Partially Observable Hierarchical Reinforcement Learning with AI Planning (Student Abstract)

Brandon Rozek[1], **Junkyu Lee**[2], **Harsha Kokel**[2], **Michael Katz**[2], **Shirin Sohrabi**[2]

[1]Rensselaer Polytechnic Institute
[2]IBM Research AI
rozekb@rpi.edu, {Junkyu.Lee,harsha.kokel,michael.katz1}@ibm.com, ssohrab@us.ibm.com

Abstract

Partially observable Markov decision processes (POMDPs) challenge reinforcement learning agents due to incomplete knowledge of the environment. Even assuming monotonicity in uncertainty, it is difficult for an agent to know how and when to stop exploring for a given task. In this abstract, we discuss how to use hierarchical reinforcement learning (HRL) and AI Planning (AIP) to improve exploration when the agent knows possible valuations of unknown predicates and how to discover them. By encoding the uncertainty in an abstract planning model, the agent can derive a high-level plan which is then used to decompose the overall POMDP into a tree of semi-POMDPs for training. We evaluate our agent's performance on the MiniGrid domain and show how guided exploration may improve agent performance.

Introduction

Current techniques in reinforcement learning (RL) attempt to solve a partially observable problem by approximating the Markovian state through a form of agent memory. This is commonly a stack of sequential observations or hidden units of a recurrent function approximator. Assuming that uncertainty is monotonic, one strategy is to fully explore the environment. However, there could be many irrelevant variables; leading to wasted exploration efforts. In this work, to improve sample complexity, we examine the use of AI planning through a fully observable non-deterministic (FOND) model to guide exploration for a given goal.

For example, consider an environment where an agent navigates through rooms with locked doors shown in Figure 2a. To reach the goal location, shown in green, the agent only needs a yellow key. An agent that is motivated to fully explore the environment would first explore all the rooms and then enter the goal location. In contrast, an AI planning guided agent, would only explore the environment until it finds the yellow key. By capturing the goal, possible valuations of unknown variables, and some high-level transition dynamics, an AI planning model can provide a plan for discovering only the necessary information, and hence reduce exploration efforts. We present an approach that leverages a FOND planner to guide the exploration and execution of RL agents to solve the POMDP task.

Background

We address a sparse-reward, goal-oriented POMDP problem represented as $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, O, s_0, G \rangle$, with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function T, a sparse reward function R, observation space O, initial state s_o, and goal condition G. The objective is to find the optimal policy π^* which maximizes the expected return.

A FOND planning task is presented as $\Pi = \langle \mathcal{V}, \mathcal{O}, s'_0, s'_* \rangle$, where $\mathcal{V}$ is a finite set of state variables, $\mathcal{O}$ is a finite set of operators which consists of preconditions (pre(o)) and a list of deterministic effects often expressed in the form eff(o) $=$ $oneof(e_{o,1}, \dots, e_{o,n})$. A planning state s' is a partial state over $\mathcal{V}$. An operator o can be executed at state s'_t if $s'_t \models \text{pre}(o)$ and upon executing the operator an effect $e_{o,i}$ is non-deterministically chosen for the next state s'_{t+1}. Each $e_{o,i}$ is a set of conditional effects. Each conditional effect is represented as $\langle c, l \rangle$ where effect-literals l are applied to a state if the conditional-literals c are satisfied (i.e. $s'_t \models c \implies s'_{t+1} \models l$). Additionally, s'_0 is the initial state, and s'_* is a partial state representing the goal states.

This work extends the Planning Annotated Reinforcement Learning (PaRL) framework (Lee et al. 2022). A PaRL task E is defined as a triple $\langle \mathcal{M}, \Pi, L \rangle$ where $\mathcal{M}$ is a goal-oriented MDP, Π is a $\texttt{SAS}^+$ planning task, and L is a surjective mapping from MDP states $\mathcal{S}$ to planning states $\mathcal{S}'$. The PaRL task is solved by decomposing the MDP $\mathcal{M}$ into semi-MDPs using the planning task Π and defining a separate option policy for each semi-MDP. For each operator $o \in \mathcal{O}$ within Π there is a corresponding operator option $O_o := \langle I_o, \pi_o, \beta_o \rangle$ with $I_o := \{s \in \mathcal{S} | L(s) \models \text{pre}(o)\}$ and $\beta_o := \{s \in \mathcal{S} | L(s) \models (\text{prv}(o) \cup \text{eff}(o))\}$. The prevail prv($o$) is the subset of variables in pre(o) that does not appear in eff(o).

Proposed Approach

In order to extend the PaRL framework to handle partial observability, we look at how to encode uncertainty in AI planning, and from the model how to derive our options for hierarchical reinforcement learning.

