TLdR: Policy Summarization for Factored SSP Problems Using Temporal Abstractions

Sarath Sreedharan, Siddharth Srivastava, and **Subbarao Kambhampati**
CIDSE, Arizona State University, Tempe, AZ 85281 USA
ssreedh3@asu.edu, siddharths@asu.edu, rao@asu.edu

Abstract

As more and more people are expected to work with complex AI-systems, it becomes more important than ever that such systems provide intuitive explanations for their decisions. A prerequisite for holding such explanatory dialogue is the ability of the systems to present their proposed decisions to the user in an easy-to-understand form. Unfortunately, such dialogues could become hard to facilitate in real-world problems where the system may be planning for multiple eventualities in stochastic environments. This means for the system to be effective, it needs to be able to present the policy at a high-level of abstraction and delve into details as required. Towards this end, we investigate the utility of temporal abstractions derived through analytically computed landmarks and their relative ordering to build a summarization of policies for *Stochastic Shortest Path Problems*. We formalize the concept of policy landmarks and show how it can be used to provide a high level overview of a given policy. Additionally, we establish the connections between the type of hierarchy we generate and previous works in temporal abstractions, specifically MaxQ hierarchies. Our approach is evaluated through user studies as well as empirical metrics that establish that people tend to choose landmarks facts as subgoals to summarize policies and demonstrates the performance of our approach on standard benchmarks.

1 Introduction

Recent successes in AI have helped fuel an increasing interest in developing AI-based systems that can assist us in our daily lives. As more and more users start working with these systems, it becomes essential that these systems can facilitate intuitive and fluent interaction with their end-users. A prerequisite for allowing such interaction would be the agent's ability to hold easy to follow explanatory dialogue that helps users understand the rationale behind agent decisions. Unfortunately, such explanatory dialogue is complicated by the fact that in real-world scenarios the agent is expected to plan its actions over long time horizons while taking into account the stochasticity of the environment. This means the agent would have to compute contingencies for each possible eventualities and its decisions may be best represented as a policy. Large policies are extremely hard for lay users to understand and the very act of presenting the agent policy can hinder the deployment and adoption of AI systems.

Our proposed strategy for handling such scenarios is to equip the system with mechanisms that allow the users to view a high-level summary of the policy that the agent may have chosen to follow. Our choice of summarization techniques were motivated by three main factors (1) People tend to decompose long horizon planning into sequences of subgoals. This is a well-known fact in psychological literature and has been validated by number of studies (c.f. (Donnarumma *et al.* 2016; Cooper and Shallice 2006; Simon and Newell 1971)); (2) The approach shouldn't assume that the user is an expert in the domain or has prior knowledge about the dynamics of the domain, since the method is a preceding step for explanations; and (3) The approach shouldn't assume it is summarizing optimal policies.

With these design principles in mind we introduce our approach: **T**emporal Abstraction Through **L**an**d**mark **R**ecognition or **TLdR**. TLdR hypothesizes that one way to extract useful temporal abstractions for a given policy is to identify sets of bottleneck or *landmarks facts* and their relative ordering that needs to be satisfied by any valid execution of the given policy. We believe our work represents the first formalization of landmarks for stochastic domains. We also introduce the idea of policy landmarks along with compilation-based methods to generate these landmarks with formal guarantees. This paper will focus on Stochastic Shortest Path problems (SSPs) since their goal-directed nature is more natural for everyday users (Newell *et al.* 1972) as well as being more general than infinite horizon discounted MDPs. Once identified, end users can use these landmarks as the basis for generating their explanatory queries (of the form discussed in (Miller 2018)) or can further drill down by focusing on specific landmarks to get more details.

The rest of the paper is structured as follows: Section 2 starts with a brief overview of related works in this space and then Section 3 introduces the setting and some of the formalisms we will be using throughout the paper. Section 4 introduces a simple illustrative scenario that will act as our running example and Section 5 will delve into the details of our methods. Through Section 6 we will further investigate the specifics of the hierarchy we are generating and we discuss the evaluations we performed in Section 7. Finally, Section 8 concludes the paper with a discussion of future directions.

2 Related Work

In recent years there has been increasing interest in the problem of explainable AI in general and also specifically for explaining/summarizing policies. While many of these works focus on policies learned through neural networks (c.f. (Greydanus *et al.* 2017)), there are a few works that have considered factored MDP settings as well (c.f. (Khan *et al.* 2009)).

In the context of policy summarization, (Lage *et al.* 2019) presents an approach that tries to identify a subset of state action tuples that can help users guess the rest of the policy. The tuple is selected under some assumptions about the computational model the user may be making. Unfortunately, such works that aims to generate summaries optimized for policy completion assume the user has some prior knowledge about the task. Our motivation in this work is to provide summaries that could act as a first step before providing more explanations about the task. Consequently, we address scenarios where the user may be unaware of task details or may misunderstand the the task (similar to explanatory settings studied in (Chakraborti *et al.* 2017) and (Sreedharan *et al.* 2018)).

Topin and Veloso (2019), utilize state abstractions to simplify policies. This is completely complementary to our approach and we can use methods described in that paper along with ours to identify landmarks in abstract state spaces (c.f. (Sreedharan *et al.* 2019) for similar strategies applied to classical planning setting). Hayes and Shah (2017) have also looked at producing summaries specific to user questions.

