $X \subseteq \mathcal{F}$ reachable if there exists a sequence of actions $p \in \mathcal{Q}$ applicable in I such that $X \subseteq I[\![p]\!]$. And we call a subset of soft goals $G \subseteq G^{\text{soft}}$ **solvable** if there exists a plan $p \in \mathcal{P}$ such that $G \subseteq I[\![p]\!]$.

We consider probabilistic state-dependent policies. A **policy** $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ maps each state and action to a value between zero and one, and for every state $s \in \mathcal{S}$ it holds that $\sum_{a \in \mathcal{A}} \pi(s, a) = \sum_{a \in \mathcal{A}(s)} \pi(s, a) = 1$, i.e., it returns zero for every inapplicable action and the sum over applicable actions is always one in any given state. We also consider deterministic policies induced by π that always select the action with the highest probability: Given a policy π, $\hat{\pi} : \mathcal{S} \mapsto \mathcal{A}$ denotes the deterministic policy defined as $\hat{\pi}(s) = \arg\max_{a \in \mathcal{A}(s)} \pi(s, a)$ for every state $s \in \mathcal{S}$ (where we assume $\arg\max$ breaks ties deterministically in each state). We also call $\hat{\pi}$ a **determinization** of π. In this work, we focus on policies that are learned once for a whole (PDDL) domain, and then re-used for every task instance from the domain, i.e., the learning effort is amortized over all instances that can be addressed.

Following Eifler et al. [4], we do not define a plan utility over G^{soft}, but we focus on the analysis of the conflicts between different goals from G^{soft} using so-called minimal unsolvable goal subsets (MUGS). Eifler et al. defined a MUGS as a set of soft goals that cannot be achieved by any plan, but all of its subsets can. Here, we intend to restrict the state space by considering a vicinity of the given policy only. Therefore, in contrast to their work, we consider minimal unsolvable goal subsets with respect to a subset of possible plans:

Given a task τ and a subset of plans $\mathcal{P}' \subseteq \mathcal{P}$, a set of soft goals $C \subseteq G^{\text{soft}}$ is called a **minimal unsolvable goal set** (MUGS) for $\mathcal{P}'$ if $C \not\subseteq I[\![p]\!]$ for every plan $p \in \mathcal{P}'$, but for every $C' \subsetneq C$ there exists a plan $p \in \mathcal{P}'$ such that $C' \subseteq I[\![p]\!]$. The set of all MUGS for the set of plans $\mathcal{P}' \subseteq \mathcal{P}$ is denoted by $\mathcal{G}^{\text{MUGS}}(\mathcal{P}')$. We also call MUGS $\mathcal{G}^{\text{MUGS}}(\mathcal{P})$ over all plans **exact** MUGS, and $\mathcal{G}^{\text{MUGS}}(\mathcal{P}')$ for $\mathcal{P}' \subseteq \mathcal{P}$ **approximate** MUGS.

We build on the framework implementing an iterative explanation process for OSP tasks proposed by Eifler et al. [4] that uses MUGS to represent conflicts between soft goals. Given an OSP task τ with hard goals G^{hard} and soft goals G^{soft}, each iteration is conducted as follows.

(*1*) The user selects a subset $G^{\text{enf}} \subseteq G^{\text{soft}}$ of soft goals that they want to enforce.

(*2*) The system tries to find a plan $p \in \mathcal{P}$ for τ such that (at least) G^{enf} is satisfied, i.e., $G^{\text{enf}} \subseteq G^{\text{true}}(p)$. If no plan is found, the system reports it to the user and the process moves back to (1) allowing the user to select a different G^{enf} or abort the process.

(*3*) The user can ask questions of the form "Why is Q not satisfied in p?", where $Q \subseteq G^{\text{false}}(p)$ can be any subset of soft goals not satisfied by p.

(*4*) 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 p that cannot be satisfied any more by plans satisfying Q.

Formally the answer $A(Q)$ is defined as follows.

Definition 1 (Explanation). *Let τ denote an OSP task with plans $\mathcal{P}$. Given a plan $p \in \mathcal{P}$ and **question** $Q \subseteq G^{\text{false}}(p)$, the **answer** is defined as $A(Q) = \min_{\subseteq}\{C \setminus Q \mid C \in \mathcal{G}^{\text{MUGS}}(\mathcal{P}), C \subseteq Q \cup G^{\text{true}}(p)\}$ where $\min_{\subseteq} X$ denotes the inclusion-wise minimal set of elements from X such that for every $s \in X$ there exists $s' \in \min_{\subseteq} X$ such that $s' \subseteq s$.*

3 Approximate MUGS

For iterative planning with conflict explanations it is necessary to provide sample plans without long waiting times and to compute conflicts between soft goals. To compute the sample plans, Eifler et al. [4] use a satisficing planner that tries to solve the task $\tau' = \langle \mathcal{F}, \mathcal{A}, c, I, G^{\text{hard}} \cup G^{\text{enf}}, G^{\text{soft}} \setminus G^{\text{enf}}, b\rangle$. If τ' is solvable then the planner provides a plan p where $G^{\text{hard}} \cup G^{\text{enf}} \subseteq I[\![p]\!]$ and otherwise no plan is provided. To answer any potential user question they precompute $\mathcal{G}^{\text{MUGS}}(\mathcal{P})$ using an exhaustive state space search with branch-and-bound style pruning. This computation strongly limits the size of the feasible tasks.

