Learning Generalizable Symbolic Options for Transfer in Reinforcement Learning

Rashmeet Kaur Nayyar, Shivanshu Verma, and Siddharth Srivastava
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ 85281
{rmnayyar,sverma76,siddharths}@asu.edu

Abstract

This paper presents a new approach for Transfer Reinforcement Learning (RL) for Stochastic Shortest Path (SSP) problems in factored domains with unknown transition functions. We take as input a set of problem instances with sparse reward functions. The presented approach first learns a semantically well-defined state abstraction and then uses this abstraction to invent high-level options, to learn abstract policies for executing them, as well as to create abstract symbolic representations for representing them. Given a new problem instance, our overall approach conducts a novel bi-directional search over the learned option representations while also inventing new options as needed. Our main contributions are approaches for continually learning transferable, generalizable knowledge in the form of symbolically represented options, as well as for integrating search techniques with RL to solve new problems by efficiently composing the learned options. Empirical results show that the resulting approach effectively transfers learned knowledge and achieves superior sample efficiency compared to SOTA methods.

1 Introduction

At the heart of creating intelligent AI agents is the ability to *learn representations* for efficient reasoning and *generalize behaviors* to new, unseen situations. As Reinforcement Learning (RL) can be sample-inefficient and difficult to scale in long horizon sparse settings (Dadvar et al., 2023), much research has extensively investigated Transfer Learning (TL) from source tasks to improve sample-efficiency of RL in related but distinct target tasks (Taylor and Stone, 2009). Hierarchical Reinforcement Learning (HRL) (Barto and Mahadevan, 2003), in particular the options framework (Sutton et al., 1999), provides a principled way of using macro-actions for temporal abstraction to accelerate RL. However, despite much effort in autonomously learning options (Menache et al., 2002; Frans et al., 2017), these approaches suffer from limited generalizability and sample-inefficiency.

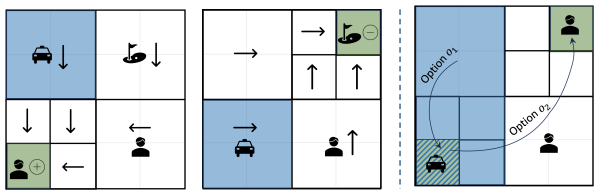
We present a novel approach for sample-efficient transfer learning in RL for factored domains by learning *composable* and *generalizable* options with abstract symbolic representations (illustrated in Fig. 1). Such problems include many practical real-world scenarios (e.g., pick-up and drop-off taxi service) where states can be intuitively expressed in terms of values of state variables. Our approach starts by learning a semantically well-defined state abstraction from source problems and then uses this abstraction to learn high-level generalizable options with abstract symbolic representations and policies for executing them. Given a target problem, our approach conducts a novel bi-directional search over these options to compose and invent new options if needed.

Through an extensive evaluation on a diverse range of challenging domains, we demonstrate that our approach significantly outperforms state-of-the-art RL baselines in terms of sample efficiency on target problems with the same state variables but different tasks using a small set of source problems.

NeurIPS 2023 Workshop on Generalization in Planning (GenPlan 2023).

Furthermore, the learned symbolic options are semantically meaningful and have composable and generalizable properties that aid transfer.

The presented approach is related to research in option discovery (Bacon et al., 2017; Riemer et al., 2018; Klissarov and Precup, 2021). However, unlike conventional approaches, we learn composable and generalizable options with abstract symbolic representations without the need to prespecify the number of options to be learned. Our approach also provides the benefits of transfer to environments that are larger and more cluttered than those used in the source problems. For instance, most self-driving cars today train and over-fit to known scenarios which makes it difficult to start operations in new, more crowded cities and countries. Effective transfer in these scenarios requires learning transferable knowledge, identifying useful knowledge during transfer, as well as discovering and learning new relevant knowledge.



(a) Option o_1 (b) Option o_2 (c)

Figure 1: Illustration of options in taxi world with passengers and a destination. The images (a) and (b) show abstract policies and initiation sets for option o_1 (to navigate and pickup passenger) and option o_2 (to navigate and dropoff passenger) respectively that are generalizable due to state abstraction. The image (c) shows composability of the options as abstract state in the termination of o_1 (shown in green) is in the initiation of o_2 (shown in blue).

To our knowledge, this is the first end-to-end approach for learning composable and generalizable options with abstract symbolic representations in a transfer learning setting without the need for any hand-engineered inputs and a predefined number of options. Our key contributions are:

- **Learning composable and generalizable options:** We present a novel approach for learning options with abstract symbolic representations that exhibit composable and generalizable properties from a set of source problems without any hand-engineered inputs.
- **Continually inventing options:** We present a novel bi-directional planner that uses an initial library of learned options to continue to invent new options for target problems by prioritizing search over the previously learned options.
- **Integrating planning with reinforcement learning:** We present a general framework that identifies relevant transferred options using the learned abstract representations that express their initiation and termination conditions, and effectively recomposes these learned options to find a plan to solve a target problem.

The rest of this paper is organized as follows: We begin by reviewing related work in Sec. 2. We establish the relevant notation and review essential key concepts in Sec. 3. Then, we describe the proposed novel integrated bi-directional planning and learning framework for transfer in Sec. 4. We present the empirical results in Sec. 5. Finally, we conclude and discuss the future work in Sec. 6.

2 Related Work

Symbolic planning and reinforcement learning A line of research has emerged trying to combine symbolic planning with RL (Grounds and Kudenko, 2005; Yang et al., 2018; Lyu et al., 2019; Illanes et al., 2020; Kokel et al., 2021). Taskable RL (Illanes et al., 2020) assumes that a high-level model of the environment is provided and uses a planner to provide high-level actions to a low-level RL. RePRel (Kokel et al., 2021) requires hand-crafted abstractions as input for integrating planning with learning. In contrast, we learn composable options and abstract representations without requiring any human-engineered inputs that allow planning at the high level and RL at the low level.