In MDP literature, people have previously used landmarks in different contexts, for example (Ramesh *et al.* 2019) uses the term landmark as a way to denote prototypical state and (Kaelbling 1993) uses landmarks to refer to centers of a region of the environment. Options learning literature (c.f (McGovern and Barto 2001; Stolle and Precup 2002)) have also used the term bottleneck states to refer to states that appear frequently in valid execution traces. They utilize such states as a basis for learning options and as we will see this is closely related to the techniques discussed in the paper. The idea of bottleneck states are also related to landmarks as discussed in classical planning literature (Hoffmann *et al.* 2004).

3 Background

We will focus on cases where the planning problem corresponds to an SSP_{s_0}, i.e a stochastic shortest path problem with a single initial state (Kolobov 2012). In the rest of the paper when we refer to MDPs we will be in fact referring to an SSP_{s_0}. Such models can be formally defined by the tuple $\mathcal{M} = \langle S, A, T, C, G, s_0 \rangle$, where S is the set of states, A is the set of actions, $T : S \times A \rightarrow [0, 1]$ defines the transition function corresponding to the MDP, while C captures the cost function, G the set of goal states, and s_0 the initial state. For this setting, we are generally interested in policies that guarantee that any history sampled from the policy will eventually lead to a goal state with probability one, such policies are generally referred to as proper policies. Moreover, the fact that we are given an initial state means that we only need to identify actions for states that are reachable from from s_0 under π_0, hence we can restrict our attention to partial proper policies (Kolobov 2012).

We will assume that the state space S can be specified by a set of propositional fluents F, where $|S| = 2^{|F|}$ and each state can be uniquely identified by the set of propositional fluents that are true in that given state. Additionally, we assume the set of goal states G can be concisely described by a subset of propositions $\mathcal{G}$. For example, if we are considering a travel planning task, $\mathcal{G}$ might just include the proposition onboard_plane and the corresponding goal state set G consists of all states where the fluent onboard_plane will be true.

The expected cost of a state is defined here similarly to the standard undiscounted indefinite horizon SSPs and the Bellman optimality equation is provided as

$$V(s) = C(s, a) + \Sigma_{s' \in S} T(s, a, s') * V(s')$$

and all goal states are absorbing states.

A partial policy π_0^* is defined to be optimal if there exists no other partial policy whose expected cost for the initial state is smaller than π_0^* (i.e. $V^{\pi_0^*} \leq V^{\pi}$ for all proper policies π).

We will consider a setting where the underlying MDP $\mathcal{M}$ is known to the explainer and it is tasked with explaining a given policy π. Note that we don't assume π to be optimal, but rather it can be any partial proper policy. To be succinct in the rest of the paper we will use policy in place of partial proper policies and we will specifically note any exceptions to the case. Before we delve further into the problem, let us take a quick look at a travel planning domain that will act as our illustrative example for the rest of the paper.

4 Motivating Example: Travel Planning Domain

Consider an intelligent personal assistant that is being used to track and plan various daily activities of its users. The personal assistant is capable of gathering information from multiple sources and generating probabilistic models for various events like weather, vehicle delays, traffic, etc.. Many of these models may be too complex for the user to easily understand or the information sources too rapidly changing for the user to keep track off. Now if the user was to ask the agent to come up with a plan that allows them to get to their flight from their home, the agent could easily use the models it generated for various sources to create an MDP and compute a policy that is guaranteed to take the user from their home to the designated flight.

Even in this setting, challenges will arise when the user wants to actually make sense of the policy suggested by the system. The policy could be extremely large, with many branches to handle various contingencies (Figure 1 (A) presents a simplified version of such a policy). The system could hardly expect the user to make sense of the policy if it merely dumped the entire policy-graph. Given the fact that the intelligent assistant is almost always available to the user on various devices, the agent could decide to give the user the policy one step at a time, always coming back to the user with the next action to perform given where the user is. This