Here, we propose to address larger tasks, where computing $\mathcal{G}^{\text{MUGS}}(\mathcal{P})$ exactly is infeasible, and where finding a plan can be time-consuming. We leverage learned problem knowledge to provide approximations by using a learned action policy π for the task. This knowledge is then used to restrict the state space in which we search for plans. We do so, by defining a *policy distance function* that maps each sequence of actions to a numerical value expressing its distance from the given policy, i.e., it quantifies how much the given plan *diverges* from the policy. With such a distance function δ at hand, we only consider plans P_r^δ whose distance from the policy π is within a *radius* of at most $r \in \mathbb{R}^+$ according to the distance function δ. In other words, we restrict the whole iterative process to consider only plans P_r^δ.

The radius allows going between a fast plan generation strictly following the policy π (in our case its determinization $\hat{\pi}$), and obtaining possibly cheaper plans by diverging from the policy (thus exploring a larger portion of the state space). Moreover, increasing the radius may also allow to remedy the situation where the policy does not find any plan, but there might be a plan in the vicinity of the policy. In this setting, MUGS for P_r^δ are simply approximations of MUGS for all plans $\mathcal{P}$.

Formally, a *policy distance function* is any function mapping sequences of actions to non-negative numbers that is monotonically non-decreasing along every sequence of actions. 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 2 (Policy Distance Function). *Let τ denote an OSP task with action sequences $\mathcal{Q}$, and let π denote a policy for τ.*

A function $\delta : \mathcal{Q} \mapsto \mathbb{R}_0^+$ mapping sequences of actions to non-negative numbers is called a ***policy distance function*** *for π if for every action sequence $p \in \mathcal{Q}$ and every prefix p' of p it holds that $\delta(p') \leq \delta(p)$.*

Given a number $r \in \mathbb{R}^+ \cup \{\infty\}$ and a policy distance function δ for π, P_r^δ denotes the ***plans within the radius*** *r defined as $P_r^\delta = \{p \in \mathcal{P} \mid \delta(p) \leq r\}$.*

We define distance functions to be monotonically non-decreasing along every action sequence because we want to avoid situations where a sequence of actions has a smaller distance from the policy than its prefix. It is easy to see that $P_\infty^\delta = \mathcal{P}$.

Although not necessary, it is reasonable to design the policy distance function so that it returns zero when strictly following the policy. In the next section, we introduce two policy distance functions that assign zero to action sequences strictly following the determinization $\hat{\pi}$ of the given policy π, i.e., they assign zero to action sequences resulting from selecting the action with the highest probability in each state. In these cases, P_0^δ contains all plans achieved by executing $\hat{\pi}$, but P_0^δ can also contain other plans, because the distance function δ is allowed to assign zero to plans that diverge from $\hat{\pi}$. When this is the case and why this behavior is intended is discussed in the next section.

The radius r can be used to phase between computational effort and accuracy, because $P_r^\delta \subseteq P_{r'}^\delta$ whenever $r \leq r'$. In other words, expressiveness of MUGS monotonically increases with increasing radius r, because the larger the radius becomes, the more alternative plans are considered and thus more soft goal subsets become solvable, and therefore MUGS cover more facts.

Proposition 1 (Monotonicity). *Let $\tau = (\mathcal{F}, \mathcal{A}, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ denote an OSP task such that $G^{\text{hard}} \cup G^{\text{soft}}$ is not reachable, let π denote a policy for τ, let δ denote a policy distance function for π, and let $r, r' \in \mathbb{R}^+$ be two radii such that $r \leq r'$. Then for every $C \in \mathcal{G}^{\text{MUGS}}(P_r^\delta)$ there exists $C' \in \mathcal{G}^{\text{MUGS}}(P_{r'}^\delta)$ such that $C \subseteq C'$.*

Proof. Since $G^{\text{hard}} \cup G^{\text{soft}}$ is unreachable, $\mathcal{G}^{\text{MUGS}}(P_{r'}^\delta)$ is non-empty. Let $C \in \mathcal{G}^{\text{MUGS}}(P_r^\delta)$. So we have that for every $G \subsetneq C$ there is $p \in P_r^\delta$ such that $G \subseteq I[\![p]\!]$. Since $r \leq r'$ it follows that $P_r^\delta \subseteq P_{r'}^\delta$ and therefore either $C \in \mathcal{G}^{\text{MUGS}}(P_{r'}^\delta)$ or C is reachable by some plan from $P_{r'}^\delta$ and therefore there must be $C' \in \mathcal{G}^{\text{MUGS}}(P_{r'}^\delta)$ such that $C \subseteq C'$ because $\mathcal{G}^{\text{MUGS}}(P_{r'}^\delta)$ is non-empty. □

Note that, with $P_\infty^\delta = \mathcal{P}$ in particular, this shows that the MUGS for any radius are subsets of the task's MUGS.

Now, we can define approximate explanations restricted to the part of the state space within a given radius of a chosen policy distance function.

Definition 3 (Approximate Explanation). *Let τ denote an OSP task, let π denote a policy for τ, and let δ denote a policy distance function for π. Given a radius $r \in \mathbb{R}^+$ and a plan $p \in P_r^\delta$ and a* ***question*** *$Q \subseteq G^{\text{false}}(p)$, the* ***answer*** *is defined as $A(Q, r) = \min_\subseteq \{C \setminus Q \mid C \in \mathcal{G}^{\text{MUGS}}(P_r^\delta), C \subseteq Q \cup G^{\text{true}}(p)\}$.*

Restricting the reachable part of the state space to the policy radius may result in MUGS indicating conflicts that actually do not exist when considering the whole, unrestricted, state space. How accurate the MUGS approximation is depends on the quality of the policy and on the concrete distance function used. Two possible implementations of policy distance functions are discussed next.