State and Action Abstractions The majority of the current literature tackles either state or action abstraction in isolation. However, the problem of jointly learning both abstractions remains largely unaddressed (Konidaris, 2019). Dietterich (1999) show the importance of combining state abstractions with action hierarchies, however, assume that state abstractions are provided. Jonsson and Barto (2000) learn state abstractions for options, similar to this work, however, assume options are provided. Ravindran and Barto (2003) introduce homomorphisms between original MDP and minimized MDP

obtained by collapsing state-action pairs. Abel et al. (2020) analyze value-preserving properties of provided state-action abstractions. Walsh et al. (2006) infer and transfer state abstractions while Silver et al. (2023) learn predicates, assuming that the model of the environment is known. In contrast, our approach exploits the benefits of both temporal and state abstractions by autonomously learning them for transfer RL. We first learn state abstractions and then options from source problems, then continually learn new options and option-driven state abstractions for target problems.

Discovering Task Hierarchies Hierarchical Reinforcement Learning (HRL) (Barto and Mahadevan, 2003) frameworks such as Options (Parr and Russell, 1997), MAXQ (Dietterich, 1999), and HAMs (Sutton et al., 1999) provide a way of decomposing the original problem into a hierarchy of subproblems to make RL more sample-efficient. HEXQ (Hengst et al., 2002) and VISA (Jonsson and Barto, 2006) learn subtasks based on changing values of state variables. Mehta et al. (2008) learn hierarchies by discarding irrelevant state variables.

Discovering Temporal Abstractions Recent work has developed methods for learning options based on policy gradients (Bacon et al., 2017; Riemer et al., 2018). However, these approaches need to prespecify the number of options to be learned and lack diversity in the learned options. Bagaria and Konidaris (2020) discover options by chaining but do not learn or employ state abstractions to represent or discover options. Konidaris and Barto (2007) reframe the state space into agent-centric and problem-centric representations and transfer portable options in the agent-space.

Discovering Subgoals Another avenue of research has explored a combination of graph-partitioning (Menache et al., 2002; Şimşek and Barto, 2007; Machado et al., 2017; Bacon and Precup, 2013), clustering (Mannor et al., 2004), and frequency-based (McGovern and Barto, 2001; Stolle and Precup, 2002) techniques to identify subgoals or bottleneck states. Unlike these approaches, this paper's primary focus is on transfer.

3 Preliminaries

In this section, we establish relevant notation and review required key concepts.

Reinforcement Learning In this work, we formalize each problem as a factored *Stochastic Shortest Path* (SSP) problem $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, s_o, s_g, \gamma \rangle$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function, s_o is a start state, s_g is a goal state, and $\gamma \in (0, 1]$ is a discount factor. We define each state $s \in \mathcal{S}$ using a set of n variables $\mathcal{V} = \{v_1, \ldots, v_n\}$ where each v_i takes on values in some interval $\text{Dom}(v_i) = [v_i^{min}, v_i^{max}]$, where v_i^{min} and v_i^{max} denote the lower and upper bounds on the value of v_i respectively.

A solution to an SSP is a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that prescribes an action that should be taken in each state. Generally, dynamic programming methods such as Value Iteration and Policy Iteration can be used to compute a solution. However, in most real-world scenarios, these methods fail as $\mathcal{T}$ and $\mathcal{R}$ are unknown. In such cases, RL methods (Sutton and Barto, 2018) such as Q-learning (Watkins and Dayan, 1992) and DQN (Mnih et al., 2013) can be used to solve such problems, however, they are sample inefficient. In this paper, we consider SSPs with unknown $\mathcal{T}$ and $\mathcal{R}$.

Conditional State Abstractions We define a state abstraction function as $\phi : \mathcal{S} \rightarrow \overline{S}$ where each ground state $s \in \mathcal{S}$ is mapped to an abstract state $\overline{s} \in \overline{S}$. Given a set of variables $\mathcal{V}$, we define an abstract state $\overline{s} = \{\theta_i | v_i \in \mathcal{V}\}$ as a set of partitions of the form $\theta_i = [v_i^{low}, v_i^{high}]$ for each variable $v_i \in \mathcal{V}$ such that v_i^{low} and $v_i^{high} \in [v_i^{min}, v_i^{max}]$ and $v_i^{low} \leq v_i^{high}$. E.g., [1,2] is a partition of the original value range [1,4] for a variable. The most trivial abstract state $\overline{s}_{init}$ has the original value range itself as the partition for each variable, i.e., $\overline{s}_{init} = \{\theta_i | \forall v_i \in \mathcal{V}, v_i^{low} = v_i^{min}, v_i^{high} = v_i^{max}\}$.

A hierarchy of state abstractions in the form of a Conditional Abstraction Tree (CAT) (Dadvar et al., 2023) is defined as $\xi = \langle \mathcal{N}, \mathcal{E} \rangle$ where $\mathcal{N}$ is a set of possible abstract states where the root node represents $\overline{s}_{init}$ and $\mathcal{E}$ is a set of directed edges connecting possible abstract states. Each edge $\in \mathcal{E}$ from a parent abstract state $\overline{s}_p$ to a child abstract state $\overline{s}_c$ exists iff $\overline{s}_c$ can be obtained by directly refining $\overline{s}_p$ denoted by $\overline{s}_c \trianglerighteq \overline{s}_p$.