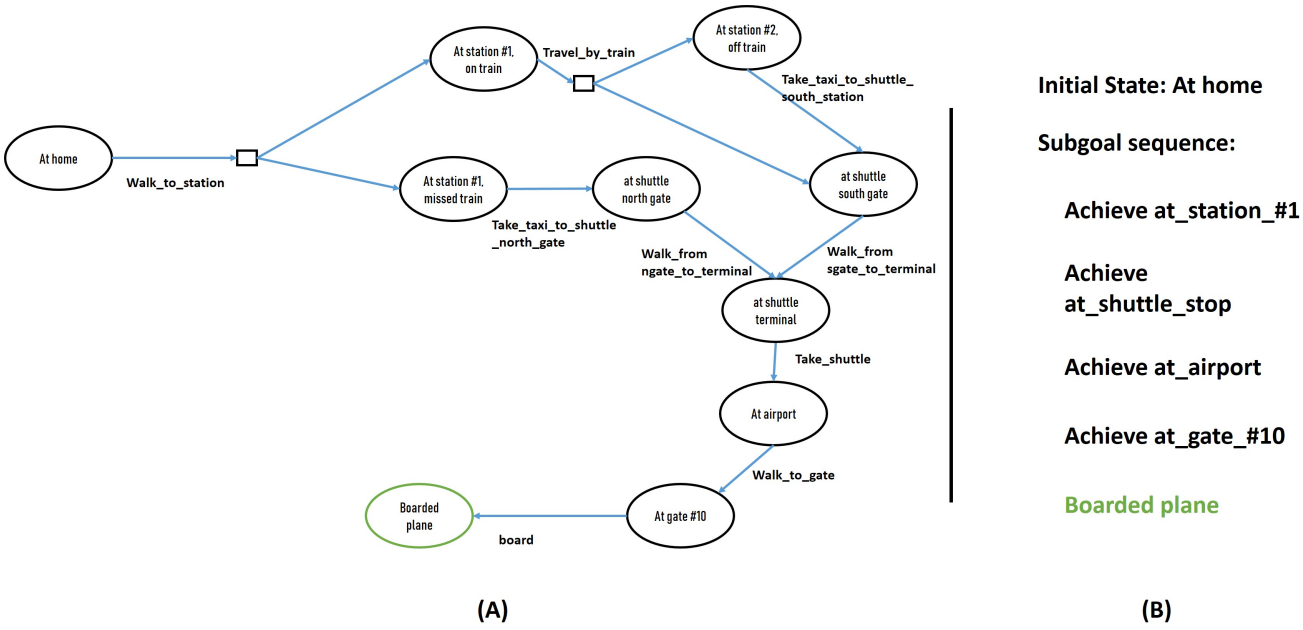

Figure 1: The subfigure (A) presents policy graph corresponding to example detailed in the illustrative domain. Here the edges with boxes represents actions with stochastic effects and the goal is to board the plane. Subfigure (B) presents one possible summarization that describes a sequence of subgoals that needs to achieved under the given policy.

approach could also be extremely unsatisfying as the user would want to know if the policy as a whole aligns with their preferences; being fed the decisions one at a time, prevents them from getting an overall idea about the policy.

A more reasonable strategy would involve first presenting the user with a summary of the policy that highlights some of the important waypoints on the way to the airport, along with the order in which they should be crossed. Figure 1 (B) presents one possible set of such waypoints and their order in which they are to be achieved. If the agent were to follow this summarization scheme, the system could report that: you would first need to get to "the station #1" and then get to "airport shuttle station", then get to the "airport", then to "gate 10" and finally "board the flight".

Now given these abstract subgoals, the user can the possibly raise contrastive questions (c.f (Miller 2018)) in terms of these subgoals, for example, "What if you try to avoid the shuttle staion?" or the user could further drill down further to get more information. In the next sections, we will discuss how for a given model and policy, we can automatically generate such subgoals and their relative ordering to summarize the policy.

5 Our Approach: TLdR

As hinted in earlier sections, we will use partially ordered landmarks as our summary. Since we are not aware of any earlier works that have formalized landmarks for MDPs, we will start by introducing and formalizing the notion of landmarks in this setting. With the basic notions of landmarks in place, we can define policy landmarks and establish some basic properties. To generate these policy landmarks, we will propose compilation based methods with soundness and completeness guarantees.

5.1 Landmarks in MDPs

In the simplest terms, a landmark can be understood as formulas over propositional fluents that need to be satisfied by all paths from the initial state to goal states. More formally we can define landmarks as as

Definition 1. *For a given model $\mathcal{M}$ whose state space is defined over a set of factors F, let's define a tuple $\mathcal{L} = \langle \Phi, \prec \rangle$, where Φ is a set of formulas specified over F and $\prec$ defines some ordering over them. Now $\mathcal{L}$ is said to be a landmark set for $\mathcal{M}$, if all transition sequences from start state to goal state of non-zero probability in $\mathcal{M}$ satisfy the formulas in Φ and their relative ordering, i.e.,*
Let $\tau = \langle s_0, a_0, ..., s_g \rangle$ be a transition sequence from s_0 to $s_g \in G$, if $\mathcal{L}$ specifies landmarks for $\mathcal{M}$ then for all $\varphi \in \Phi$ there exists $s_j \in \tau$ such that $s_j \models \varphi$ and for all other formulas φ_1, φ_2 in Φ such that $\varphi_1 \prec \varphi$ and $\varphi \prec \varphi_2$ there must exist $s_i, s_k \in \tau$ $(i < j < k)$ such that $s_i \models \varphi_1$ and $s_k \models \varphi_2$.

So when identifying landmarks, we expect not only to identify a single formula, but rather a partially ordered set of formulas. For example the set $\{(\text{at\_airport}) \wedge (\text{has\_ticket}), (\text{onboard\_plane})\}$ along with the ordering $(\text{at\_airport}) \wedge (\text{has\_ticket}) \prec (\text{onboard\_plane})$ would constitute landmarks for the aforementioned travel task. This definition of landmarks parallels their usage in classical planning literature (c.f (Hoffmann *et al.* 2004)) where they have been identified as an extremely useful information for guiding the planner during plan generation. Throughout the text we will use 'landmark' in singular to

denote individual formulas while we will reserve the use of 'landmarks' to denote the partially ordered set of formulas.

While such landmarks on their own could still act as useful subgoals for policy summarization, they may be too few and far between and may not capture the specifics of the policy being pursued (unless the user choose to focus on specific intermediate states). Instead, we will consider landmarks that are specific to the policy at hand, which would lead us to define *policy landmarks*

Definition 2. *For a given model $\mathcal{M}$, fluent set F and a policy π, a tuple $\mathcal{L}^{\pi}_{s_0,G} = \langle \Phi, \prec \rangle$ is a policy landmarks from s_0 to the goal set G if and only if, the formulas in Φ and their corresponding ordering can be satisfied by every execution trace $\tau \sim \pi(s_0)$ from s_0 to G.*

Note that while any landmarks for the model as a whole will also be a policy landmark, but the reverse is not true. For example, in the travel scenario in_taxi might be a landmark for some policy but may not be a landmark for the task as a whole since it may be possible to get to the flight without ever boarding a taxi. We include the initial state and goal state set in the definition to allow for the possibility of recursively generating landmarks from any two reachable states for the given policy.