4 Policy Distance Functions

A policy distance function δ for a policy π, quantifies how far an action sequence p diverges from the policy. A distinction can be made between focusing on local policy behavior and global behavior. In the following, we define one distance function for each.

4.1 Action-Probability Distance Function

We start with the *action-probability distance function* that we define as the maximum difference between the highest probability assigned by π to any action and the probability of the action actually used in the given sequence of actions.

Definition 4 (Action-Probability Distance Function). *Given an OSP task τ and a policy π for τ, the* ***action-probability distance function*** *$\delta_P : \mathcal{Q} \mapsto \mathbb{R}^+$ for π is defined as*

$$\delta_P(p) = \max_{i \in \{1,\dots,n\}} \big(\max_{a \in \mathcal{A}(s_{i-1})} (\pi(s_{i-1}, a)) - \pi(s_{i-1}, a_i) \big)$$

for every sequence of actions $p = \langle a_1, \dots, a_n \rangle$ with its intermediate state sequence $\langle s_0, s_1, \dots, s_n \rangle$.

Clearly, the action-probability distance function for π is a policy distance function for π because its value for any action sequence p cannot be smaller than its value for any prefix of p. Meaningful radii for the action-probability distance function lie between zero and one. Whether a sequence of actions is within a probability radius is a local decision. It depends individually on each state s and $\pi(s, a)$ over $a \in \mathcal{A}(s)$.

It is also easy to see that δ_P assigns zero to action sequences resulting from strictly following the determinization $\hat{\pi}$ of π, i.e., $P_0^{\delta_P}$ contains plans resulting from selecting the action with the highest probability in every state. However, if there are states assigning the same highest probability to multiple actions, then $P_0^{\delta_P}$ contains also sequences of actions that diverge from $\hat{\pi}$. We will look more closely on the ramifications of this property when we experimentally evaluate this policy distance function.

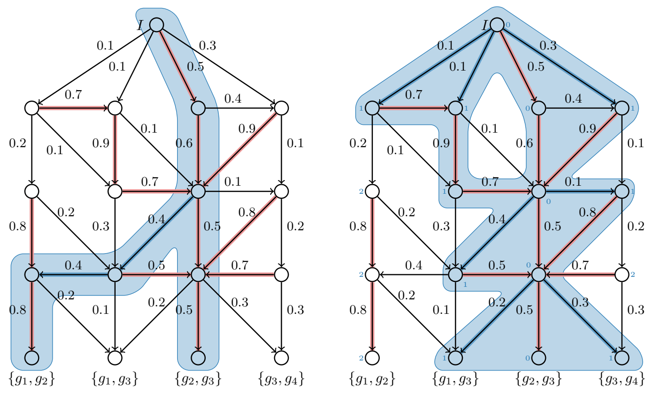
Figure 1. Example for action-probability (left) and diverging action (right) radius function; $G^{\text{soft}} = \{g_1, g_2, g_3, g_4\}$, top: initial state, bottom: goal states, edges correspond to actions and labels are $\pi(s, a)$; $\mathcal{G}^{\text{MUGS}}(\mathcal{P}) = \{\{g_1, g_2, g_3\}, \{g_1, g_4\}, \{g_2, g_4\}\}$; red: action with maximal probability assigned by π, blue: states reachable within $P_{0.1}^{\delta_P}$ (left) and $P_1^{\delta_\#}$ (right).

Figure 1 shows on the left an example state space and policy π. It depicts which portion of the state space would be considered when using an action-probability distance function δ_P with the radius $r = 0.1$. The example policy results in $\mathcal{G}^{\text{MUGS}}(P_0^{\delta_P}) = \{\{g_1\}, \{g_4\}\}$ for radius $r = 0$. Increasing the radius to $r = 0.1$ includes one additional plan, resulting in $\mathcal{G}^{\text{MUGS}}(P_{0.1}^{\delta_P}) = \{\{g_1, g_3\}, \{g_4\}\}$. Note that MUGS for the unrestricted task is $\mathcal{G}^{\text{MUGS}}(\mathcal{P}) = \{\{g_1, g_2, g_3\}, \{g_1, g_4\}, \{g_2, g_4\}\}$.

Diverging based on the probability distribution over actions makes most sense if the policy itself is not sure which action to select in which state, i.e., if the policy assigns a similar (high) probabilities to multiple applicable actions. This behaviour may be caused not only by the policy being uninformative (e.g., insufficiently trained), but it also may be that it does not matter which of the actions with high probability is selected as all of them lead to a goal state. Next, we consider a global distance measure that counts the absolute number of used actions not following $\hat{\pi}$.

4.2 *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 strictly following $\hat{\pi}$.

Definition 5 (Diverging Actions Distance Function)**.** *Given an OSP task τ and a policy π for τ, the **diverging actions distance function** $\delta_\# : \mathcal{Q} \mapsto \mathbb{R}^+$ for π is defined as*

$$\delta_\#(p) = \sum_{i \in \{1,\dots,n\}} [\hat{\pi}(s_{i-1}) \neq a_i]$$

for every sequence of actions $p = \langle a_1, \dots, a_n \rangle$ with its intermediate state sequence $\langle s_0, s_1, \dots, s_n \rangle$, where $[a' \neq a]$ denotes the characteristic function of inequality predicate, i.e., $[a' \neq a]$ is 1 if $a' \neq a$ and it is 0 otherwise.