Fig. 2 shows an example CAT for x and y variables where [1-4] represents the complete range of values for the variables. Given a CAT ξ, the set of leaves represents the complete set of abstract states $\overline{S}$ of the most refined state abstraction in ξ.

The CAT+RL algorithm takes as input an SSP $\mathcal{M}$ and learns a CAT ξ and an abstract policy π. However, it does not address the problem of learning options, which is the focus of this paper. In this work, we use CAT+RL as an underlying algorithm to learn state abstractions.

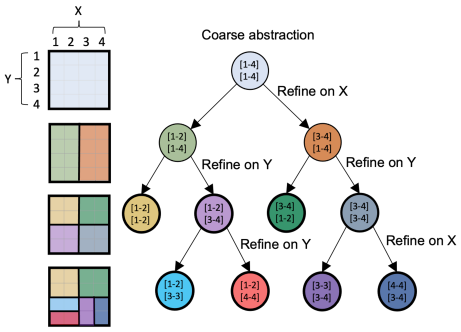


Figure 2: CAT

Temporal abstractions The standard formulation of the options framework (Sutton et al., 1999) does not involve state abstractions. In this work, we define an option over a set of abstract states $\overline{S}$ as $o = \langle \mathcal{I}, \beta, \pi \rangle$ where $\mathcal{I} \subset \overline{S}$ is a subset of abstract states where the option o can initiate, $\beta \subset \overline{S}$ is a subset of abstract states where the option o terminates, and $\pi : \overline{S} \rightarrow \mathcal{A}$ is a prescribed abstract policy.

Given a set of source problems $\boldsymbol{\mathcal{M}}$ and a target problem $\mathcal{M}$, we aim to efficiently learn a solution for $\mathcal{M}$. In the next section, we present a novel end-to-end framework that achieves this by autonomously learning and transferring generalizable knowledge in the form of symbolically represented options and a CAT from $\boldsymbol{\mathcal{M}}$, while also continually inventing new options as needed to solve $\mathcal{M}$.

4 Integrated Planning and Learning for Transfer

The key major contributions of this work are that our approach automatically (i) identifies existing options relevant for a target problem, as well as (ii) identifies new options needed to solve the target problem. This focused learning of new abstract policies aids transferability. The learned symbolic option representations are *composable* and *generalizable*, enabling effective planning and transfer.

We present a novel general framework called **C**AT **O**ptions **Plan**ning and **Learn**ing -- COPlanLearn -- that inputs a set of source problems $\boldsymbol{\mathcal{M}}$ and integrates planning with learning to solve a new target problem $\mathcal{M}_T$ (Alg. 1). COPlanLearn first learns a CAT ξ and a library of reusable options called CAT-options $\mathcal{O}$ from a set of source problems $\boldsymbol{\mathcal{M}}$ (lines 2-6, Sec. 4.1). COPlanLearn then transfers these abstractions while inventing new options called bridge-options if needed to compute an option plan Π for a new target problem $\mathcal{M}_T$. We present a novel bi-directional search algorithm to compute these bridge-options (lines 8-9, Sec. 4.2). Then, COPlanLearn learns or fine-tunes policies for options in the option plan Π while updating the CAT ξ to find a complete solution for $\mathcal{M}_T$. If such a plan is not found or fails, COPlanLearn instead learns an updated CAT ξ. Finally, it learns new CAT-options for $\mathcal{M}_T$ and updates the option library $\mathcal{O}$ (lines 11-16, Sec. 4.3). The framework returns the solution found for $\mathcal{M}_T$ along with the learned option library $\mathcal{O}$ and the CAT ξ.

Algorithm 1: COPlanLearn

Input: Source SSPs $\boldsymbol{\mathcal{M}}_S$, Target SSP $\mathcal{M}_T$ (initial state: s_o, goal state: s_g)
Output: Solution Π for $\mathcal{M}_T$, Option library $\mathcal{O}$, CAT ξ

1 /* Invent CAT-Options from Source Problems */
2 Initialize $\mathcal{O}$, ξ
3 **for** $\mathcal{M}_S \in \boldsymbol{\mathcal{M}}_S$ **do**
4 $\xi, \pi, \tau \leftarrow$ CAT+RL($\mathcal{M}_S$,ξ)
5 $\Pi \leftarrow$ inventCATOptionsPlan($\mathcal{M}_S$,ξ,π,τ)
6 $\mathcal{O}$.update(Π)
7 /* Invent Bridge-Options for target problem */
8 $\overline{s}_o, \overline{s}_g \leftarrow$ getAbstractStates(ξ,s_o,s_g)
9 $\Pi \leftarrow$ inventBridgeOptionsPlan($\mathcal{M}_T$,$\mathcal{O}$,$\overline{s}_o$,$\overline{s}_g$)
10 /* Learning Abstract Policies for Invented Options */
11 **if** Π *found* **then**
12 $\Pi, \xi \leftarrow$ LearnOrFinetuneOptionPolicies(Π,$\mathcal{M}_T$,ξ)
13 **if** Π *not found or fails* **then**
14 $\xi, \pi, \tau \leftarrow$ CAT+RL($\mathcal{M}_T$,ξ)
15 $\Pi \leftarrow$ inventCATOptionsPlan($\mathcal{M}_T$,ξ,π,τ)
16 $\mathcal{O}$.update(Π)
17 **return** Π, $\mathcal{O}$, ξ

We now discuss these major components of Alg. 1 in detail in the sections below.