Modeling Uncertainty We do not seek to model the entire POMDP using AI planning, but instead we look at capturing the agent's knowledge on the list of unknown predicates and how it might discover them. To do this, we use the FOND

```
(:action move-room
  :parameters (?d - door ?r1 - room ?r2 - room)
  :precondition (and (at-agent ?r1) (unlocked ?d) (
        CONNECTED-ROOMS ?r1 ?r2) (LINK ?d ?r1 ?r2) )
  :effect (and
    (not (at-agent ?r1)) (at-agent ?r2)
    (forall (?k -key)
    (when (not (entered-room ?r2))
    (when (not (discovered ?k))
    (oneof
      ; Yellow Key Present
      (and (at ?k ?r2) (color ?k yellow) (discovered ?k) (
           entered-room ?r2))
      ; Purple Key Present
      (and (at ?k ?r2) (color ?k purple) (discovered ?k) (
           entered-room ?r2))
      ; Key not present
      (entered-room ?r2)
))))))
```

Figure 1: PDDL with discovery effects

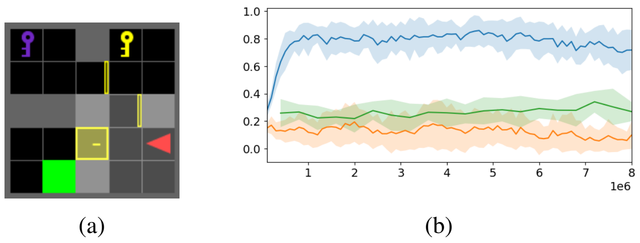

Figure 2: (a) Example environment where it's unnecessary to explore the top left room in order to reach the goal. (b) Success rate (y-axis) over training samples (x-axis) for HFondPlan, PPO, A2C (top to bottom)

planning model. The intuition is that in the incomplete abstract level, the agent's knowledge of unknown predicates is fully observable, and non-deterministic effects encode the potential discovery of subsets of these predicates.

For each predicate $v \in V$, if $v \in s'_0$ then we say that v is true in the initial belief state. Otherwise, we say that v is either false or unknown. To disambiguate between false and unknown, we make use of *discovery predicates*. These special predicates are true when an agent has discovered a subset of unknown state variables in the environment. For example, the agent has a key within their field of view, so it knows its color and location. We can model how an agent might discover these unknown predicates by augmenting existing actions with *discovery effects*. Let $e_{o,*}$ represent the deterministic effects of an operator o before encoding discovery. Then the discovery augmented o_d will have $\mathit{eff}(o_d) = \mathit{oneof}(e_{o,1}, \ldots, e_{o,n})$ where $e_{o,*} \subset e_{o,i}$. A discovery effect is a conditional effect $\langle c, l \rangle$ with a couple additional restrictions on c. First, a discovery effect can only have its conditional-literals satisfied once. This is to prevent cyclic policies in an attempt to obtain a desired non-deterministic effect. In practice, we add $\langle \top, c_{o,f} \rangle$ to every $e_{o,i}$ to account for this. Secondly, every state variable within l must be unknown in s' to prevent inconsistencies in the belief state. Discovery predicates are used to track this. Since it's possible that an agent might not discover information after executing an option, l may be empty.

For example, for any given key in the MiniGrid environment, if we move to a room we have not visited before, then the key can either be present and yellow, present and purple, or not present in the room. The corresponding PDDL encoding is shown in Figure 1. There `(entered-room ?r2)` represents the $c_{o,f}$ condition and `(discovered ?k)` is the discovery predicate associated with the key color and locations.

Defining Options In the POMDP setting, the agent does not have access to a state. In order to derive our planning belief state s', we need to pass a history of observations h into a mapping function L_h. For each operator $o \in \mathcal{O}$, we have a corresponding operator option $\mathcal{O}_o = \langle I_o, \pi_o, \beta_o \rangle$. The initiation set is similar to the PaRL setting and is defined as $I_o := \{s \in \mathcal{S} | L_h(h) \models \text{pre}(o)\}$. We adapt the termination set to account for issues of non-determinism. For an operator o with non-deterministic effects, its execution leads to multiple possible s'. To account for this, we terminate when $L_h(h) \models \bigvee_i \text{prv}(o) \cup e_{o,i}$. The issue comes with empty discovery effects. Lets say during the execution of o, that $L_h(h) \models e_{o,e}$ where $e_{o,e}$ is an empty discovery effect. The option terminates, however, the agent has not learned anything additional about the relevant unknown predicate(s). Including whether they don't hold in the environment. It is possible that at some later time step, the history of observations $L_h(h') \models e_{o,i}$ where $i \neq e$. This will violate our policy, since the compilation assumed a deterministic observation model and the discovery effects cannot execute more than once. For our approach, we rely on replanning to correct this issue. There are alternate approaches, however, we leave experimentation of those to future work.

Preliminary Evaluations and Conclusion

We evaluate our agent on an instance of the MiniGrid environment shown in Figure 2a. The agent only sees a 3x3 window in front of it and the key locations and colors are unknown. For our implementation we use the stable-baselines library, and for the experiment we compare against two reinforcement learning agents PPO and A2C while keeping the default hyperparameters for each. For all agents, we pass in a stack of the last ten observations and set the horizon H to 2048. For our agent, we provide a -0.9 reward on the terminal state of options that violate the prevail, and a reward of $1 - 0.4 * (t_o/H)$ on option success where t_o is the number of steps taken within that option. We generate 4000 environmental seeds for each run, 3900 for training and 100 for evaluation. We perform ten runs with different starting seeds for each agent.

Overall our preliminary evaluation in Figure 2b shows that the existence of a discovery model helps guide the exploration process of a HRL agent and may lead to improved performance in a partially observable task.

References

Lee, J.; Katz, M.; Agravante, D. J.; Liu, M.; Tasse, G. N.; Klinger, T.; and Sohrabi, S. 2022. Hierarchical Reinforcement Learning with AI Planning Models. arXiv:2203.00669.