PEORL: Integrating Symbolic Planning and Hierarchical Reinforcement Learning for Robust Decision-Making

Fangkai Yang[1], **Daoming Lyu**[2], **Bo Liu**[2], **Steven Gustafson**[1]
[1] Maana Inc., Bellevue, WA, USA
[2] Auburn University, Auburn, AL, USA
fyang@maana.io, daoming.lyu@auburn.edu, boliu@auburn.edu, sgustafson@maana.io

Abstract

Reinforcement learning and symbolic planning have both been used to build intelligent autonomous agents. Reinforcement learning relies on learning from interactions with real world, which often requires an unfeasibly large amount of experience. Symbolic planning relies on manually crafted symbolic knowledge, which may not be robust to domain uncertainties and changes. In this paper we present a unified framework *PEORL* that integrates symbolic planning with hierarchical reinforcement learning (HRL) to cope with decision-making in a dynamic environment with uncertainties. Symbolic plans are used to guide the agent's task execution and learning, and the learned experience is fed back to symbolic knowledge to improve planning. This method leads to rapid policy search and robust symbolic plans in complex domains. The framework is tested on benchmark domains of HRL.

1 Introduction

Reinforcement learning (RL) [Sutton and Barto, 1998] and symbolic planning [Cimatti *et al.*, 2008] have both been used to build autonomous agents that behave intelligently in the real world. An *RL agent* relies on interactions with the environment to achieve its optimal behavior, without the need of prior knowledge. Building upon the model of Markov Decision Process (MDP), the *policy*, i.e., a mapping from a state to an action, can be learned via a number of trial-and-error. With the recent development of deep learning, such methods can lead to a highly adaptive and robust agent [Mnih *et al.*, 2015], but may often rely on an unfeasibly huge amount of experience. On the other hand, a large body of work on symbolic task planning of mobile robots exists [Hanheide *et al.*, 2015; Chen *et al.*, 2016; Khandelwal *et al.*, 2017], where a *planning agent* carries prior knowledge of the dynamic system, represented in a formal, logic-based language such as PDDL [McDermott *et al.*, 1998] or an action language [Gelfond and Lifschitz, 1998] that relates to logic programming under answer set semantics (answer set programming) [Lifschitz, 2008]. The agent utilizes a symbolic planner, such as a PDDL planner FastDownward [Helmert, 2006] or an answer set solver Clingo [Gebser *et al.*, 2012] to generate a sequence of actions to achieve its goal. The pre-defined, manually crafted symbolic representation may not completely capture all domain details, and domain uncertainties and execution failures are handled by execution monitoring and re-planning. A planning agent does not require a large number of trial-and-error (which is quite expensive for real robots) to behave reasonably well, but it may not be robust enough to domain uncertainties, change and some reward structure that is only available through execution and learning. As planning and reinforcement learning are important and complementary aspects of intelligent behavior, combining the two paradigms to bring out the best of both worlds is quite attractive. In this paper, we focus on developing such a framework where an agent (i) utilizes a symbolic representation to generate plans that guide reinforcement learning , and (ii) leverages learned experience to enrich symbolic knowledge and improve planning.

Topics (i) and (ii) above have been studied separately by researchers from different communities. (i) has been studied in a broader sense by the RL community based on the notion of hierarchical reinforcement learning (*HRL*) [Barto and Mahadevan, 2003]. The HRL framework has a two-level structure: the lower level is called the (primitive) action level, with the primitive actions as defined in the MDP setting, and the higher level is called the task level, or termed as the *option* level [Sutton *et al.*, 1999]. An option is a temporal abstraction of actions specified by a policy and a termination condition, and guides learning such that at any state, only the primitive actions specified by the option are considered. Symbolic plans play similar roles [Ryan, 2002; Leonetti *et al.*, 2016], but these work does not embrace the hierarchical aspect of options: symbolic actions are 1-1 corresponding to primitive actions in the underlying MDP. Nor do they dynamically discover new plans and options to improve learning. For (ii), basic learning models such as relational decision trees [Jiménez *et al.*, 2013] or weighted exponential average [Khandelwal *et al.*, 2014] were used to improve planning through execution experiences, but they are not as expressive or general as a general RL framework.

Aiming at building an agent that can unify planning and RL for robust decision-making, this paper advances both lines of research by integrating symbolic planning using action language $\mathcal{BC}$ [Lee *et al.*, 2013] with hierarchical R-learning [Schwartz, 1993; Mahadevan, 1996]

through a *Planning–Execution–Observation–Reinforcement-Learning* (PEORL) framework. R-learning is an important family of reinforcement learning paradigm that characterizes finite horizon average reward, and is shown to be particularly suitable for planning and scheduling tasks. In PEORL, we use $\mathcal{BC}$ to represent commonsense knowledge of actions and constraint answer set solver CLINGCON [Banbara *et al.*, 2017] to generate a symbolic plan, given an initial state and a goal. The symbolic plan is then mapped to a deterministic sequence of stochastic options to guide RL. R-learning iterates on two values: the average-adjusted reward R and the cumulative average reward, or termed as the gain reward ρ. While R-values indicate the learned policy, ρ-values can be effectively used by CLINGCON to generate an improved symbolic plan with better quality, in terms of the cumulative gain reward. The improved plan is mapped to new options, which further guide R-learning to continue, until no better symbolic plan can be found. The framework is empirically evaluated on two benchmarks: Taxi domain [Barto and Mahadevan, 2003] and Gridworld [Leonetti *et al.*, 2016]. In our experiments, when the algorithm terminates, the optimal symbolic plan is returned.