4.1 Inventing CAT-options

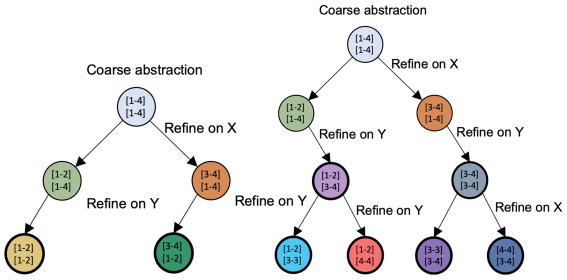
We present a novel approach based on CATs to first extract subgoal abstract states and then invent CAT-options. The structure of abstraction in well-developed CATs reveal subgoal-based temporal abstractions or options. As the subtask (unknown) of current focus changes for a given problem, the abstraction that is relevant to achieve the subtask changes. We use the structure of a CAT to identify the points of change in the relevant abstraction for a solution trajectory and determine the subgoal abstract states corresponding to the subtasks. We then leverage these subgoals and the learned CAT to invent options called CAT-options, updating the option library for transfer to other problems. We first describe the approach for learning a well-defined CAT for a given problem and extracting subgoals. We then describe our approach for inventing corresponding CAT-options with abstract symbolic representations.

Learning CAT and Abstract Policy We first learn a semantically meaningful state abstraction as a CAT and an associated policy over that CAT abstraction for a source SSP using CAT+RL.

Extracting subgoals from CAT Given a CAT and abstract trajectory, we identify subgoals as abstract states that alter the relevant CAT abstraction. For example, in Taxi World, the abstraction shifts when a passenger is picked up - from subtrees that are refined more on the passenger location variables to subtrees that are refined more on the destination location variables. Given a concrete trajectory, whenever the abstraction distance between two states s and s' is beyond a certain threshold or is non-zero, the subtask in focus changes and the abstract state for s' is extracted as a subgoal.

To detect change in abstraction between two consecutive states, we construct a pruned CAT for each state by abstracting high frequency variables and fixing values of other variables. Variable change frequencies can be readily calculated from concrete trajectories during CAT learning. Fig. 3 displays pruned CATs of the CAT in Fig. 2 for the states y=2 and y=3. In both CATs, the variable x has been abstracted—all values of x are considered when constructing these pruned variants. We now describe the Abstraction distance metric for comparing state abstractions of two pruned CATs to identify subgoal abstract states, as described below.

Abstraction distance between pruned CATs for two states in a transition captures the change in the abstraction relevant for the two states. This distance is computed by observing the change in the structure of abstraction encoded by the two pruned CATs. This is computed by adding the depths of all the dissimilar nodes in the two pruned CATs.



(a) For y = 2 (b) For y = 3

Figure 3: Pruned CATs

Inventing CAT-options from subgoals We extract partial trajectories for each subgoal, using only segments of available trajectories that originate either from the initial state or the previous subgoal. We then develop terminations for options by calculating the distance between consecutive abstract states and their common ancestor in the CAT. When this distance exceeds a threshold, that abstract state initializes the termination set of a new option. After defining these option terminations, we determine initiations by identifying abstract states in the partial trajectories that lead to those terminations. Since these options are derived from the learned CAT structure, we call them CAT-options.

4.2 Inventing bridge-options

Our approach invents new options for a target problem $\mathcal{M}_T$ using a novel bi-directional search over the existing option library (Alg. 2) if needed. It chains the existing options in $\mathcal{O}$ from the initial abstract state $\overline{s}_o$ and the goal abstract state $\overline{s}_g$, and invents novel options called bridge-options to compute a complete option plan Π. The approach identifies relevant existing options and discovers declarative abstract representation or option signature for new options capturing initiation and termination conditions.

The search initializes a forward fringe from the initial abstract state $\overline{s}_o$ and a backward fringe from the goal abstract state $\overline{s}_g$. As search expands the fringes, the forward fringe main-

tains all termination sets of options in $\mathcal{O}$ reachable from the initial abstract state and the backward fringe maintains all initiation sets of options in $\mathcal{O}$ from which you can reach the goal abstract state. Given an arbitrary distance measure between abstract states, states reachable from a state within a threshold ϵ are used to expand the forward and backward fringes.

Alg. 2 repeatedly pops the most promising node $sBest$ from the forward fringe and expands the forward fringe (Alg. 2 lines 6-7). Similarly, it repeatedly pops the most promising node $gBest$ from the backward fringe and expands the backward fringe (Alg. 2 lines 9-10). Alg. 3 expands the forward fringe by adding states from the termination of options whose initiation sets are reachable from $sBest$ (Alg. 3 lines 1-4) and expands the backward fringe by adding states from the initiation of options whose termination sets are reachable from $gBest$ (Alg. 3 lines 8-14).

If the fringes intersect at some state $mBest$, an option plan is reconstructed by tracking parent states back to the initial and goal abstract states (Alg. 2 line 11). The search also checks for a partial plan from $sBest$ to any state in the entire backward fringe by calling any off-the-shelf planner (Alg. 2 line 12). If such a partial plan is found, a complete option plan from from $\overline{s}_o$ to $\overline{s}_g$ is reconstructed. Finally, if an option plan is not found and $sBest$ or $gBest$ differ from the initial and goal abstract states respectively, then a new bridge-option is constructed with $sBest$ in the initiation and $gBest$ in the termination sets, to find a complete composable option plan Π from $\overline{s}_o$ to $\overline{s}_g$. In the next section, we discuss how abstract policies for the newly discovered bridge-option is learned.

