Guiding Hiearchical Reinforcement Learning in Partially Observable Environments with AI Planning

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 challenge reinforcement learning agents since observations provide an limited view of the environment. This often requires an agent to explore collecting observations to form the necessary state information to complete the task. Even assuming knowledge is monotonic, it is difficult to know when to stop exploration. We integrate AI planning within hierarchical reinforcement learning to aide in the exploration of partially observable environments. Given a set of unknown state variables, their potential valuations, along with which abstract operators may discover them, we create an abstract fully-observable non-deterministic planning problem which captures the agent's abstract belief state. This decomposes the POMDP into a tree of semi-POMDPs based on sensing outcomes. We evaluate our agent's performance on a MiniGrid domain and show how guided exploration may improve agent performance.

Introduction

Observations from an agent's sensors may be noisy or incomplete. Within sequential decision making, this may be represented as a partially observable Markov decision process (POMDP). This has been studied, for the most part, independently in the reinforcement learning (RL) and AI planning (AIP) communities. Within the former, algorithms approximate the Markovian state by introducing some form of memory. For example, providing sequential observations into function approximators (Singh, Jaakkola, and Jordan 1994) and using recurrent function approximators (Bakker 2001). Model-free methods are then used without prior knowledge of transition dynamics to learn the task; mainly at the cost of environment samples. In AI planning, transition dynamics and uncertainty has to be exhaustively modeled in the initial belief state. Contingent planners then finds exact solutions without the need of any samples (Bonet and Geffner 2000). Crafting the initial belief state, however, is often difficult due to complex interactions between unknown state variables. Also, the model may be incomplete when there are exogenous actions. In this paper, we aim to incorporate the advantages of AIP to alleviate the sample complexity of HRL within partially observable environments by decomposing the POMDP into a tree of semi-POMDPs.

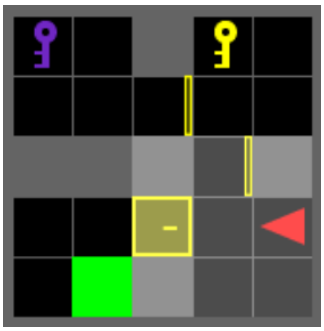

Figure 1: Environment where it's unnecessary to enter the top left room for the needed yellow key.

Assuming that uncertainty in the POMDP is monotonic, one strategy is to explore the environment until the problem becomes fully-observable with respect to the agent's memory. This approach, however, has two issues: irrelevant unknown state variables with respect to the task leads to wasted exploration; and environment deadends aren't guarded against. To alleviate these concerns, we make use of AIP through a fully-observable non-deterministic (FOND) model to guide discovery of relevant state variables for a given goal. For example, consider an environment where an agent navigates through rooms with locked doors as shown in Figure 1. To reach the bottom-left goal, shown in green, the agent needs the yellow key in the top-right room. An agent motivated to fully explore would first enter all the rooms and then enter the goal location. In contrast, an AIP guided agent, would only explore until it finds the yellow key. By capturing the goal, possible valuations of unknown state variables, and some high-level transition dynamics, an AIP model can provide a plan for discovering only the necessary information, hence reducing exploration efforts.

Background

We address a sparse-reward, goal oriented POMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, O, s_0, G \rangle$, with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T : S \times A \times S \mapsto [0, 1]$, sparse reward function $R : S \times A \times S \mapsto \mathbb{R}$, observation space O, initial state s_o, and goal condition G. The objective is to find the optimal policy π^* which maximizes the expected cumulative

discounted reward. POMDPs model time with actions only persisting for a single discrete step, while semi-POMDPs introduce temporally extended actions of various lengths. The Options framework (Sutton, Precup, and Singh 1998) is a HRL approach for modeling temporally extended actions known as *options*. An option is defined as $p = \langle I_p, \beta_p, \pi_p \rangle$ with the set of option initiation states I_p, the set of option termination states β_p, and the option policy π_p.