The contribution of this paper is summarized as follows:

- To advance learning capability of agents, to the best of our knowledge, this is the first work using symbolic planning for option discovery in HRL. The PEORL agent outperforms RL agent and HRL agent by returning policies of significantly larger cumulative reward.
- To advance planning capability of agents, to the best of our knowledge, this is the first work where symbolic planning leverages R-learning to improve its robustness. The PEORL agent outperforms planning agent by discovering a new state that leads to extra reward and reducing the number of execution failure.

The paper is organized as follows. After a brief review of action language $\mathcal{BC}$ and HRL in Section 2, we present the framework formulation in Section 3 and the main algorithm in Section 4. Experimental evaluation results are shown in Section 5. Related work is discussed in Section 6.

2 Preliminaries

Action Language and Symbolic Planning. An *action description* D in the language $\mathcal{BC}$ [Lee *et al.*, 2013] includes two kinds of symbols, *fluent constants* that represent the properties of the world, denoted as $\sigma_F(D)$, and *action constants*, denoted as $\sigma_A(D)$. A fluent atom is an expression of the form $f = v$, where f is a fluent constant and v is an element of its domain. For boolean domain, denote $f = \mathbf{t}$ as f and $f = \mathbf{f}$ as $\sim f$. An action description is a finite set of *causal laws* that describe how fluent atoms are related with each other in a single time step, or how their values are changed from one step to another, possibly by executing actions. For instance, (A **if** $A_1, \ldots, A_m$) is a *static law* that states at a time step, if $A_1, \ldots, A_m$ holds then A is true. Another static law (**default** $f = v$) states that by default, the value of f equals v at any time step. (a **causes** A_0 **if** $A_1, \ldots, A_m$) is a *dynamic law*, stating that at any time step, if $A_1, \ldots, A_m$ holds, by executing action a, A_0 holds in the next step. (**nonexecutable** a **if** $A_1, \ldots, A_m$) states that at any step, if $A_1, \ldots, A_m$ holds, action a is not executable. Finally, the dynamic law (**inertial** f) states that by default, the value of fluent f does not change from one step to another, formalizing the *commonsense law of inertia* that addresses the frame problem.

An action description captures a dynamic transition system. A *state* s is a complete set of fluent atoms, and a transition is a tuple $\langle s_1, a, s_2 \rangle$ where s_1, s_2 are states and a is a (possibly empty) set of actions. The semantics of D is defined by a translation into a set of answer set programs $PN_l(D)$, for an integer $l \geq 0$ stating the maximal steps of transition. It is shown that all answer sets of $PN_0(D)$ correspond to all states in the transition system, and all answer sets of $PN_l(D)$ correspond to all transition paths Π of length l, of the form $\langle s_1, a_1, \ldots, a_{l-1}, s_l \rangle$ (or equivalently, $\Pi = \bigcup_1^{l-1} \langle s_i, a_i, s_{i+1} \rangle$) [Lee *et al.*, 2013, Theorems 1, 2]. Let I and G be states. The triple (I, G, D) is called a planning problem. (I, G, D) has a plan of length $l-1$ iff there exists a transition path of length l such that $I = s_1$ and $G = s_l$. Throughout the paper, we use Π to denote both the plan and the transition path by following the plan.

Due to the semantic definition above, automated planning with an action description in $\mathcal{BC}$ can be achieved by an answer set solver, and the output answer sets encode the transition paths that solve the planning problem.

R-learning for Average Reward. A Markov Decision Process (MDP) is defined as a tuple $(\mathcal{S}, \mathcal{A}, P^a_{ss'}, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of symbols denoting states and actions, the transition kernel $P^a_{ss'}$ specifies the probability of transition from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$, $r(s, a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is a reward function bounded by $r_{\max}$, and $0 \leq \gamma < 1$ is a discount factor. A solution to an MDP is a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ that maps a state to an action. RL concerns on learning a near-optimal policy by executing actions and observing the state transitions and rewards, and it can be applied even when the underlying MDP is not explicitly given, a.k.a, model-free policy learning.