Landmarks as defined above are quite expressive and extensible. By leveraging disjunctive formulas we can even generate subgoals in cases where there are no state facts that are shared by all the paths. For example, in the case of the travel domain described in Figure 1 at_shuttle_north_gate $\vee$ at_shuttle_north_gate is a landmark. In fact, we can show that there exists a landmark set that can capture the entirety of any given policy.

Proposition 1. *For a given model $\mathcal{M}$, fluent set F and a policy π, there exists a policy landmark $\mathcal{L}^{\pi}_{s_0,G} = \langle \Phi, \prec \rangle$ such that for every reachable state s from s_0 under π, there exists a non-trivial formula $\phi \in \Phi$ such that $s \models \phi$.*

Proof Sketch. We will show that the property is true by constructing such a landmark set. Let's start with the policy graph corresponding to the given policy and partition the states in the graph to levels based on number of hops from the initial state (in case of policies with loops we assign each state the level corresponding to the shortest number of steps in which it can be reached by an execution of the policy). Now we will generate a DNF formula ϕ for each level, such that formula for level l contains a clause corresponding to each state reachable within l steps. Now we can create landmarks by ordering these DNF formulas according to the levels (and limit ordering to $\preceq$ starting from the level where the first goal state is reached). Such a landmark set should satisfy the requirements based on their construction. □

The above property merely demonstrates the extensibility of landmarks as a concept and is not an endorsement for using more complex landmark formulas for generating summaries. In fact, in this paper we will focus on *fact landmarks*, where each formula corresponding to a landmark consists of a single proposition. So when viewing these landmarks as subgoals, they can be thought of as achieving the fact corresponding to that proposition.

Now that we have defined partially ordered landmarks for a policy, our next step would be to identify methods that allow us to generate such landmarks.

5.2 Generating Landmarks

Our proposed algorithm for generating landmarks would rely on compilation into a corresponding classical planning problem, so we will start with a quick definition of classical planning problem (Geffner and Bonet 2013). Classical planning problems are generally describe by a tuple of the form $\mathcal{M}^c = \langle F^c, A^c, \mathcal{I}^c, \mathcal{G}^c \rangle$, where F^c provides the set of propositional fluents, A^c the list of deterministic actions, $\mathcal{I}$ the initial state and $\mathcal{G}^c$ is the goal specification (similar to how we defined $\mathcal{G}$ for the MDP). Each action is defined by a tuple $<$ prec$^+$, prec$^-$, add, del $>$, where prec$^+$ and prec$^-$ respectively provides the set of facts that must be true and false for the applicability of actions, add specifies the fluents it will set true on execution and del specifies the fluents that it will set false. For our purposes we will assume all the individual components of the action definition can be represented as sets.

It is well known that one way we can approximate MDPs is to determinize the MDP to an equivalent classical planning problem (Yoon *et al.* 2007). One of the popular forms of determinization is what's called an all outcome determinization (Yoon *et al.* 2007) where for every possible effect of the original MDP action, the classical planning problem would include a separate action with that specific effect, i.e. if for a given state s executing an action a could either result in s_1 or in s_2, then the determinization should produce two actions, one which will result in s_1 and another action that result in s_2. Such determinization procedures become simpler when the original MDP is specified in a factored form such as PPDDL (Younes and Littman 2004). While a naive all outcome determinization could result in an exponential sized planning model, we can get more concise models by relying on more expressive representations (c.f (Keller and Eyerich 2011)). As we will see in our particular setting if we relax the need for generating all landmarks we can still rely on simple models.

Now given such an all outcome determinization of our MDP $\mathcal{M}$ we can see that planning landmarks in the determinized model should be valid landmarks for $\mathcal{M}$. This property holds since landmarks for classical planning models are defined over all possible plans and plans in the determinized model have a one to one correspondence with traces in the original model. Unfortunately, we aren't interested in just generating landmarks for the model as a whole, but rather for the specific policy. One way to get there would be to restrict the model only to produce plans that correspond to traces we can sample from the given policy. We can create a compiled version of the classical planning problem which meets the above requirement if the policy is specified as a finite state machine (see (Sreedharan *et al.* 2019) or (Baier *et al.* 2007) for possible compilation). More often than not, many of the popular offline planners for MDPs generate policies in tabular form, i.e., they explicitly enumerate the reachable states and their corresponding actions. So in this paper we will focus on cases where the policy is provided in this form.

Instead of relying on a compilation of policies to FSAs and then to planning problem, we will develop a method to create a compiled classical planning problems from the given policy, such that, the landmarks for that model aligns with policy landmarks of the given tabular policy.

Specifically, given our original MDP $\mathcal{M}$ and a policy π with a reachable state set $\mathcal{R}(s_0, \pi)$ let's consider the following classical planning model, $\mathcal{D}(\mathcal{M}) = \langle F, A^{\mathcal{D}}, I^{\mathcal{D}}, \mathcal{G} \rangle$. Here the new classical planning model makes use of the same fluent set, initial state ($I^{\mathcal{D}} = s_0$) and goal as the MDP $\mathcal{M}$. The new model contains one action a^s for each reachable state s, such that the precondition of the action is $\text{prec}^+_{a^s} = s$ (assuming set representation for the state), the negative preconditions and delete effects is empty (i.e $\text{prec}^-_{a^s} = \text{del}_{a^s} = \{\}$), and finally the add effect is the set of all new fluents that can be set true by the corresponding policy action, i.e.,

$$\text{add}_{a^s} = \Big(\bigcup_{s' \in \{s' | s \in S \wedge T(s,\pi(s),s') > 0\}} s' \Big) \setminus s$$

The above model is polynomial in the size of the given policy even for factored model representation, and the reason why we do not need to worry about delete effects is because most established methods for landmark generation in classical planning rely on approximations that ignore deletes. We will restrict ourselves to landmark generation methods that focus on generating *causal landmarks* (Zhu and Givan 2003), where causal landmarks are defined to be facts that are required as preconditions for every possible plan. All that remains to be shown is that the causal landmarks derived from this model are sound policy landmarks for original model.