This work extends the Planning annotated Reinforcement Learning (PaRL) framework defined in Lee et al. (2022). A PaRL task E is the triple $\langle \mathcal{M}, \Pi, L \rangle$ where $\mathcal{M}$ is a goal-oriented MDP, Π is a $\mathtt{SAS}^+$ planning task, and L is a surjective function mapping MDP states $\mathcal{S}$ to planning states $\mathcal{S}'$. An operator $o \in \mathcal{O}$ within Π is the tuple $\langle \mathtt{pre}, \mathtt{eff} \rangle$ where the state variables within $\mathtt{pre}$ denote the preconditions of o while $\mathtt{eff}$ are the state variables assigned after o is applied. The prevail of an operator $\mathtt{prv}(o)$ is the subset of variables in $\mathtt{pre}(o)$ that do not appear in $\mathtt{eff}(o)$. The PaRL task is solved by decomposing the MDP $\mathcal{M}$ into semi-MDPs through the planning task Π. For every operator $o \in \mathcal{O}$ within Π there is a corresponding operator option $O_o := \langle I_o, \pi_o, \beta_o \rangle$ with $I_o := \{s \in \mathcal{S} \mid L(s) \models \mathtt{pre}(o)\}$ and $\beta_o := \{s \in \mathcal{S} \mid L(s) \models (\mathtt{prv}(o) \cup \mathtt{eff}(o))\}$. Additionally, a single goal option is defined with the initiation set $I_G = \{s \in S \mid L(s) \vDash G'\}$ where G' is the goal in Π, and the option terminates when the MDP goal G is met.

Instead of using a classical $\mathtt{SAS}^+$ planning task, we model the abstract belief state of the agent through a FOND planning model. A FOND planning task is defined as $\Pi = \langle \mathcal{V}, \mathcal{O}, s'_0, s'_* \rangle$, with the finite set of state variables $\mathcal{V}$, finite set of operators $\mathcal{O}$, initial state s'_0, and partial state representing the goal s'_*. A planning state s' is a partial state over $\mathcal{V}$. An operator $o \in \mathcal{O}$ consist of preconditions ($\mathtt{pre}(o)$) and a list of deterministic effects expressed as $\mathtt{eff}(o) = \mathit{oneof}(e_{o,1}, \ldots, e_{o,n})$. An operator o is *applicable* at state s'_t iff $s'_t \models \mathtt{pre}(o)$. After executing o at state s'_t, the effect $e_{o,i}$ is non-determistically chosen for the next state s'_{t+1}. Each $e_{o,i}$ is a set of conditional effects $\langle c, l \rangle$ where the effect-literals l are applied to a state if the conditional-literals c are satisfied. That is, $s'_t \models c \implies s'_{t+1} \models l$.

Related Work

Reward Machines (RMs) in Icarte et al. (2022) is a HRL approach which represents the reward function as a finite state automata (FSA). The FSA is then used to automatically create subproblems for offline learning. A closer look at RMs under partial observability was presented in Icarte et al. (2023). Within it, they define a *perfect reward machine* which is a RM whose state combined with an environment observation turns the POMDP Markovian. When defining an RM, the user needs to provide a labeling function L which takes an observation and detects the presence of propositional symbols. Their algorithm takes L and traces from the environment to automatically find a RM which captures the necessary (but not sufficient) conditions for being perfect if one exists. In our work, the user provides high-level transition dynamics via a FOND model. This defines the conditions in which operators are applicable, and the effects of executing operators on an abstract planning state. While our approach does not guarantee a memoryless decomposition, we do not place any additional restrictions on the FOND model nor require samples to learn an ideal representation.

Similar to our work, Gordon, Fox, and Farhadi (2019) provide a HRL approach that uses AIP. They have a hierarchical meta-controller which chooses between direct controllers such as the planner, stopper, explorer, navigator, and scanner. While their work was tested under partially observable environments, their planner controller treats the problem as fully-observable and relies on replanning which does not account for deadends in the environment.