To evaluate a policy π, there are two types of performance measures: the expected discounted sum of reward for infinite horizon problems and the expected un-discounted sum of reward for finite horizon problems. In this paper we adopt the latter metric defined as $J^\pi_{\text{avg}}(s) = \mathbb{E}[\sum_{t=0}^{T} r_t | s_0 = s]$. We define the *gain reward* $\rho^\pi(s)$ reaped by policy π from s as

$$\rho^\pi(s) = \lim_{T \to \infty} \frac{J^\pi_{\text{avg}}(s)}{T} = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}[\sum_{t=0}^{T} r_t].$$

R-learning [Schwartz, 1993; Mahadevan, 1996] is a model-free value iteration algorithm that can be used to find the optimal policy for average reward criteria. At the t-th iteration (s_t, a_t, r_t, s_{t+1}), update:

$$R_{t+1}(s_t, a_t) \xleftarrow{\alpha_t} r_t - \rho_t(s_t) + \max_a R_t(s_{t+1}, a),$$
$$\rho_{t+1}(s_t) \xleftarrow{\beta_t} r_t + \max_a R_t(s_{t+1}, a) - \max_a R_t(s_t, a)$$

where α_t, β_t are the learning rates, and $a_{t+1} \xleftarrow{\alpha} b$ denotes the update law as $a_{t+1} = (1 - \alpha)a_t + \alpha b$.

Hierarchical Reinforcement Learning. Compared with regular RL, hierarchical reinforcement learning (HRL) [Barto and Mahadevan, 2003] specifies on real-time-efficient decision-making problems over a series of tasks. An MDP can be considered as a flat decision-making system where the decision is made at each time step. On the contrary, humans make decisions by incorporating temporal abstractions. An option is temporally extended course of action consisting of three components: a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, a termination condition $\beta : \mathcal{S} \mapsto [0, 1]$, and an initiation set $\mathcal{I} \subseteq \mathcal{S}$. An option (I, π, β) is available in state s_t iff $s_t \in I$. After the option is taken, a course of actions is selected according to π until the option is terminated stochastically according to the termination condition β. With the introduction of options, the decision-making has a hierarchical structure with two levels, where the upper level is the option level (also termed as task level) and the lower level is the (primitive) action level. Markovian property exists among different options at the option level.

3 PEORL Framework

In this section we formally define PEORL framework. A PEORL theory is a tuple $(I, G, D, \mathcal{S}, \mathcal{A}, r, \gamma, \mathbb{F}_A)$. It contains the elements from a symbolic planning problem, an MDP and how they are linked with each other:

- I, G, D form a symbolic planning problem, where I is the initial state, G is a *PEORL goal* that consists of a goal state condition and a linear constraint, and D is a *PEORL action description* in the language of $\mathcal{BC}$.
- $\mathcal{S}, \mathcal{A}, r, \gamma$ form part of an MDP. $\mathcal{A}$ is a set of action symbols in MDP space. We use small letters with tilde, such as $\tilde{a}$ to denote its element, and assume $|\sigma_A(D)| \leq |\mathcal{A}|$. $\mathcal{S}$ is a set of state symbols in MDP space. It contains *simple state* symbols of form s which are 1-1 correspondent to (symbolic) states of $T(D)$. Due to such correspondence, we also use a state of $T(D)$, i.e., a set of fluent atoms in $\sigma_F(D)$ to denote a simple state symbol in $\mathcal{S}$. Furthermore, $\mathcal{S}$ also contains the *MDP state symbols*, denoting a state obtained by applying MDP action $\tilde{a}$. r is a reward function such that $r(s, a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. $0 \leq \gamma \leq 1$ is a discount factor.
- A *symbolic transition–option* mapping $\mathbb{F}_A$ that translates a symbolic transition path $\Pi \subseteq T(D)$ into a set of options.

Some components are further explained as follows.

3.1 Symbolic Planning Problem

A *PEORL action description* D is written in $\mathcal{BC}$ and contains a specific set of causal laws formulating *plan quality* accrued from executing a course of actions:

- For any state of $T(D)$ that contains atoms $\{A_1, \ldots, A_n\}$, D contains static laws of the form

$$s \textbf{ if } A_1, \ldots, A_n, \text{ for simple state } s \in \mathcal{S}. \quad (1)$$

- Introduce new fluent symbols of the form $\rho(s, a)$ to denote the gain reward at state s following action a. D contains a static law stating by default, the gain reward is a sufficiently large number, denoted as *INF*, to promote exploration when necessary:

$$\textbf{default } \rho(s, a) = \mathit{INF}, \text{ for simple state } s \in \mathcal{S}, a \in \sigma_A(D).$$

- Use fluent symbol *quality* to denote the cumulative gain reward reward of a plan, termed as *plan quality*. D contains dynamic laws of the form

$$a \textbf{ causes } \mathit{quality} = C + Z \textbf{ if } s, \rho(s, a) = Z, \mathit{quality} = C. \quad (2)$$

- D contains a (possibly empty) set P of facts of the form $\rho(s, a) = z$.

A *PEORL initial state* contains a state I. In particular, the initial plan quality is 0. A *PEORL goal* $G = (A, L)$ where A is a goal state, and L is a linear constraint of the form $(\mathit{quality} \geq n)$ where n is an integer. The negation of L is defined in the usual way.

The triple $(I, (A, L), D)$ forms a symbolic planning problem with linear constraints: the plan is encoded by a transition path of $T(D)$ that starts from state I and ends in state A with L satisfied. A plan Π of (I, G, D) is *optimal* iff $\sum_{\langle s,a,t\rangle \in \Pi} r(s, a)$ is maximal among all plans. However, note that reward function r is not a part of the planning problem because we assume that reward is part of the specific domain details not captured as prior knowledge. We later use PEORL learning algorithm to interact with the environment and generate optimal plan when the algorithm terminates.