Algorithm 2: Bi-directional Search to Invent Bridge Options and Compute Option Plan

Input: Target SSP $\mathcal{M}_T$, Option library $\mathcal{O}$, Initial abstract state $\overline{s}_o$, Goal abstract state $\overline{s}_g$
Output: Plan of options Π

```
1  Π ← Initialize empty
2  add s̄_o to fwdFringe and fwdVisited
3  add s̄_g to bwdFringe and bwdVisited
4  repeat
5      if fwdFringe then
6          sBest ← fwdFringe.getBest()
7          expandFwdFringe(O, sBest)
8      if bwdFringe then
9          gBest ← bwdFringe.getBest()
10         expandBwdFringe(O, gBest)
11     mBest ← fwdVisited.intersection(bwdVisited)
12     plan ← callPlanner(O, sBest, bwdFringe)
13     if plan or mBest ≠ ∅ then
14         Π ← reconstruct path of options
15         break
16 until not fwdFringe and not bwdFringe;
17 if Π not found and (sBest ≠ s̄_o or gBest ≠ s̄_g) then
18     O += inventOptionSignature(sBest, gBest)
19     Π ← reconstruct path of options
20 return Π
```

Algorithm 3: Bi-directional Search Fringe Expansion

```
1  Function expandFwdFringe(O, sBest):
2      for o in O do
3          if sBest ∈ o.I(ϵ) and o.β ∉ fwdVisited then
4              add o.β to fwdFringe and fwdVisited
5  Function expandBwdFringe(O, gBest):
6      for o in O do
7          if gBest ∈ o.β(ϵ) and o.β ∉ bwdVisited then
8              add o.I to bwdFringe and bwdVisited
```

4.3 Learning Abstract Policies for Options

We now discuss our approach for learning or fine-tuning abstract policies for options in the plan Π to solve the target problem $\mathcal{M}_T$ (Alg. 1 lines 11-12).

Alg. 1 loops over the option plan, executing each option if its policy is already learned. For newly learned options without a policy, Alg. 1 constructs an SSP specific to that option and uses CAT+RL to learn a refined state abstraction by expanding the CAT ξ. It also learns an abstract policy specific to the option. If an option fails, it's abstraction and/or policy is fine-tuned for the target problem similarly. The option's declarative abstract representation is then updated to reflect the state abstraction encoded in the learned CAT ξ. Finally, the option plan Π and the CAT ξ are updated with the learned option policies and abstractions.

Throughout the approach, the most refined CAT ξ is maintained. The declarative representations of all options must reflect the state abstraction defined by this CAT to enable flexible planning. As optimization, concrete states are cached when options initiate and terminate during learning/execution. When the CAT is further refined, these states are used to identify relevant new abstractions to update the options' declarative representations. Additionally, each option maintains its own internal CAT

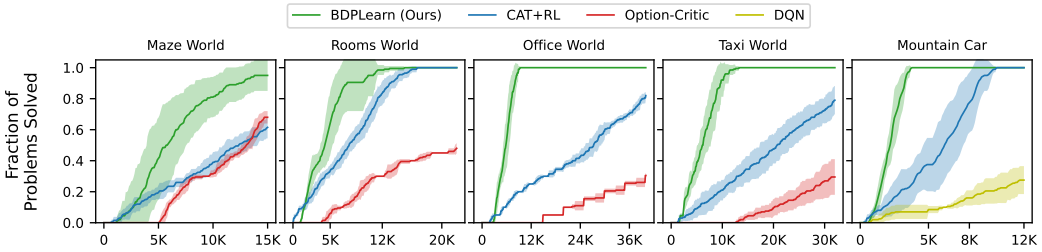


Figure 4: An average of the fraction of problems solved vs learning episodes required by each approach in each domain, computed from 10 independent trials. A total of 20 problems (2 source, 18 target) were solved sequentially by each approach.

to determine the abstract state and action prescribed by its learned abstract policy during execution. This allows learning an abstraction tailored specifically to the subtask defined by that option.

If an option plan is not found or it fails, the CAT ξ is refined for $\mathcal{M}_T$ using CAT+RL and CAT-options are learned eventually as described in 4.1, updating the option library $\mathcal{O}$ (Alg. 1 lines 13-16). Finally, Alg. 1 returns the learned option plan Π for $\mathcal{M}_T$, the option library $\mathcal{O}$, and the CAT ξ.

5 Empirical Evaluation

We now evaluate COPlanLearn on a diverse range of domains in a transfer learning setting. We implemented the method in Python and ran all experiments on 5.0 GHz Intel i9 CPUs with 64 GB RAM running Ubuntu 22.04. The details regarding the environments and hyper-parameters are included in the appendix. We sought to empirically answer the following questions:

Q1 Does COPlanLearn outperform state-of-the-art RL and transfer RL approaches to solve a diverse set of problems that differ in initial and goal configurations?

Q2 Does COPlanLearn learn abstract symbolic options that are semantically meaningful?

Q3 Does COPlanLearn transfer to larger and more cluttered or obstructed environments?

Environments We performed an extensive evaluation of the literature to ensure that the selected domains are challenging for SOTA methods. We conducted our investigation of COPlanLearn on these *stochastic* versions of environments: (1) Maze World (Ramesh et al., 2019) with randomly placed walls as obstacles; (2) Four Rooms World (Sutton et al., 1999) where an agent can navigate within rooms and through hallways to other rooms; (3) Office World (Icarte et al., 2018) requires an agent to pick-up coffee, then pick-up mail, and deliver them both to an office, (4) Taxi World introduced by Dietterich (2000) has a taxi that picks up a passenger from its pick-up location and drops it off at its destination location, and (5) Continuous state Mountain Car domain from Brockman et al. (2016).

Baselines We compare with the state-of-the-art transfer RL Option-Critic (Bacon et al., 2017) and the state-of-the-art RL CAT+RL (Dadvar et al., 2023) approaches that learn abstractions without any hand-engineered inputs. Option-Critic is a hierarchical RL approach that automatically discovers and learns options that can be transferred to related problems, whereas, CAT+RL is a top-down RL approach that automatically learns hierarchical state abstractions. For continuous domain, we also compare with DQN since multiple layers in the neural network successively learn state abstractions. We use its implementation from Stable-Baselines3 [1] framework by Raffin et al. (2019).