The following related work only consider the fully-observable case. The Logical Options framework (Araki et al. 2021) is a HRL framework that uses Linear Temporal Logic (LTL) to automatically create subproblems from the MDP. It does this by partitioning the propositions within the LTL specification into subgoals, safety propositions, and event propositions. The subgoals are then used to derive the options. There is a strong line of research in goal-augmented MDPs (Qiu, Mao, and Zhu 2023; Kantaros 2022; Kalagarla et al. 2022) where the objective is to satisfy some LTL formula at the goal condition with a high probability. Lastly, other research using AIP within HRL under fully observable settings include (Lyu et al. 2019; Illanes et al. 2020; Yang et al. 2018; Kokel et al. 2023).

Approach

Adapting the PaRL framework to handle partial observability, we define a PO-PaRL task. We'll discuss how the user encodes uncertainty within the AIP annotation, and from that how options will be derived for use in HRL.

Modeling Uncertainty We do not need to model the entire POMDP within the FOND planning model. Instead, we seek to capture some high-level dynamics, the agents knowledge on a set of relevant unknown state variables, and how it might discover these variables valuation.

Contingent planning takes an incomplete initial state, along with sensing operators which reveal a previously unknown state variable, and produces a branching plan which splits based on the outcome of sensing. A line of research shows how to compile a contingent problem into a FOND problem (Albore, Palacios, and Geffner 2009; Bonet and Geffner 2011; Muise, Belle, and McIlraith 2014). In these works, sensing outcomes are always assumed to succeed. The agent is guaranteed to know the valuation of a previously unknown state variable. However in our work sensing outcomes may fail as the planning model may be incomplete with respect to the POMDP. Therefore, we present an alternative encoding which uses the discovery set $\mathcal{D}$ to augment deterministic operators with *discovery effects*. These effects capture the potential non-deterministic assignment of unknown state variables. For example, returning to the maze environment in Figure 1, consider the operator `(move-room r1 r2)`. When an agent moves between rooms, it may discover in the new room that (1) a yellow key exists, (2) a purple key exists, or (3) nothing. These potential discoveries are then encoded as non-deterministic effects for the augmented operator.

The intuition behind using a FOND planning model is that the agent's knowledge of unknown state variables are fully observable. In other words, the agent knows what it does not know. One approach to encoding an agent's knowledge of a state variable's v assignment to l is to create two new ground atomic formulae Kv_l and $K\neg v_l$ which represents whether the agent knows that the assignment to l is true or false respectively. For simplicity, we rely on an alternate encoding. For each variable $v \in V$, we say that it's assignment v_l is true in the initial abstract belief state if $s'_0 \models v_l$. Otherwise it's assignment is either false or unknown. To disambiguate between false and unknown, we make use of *discovery predicates*. These special predicates are defined for every unknown state variable. The discovery predicate for a given state variable is marked as true if the agent has discovered the valuation of that state variable. For example, let's consider the color of the key k1. In our example this is initially unknown, so the discovery predicate (`discovered-color k1`) is created and within the initial state $s'_0 \models \neg$(`discovered-color k1`). Additionally given the potential colors of yellow and purple, $s'_0 \models \neg$(`color k1 yellow`) $\wedge$ $\neg$(`color k1 purple`). Now let's say that the agent discovers that k1 is yellow at the abstract planning state s'_t. Then, $s'_t \models$ (`discovered-color k1`) $\wedge$ (`color k1 yellow`) $\wedge \neg$(`color k1 purple`).