Theorem 1. *If f is a causal fact landmark extracted from $\mathcal{D}(M)$ then f must be a policy landmark for $\mathcal{M}$, π, initial state s_0 and goal set G. Also for another landmark f_2 (also causal in $\mathcal{D}(M)$), if the precedence $f \prec f_2$ holds for $\mathcal{D}(M)$ then it must also hold for the original MDP policy.*

Proof Sketch. We will prove this through contradiction. First off, it is easy to see that for any trace $\tau \sim \pi$, where τ is of the form $\langle s_0, a_0, ..., s_k \rangle$, there exists a corresponding executable plan trace in $\mathcal{D}(M)$, $\hat{\pi} = \langle s_0, a_0^{s_0}, ..., \hat{s}_k \rangle$ with the property that for any state s_i in τ, the corresponding state in the plan trace $\hat{s}_i$ exhibits the property that $s_i \subseteq \hat{s}_i$ (this can be easily established through induction). If the landmark f in $\mathcal{D}(M)$ was not a policy landmark for the original MDP, then there must be a trace from s_0 to some state in G we can sample from π where f never appears in any of the states. This means that in the corresponding valid plan for $\mathcal{D}(M)$ (where it goes from I to some state that satisfy $\mathcal{G}$), f can't appear in the precondition of any of the actions which contradicts the definition of causal landmark. Thereby proving our initial assertion. The soundness of ordering can be established following a similar line of reasoning. □

Another interesting property with this setting is the fact that we can exhaustively generate all policy landmarks from delete relaxation of the determinized model. Though this requires us to move away from the relatively concise representation used for $\mathcal{D}(M)$ to a more traditional all outcome determinization where we have a separate action for each possible transition in $\mathcal{M}$. That is, for a reachable state s, we add an action $a^{s,s'}$ for every s' such that $T(s, \pi(s), s') > 0$, where the action is defined as

$$a^{s,s'} = \langle \text{prec}^+ = s, \text{prec}^- = F \setminus s, \text{add} = s' \setminus s, \text{del} = s \setminus s' \rangle$$

We will call this encoding $\mathcal{D}(M)^c$

Theorem 2. *If f is a policy landmark for $\mathcal{M}$, policy π, initial state s_0 and goal set G then f must be a causal fact landmark for $\mathcal{D}(M)^c$.*

Proof Sketch. We can again show this through contradiction. Since $\mathcal{D}(M)^c$ is a standard all outcome determinization there should be a one to one correspondence between all possible traces from π and plans in $\mathcal{D}(M)^c$. Assume that f is a policy landmark that is not a causal landmark for $\mathcal{D}(M)^c$. This means that there must be a valid plan for $\mathcal{D}(M)^c$, where f won't appear in any of the preconditions. This means that there must be a trace from s_0 to a state in G that doesn't result in the generation of the fact f. This means that f can't be a policy landmark, hence resulting in a contradiction. This proves our assertion. □

Since we are not as focused on ensuring completeness for our evaluation we will just use the more concise compilation. While our formulation was based on proper policies to simplify formulation, the methods proposed in this paper do not require the policy itself to be proper. In fact, all we require is that there exists at least one possible trace from initial state to goal to generate the possible landmarks.

6 Formal Semantics for Hierarchies Generated Using TLdR

While in the earlier sections we alluded to the fact that the landmarks can be viewed as providing a form of temporal abstraction, we have yet to discuss specifics of the abstraction hierarchy induced by the landmarks. One way to understand the hierarchy induced is to map it to a MaxQ hierarchy and a related hierarchical policy (Dietterich 2000) (we consider a slightly modified version to allow application to SSP as opposed to just an infinite horizon discounted MDPs).

MaxQ is a particular method for specifying temporal abstractions for MDPs that involve specifying a task hierarchy for the given MDP. Usual MaxQ task hierarchies start with a root task and each task is recursively composed of subtasks. While the general MaxQ framework allows for parameterized tasks, we will just focus on a non-parameterized version where each subtask can be defined by the tuple $\langle T, A_i \rangle$ where T specifies the termination predicate or condition and A_i specifies the set of subtasks (including primitive actions) that can be performed as part of the execution of the subtask. Since we do not concern ourselves with learning the policies for these subtasks, we can easily skip the pseudo reward component that is usually also included in the subtask definition.

One of the main challenges of mapping landmarks we generate to a MaxQ task hierarchy is the fact that they may be partially ordered. So let us start from a partially ordered set of landmarks $\mathcal{L} = \langle \Phi, \prec \rangle$ to construct a totally ordered set $\mathcal{L}^{tot} = \langle \Phi^{tot}, \prec \rangle$ such that Φ^{tot} is the maximal subset of Φ that allows for total ordering and still contains the goal.