Experimental setup and metrics reported We provide a total of 20 randomly selected diverse problems with different initial and goal configurations to all the approaches and investigate their ability to solve a maximum number of problems using a minimum amount of learning experience. For methods that transfer, we use two randomly selected problems as source problems and the rest of them as target problems. This setting allows us to assess the combined ability of the methods to

[1] https://github.com/DLR-RM/stable-baselines3

transfer options from a small set of source problems and discover new options if needed in a large set of target problems. The approaches first jointly learn to solve all source problems and then transfer independently to solve each of the target problems.

We report an average of the fraction of total problems solved and the number of learning episodes required to solve them by each approach for each domain, along with the standard deviations computed from 10 independent runs. A problem is considered as solved only if an approach achieves a mean evaluation success rate of 0.9 or above, computed by evaluating the learned greedy policy for 100 episodes. We provide each approach a maximum allowed learning experience of n episodes to solve each problem (n = 7000 for Taxi World and n = 5000 for the rest of the domains). Upon reaching this limit or solving the problem, whichever happens first, each approach moves on to solve the next problem. We now discuss our results and analysis in detail below.

5.1 Results

Fig. 4 shows that COPlanLearn consistently solves all of the problems faster compared to the baselines in all the domains. This is reflected in the fraction of the total problems solved by COPlanLearn compared to the baselines. Option-Critic performs poorest in all the experiments, even when compared with CAT+RL which does not transfer. The reasons can be attributed to the benefits gained from learning state abstractions by CAT+RL, and insufficient diversity in reusable options learned by Option-Critic. Our approach COPlanLearn significantly outperforms the baselines in terms of sample efficiency as it learns both temporal as well as state abstractions, and as a result solves a higher number of problems using the same amount of learning experience as used by all the baselines. Empirically, we found that COPlanLearn learns one successful abstract trajectory within a few learning episodes and hence use it to compute an abstract state distance heuristic and invent new options. The learning episodes for COPlanLearn also includes the steps taken for execution of options.

5.2 Analysis

We now present our analysis and answer each of the three key questions raised earlier in Sec. 5.

Improved Sample efficiency The results in Fig. 4 corroborate that the symbolically represented options learned by our approach COPlanLearn have high utility in a transfer learning setting. COPlanLearn solves all the problems faster compared to the baselines in all the domains. The superior sample efficiency achieved by COPlanLearn demonstrates that it (i) derives benefits from learning both temporal and state abstractions, and (ii) effectively identifies reusable abstractions while discovering new relevant abstractions if needed.

Semantically meaningful options An important property of the learned options is that they express meaningful high-level behaviors and are interpretable due to their abstract representations. Fig. 5 shows a visualization of options learned in Taxi World highlighting their initiation and termination sets. The images show two options that clearly express: taxi navigation to the pickup location to pick up a passenger, and taxi navigation to the destination to drop off the passenger.

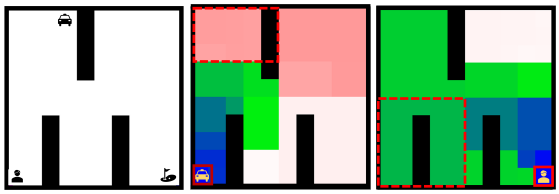
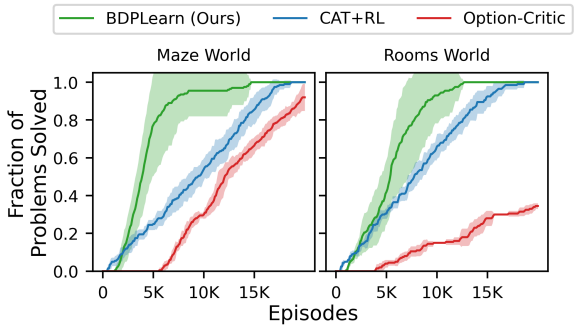


Figure 5: Visualization of options for Taxi World. Dashed and solid lines show the initiation and termination sets for an option respectively. Left: the initial state and the destination. Middle: an option for navigation to pickup the passenger. Right: an option for navigation to dropoff the passenger.

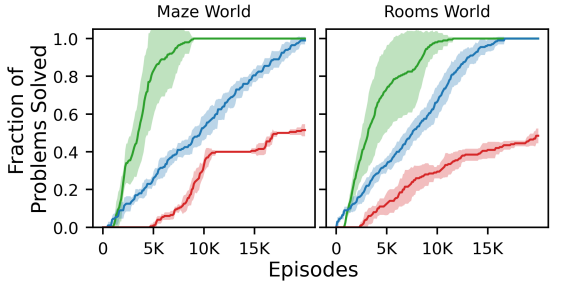
Transfer to more cluttered environments We conducted an additional study to investigate whether the abstractions learned by COPlanLearn express generalizable knowledge that can be transferred to more obstructed environments in Maze and Rooms domains. Fig.6 shows that COPlanLearn achieves superior performance even with increased obstacles in the environment. This demonstrates that COPlanLearn has a broader applicability and capability to transfer.

Transfer to larger environments We also conducted an additional study to test COPlanLearn's ability to transfer to environments that are much larger than the source environment. Fig. 6 clearly shows that options learned by COPlanLearn are generalizable as it effectively fine-tunes abstract policies for options in environments that are 9 times larger than the source environment.