Let us formalize the discovery set $\mathcal{D}$ as the set of discovery tuples $d = \langle o, V_o, F_o \rangle$ where o is the operator to be augmented, V_o is the tuple of discoverable state variables after executing o, and F_o is the set of potential assignments for each $v \in V_o$. As an example, consider $o =$ (`move-room r1 r2`), $V_o = \{$k1-color, k1-loc$\}$, and $F_o = \{$(yellow, r2), (purple, r2), (_, _)$\}$. This means that after executing the o operator, the agent may discover that k1 is yellow and at room r2, k1 is purple and at room r2, or discover no new information. From our discovery set $\mathcal{D}$, we augment operators o within the discovery tuples with *discovery effects*. Let $e_{o,*}$ be the deterministic effects of an operator $o \in \langle o, V_o, F_o \rangle$ before encoding discovery. In our example, after executing (`move-room r1 r2`) the agent is no longer at r1, and is at r2. That is, $e_{o,*} = \{\neg$(`at-agent r1`), (`at-agent r2`)$\}$. The discovery augmented operator $o_d \in \Pi$ will then have $\mathit{eff}(o_d) = \mathit{oneof}(e_{o,1}, \ldots, e_{o,n})$ where $e_{o,*} \subset e_{o,i}$. Each discovery effect $e_{o,i}$ is a conditional effect $\langle c, l \rangle$ that corresponds to a set of potential valuations f within F_o. The effect-literals l make true the valuations in f as well as any discovery predicates corresponding to a nonempty assignment of a state variable $v \in V_o$ within f. For example, let $e_{o,1}$ be the discovery effect that captures finding that k1 is yellow and is within room r2. Then, $l \models$ (`at k1 r2`) $\wedge$ (`color k1 yellow`). Additionally, since the k1's location and color have a nonempty assignment $l \models$ (`discovered-loc k1`) $\wedge$ (`discovered-color k1`). The conditional-literals c must ensure the following two restrictions. First, a discovery tuple may only have one of its corresponding discovery effects fired. This is to prevent cyclic policies in an attempt to obtain a desired nondeterministic effect. In practice, we create a new ground atomic formula $c_{o,f}$ which we add to the l-literals of every $e_{o,i}$ and we require that $c_{o,f}$ is false for all c-literals

```
(: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-loc ?k))
    (when (not (discovered-color ?k))
    (oneof
      ; Yellow Key Present
      (and (at ?k ?r2) (color ?k yellow) (discovered-loc ?k) (
           discovered-color ?k) (entered-room ?r2))
      ; Purple Key Present
      (and (at ?k ?r2) (color ?k purple) (discovered-loc ?k) (
           discovered-color ?k) (entered-room ?r2))
      ; Key not present
      (entered-room ?r2)
))))))
```

Figure 2: PDDL with discovery effects

in $e_{o,i}$ that have a non-empty assignment. For our example, $c_{o,f} =$ (`entered-room r2`). Secondly, every state variable within l must be unknown in the current state when the conditional effect fires to prevent inconsistencies in the planning state. This means that the current planning state cannot make true any discovery predicates corresponding to the state variables within V_o. The corresponding PDDL encoding for our example is shown in Figure 2.

Definition 1 *A PO-PaRL task E is the triple $\langle \mathcal{M}, \Pi, L, \mathcal{D} \rangle$ with POMDP $\mathcal{M}$, FOND planning task Π, surjective function mapping history of observations to an abstract planning state L, and discovery set $\mathcal{D}$.*

Deriving Options Given a FOND planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s'_0, s'_* \rangle$ within the PO-PaRL task $E = \langle \mathcal{M}, \Pi, L, \mathcal{D} \rangle$, we derive the options used within our HRL algorithm. Let h be a history, a sequence of observations, in $\mathcal{M}$. With a slight abuse of notation, we use $h \in S$ to represent the state after said sequence of observations. As is the case in PaRL, we define a goal option p_g with the initiation set $I_g = \{h \in S \mid L(h) \models s'_*\}$. The option p_g terminates when the POMDP goal G is reached. Additionally for each operator $o \in \mathcal{O}$, there is a corresponding operator option $p_o = \langle I_o, \pi_o, \beta_o \rangle$. The initiation set is defined as $I_o := \{h \in S \mid L(h) \models \mathtt{pre}(o)\}$. If the operator is not augmented with discovery effects, then it will have a deterministic effect. The termination set for such an operator is $\beta_o = \{h \in S \mid L(h) \models (\mathtt{prv}(o) \cup \mathtt{eff}(o))\}$. On the other hand if an operator o is augmented with discovery effects, then it will have a non-deterministic effect. To account for this, our current approach is to terminate the option when it encounters one of the deterministic effects $e_{o,i}$ within $\mathit{oneof}(e_{o,1}, \ldots, e_{o,n})$. That is, $\beta_o = \{h \in S \mid L(h) \models \bigvee_i (\mathtt{prv}(o) \cup e_{o,i})\}$. Recall that since the planning model may be incomplete with respect to the POMDP that sensing may fail. A discovery effect $e_{o,f}$ which captures this will have effect-literals that lead to no new information gain. This means that the effect-literals only contains $c_{o,f}$ and the literals from the original deterministic effect $e_{o,*}$. A consequence is that an operator may terminate prematurely since $e_{o,f}$ is a subset of all other discovery effects $e_{o,i}$. For exam-