The root task itself would be a task corresponding to achieving the goal, and we can add a subtask corresponding to each fact in Φ^{tot} to the root task. The termination condition of each subtask is specified by the landmark formula (which for our case is just the individual fact), the action set consists of all the individual atomic actions. Now for each formula dropped (i.e formulas in $\Phi \setminus \Phi^{tot}$) we will add a subtask node to:

1. that are it's closest remaining successor, i.e., add a subtask to the node corresponding to ϕ', such that $\phi' \in \Phi^{tot}$, $\phi \prec \phi'$ and $\nexists \hat{\phi} \in \Phi^{tot}$ and $\phi \prec \hat{\phi} \prec \phi'$.
2. to any node in $\mathcal{L}^{tot}$ that is not comparable (as per $\mathcal{L}$) with the dropped node.

Addition of these new subtasks follows from the fact that when there are partial ordering among landmarks, each of their possible linearizations could occur under different traces. To formally describe this, let's introduce the concept of a completion of a landmark. We can define the completions of a landmark ϕ (denoted as $\mathcal{C}(\phi)$) as the set of states where the formula is satisfied for the first time ,i.e.,
$\mathcal{C}(\phi) = \{s_k | s_k \in \mathcal{R}(s_0, \pi) \wedge \ \phi \models s \wedge \exists \tau \sim \pi(s_0), \forall s_j \in \tau, (s_j \models \phi \implies k \leq j)\}$.
Now given this concept, we will assert

Proposition 2. Let f_0, f_1, f_2 and f_3 be landmarks such that $f_0 \prec f_1$, $f_0 \prec f_2$, $f_1 \prec f_3$ and $f_2 \prec f_3$. If $|\mathcal{C}(f_1) \cap \mathcal{C}(f_2)| > 0$ and the ordering is complete, then for any state $s_{f_2} \in \mathcal{C}(f_2)$, there either exists a states $s_{f_0} \in \mathcal{C}(f_0)$ such that f_1 is a landmark for paths from s_{f_0} to s_{f_2} or there exists a completion for f_3 such that f_1 is a landmark for paths from the completion of f_2 to f_3.

The above proposition states that if f_1 and $f2$ are not fact landmark representations of a larger conjunctive landmark (thus $|\mathcal{C}(f_1) \cap \mathcal{C}(f_2)| > 0$) and the ordering is complete (i.e partial ordering is not due to missing ordering), then for any state corresponding to achievement of f_2, the landmark f_1 needs to be achieved before reaching the current state or needs to be achieved before the successor state (after crossing the current state). This proposition trivially follows from the nature of ordering between landmarks and shows the landmarks excluded from the original total-ordering can still be used as a subtask to achieving the landmarks listed in $\mathcal{L}^{tot}$. For conjunctive landmarks, the dropping of one of the facts shouldn't result in any information loss since the completion set of one fact contains all the states where the other fact is also established. Moreover such partial ordering should dissappear once we start generating conjunctive landmarks.

The landmarks also allows us to convert the current policy into a hierarchical one that can be executed in the context of MaxQ (we will denote this policy as $\pi^{\mathcal{L}}$). We can use the following rules to recursively define the hierarchical policy

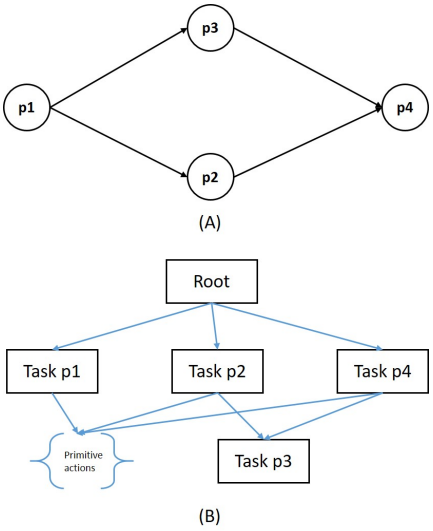

Figure 2: (A) The Hasse diagram (with arrow directions inverted) corresponding to a partially ordered landmark set and (B) an equivalent MaxQ hierarchy.

1. For root node: The initial state gets mapped to the subtask corresponding to the first landmark. For completeness, we will map the remaining states to the primitive actions specified in the policy, though those should never get executed.
2. For other subtasks: if the state corresponds to one of the completions of a child subtask, then map it to the corresponding subtask. For any other state execute action specified by the current policy for that state.

Proposition 3. The hierarchical value of the policy $\pi^{\mathcal{L}}$ ($V^{\pi^{\mathcal{L}}} = \langle s_0, nil \rangle$) is equal to the value of the original $V^{\pi}(S)$

If we follow the hierarchical policy execution procedure specified in (Dietterich 2000), it should be see that the hierarchical cost is the same as the original policy cost as the hierarchical policy would only be executing the primitive action as per the original policy.

7 Evaluation

7.1 User Studies

Concerning evaluation, our first priority was to perform an assessment of our central hypothesis, namely, that landmarks constitute useful subgoals. So one of the questions, that can be raised is whether people would choose landmarks when they use subgoals to summarize policies. The specific hypothesis we were interested in evaluating was

Hypothesis: *When presented with a policy for a task with non-deterministic/stochastic dynamics, people will choose landmark facts as high-level subgoals over non-landmark facts (where the landmarks are as defined in earlier sections).*

The hypothesis was tested by presenting different planning scenarios to participants from Amazon Mechanical Turk. Each participant was then asked to choose from a set