As the target problem's state space differs from the source problem, COPlanLearn extrapolates the CAT learned from the source problem to build a new CAT for the target problem and creates a mapping between abstract states in these two CATs. It then uses this mapping to update the declarative abstract representations of the transferred options. The superior sample efficiency achieved clearly demonstrates it's capabilities to effectively generalize.



(a) Transfer from less to more cluttered environments.



(b) Transfer from smaller to larger environments.

Figure 6: An average of fraction of problems solved vs learning episodes when transferred from one source problem in a less cluttered (top) (Maze: 24×24, Rooms: 33×33) or smaller (down) (Maze: 8×8, Rooms: 11×11) environment to 20 target problems in a more cluttered (top) or larger (down) environment (Maze: 24×24, Rooms: 33×33) with different initial and goal configurations for Maze World and Rooms World domains.

6 Conclusions and Future Work

We presented novel approaches for learning symbolic options with abstract representations and composing the learned options to solve a target problem in a transfer learning setting. Extensive empirical evaluation demonstrated that the presented approach achieves superior sample efficiency compared to the baselines, even in environments that are more cluttered and larger than the source environments. The learned options are generalizable, composable, as well as express semantically meaningful high-level behaviors. In the future, we plan to investigate learning hierarchies of temporal and state abstractions in tandem.

6.1 Acknowledgement

This work was supported in part by the NSF Grant 1909370 and the ONR grant N000142312416.

References

D. Abel, N. Umbanhowar, K. Khetarpal, D. Arumugam, D. Precup, and M. Littman. Value preserving state-action abstractions. In *International Conference on Artificial Intelligence and Statistics*, pages 1639–1650. PMLR, 2020.

P.-L. Bacon and D. Precup. Using label propagation for learning temporally abstract actions in reinforcement learning. In *Proceedings of the Workshop on Multiagent Interaction Networks*, pages 1–7, 2013.

P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, 2017.

A. Bagaria and G. Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2020.

A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

M. Dadvar, R. K. Nayyar, and S. Srivastava. Conditional abstraction trees for sample-efficient reinforcement learning. In *Uncertainty in Artificial Intelligence*, pages 485–495. PMLR, 2023.

T. Dietterich. State abstraction in maxq hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 12, 1999.

T. G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

M. Grounds and D. Kudenko. Combining reinforcement learning with symbolic planning. In *European Symposium on Adaptive Agents and Multi-Agent Systems*, pages 75–86. Springer, 2005.

B. Hengst et al. Discovering hierarchy in reinforcement learning with hexq. In *Icml*, volume 19, pages 243–250. Citeseer, 2002.

R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116. PMLR, 2018.

L. Illanes, X. Yan, R. T. Icarte, and S. A. McIlraith. Symbolic plans as high-level instructions for reinforcement learning. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, pages 540–550, 2020.

A. Jonsson and A. Barto. Automated state abstraction for options using the u-tree algorithm. *Advances in neural information processing systems*, 13, 2000.

A. Jonsson and A. Barto. Causal graph based decomposition of factored mdps. *Journal of Machine Learning Research*, 7(11), 2006.

M. Klissarov and D. Precup. Flexible option learning. *Advances in Neural Information Processing Systems*, 34:4632–4646, 2021.

H. Kokel, A. Manoharan, S. Natarajan, B. Ravindran, and P. Tadepalli. Reprel: Integrating relational planning and reinforcement learning for effective abstraction. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pages 533–541, 2021.

G. Konidaris. On the necessity of abstraction. *Current opinion in behavioral sciences*, 29:1–7, 2019.

G. D. Konidaris and A. G. Barto. Building portable options: Skill transfer in reinforcement learning. In *Ijcai*, volume 7, pages 895–900, 2007.

D. Lyu, F. Yang, B. Liu, and S. Gustafson. Sdrl: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2970–2977, 2019.

M. C. Machado, M. G. Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, pages 2295–2304. PMLR, 2017.

S. Mannor, I. Menache, A. Hoze, and U. Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 71, 2004.

A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning, 2001*, 2001.

N. Mehta, S. Ray, P. Tadepalli, and T. Dietterich. Automatic discovery and transfer of maxq hierarchies. In *Proceedings of the 25th international conference on Machine learning*, pages 648–655, 2008.

I. Menache, S. Mannor, and N. Shimkin. Q-cut—dynamic discovery of sub-goals in reinforcement learning. In *Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings 13*, pages 295–306. Springer, 2002.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, 10, 1997.

A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann. Stable baselines3, 2019.

R. Ramesh, M. Tomar, and B. Ravindran. Successor options: An option discovery framework for reinforcement learning. *arXiv preprint arXiv:1905.05731*, 2019.

B. Ravindran and A. G. Barto. Smdp homomorphisms: an algebraic approach to abstraction in semi-markov decision processes. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1011–1016, 2003.

M. Riemer, M. Liu, and G. Tesauro. Learning abstract options. *Advances in neural information processing systems*, 31, 2018.

T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. Kaelbling, and J. B. Tenenbaum. Predicate invention for bilevel planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12120–12129, 2023.

Ö. Şimşek and A. G. Barto. Betweenness centrality as a basis for forming skills. Technical Report 07-26, University of Massachusetts, 2007.

M. Stolle and D. Precup. Learning options in reinforcement learning. In *Abstraction, Reformulation, and Approximation: 5th International Symposium, SARA 2002 Kananaskis, Alberta, Canada August 2–4, 2002 Proceedings 5*, pages 212–223. Springer, 2002.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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.