Solving the planning problem follows the method of translating I, G and D into the input language of CLINGO [Khandelwal *et al.*, 2014]. However, in this paper, we use a slightly different but equivalent translation into the input language of CLINGCON to handle linear constraints more efficiently[1].

3.2 From Symbolic Transitions to Options

We assume that in the transition system $T(D)$, for each transition $\langle s, a, t\rangle \in T(D)$, a contains exactly one action symbol, i.e, concurrent execution of actions is not allowed. $\mathbb{F}^A$ maps a symbolic transition $\langle s, a, t\rangle$ to an option in the sense of [Barto and Mahadevan, 2003]. $\mathbb{F}^A(\langle s, t, a\rangle) = (\pi, \beta, \mathcal{I})$ where $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, $\beta : \mathcal{S} \mapsto [0, 1]$, and $\mathcal{I} \subseteq \mathcal{S}$. In particular, we enforce that option $\mathbb{F}^A(\langle s, t, a\rangle)$ is available for transition $\langle s, a, t\rangle$ iff $s = \mathcal{I}$ and $\beta(t) = 1$. This condition guarantees that the right option is chosen to realize the symbolic transition $\langle s, a, t\rangle$ at its starting state, and terminates when it fulfills the symbolic transition.

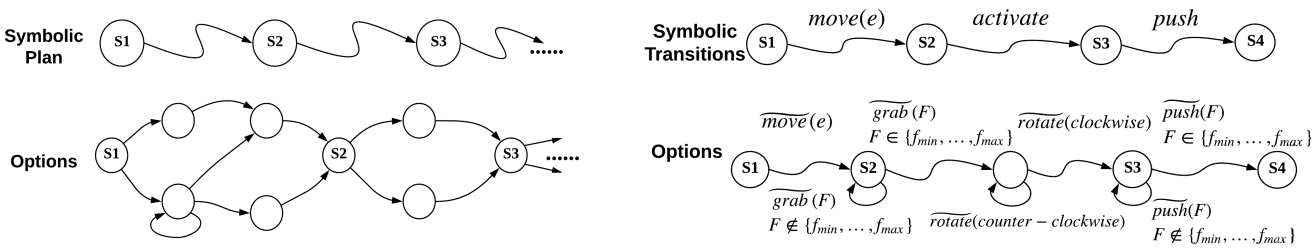
We further build one more deterministic layer by mapping a transition path defined by a symbolic plan to a set of options. For a transition path $\Pi = \langle s_1, a_1, \ldots, a_{l-1}, s_l\rangle$,

$$\mathbb{F}_A(\Pi) = \bigcup_{\langle s_{i-1}, a_{i-1}, s_i\rangle \in \Pi} \mathbb{F}^A(\langle s_{i-1}, a_{i-1}, s_i\rangle).$$

[1] To use CLINGCON, (2) is translated to

$$\&\texttt{sum}\{\texttt{quality(k}-1\texttt{)};\texttt{Z}\} = \texttt{quality(k)} :- \texttt{a(k}-1\texttt{)}, \texttt{s(k}-1\texttt{)}, \texttt{R(s, a, Z)}.$$

where k stands for time step.



(a) The mapping from a symbolic transition path to options

(b) The option mapping for transitions t_1, t_2, t_3

Figure 1: Mappings from symbolic transitions to options

It is easy to see that the execution of a symbolic plan is deterministically realized by executing their corresponding options sequentially. Such hierarchical mapping is illustrated in Figure 1a.

3.3 Example: Grid World

We use the Grid World adapted from [Leonetti *et al.*, 2016] as an example. In a 20×20 grid, there is an agent that needs to navigate to (9,10), which can only be entered through (9,9). At (9,9) there is a door that the agent needs to activate first, and then push to enter. The action description consists of causal laws formulating effects of *move*(E) where $E \in \{e, s, w, n\}$, *push* and *activate*, for instance,

move(e) **causes** *pos*$(X, Y+1)$ **if** *pos*(X, Y)
nonexecutable *move*(e) **if** *pos*$(X, 20)$
nonexecutable *move*(e) **if** *pos*$(9, 9)$, ∼*dooropen*
activate **causes** *dooractive* **if** *pos*$(9, 9)$, ∼*dooractive*
push **causes** *dooropen* **if** *pos*$(9, 10)$, *dooractive*.

Declare the following fluents are inertial:

inertial *pos* **inertial** *dooropen* **inertial** *dooractive*.