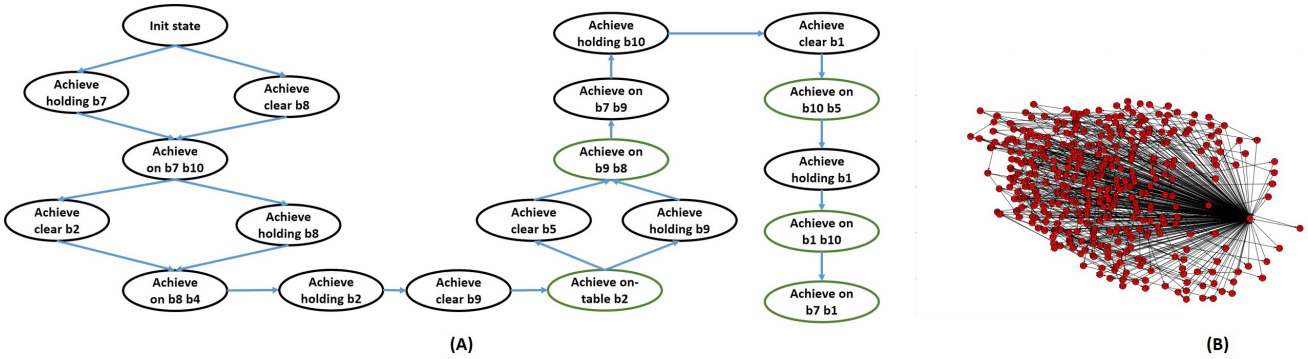

Figure 3: (A) The Hasse diagram corresponding to the policy landmarks generated for the fifth exploding blocksworld problem presented as part of IPC 2006. (B) A simple visualization of the entire policy.

of possible facts they believe would be the most appropriate subgoal to be included as part of a summary. In particular, we used a modified version of the travel scenario discussed earlier and a logistics planning scenario that dealt with the problem of delivering a package to a pre-specified location. We used 30 participants per scenario, and the scenario description included a description of the policy graph and no details on the actual transition probabilities or the rewards. After reading the scenario, the participants were provided a list of facts about reachable states under the given policy. They are then asked to select four facts from the list they believe can be provided as subgoals as part of a summary of the given policy. The list consisted of 13 facts in total, six of which were landmarks in the travel domain questionnaire, and five of them were landmarks for the logistics domain questionnaire. The users were presented the facts in a randomized order to make sure the results are counterbalanced. We also filtered the answers from the participants based on their answers on some factual questions about the policy. For filtering the response, each participant was presented two questions and their entire response was filtered out if they got any of the answers incorrect. We also changed the questions for every 15 participants to make sure the questions didn't introduce any additional bias in the participants' reply. After filtering we were left with 41 participants and 164 subgoal selections. Out of this 164 selection, 125 were landmark selections, which puts the number of landmarks selected at 76.2%. While non-landmark facts were selected by participants, the results show that the majority did in fact choose landmark facts to summarize the give policy. To test the statistical significance of our result we ran a paired two tailed t-test with the fact type (landmark or not) as the independent variable and the number of responses per group as the measure. We were able to establish that there exists a statistically significant difference between the groups with a level of significance set at 0.001 (the p-value was 0.0000093 for travel domain and 0.000533 for logistics). A PDF copy of the exact surveys has been included in the supplementary files (all of them are anonymized).

Domain	$\lvert \mathcal{R}^{\pi}(I) \rvert$	$\lvert \Psi^{\pi} \rvert$	$\lvert \Psi^{\mathcal{M}} \rvert$
Ex-Blocksworld	104.8	8.8	5.2
Elevator	13.2	7	4
Tireworld	13.6	1.8	0.8
Tri-Tireworld	3973	6.25	0

Figure 4: Table showing the reachable state set and landmark sizes for benchmark domains. The second column of the table presents the number of reachable states, while the third and fourth columns list the average number of policy and model landmarks (excluding goals). All domain and problem instances were taken from IPC 2006 and 2008.

7.2 Empirical Evaluation

As for the empirical evaluation, we were interested in understanding whether policy landmarks provided any advantage over problem landmarks in terms of identifying more subgoals. Since model landmarks are always a subset of policy landmarks, any additional facts part of the policy landmark should hopefully capture policy specific characteristics. Moreover, we were interested in seeing how many fact landmarks were, in fact, extracted for some prototypical problems. So we selected four standard PPDDL domains from some earlier probabilistic tracks of IPC competitions (International Planning Competition 2011) and five problem instances per domain (except triangle tireworld for which we only used four). We used compilation methods discussed in earlier section to prepare the deterministic domain for policy landmark extraction. Table 4 presents results from this analysis and presents the average policy size, the number of non-initial and non-goal causal landmarks extracted (i.e landmark facts aren't part of the initial state or the goal state) for the policy and for the model as a whole. All policies were generated using an LAO^{*}(Hansen and Zilberstein 2001) implementation and because some of these domains we considered included dead ends, we didn't enforce the proper policy requirement during the evaluation. The landmarks were generated using FastDownward (Helmert 2006) implementation of Keyder *et al.* (2010). For all the domains we considered, we found that policy landmarks do lead to identifying more fact as compared to model landmarks. In fact, for triangle

tireworld problems there were no non-init, non-goal landmarks, while our method was able to identify multiple policy landmarks. In all the domains, we see that the landmark sets does lead to more concise policy summary when compared to the size of the reachable state set.

As an illustration of the kind of summaries we can generate from such PPDDL domain, consider one of the problems from the exploding blocksworld domain. The domain is quite similar to the deterministic blocks world domain, except that now putting blocks on top of another or on the table could potentially lead to it exploding. Once a block explodes you can't place another block on top of it. The specific instance we looked at contained, 10 blocks and had a goal consisting of five facts. Figure 3 (A) presents the policy landmarks and their ordering for the specific policy we considered which allowed for 362 reachable states (Figure 3 (B)). In addition to showing the various intermediate facts and the order in which to achieve them, one interesting fact the landmarks highlight is that it presents the order in which the policy expects the various goal facts to be achieved (highlighted in green).