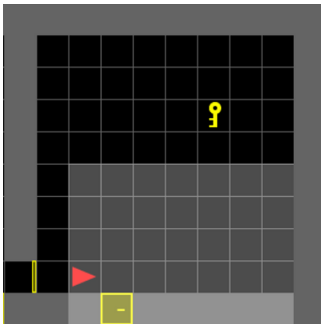

Figure 3: Environment where the agent may prematurely terminate the option concluding that no unknown state variables were discovered.

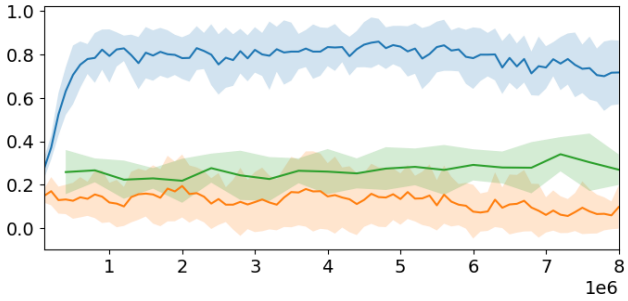

Figure 4: Success rate (y-axis) over training samples (x-axis) for HFondPlan, PPO, A2C (top to bottom)

ple in Figure 3, the agent walks into a room and a key exists in that room; however, the key is not within it's field of view. With our current option termination approach, the agent will conclude that there is not a key in that room, and proceed onto the next option within the HRL algorithm.

There are two alternatives to the termination set of discovery-augmented operator options that are worth investigating. The first is to include in the mapping function L a mechanism that guarantees given a history of observations that some state variable would not be exogenously discovered. For example in our maze problem, we can introduce a mapping which makes true `(fully-explored ?r)` so that the option only terminates after the agent fully explores the room. This may be difficult for the domain modeler to include depending on the environment. The second alternative is to include some number of time steps t that must pass after satisfying the other conditions in order for the option to terminate. This does not guarantee that the agent wouldn't later exogenously discover the state variable, however, it may be beneficial empirically.

Collecting Rollouts We start by calling a FOND planner over Π within the PO-PARL task E. In our experiments, we use the planner from Muise, McIlraith, and Belle (2014). From the generated policy, we create an option for every operator listed. We additionally create the goal option p_g. These options are trained online. If the agent is not already within an option, then given the current history h, the abstract belief state $L(h)$ determines the next option. This results in four cases. (1) The FOND planner fails to generate an abstract policy. This may be due to *observability deadends* which arise from incomplete discovery models or premature option termination. From here, a random primitive action $a \in \mathcal{A}$ is selected. (2) The planning goal is satisfied from the current abstract belief state. From here, the goal option p_g is initiated. (3) The current abstract belief state has a corresponding operator p_o within the FOND policy. This initiates the option p_o. (4) The current abstract belief state is not captured within the FOND policy. This may be due to state variables being exogenously discovered. From here, we call the FOND planner setting the initial belief state to the current abstract belief state.

Within an option, each step outside of option termination has zero reward. When an option terminates, if the option is p_g, then the reward is the environment reward. Otherwise, the terminating reward includes a positive reward modulated by the number of steps taken in addition to a negative reward if the prevail isn't maintained. We store the transitions from each option in their corresponding buffers for use in training and repeat until the size matches the rollout length.

Preliminary Evaluations and Conclusion

We evaluate our agent HFondPlan on an instance of the MiniGrid environment shown in Figure 1. The agent only sees a 3x3 window ahead of it and the key locations and colors are initially unknown. We use the stable-baselines3 library by Raffin et al. (2021) as the basis for HFondPlan in which the underlying PPO implementation is used to train each of the options. We also use the reinforcement learning agents PPO and A2C from the library for our empirical evaluation 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 terminal option states which violate the prevail, and a reward of $1 - 0.4 * (t_0/H)$ on option success with t_o being the number of steps taken within that option. We generate 4000 environmental seeds for each run, with 3900 for training and 100 for evaluation. We perform ten runs with different starting seeds for each agent. Overall our preliminary results shown in Figure 4 show an improved performance with the presence of a discovery model.