The following causal laws are instantiation of (1) and (2). They formulate the effects on plan quality by executing *move* for a particular state, and similar causal laws can be defined for *activate* and *push*:

s(X, Y) **if** *pos*(X, Y), ∼*dooractive*, ∼*dooropen*
move(E) **causes** *quality* $= C + Z$ **if**
$s(X, Y), R(s(X, Y), \textit{move}(E)) = Z, \textit{quality} = C.$

Assuming initially the agent is located at $(9, 8)$ with door closed and inactive, the action description D, initial state $I = \{\textit{pos}(9, 8), \sim \textit{dooractive}, \sim \textit{dooropen}\}$ and goal state $G = \{\textit{pos}(9, 10), \textit{dooractive}, \textit{dooropen}\}$ are translated into the input language of CLINGCON and a plan is

$$\begin{aligned}
t_1 &: \langle\{\textit{pos}(9,8), \sim\textit{dooractive}, \sim\textit{dooropen}\}, \textit{move}(e),\\
&\quad \{\textit{pos}(9,9), \sim\textit{dooractive}, \sim\textit{dooropen}\}\rangle\\
t_2 &: \langle\{\textit{pos}(9,9), \sim\textit{dooractive}, \sim\textit{dooropen}\}, \textit{activate},\\
&\quad \{\textit{pos}(9,9), \textit{dooractive}, \sim\textit{dooropen}\}\rangle\\
t_3 &: \langle\{\textit{pos}(9,9), \textit{dooractive}, \sim\textit{dooropen}\}, \textit{push},\\
&\quad \{\textit{pos}(9,9), \textit{dooractive}, \textit{dooropen}\}\rangle\\
t_4 &: \langle\{\textit{pos}(9,9), \textit{dooractive}, \textit{dooropen}\}, \textit{move}(e),\\
&\quad \{\textit{pos}(9,10), \textit{dooractive}, \textit{dooropen}\}\rangle.
\end{aligned} \tag{3}$$

Now we map symbolic transitions t_1, t_2, t_3 to options. As options talk about the realization of symbolic actions in terms of MDP actions, we assume that each symbolic action *move*(E) for a direction E is executed in the same way in MDP, denoted as $\widetilde{move}(E)$. Symbolic action *push* can be executed in a variety of ways: the agent needs to use proper force to push the door such that the door can be opened without any damage. Therefore, *push* is executed in finite number of options, denoted as $\widetilde{push}(F)$ where $F \in \{f_{min}, \ldots, f_{max}\}$. Executing symbolic action *activate* as an option involves two steps: first, the agent needs to grab the doorknob using proper force, denoted by $\widetilde{grab}(F)$, where $F \in \{f_{min}, \ldots, f_{max}\}$. Second, after the door knob is successfully grabbed, it can be turned either clockwise or counter-clockwise, and turning it clockwise can activate the door. This action is denoted as $\widetilde{rotate}(E)$ for $E \in \{\textit{closewise}, \textit{counter-clockwise}\}$. The mapping from t_1, t_2, t_3 to options is demonstrated as Figure 1b.

4 PEORL Learning

Given any transition $\langle s_{i-1}, a_{i-1}, s_i\rangle$ in a plan Π, hierarchical R-learning involves the updates of R and ρ in two steps. Since every symbolic transition is 1-1 correspondence to its option $\mathbb{F}^A(\langle s_{i-1}, a_{i-1}, s_i\rangle)$, we also use a_{i-1} to denote the option. Before an option terminates, execute actions following the option, and for any transition $\langle x, \tilde{a}, y\rangle$ where $\tilde{a} \in \mathcal{A}$, update

$$\begin{aligned}
R_{t+1}(x, \tilde{a}) &\overset{\alpha}{\leftarrow} r(x, \tilde{a}) - \rho_t^{\tilde{a}}(x) + \max_{\tilde{a}} R_t(y, \tilde{a})\\
\rho_{t+1}^{\tilde{a}}(x) &\overset{\beta}{\leftarrow} r(x, \tilde{a}) + \max_{\tilde{a}} R_t(y, \tilde{a}) - \max_{\tilde{a}} R_t(x, \tilde{a}).
\end{aligned} \tag{4}$$

When option terminates, update

$$\begin{aligned}
R_{t+1}(s_{i-1}, a_{i-1}) &\overset{\alpha}{\leftarrow} r(s_{i-1}, a_{i-1}) - \rho_t^{a_{i-1}}(s_{i-1})\\
&\quad + \max_a R(s_i, a)\\
\rho_{t+1}^{a_{i-1}}(s_{i-1}) &\overset{\beta}{\leftarrow} r(s_{i-1}, a_{i-1})\\
&\quad + \max_a R_t(s_i, a) - \max_a R_t(s_{i-1}, a)
\end{aligned} \tag{5}$$

where α and β are learning rates for R and ρ, r denotes the cumulative reward accrued by executing option mapped from symbolic action a_{i-1}. Given a plan Π, the quality of Π is defined by summing up all gain rewards for the transitions in Π:

$$\textit{quality}_t(\Pi) = \sum_{\langle s_{i-1}, a_{i-1}, s_i\rangle \in \Pi} \rho_t^{a_{i-1}}(s_{i-1}) \tag{6}$$

Given a PEORL theory $(I, G, D, \mathbb{F}_A)$, its learning algorithm is shown in Algorithm 1.