Reasonable values of radii r for the diverging actions distance function are natural numbers. Setting r to zero only considers plans induced by $\hat{\pi}$. Any r higher than zero expresses that the system is allowed to consider plans that differ in at most r actions from $\hat{\pi}$.

Figure 1 depicts on the right the same example state space and policy as on the left, but this time we show the part of the state space within radius 1 with $\delta_\#$. By following the deterministic policy $\hat{\pi}$ (i.e., with radius $r = 0$), we would get $\mathcal{G}^{\text{MUGS}}(P_0^{\delta_\#}) = \{\{g_1\}, \{g_4\}\}$, and diverging at most once would lead to $\mathcal{G}^{\text{MUGS}}(P_1^{\delta_\#}) = \{\{g_1, g_2\}, \{g_1, g_4\}, \{g_2, g_4\}\}$. In this example, a diverging action distance of 1 includes more plans and is therefore a more accurate approximation than using action-probability distance with radius 0.1. However, this comes at the cost of exploring a larger part of the state space. A large branching factor of the task may be detrimental for diverging actions distance function even with a small radius.

We remark that our concept of policy distance function is related to other distance measures. Plan distance functions are used in the context of diverse planning [e.g., 2, 11, 16] to measure plan diversity, which is maximized within a set of output plans. Our objective here is very different. We measure not the distance between pairs of plans but between plans and a fixed policy. For goal recognition, Amado et al. [1] introduce different distance measures between an observation (i.e., a plan prefix) and a policy. These are then used to identify the intended goal based on the policy that minimizes the distance. Ideas from both approaches may be applicable in our context, and exploring this remains a topic for future work.

5 Experiments

The experiments were conducted on a cluster with Intel E5-2695v4@2.1GHz processors and 4 GB memory limit per process. We conduct our experiments for a range of radii values. For the action-probability distance function δ_P, we scale from 0 to 1 in steps of 0.1. Clearly, the largest radius $r = 1$ is equivalent to the full exploration. For the diverging actions distance function $\delta_\#$, we scale from 0 to 10 in steps of 1.

Policy We use our own implementation of ASNets policies [23, 24] using the DyNet library [18], which was trained with two propositional layers, 16 output channels, batch size of 64, 700 training steps within an epoch, at most 300 training epochs and time limit of 8 hours per domain. We used Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$) with learning rate 10^{-3}, dropout rate 0.1 and l_2 regularizer of $2 \cdot 10^{-4}$. Our variant of ASNets does not use landmarks or action history, i.e., we use the simplest variant of ASNets described by Toyer et al. [24]. Note that each policy is trained only once per domain and then used for all tasks from the domain.

Policy Radius Search The implementation of the search algorithms is based on Fast Downward [13]. To compute *approximate MUGS* $\mathcal{G}^{\text{MUGS}}(P_r^{\delta})$ we adapt the algorithm introduced by Eifler et al. [5] to compute *exact MUGS* $\mathcal{G}^{\text{MUGS}}(\mathcal{P})$. It performs an exhaustive state space exploration while keeping track of the *maximal solvable goal subsets* (MSGS) that are reachable. Those are then used to compute the MUGS. The exploration of the state space restricted to the policy radius is implemented as a simple pruning function. Given a policy π, a policy distance function δ for π, and a radius r, we prune state s reached by a sequence of actions p whenever $\delta(p) > r$.

Moreover, since we run experiments with time (and state) limits and we know that the state space restricted to a radius r only grows with increasing r, we use an iterative method where we increase the radius step by step until the time (or state) limit is reached. That is, given sequence of increasing radii $r_1 < r_2 < \cdots$, state s reached by a sequence of actions p is considered in iteration i only if $\delta(p) \leq r_i$, and if $\delta(p) > r_i$, then $\langle p, s \rangle$ is stored for subsequent iterations. When we explore the whole state space within the radius r_i and there is some time (or state) budget left, we set the radius to r_{i+1} and

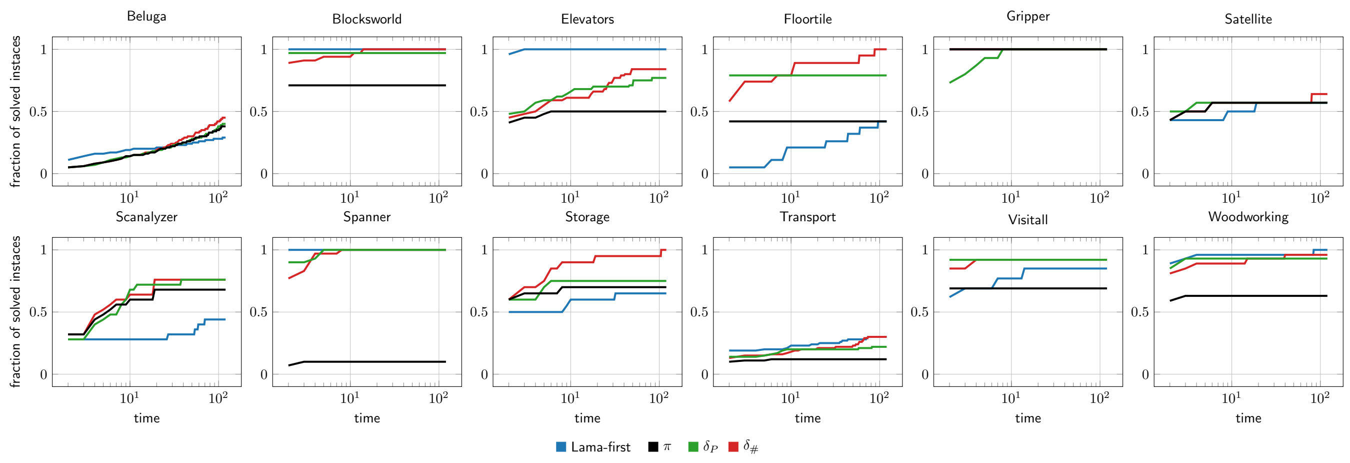

Figure 2. Fraction of solved instances over time. blue: LAMA-first without pruning, black: execution of $\hat{\pi}$, green: Lama-first with pruning with action-probability distance function, red: Lama-first with pruning with diverging action distance function.

continue the search from the stored pairs $\langle p, s\rangle$ s.t. $\delta(p) \leq r_{i+1}$. Note that this approach can be applied to any forward search algorithm.