In this paper we presented the PO-PaRL framework, an augmentation of the PaRL framework that handles partially observable tasks. We discussed how the discovery set $\mathcal{D}$ creates a FOND planning task from a set of deterministic operators, and how our approach can also model incomplete sensing. Future work includes further investigations into handling sensing failures. That is, other alternatives to option termination, and what policy an agent should follow when it hits an observability deadend.

References

Albore, A.; Palacios, H.; and Geffner, H. 2009. A Translation-Based Approach to Contingent Planning. In Boutilier, C., ed., *IJCAI 2009, Proceedings of the 21st*

International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009, 1623–1628.

Araki, B.; Li, X.; Vodrahalli, K.; DeCastro, J. A.; Fry, M. J.; and Rus, D. 2021. The Logical Options Framework. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, 307–317. PMLR.

Bakker, B. 2001. Reinforcement Learning with Long Short-Term Memory. 1475–1482.

Bonet, B.; and Geffner, H. 2000. Planning with Incomplete Information as Heuristic Search in Belief Space. In Chien, S. A.; Kambhampati, S.; and Knoblock, C. A., eds., *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, Breckenridge, CO, USA, April 14-17, 2000*, 52–61. AAAI.

Bonet, B.; and Geffner, H. 2011. Planning under Partial Observability by Classical Replanning: Theory and Experiments. In Walsh, T., ed., *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 1936–1941. IJCAI/AAAI.

Chevalier-Boisvert, M.; Dai, B.; Towers, M.; Perez-Vicente, R.; Willems, L.; Lahlou, S.; Pal, S.; Castro, P. S.; and Terry, J. 2023. Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks.

Gordon, D.; Fox, D.; and Farhadi, A. 2019. What Should I Do Now? Marrying Reinforcement Learning and Symbolic Planning. *CoRR*, abs/1901.01492.

Icarte, R. T.; Klassen, T. Q.; Valenzano, R.; Castro, M. P.; Waldie, E.; and McIlraith, S. A. 2023. Learning reward machines: A study in partially observable reinforcement learning. *Artif. Intell.*, 323: 103989.

Icarte, R. T.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2022. Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning. *J. Artif. Intell. Res.*, 73: 173–208.

Illanes, L.; Yan, X.; Icarte, R. T.; and McIlraith, S. A. 2020. Symbolic Plans as High-Level Instructions for Reinforcement Learning. In Beck, J. C.; Buffet, O.; Hoffmann, J.; Karpas, E.; and Sohrabi, S., eds., *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, 540–550. AAAI Press.

Kalagarla, K. C.; Dhruva, K.; Shen, D.; Jain, R.; Nayyar, A.; and Nuzzo, P. 2022. Optimal control of partially observable Markov decision processes with finite linear temporal logic constraints. In Cussens, J.; and Zhang, K., eds., *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, 949–958. PMLR.

Kantaros, Y. 2022. Accelerated Reinforcement Learning for Temporal Logic Control Objectives. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*, 5077–5082. IEEE.

Kokel, H.; Natarajan, S.; Ravindran, B.; and Tadepalli, P. 2023. RePReL: a unified framework for integrating relational planning and reinforcement learning for effective abstraction in discrete and continuous domains. *Neural Comput. Appl.*, 35(23): 16877–16892.

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.

Lyu, D.; Yang, F.; Liu, B.; and Gustafson, S. 2019. SDRL: Interpretable and Data-Efficient Deep Reinforcement Learning Leveraging Symbolic Planning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 2970–2977. AAAI Press.

Muise, C.; McIlraith, S. A.; and Belle, V. 2014. Non-Deterministic Planning With Conditional Effects. In *The 24th International Conference on Automated Planning and Scheduling*.