8 Discussion and Conclusion

In summary, our work presents a policy summarization technique that tries to automatically identify subgoals for a given policy by identifying landmark facts. Towards this end, we formalized the concept of landmarks for MDPs and proposed the idea of policy landmarks. We introduced compilation based procedures for extracting these landmarks and established the relationship between the hierarchy induced by the landmarks with previously studied methods for reasoning with task hierarchies in MDPs. Our user studies suggests that in the absence of task details, people do tend to choose landmarks when asked to select intermediate objectives. Additionally, we also found that in many of the domains choosing to extract policy landmarks does provide us with more information.

One point hinted at but not expanded upon in the current paper is how the summary could be expanded based on user response. For example, from the given set of subgoals, the user might want to know how exactly the policy plays out. Interestingly, we can leverage the exact methods discussed in the paper to generate these lower-level subgoals. When the user wants to drill down, they can be asked to choose from the list of completions for source subgoal. Then we can use the compilation methods developed in the paper to generate new landmarks from the new initial state to the destination subgoal.

One of the issues that we didn't quite cover in the paper is the fact that users may not be interested in all landmarks, but rather ones related to a subset of fluents. For example, an engineer trying to view the policy of an extra-planetary rover may be more interested in seeing landmarks related to fuel-levels and engine performance, rather than ones related to samples collected by the robot. On the other hand, a geologist may be completely oblivious to the details about the robot's engines or batteries. To allows for such differing views, we would need to marry our landmark extraction methods with methods for state abstraction.

Another point to note is that as the underlying model becomes more deterministic, it should start introducing more policy landmarks, with the entire policy turning into a single sequence of states and actions in the extreme case. This would mean that each state in the sequence would be technically a policy landmark. For cases where the task is known to be nearly deterministic, we would recommend starting with model landmarks and switching to policy landmarks only when the user needs more information.

References

Jorge A Baier, Christian Fritz, and Sheila A McIlraith. Exploiting procedural domain control knowledge in state-of-the-art planners. In *ICAPS*, 2007.

Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *IJCAI*, pages 156–163, 2017.

Richard P Cooper and Tim Shallice. Hierarchical schemas and goals in the control of sequential behavior. *Psychological Review*, 2006.

Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

Francesco Donnarumma, Domenico Maisto, and Giovanni Pezzulo. Problem solving as probabilistic inference with subgoaling: explaining human successes and pitfalls in the tower of hanoi. *PLoS computational biology*, 12(4):e1004864, 2016.

Hector Geffner and Blai Bonet. A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(1):1–141, 2013.

Sam Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. *arXiv preprint arXiv:1711.00138*, 2017.

Eric A Hansen and Shlomo Zilberstein. Lao: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62, 2001.

Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 303–312. IEEE, 2017.

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

Jörg Hoffmann, Julie Porteous, and Laura Sebastia. Ordered landmarks in planning. *JAIR*, 22:215–278, 2004.

International Planning Competition. IPC Competition Domains. `https://goo.gl/i35bxc`, 2011.

Leslie Pack Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, pages 167–173, 1993.

Thomas Keller and Patrick Eyerich. A polynomial all outcome determinization for probabilistic planning. In *Twenty-First International Conference on Automated Planning and Scheduling*, 2011.

Emil Keyder, Silvia Richter, and Malte Helmert. Sound and complete landmarks for and/or graphs. In *ECAI*, volume 215, pages 335–340, 2010.

Omar Zia Khan, Pascal Poupart, and James P Black. Minimal sufficient explanations for factored markov decision processes. In *Nineteenth International Conference on Automated Planning and Scheduling*, 2009.

Andrey Kolobov. Planning with markov decision processes: An ai perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–210, 2012.

Isaac Lage, Daphna Lifschitz, Finale Doshi-Velez, and Ofra Amir. Exploring computational user models for agent policy summarization. In *IJCAI*, 2019.

Amy McGovern and Andrew G Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. 2001.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 2018.

Allen Newell, Herbert Alexander Simon, et al. *Human problem solving*, volume 104. Prentice-hall Englewood Cliffs, NJ, 1972.

Rahul Ramesh, Manan Tomar, and Balaraman Ravindran. Successor options: An option discovery framework for reinforcement learning. In *IJCAI*, 2019.

Herbert A Simon and Allen Newell. Human problem solving: The state of the theory in 1970. *American Psychologist*, 26(2):145, 1971.

Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Hierarchical expertise level modeling for user specific contrastive explanations. In *IJCAI*, pages 4829–4836, 2018.

Sarath Sreedharan, Siddharth Srivastava, David Smith, and Subbarao Kambhampati. Why can't you do that hal? explaining unsolvability of planning tasks. In *IJCAI*, 2019.

Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer, 2002.

Nicholay Topin and Manuela Veloso. Generation of policy-level explanations for reinforcement learning. In *AAAI*, 2019.

Sung Wook Yoon, Alan Fern, and Robert Givan. Ff-replan: A baseline for probabilistic planning. In *AAAI*, 2007.

Håkan LS Younes and Michael L Littman. Ppddl1. 0: An extension to pddl for expressing planning domains with probabilistic effects. 2004.

Lin Zhu and Robert Givan. Landmark extraction via planning graph propagation. 2003.