To decrease the search space size, Eifler et al. [5] used underapproximation of the reachable soft goals to prune states from which no superset of any set of the current MSGS is reachable. This approach was also applied here as it can be used alongside the pruning function based on policy distance functions.

Benchmarks We use 11 domains from the standard benchmark set (see the list in Figure 2) that were already used in prior works evaluating ASNets policies [8, 9]. We also used a new "Beluga" domain. It models a part of the logistics problem involved in aircraft manufacturing at Airbus for which a user survey showed that goal-conflict explanations are useful. The objective is to unload a Beluga transportation aircraft, and transport the unloaded parts via a system of storage racks to the factory in a required order. The largest instances in our benchmark set have a realistic problem size for which an exact MUGS computation is not feasible.

Since ASNets do not support action costs, we treat all domains as unit cost. To increase the number of tasks we can evaluate, we did not remove the tasks used for training ASNets. This means that the fraction of evaluated tasks that were used also for training is 0% in Beluga, Blocksworld and Spanner, 25% in Elevators, 7% in Floortile, 13% in Gripper, 31% in Satellite, 20% in Scanalyzer, 30% in Storage, 7% in Transport, 84% in Visitall, and 38% in Woodworking.

To obtain OSP variants of the tasks, we use the same approach as Katz et al. [17]: We generate OSP tasks from the non-OSP ones by adding cost bounds of 25%, 50% and 75% of the cost of the best known plans (either from http://planning.domains or using the best plan found by the LAMA planner [19] within 60 minutes), and we consider all goals as soft goals. In most domains, the cost bound of 25% resulted in tasks that are too easy to provide a valuable comparison between exact and approximate computation. So, we include here only 50% and 75% cost-bounds. Only tasks with 32 or fewer goals are included, as our implementation does not support more. The dataset is available in the repository https://github.com/r-eifler/MUGS-approximation-dataset.

5.1 Evaluation of Plan Computation

First, we evaluate the performance of policy guidance in the computation of plans, i.e., we would like to see whether restricting the state space using policy distance functions can speed up the process of finding a plan, or whether it can help us to scale up to larger tasks where state-space planners fail. In the iterative planning process, sample plans are required to satisfy the soft goals enforced by the user within the cost bound. The interaction with the user should be responsive, so a plan generation should be relatively fast. Here we use a time limit of 2 minutes per task. We use the LAMA-first planner [19] as a baseline, because it works well with short time limits. For the policy-guided configurations, we iteratively increase the radius until we are able to solve the task or reach the time limit.

Figure 2 shows how the fraction of solved tasks (relative coverage) increases over time. Searching within the radius solves at least as many tasks as the baseline in most domains (the most notable exception being Elevators). The results vary across domains. Our methods can solve significantly more tasks and faster than the baseline in domains Floortile, Scanalyzer and Storage, and the same can observed to a lesser degree in Beluga, Satellite and Visitall. On average, the search with δ_P and $\delta_\#$ solve 8% and 15% more tasks, respectively, than the baseline. Searching within a radius outperforms strictly following the policy as was expected (except for Gripper where the policy seems to be optimal). $\delta_\#$ seems to perform slightly better than δ_P overall, and $\delta_\#$ significantly outperforms δ_P in Floortile and Storage.

Given these encouraging results, we also ran a preliminary experiment with policy-guided search in the classical planning setting. We used the same domains, but without cost bounds, all goals as hard goals, and the timeout of 30 minutes. We used GBFS with h^{FF} [14] as a canonical planning baseline here. Using δ_P and $\delta_\#$ significantly increases coverage in 5 and 4 domains, respectively; overall, out of 917 tasks, δ_P ($\delta_\#$) solves 36 (65) more tasks than the baseline. These results indicate that policy-guided search has potential beyond our OSP setting here. Future work should conduct a more thorough experimental evaluation, to get a better understanding when restricting a state space by a radius around a learned policy is beneficial, how and when to decide to increase the radius during the search, or whether there are distance functions better suited for this setting.

5.2 Evaluation of MUGS Approximation

In the iterative planning process, the computation of MUGS takes place before the user interaction and therefore allows for a larger time limit, here we use 30 minutes. We also evaluated an *anytime* version

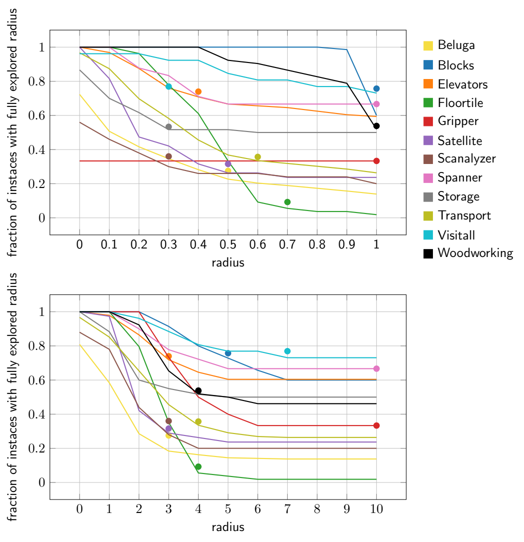

Figure 3. Fraction of instances where the whole radius could be explored scaled over radius size. Top: probability distance function δ_P and bottom diverging actions distance function $\delta_{\#}$. ● mark the coverage of the exact MUGS computation. They are placed at the radius where the coverage of the approximation drops below the exact computation.