Algorithm 1 POERL Learning Loop

Require: $(I, G, D, \mathbb{F}_A)$ where $G = (A, \emptyset)$, and an exploration probability ϵ

$P_0 \Leftarrow \emptyset, \Pi \Leftarrow \emptyset$
while True **do**
  $\Pi_o \Leftarrow \Pi$
  take ϵ probability to solve planning problem and obtain a plan $\Pi \Leftarrow$ CLINGCON.*solve*$(I, G, D \cup P_t)$
  if $\Pi = \emptyset$ **then**
    return Π_o
  end if
  for $\langle s_{i-1}, a_{i-1}, s_i \rangle \in \Pi$ **do**
    use option $\mathbb{F}^A(\langle s_{i-1}, a_{i-1}, s_i \rangle)$ to update R and ρ by (4) until the option terminates
    update R and ρ using (5).
  end for
  calculate quality of Π by (6).
  update planning goal $G \Leftarrow (A, \mathit{quality} > \mathit{quality}_t(\Pi))$.
  update facts $P_t \Leftarrow \{\rho(s_{i-1}, a_{i-1}) = z : \langle s_{i-1}, a_{i-1}, s_i \rangle \in \Pi, \rho_t^{a_{i-1}}(s_{i-1}) = z\}$
end while

While previous results show that R-learning converges, most properties of option-based hierarchical R-learning remain unknown and therefore it remains an open question that option-based hierarchical R-learning converges to optimal over a finite number of options. For this reason, we leave the theoretical study of Algorithm 1 for our future work, but will empirically show its effectiveness on two benchmark domains in next section.

5 Experiment

5.1 Taxi Domain

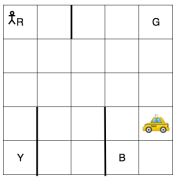
We first use Taxi domain [Barto and Mahadevan, 2003] which is a benchmark domain for studying HRL. Scenario 1 is based on Taxi-v1 in OpenAI Gym (`https://gym.openai.com/envs/Taxi-v1/`). A Taxi starts at any location in a 5×5 grid map (Figure 3a), navigates to a passenger, picks up the passenger, navigates to the destination and drops off the passenger, with randomly chosen locations for passenger and destination from marked grids. Every movement has a reward -1. Successful drop-off receives reward 20. Improper pick-up or drop-off receive reward -10. All actions are deterministic and always successful. In our experiment, we randomly set 10 initial configurations and compare the cumulative rewards received by a standard Q-learning RL-agent, a HRL-agent based on hierarchical Q-learning using the manually crafted options specified in [Barto and Mahadevan, 2003], a standard planning agent (P-agent) using CLINGO to generate plans and execute, and a PEORL-agent. For all learning rates α is annealed from 1 to 0.01, and for PEORL agent, $\beta = 0.5$. The result (Figure 2c) shows the cumulative reward of PEORL agent significantly surpasses the RL-agent and is also superior to HRL-agent. Guided by its symbolic plan, PEORL agent has a clear motivation to achieve its goal. For this reason, it never commits actions that violate its commonsense knowledge, such as an improper pick-up or drop-off, or run into the walls. For this reason, the penalty of -10 never occurs to PEORL agent, so the variance of the cumulative reward is a lot smaller than RL-agent and HRL-agent. PEORL-agent starts with the shortest plans but gradually explores longer ones. After around 1000 episodes, symbolic plans of PEORL-agent converge back to the shortest, indicating that the shortest plans are the overall optimal ones. P-agent also benefits from symbolic plans by not committing improper actions. Furthermore, since ASP-based symbolic planning is usually used to generate shortest plan, P-agent has the steadily largest cumulative reward which happens to be optimal. This result suggests that ASP-based planning can perform very well in deterministic domains where shortest plans are the most desirable.

In Scenario 2, inspired by [Kulkarni *et al.*, 2016, Section 4.1], we require that if the taxi arrives at the goal with (4,4) visited, it gets reward 30. The only information present in symbolic knowledge is when (4,4) is visited, the fluent *rewardvisited* is set to be true, so that the state representation in RL maps correctly to symbolic space. Again, PEORL-agent outperforms all others (Figure 2d). It starts by trying the shortest plan but during exploration of longer alternatives, it discovers the extra reward, and finally converge to the optimal. Figure 2b showed one solution in this scenario. By comparison, since visiting (4,4) is not a necessary condition to drop off the passenger, throughout 10 randomly generated configurations, P-agent never visits that state, behaving the same way with Scenario 1 by sticking to its shortest plan. HRL-agent and RL-agent fail to figure out the extra reward either. This scenario shows that PEORL-agent can discover state with extra reward, and its symbolic plans have leveraged the learned information from RL and become more robust and adaptive to the change of domain details.

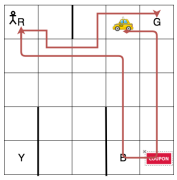
5.2 Grid World

The Grid World domain is shown in Figure 3a. Adding to the description in Section 3.3, we further assume there are both horizontal and vertical bumpers where the agent receives a penalty of -30 (grids marked as red), and -15 for grids marked with yellow, and -1 for all other grids. Actions *grab* and *push* have an integer parameter F, ranging from 0 to 60, and only if $20 \leq F < 40$ can the execution be successful. Every execution failure causes a -10 penalty. The initial state is chosen from the marked grids in the first column, and goal state is (9,10). We use this example to show that aided by RL, symbolic plans can be learned to avoid bumpers, and can be reliably executed.