Muise, C. J.; Belle, V.; and McIlraith, S. A. 2014. Computing Contingent Plans via Fully Observable Non-Deterministic Planning. In Brodley, C. E.; and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, 2322–2329. AAAI Press.

Qiu, W.; Mao, W.; and Zhu, H. 2023. Instructing Goal-Conditioned Reinforcement Learning Agents with Temporal Logic Objectives. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; and Dormann, N. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *J. Mach. Learn. Res.*, 22: 268:1–268:8.

Singh, S. P.; Jaakkola, T. S.; and Jordan, M. I. 1994. Learning Without State-Estimation in Partially Observable Markovian Decision Processes. In Cohen, W. W.; and Hirsh, H., eds., *Machine Learning, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, July 10-13, 1994*, 284–292. Morgan Kaufmann.

Sutton, R. S.; Precup, D.; and Singh, S. 1998. Intra-Option Learning about Temporally Abstract Actions. In Shavlik, J. W., ed., *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, 556–564. Morgan Kaufmann.

Yang, F.; Lyu, D.; Liu, B.; and Gustafson, S. 2018. PEORL: Integrating Symbolic Planning and Hierarchical Reinforcement Learning for Robust Decision-Making. In Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 4860–4866. ijcai.org.

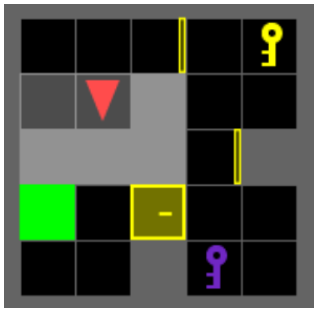

Figure 5: MiniGrid Experiment Environment

Environment Description

We use the MiniGrid environment from Chevalier-Boisvert et al. (2023) to design a task where there are four rooms in a two by two configuration as shown in Figure 5. The initial state of the environment starts the agent off in the top left room, and the goal of the POMDP is for the agent to reach the green square in the bottom left room. In order to achieve this objective, the agent needs to pick up the yellow key and use it to open the yellow locked door. In the initial abstract belief state of the agent, the agent knows that two keys exist, but not where each of the keys are located or what their colors are. The observation at a given step is a 3x3 tile view where (from the perspective of the agent) the tiles two cells ahead are visible as well as one tile on each side of the agent. A tile is represented as $\langle T, C, S \rangle$ where T is the type of object contained in that cell (if any), C is the object's color, and S is the MiniGrid specific environment state of the object at that cell. Hence for an individual observation, there are 27 dimensions. For each agent, we stacked the last ten observations for training and evaluation of the policy functions which give a total of 270 dimensions. The discrete action space has seven possible actions: left, right, forward, pickup, drop, toggle, and done. We randomly sample the agent direction, position, key locations, door locations, and goal locations when creating the environment. For our experiments, we generate 4000 environmental seeds, 3900 for training and 100 for testing.

Experimentation

We considered three agents: HFondPlan (ours), PPO, and A2C using the stable-baselines3 library version 1.2.0. For every agent, we ran ten experiments each with different starting seeds. The agents in each experiment were evaluated every 100,000 training samples testing whether the agent can reach the goal. The success rate was averaged over a hundred different testing environments. The agents were trained for eight million training samples. We use the actor-critic neural network architecture for each agent. The feature extraction layer flattens the input, then passes it to the single hidden layer of size 135, and the output layers compute the policy and value estimates. We used the default hyperparameters presented by stable-baselines3 and include them in Table 1 and 2 for posterity.

Rollout Length	9600
Batch Size	64
Discount Factor (γ)	0.99
Entropy Coeficient	0.001
Learning Rate (α)	3e-4
Epochs	10
λ_{GAE}	0.99
Clip Range	0.2
Value Coefficient	0.5
Max Grad Norm	0.5

Table 1: Hyperparameters for HFondPlan and PPO

Rollout Length	5
Discount Factor (γ)	0.99
Entropy Coefficient	0.001
Learning Rate (α)	7e-4
λ_{GAE}	1
Value Coefficient	0.5
Max Grad Norm	0.5

Table 2: Hyperparameters for A2C