of the exhaustive state space exploration: We run the exploration with an imposed time limit or state limit, collect all MSGS, and when the limit is reached we compute approximate MUGS from them. The goal here is to see in comparison whether policy-guided methods indeed carry useful information about the state space, or if the scaling they provide is merely because they explore fewer states (regardless of where these states are in the state space).

To measure the accuracy of approximate MUGS $\mathcal{G}^{\text{MUGS}}(P_r^\delta)$, we use the *fraction of solvable soft goal subsets* (*fsgs*). That is, we compute how many subsets of soft goals are solvable and divide this number by the overall number of possible subsets of soft goals, i.e., given a set of MUGS $\mathcal{M}$ we define:

$$fsgs(\mathcal{M}) = \frac{|\bigcup_{M \in \mathcal{M}} \{G \subsetneq M\}|}{2^{|G^{\text{soft}}|}}.$$

Recall that we showed in Proposition 1 that the size of MUGS increases with increasing radius (i.e., with the larger portion of state space explored). Therefore, we achieve the largest possible value of *fsgs* for the exact MUGS, and the accuracy of approximate MUGS $\mathcal{G}^{\text{MUGS}}(P_r^\delta)$ can be measured by looking at the difference between $fsgs(\mathcal{G}^{\text{MUGS}}(P_r^\delta))$ and $fsgs(\mathcal{G}^{\text{MUGS}}(\mathcal{P}))$ (i.e., the maximum achievable value is $fsgs(\mathcal{G}^{\text{MUGS}}(\mathcal{P}))$, not 1). Also note that $fsgs(\mathcal{G}^{\text{MUGS}}(P_r^\delta))$ increases monotonically and approaches $fsgs(\mathcal{G}^{\text{MUGS}}(\mathcal{P}))$ from below with increasing radius r.

5.2.1 *Coverage of Approximate MUGS Computation*

Figure 3 shows, for varying radius, in how many instances the policy-guided methods explored the whole (restricted) state space, i.e., in how many cases we were able to obtain P_r^δ in order to compute approximate MUGS. With the action-probability distance function δ_P, the smoothness of the scaling of radii highly depends on the domain. For example in Blocksworld, the action with the highest probability is almost always above 0.9, therefore alternative plans are considered only with the radius 1. Conversely, the Gripper domain has a lot of states where multiple actions are assigned similarly high probabilities, therefore the whole state space is explored even for small radii.

The action-probability distance function works best for domains laying in between these extremes. For example, Transport and Floortile have many states where one action is clearly preferred while there are also enough states that act as choice points with multiple similarly ranked actions. So, the radius in these domains scales more smoothly. In most domains, we can scale the radius from 0 up to 0.4 and still compute approximate MUGS in at least as many tasks as the exact computation. The most notable exceptions being Gripper and Scanalyzer, where δ_P does not work very well.

With the action divergence distance function $\delta_{\#}$, the scaling of the radius is more homogeneous across domains, because this distance function does not depend on the internal "confidence" of the policy. The scaling depends merely on the structure of the state space. In all domains, using radius up to 3 allows us to solve at least as many tasks with approximate MUGS than we would be able with exact MUGS. Overall, the results clearly show that, with few exceptions, computing approximate MUGS with policy distance functions scales beyond the exact method. The next question is how close approximate MUGS are to the exact MUGS.

5.2.2 *Comparison to Exact MUGS Computation*

Figure 4 shows how the average *fsgs* over all considered tasks increases with time and with the number of explored states for the approximation methods and for the anytime algorithm. Since we want to see how close we get to the exact method, we only consider the tasks where we could compute exact MUGS, and we only show a subset of the most interesting domains.

The approximation methods converge to the exact MUGS as we increase time or allow exploring more states across all domains. It seems that we are able to achieve reasonably high *fsgs* values even for small time and state (memory) limits. How quickly methods converge depends on the domain (and the quality of the corresponding policy). The anytime method converges faster than the policy-guided methods in 9 out of 12 domains, but one has to keep in mind that here we use only small tasks where we were able to compute exact MUGS. When we look at the convergence rate vs. the number of explored states, we can observe that policies are often able to provide useful information as they reach the same *fsgs* with significantly less explored states than the anytime variant. This is the case for $\delta_{\#}$ in all domains and for δ_P in 10 out of 12.

Overall $\delta_{\#}$ seems to provide better guidance than δ_P with respect to both runtime and number of explored states. For finding a plan, it is only necessary to explore one way of solving the task. However, to compute accurate approximation of MUGS, a much larger portion of the state space has to be explored (of course, preferably only the parts that are actually relevant for computing MUGS). $\delta_{\#}$ seems to be the better of the two policy distance functions in doing that.

Next we analyze whether computing approximate MUGS scales beyond the feasibility of the exact computation.