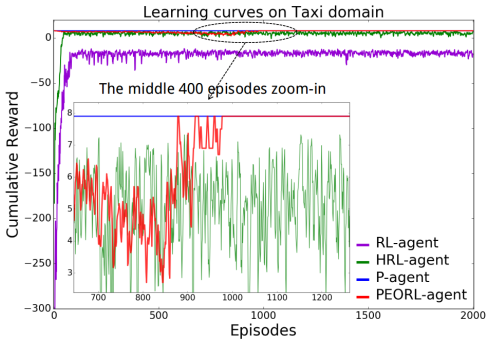
We set up RL-agent using Q-learning, PEORL-agent and P-agent. Bumper information is not captured by symbolic knowledge since we assume these are domain details that need to be learned. Learning rates are chosen as the same with Taxi domain. The learning curve is shown in Figure 3d across 1000 episodes. Similar to the Taxi domain, PEORL-agent has smaller variance in its cumulative rewards (zoomed in by Figure 3d), and achieves the optimal behavior: it avoids the bumper at its best, and reliably activates and pushes the door (e.g., Figure 3b), surpassing RL-agent. For P-agent, the shortest plans, in this case, are not ideal plans. Since P-agent has no learning capability and only relies on its symbolic knowledge, it performs the worst.



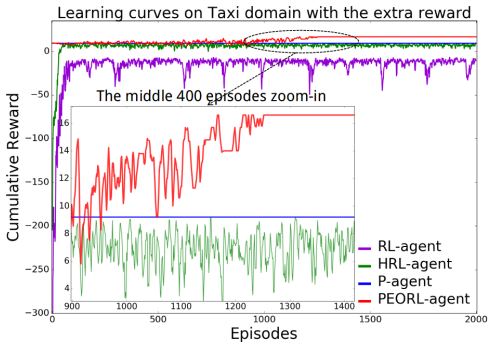
(a) Taxi domain



(b) A solution

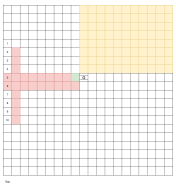


(c) Learning curves on Taxi domain

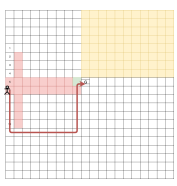


(d) Learning curves with the extra reward

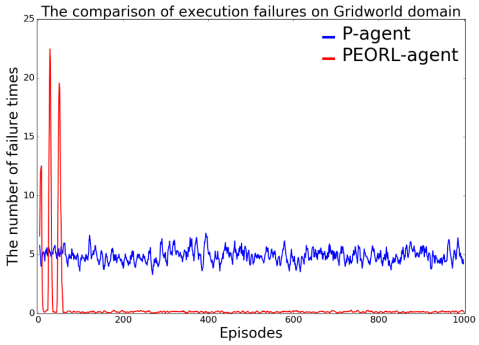
Figure 2: Taxi domain



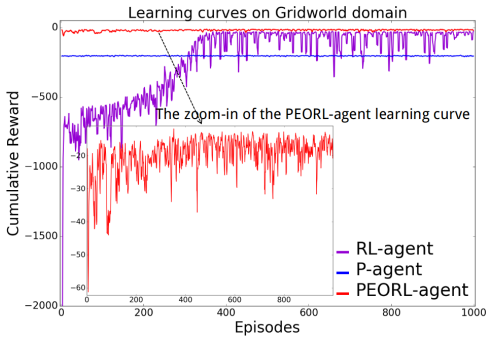
(a) Grid World



(b) A solution



(c) Execution failure of PEORL and Planning



(d) Learning curves on Gridworld domain

Figure 3: Grid World

Figure 3c shows that facing domain uncertainty, the robustness of symbolic plan of PEORL agents is improved using RL, indicated by the reduced number of execution failure. As options mapped from *activate* and *push* lead to smaller RL problems, the underlying R-learning quickly learned the right way to execute the options such that the need to replan is significantly reduced. By contrast, relying on replanning, P-agent can recover from failure and eventually achieve its goal, but it cannot improve its execution reliability from learning, leading to poor plan robustness with a relatively large number of execution failures.

6 Related Work

Integrating symbolic planning and RL has been an active research topic recently. Pre-complied symbolic plans or paths from a finite-state machine play similar roles as options [Parr and Russell, 1998; Ryan, 2002; Leonetti *et al.*, 2012]. Recent work also uses ASP to generate longer symbolic plans [Leonetti *et al.*, 2016]. In these approaches, symbolic planning is used to help RL through a one-shot plan generation and compilation. By contrast, in our work, planning is interleaved with and constantly updated by RL, and therefore new options can be explored and more meaningful ones will be selected leveraging learning. Our work is also related to automatic option discovery. Various methods have been used, including clustering [Mannor *et al.*, 2004] and Laplacian Eigenmap [Machado *et al.*, 2017]. To the best of our knowledge, this is the first work using symbolic planning for automatic option discovery. Our work is also motivated by improving symbolic planning through learning. Different learning models have been adopted in earlier work such as relational decision tree [Jiménez *et al.*, 2013] or weighted exponential average [Khandelwal *et al.*, 2014]. RL is also used to improve task decomposition in HTN planning [Hogg *et al.*, 2010]. Finally, integrating planning with probabilistic planning on POMDP was investigated [Zhang *et al.*, 2017].