5.2.3 *Beyond Feasibility of Exact Computation*

To see how well our methods scale to large tasks, we consider here only the tasks where we could *not* compute exact MUGS, and we focus on a comparison of *fsgs* values over increasing time as this is presumably the critical factor in practice. Figure 5 shows results for 8 domains, the remaining 4 domains either have no commonly solved instances, or *fsgs* values of all methods are too small for a reasonable comparison. Figure 5 includes maximum *fsgs* values from Figure 4 as a reference, because we expect the *fsgs* values over exact MUGS

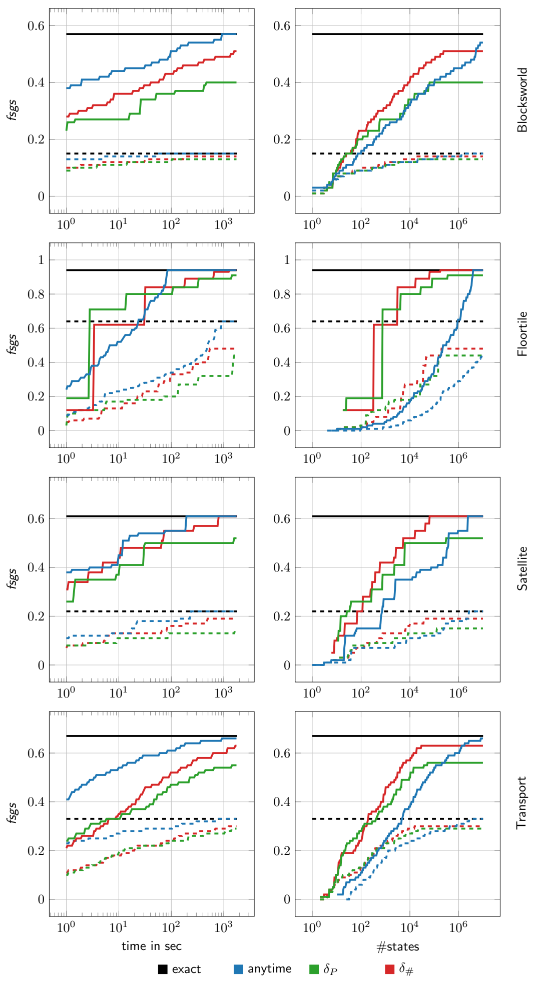

Figure 4. *fsgs* value over time (left) and explored states (right) over all commonly solved instances where the exact *fsgs* value could be computed. Dashed: tasks with cost bound of 50%; solid: cost bound of 75%.

to be about the same also in the large tasks. Nevertheless, the actual maximum *fsgs* values may be different.

We can see analogous trends as in the previous section. The *fsgs* values increase with increasing time limit, so that we obtain an increasingly more accurate MUGS approximations. There does not seem to be a significant difference for tasks with 50% cost bound. For the 75% cost bound, the baseline is slightly better than our methods in Scanalyzer. In all other domains, either $\delta_\#$ or δ_P or both outperform the baseline. The biggest advantage can be observed in Floortile, Satellite, Storage, and Woodworking where the average *fsgs* value of $\delta_\#$ is more than twice as large as the value of the baseline at the 30 minutes mark.

6 Conclusion

Explainable AI is one of the big challenges today. In the particular setting of iterative OSP planning, MUGS-based explanations can be helpful, but they are limited by the lack of scalability of exact MUGS computation, as well as, sometimes, the need to generate example plans online. Here we show that this can be alleviated by leveraging learned information in the form of action policies. Using distance functions to measure a radius around a policy execution and searching only within this radius allows us to scale both generation of example plans and the computation of MUGS to larger instances.

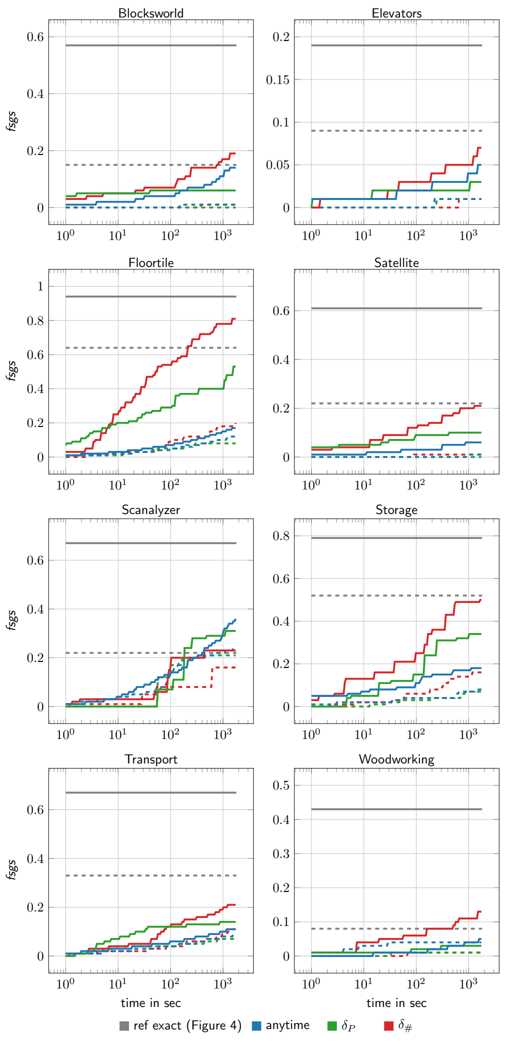

Figure 5. *fsgs* value over time over all commonly solved instances where the exact *fsgs* value could **not** be computed. Dashed: tasks with cost bound of 50%; solid: cost bound of 75%. Gray horizontal lines are maximum *fsgs* values from Figure 4, the actual maximum *fsgs* values here might differ.

Future work involves experimentation with additional kinds of action policies, evaluating their potential. Different types of policy distance functions are also thinkable.

An off-shoot of our work are policy-improvement methods. Our radii define searches around a policy path, a natural method to enhance individual policy runs. This constitutes a somewhat novel form of plan-generation method, which as our preliminary results in classical planning show does have potential. A further interesting direction is to investigate how such policy-improvement methods can feed into policy re-training.

Most interestingly for explainable planning, future work could explore the potential of policy-radius MUGS as a form of explanations *of the policy itself*, i.e., of the trade-offs the policy makes and to what extend they can be changed, similarly to the role of the classical planner in iterative OSP planning. The radius here could be viewed as a form of constraint relaxation as in [7], with higher radii being more relaxed. This makes sense if the user trusts the policy (e.g., through previous policy testing), and would prefer the radius-constraint to be tight. MUGS in this context could then explicate the trade-off between the desire to reach goals or to be close to the policy.

Acknowledgements

This work was funded by the European Union's Horizon Europe Research and Innovation program under the grant agreement TUPLES No 101070149, as well as by DFG Grant 389792660 as part of TRR 248 (CPEC, https://perspicuous-computing.science). Rebecca Eifler 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.

References

[1] L. Amado, R. Mirsky, and F. Meneguzzi. Goal recognition as reinforcement learning. In *AAAI*, pages 9644–9651, 2022.

[2] M. Boddy, J. Gohde, T. Haigh, and S. Harp. Course of action generation for cyber security using classical planning. In *ICAPS*, pages 12–21, 2005.

[3] C. Domshlak and V. Mirkis. Deterministic oversubscription planning as heuristic search: Abstractions and reformulations. *JAIR*, 52:97–169, 2015.

[4] R. Eifler, M. Cashmore, J. Hoffmann, D. Magazzeni, and M. Steinmetz. A new approach to plan-space explanation: Analyzing plan-property dependencies in oversubscription planning. In *AAAI*, pages 9818–9826, 2020.

[5] R. Eifler, M. Steinmetz, A. Torralba, and J. Hoffmann. Plan-space explanation via plan-property dependencies: Faster algorithms & more powerful properties. In *IJCAI*, pages 4091–4097, 2020.

[6] R. Eifler, M. Brandao, A. Coles, J. Frank, and J. Hoffmann. Evaluating plan-property dependencies: A web-based platform and user study. In *ICAPS*, pages 687–691, 2022.

[7] R. Eifler, J. Frank, and J. Hoffmann. Explaining soft-goal conflicts through constraint relaxations. In *IJCAI*, pages 4621–4627, 2022.

[8] J. Eisenhut, Á. Torralba, M. Christakis, and J. Hoffmann. Automatic metamorphic test oracles for action-policy testing. In *ICAPS*, pages 109–117, 2023.

[9] J. Eisenhut, X. Schuler, D. Fišer, D. Höller, M. Christakis, and J. Hoffmann. New fuzzing biases for action policy testing. pages 162–167, 2024.

[10] S. Garg, A. Bajpai, et al. Size independent neural transfer for RDDL planning. In *ICAPS*, pages 631–636, 2019.

[11] R. P. Goldman and U. Kuter. Measuring plan diversity: Pathologies in existing approaches and a new plan distance metric. In *AAAI*, pages 3275–3282, 2015.

[12] E. Groshev, M. Goldstein, A. Tamar, S. Srivastava, and P. Abbeel. Learning generalized reactive policies using deep neural networks. In *ICAPS*, pages 408–416, 2018.

[13] M. Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[14] J. Hoffmann and B. Nebel. The ff planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302, 2001.

[15] M. Issakkimuthu, A. Fern, and P. Tadepalli. Training deep reactive policies for probabilistic planning problems. In *ICAPS*, pages 422–430, 2018.

[16] M. Katz and S. Sohrabi. Reshaping diverse planning. In *AAAI*, pages 9892–9899, 2020.

[17] M. Katz, E. Keyder, D. Winterer, and F. Pommerening. Oversubscription planning as classical planning with multiple cost functions. In *ICAPS*, pages 237–245, 2019.

[18] G. Neubig, C. Dyer, Y. Goldberg, A. Matthews, W. Ammar, A. Anastasopoulos, M. Ballesteros, D. Chiang, D. Clothiaux, T. Cohn, K. Duh, M. Faruqui, C. Gan, D. Garrette, Y. Ji, L. Kong, A. Kuncoro, G. Kumar, C. Malaviya, P. Michel, Y. Oda, M. Richardson, N. Saphra, S. Swayamdipta, and P. Yin. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*, 2017.

[19] S. Richter, M. Westphal, and M. Helmert. Lama 2008 and 2011. In *International Planning Competition*, pages 117–124. ICAPS Freiburg, Germany, 2011.

[20] D. Smith. Planning as an iterative process. In *AAAI*, pages 2180–2185, 2012.

[21] D. E. Smith. Choosing objectives in over-subscription planning. In *ICAPS*, pages 393–401, 2004.

[22] S. Ståhlberg, B. Bonet, and H. Geffner. Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits. In *ICAPS*, pages 629–637, 2022.

[23] S. Toyer, F. Trevizan, S. Thiebaux, and L. Xie. Action schema networks: Generalised policies with deep learning. In *AAAI*, pages 6294–6301, 2018.

[24] S. Toyer, S. Thiébaux, F. W. Trevizan, and L. Xie. Asnets: Deep learning for generalised planning. *Journal of Artificial Intelligence Research*, 68: 1–68, 2020.