7 Conclusion and Future Work

In this paper we proposed the PEORL framework, where symbolic planning and HRL simultaneously improve each other, leading to rapid policy search and robust symbolic planning. Future work involves formal study on hierarchical R-learning, using PEORL framework to solve more complicated domains, and integrating symbolic planning with deep RL for interpretable end-to-end solutions.

Acknowledgements

Bo Liu and Daoming Lyu are funded by the startup funding at Auburn University.

References

[Banbara *et al.*, 2017] M. Banbara, B. Kaufmann, M. Ostrowski, and T. Schaub. Clingcon: The next generation. *TPLP*, 17:408–461, 2017.

[Barto and Mahadevan, 2003] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(1-2):41–77, January 2003.

[Chen *et al.*, 2016] K. Chen, F. Yang, and X. Chen. Planning with task-oriented knowledge acquisition for a service robot. In *IJCAI*, pages 812–818, 2016.

[Cimatti *et al.*, 2008] A. Cimatti, M. Pistore, and P. Traverso. Automated planning. In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, *Handbook of Knowledge Representation*. Elsevier, 2008.

[Gebser *et al.*, 2012] M. Gebser, B. Kaufmann, and T. Schaub. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187-188:52–89, 2012.

[Gelfond and Lifschitz, 1998] M. Gelfond and V. Lifschitz. Action languages. *Electronic Transactions on Artificial Intelligence (ETAI)*, 1998.

[Hanheide *et al.*, 2015] M. Hanheide, M. Göbelbecker, G. S Horn, et al. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*, 2015.

[Helmert, 2006] M. Helmert. The fast downward planning system. *JAIR*, 26:191–246, 2006.

[Hogg *et al.*, 2010] C. Hogg, U. Kuter, and H. Munoz-Avila. Learning methods to generate good plans: Integrating htn learning and reinforcement learning. In *AAAI*, 2010.

[Jiménez *et al.*, 2013] S. Jiménez, F. Fernández, and D. Borrajo. Integrating planning, execution, and learning to improve plan execution. *Computational Intelligence*, 2013.

[Khandelwal *et al.*, 2014] P. Khandelwal, F. Yang, M. Leonetti, V. Lifschitz, and P. Stone. Planning in Action Language $\mathcal{BC}$ while Learning Action Costs for Mobile Robots. In *ICAPS*, 2014.

[Khandelwal *et al.*, 2017] P. Khandelwal, S. Zhang, J. Sinapov, M. Leonetti, J. Thomason, F. Yang, I. Gori, M. Svetlik, P. Khante, and V. Lifschitz, J. Aggarwal, R. Mooney, P. Stone. Bwibots: A platform for bridging the gap between ai and human–robot interaction research. *IJRR*, 36(5-7):635–659, 2017.

[Kulkarni *et al.*, 2016] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 3675–3683, 2016.

[Lee *et al.*, 2013] J. Lee, V. Lifschitz, and F. Yang. Action Language $\mathcal{BC}$: A Preliminary Report. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[Leonetti *et al.*, 2012] M. Leonetti, L. Iocchi, and F. Patrizi. Automatic generation and learning of finite-state controllers. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 135–144. Springer, 2012.

[Leonetti *et al.*, 2016] M. Leonetti, L. Iocchi, and P. Stone. A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artificial Intelligence*, 241:103–130, 2016.

[Lifschitz, 2008] V. Lifschitz. What is answer set programming? In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1594–1597. MIT Press, 2008.

[Machado *et al.*, 2017] M. C Machado, M. G Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, 2017.

[Mahadevan, 1996] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22:159–195, 1996.

[Mannor *et al.*, 2004] S. Mannor, I. Menache, A. Hoze, and U. Klein. Dynamic abstraction in reinforcement learning via clustering. In *International Conference on Machine Learning*, 2004.

[McCarthy and Hayes, 1969] J. McCarthy and P. Hayes. Some Philosophical Problems From The Standpoint of Artificial Intelligence. In *Machine Intelligence*. Edinburgh University Press, 1969.

[McDermott *et al.*, 1998] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl-the planning domain definition language. 1998.

[Mnih *et al.*, 2015] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. Cambridge University Press, 1998.

[Parr and Russell, 1998] R. Parr and S. J. Russell. Reinforcement learning with hierarchies of machines. In *Advances in neural information processing systems*, pages 1043–1049, 1998.

[Ryan, 2002] M. R. Ryan. Using abstract models of behaviours to automatically generate reinforcement learning hierarchies. In *ICML*, 2002.

[Schwartz, 1993] A. Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proc. 10th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1993.

[Sutton *et al.*, 1999] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[Zhang *et al.*, 2017] S. Zhang, P. Khandelwal, and P. Stone. Dynamically constructed (po) mdps for adaptive robot planning. In *AAAI*, pages 3855–3863, 2017.