M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

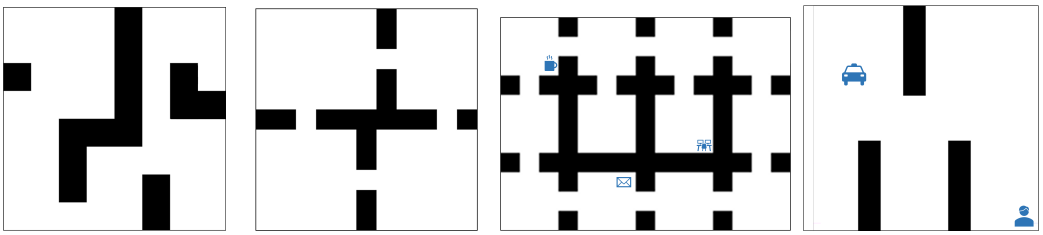
T. J. Walsh, L. Li, and M. L. Littman. Transferring state abstractions between mdps. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

F. Yang, D. Lyu, B. Liu, and S. Gustafson. Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. *arXiv preprint arXiv:1804.07779*, 2018.

7 Appendix

7.1 Environment Details



(a) Maze (24×24) (b) Four Rooms (33×33) (c) Office (11×15) (d) Taxi (30×30)

Figure 7: Office, Taxi, Maze, and Four Rooms World environments.

Maze World We consider a Maze World Ramesh et al. (2019) of dimensions 24×24 (Fig. 7a) with randomly placed walls as obstacles. The agent can take up, down, left and right actions. Upon applying any action, the agent executes the action successfully with a probability of 0.8 and may slip to one of the two adjacent cells with a probability of 0.1 each. The agent receives a reward of 500 on completing the task of reaching the goal from the start successfully, -2 on bumping against the wall, and -1 otherwise. Target problems differ from the source problems in the initial and destination locations of the agent.

We conducted additional studies to evaluate transfer from small (Fig. 7a map of size 8×8) to large (Fig. 7a map of size 24×24) environments, and from less cluttered (Fig. 8a map of size 24×24) to more (Fig. 8b map of size 24×24) cluttered environments in the Maze World domain.

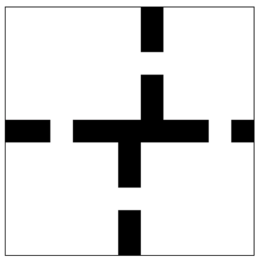
(a) Less cluttered (24×24) (b) More cluttered (24×24)

Figure 8: Transfer from less cluttered (left) to more cluttered (right) environments in Maze World.

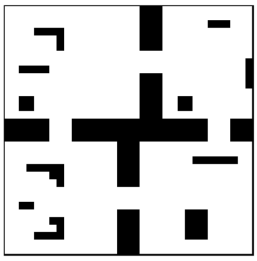
Four Rooms World We consider a Four Rooms World from Sutton et al. (1999) of dimensions 33×33 (Fig. 7b) where an agent can navigate within rooms and through hallways to other rooms. The actions, stochastic probabilities, and the reward structure are the same as in the Maze World. The agent starts from a random location in any room and needs to reach a particular destination in a different room. The initial and goal locations for all source and target problems are different.

We also conducted additional studies to evaluate transfer from smaller (Fig. 9a map of size 8×8) to larger environments (Fig. 9a map of size 24×24) as well as from less cluttered (Fig. 9a map of size 24×24) to more cluttered (Fig. 9b map of size 24×24) environments in the Four Rooms domain.

Office World We consider an Office World from Icarte et al. (2018) with dimensions 11×15 (Fig. 7c). The agent can take up, down, left and right actions. On applying any action, the agent executes the action successfully with a probability of 0.8 and may slip to one of the two adjacent cells with a probability of 0.1 each. The agent needs to pick-up coffee, then pick-up mail, and deliver them both to an office in an environment with various rooms and offices. The agent receives a reward of 500 when the agent delivers coffee and mail to the office, and 0 otherwise. Target problems differ from the source problems in the initial location of the agent and the location of the office.



(a) Less cluttered (24×24)



(b) More cluttered (24×24)

Figure 9: Transfer from less cluttered (left) to more cluttered (right) environments in Four Rooms.

Taxi World We consider a scenario of Taxi World, a classic benchmark introduced by Dietterich (2000) of dimensions 30×30 (Fig. 7d). There are four pick-up and drop-off locations, one in each corner of the grid. The agent can take up, down, left, right, pick-up, and drop-off actions. Each move action has stochastic probabilities similar to Office world. Taxi World has a taxi that starts from a random location and needs to pick up a passenger from its pick-up location and drop-off at its destination location. The agent obtains a reward of -1 on applying a move action and -100 on illegal pick-up and drop-off actions. Upon dropping the passenger at the correct destination, it receives a positive reward of 500. Target problems differ from the source problems in the initial location of the agent and the drop-off location for the passenger.

Mountain Car We consider the continuous state and discrete action environment from Open AI Gym [1]. The agent receives -1 reward on each step and 500 reward on reaching the goal position. The maximum number of steps allowed in an episode is 200. All the target and source problems differ in the initial and goal locations of the taxi.

7.2 Hyper-parameters

We used the open-source code available for the state-of-the-art baselines Option-Critic [2], CAT+RL [3], and DQN implemented in Stable-Baselines3 [4] framework by Raffin et al. (2019). The presented approach COPlanLearn does not need to predefine the number of options as Option-Critic. COPlanLearn's uses parameters n_{check}, t_{succ}, and cap k similar to CAT+RL. n_{check} is the interval of number of episodes at which abstraction is evaluated for refinement, t_{succ} is the threshold of success rate, and the cap k is the maximum number of unstable states that can be refined in each refinement. f is the factor used to control the abstraction distance heuristic of edges retained and is set to $\frac{1}{4}$ for all domains. n_{eval} is the number of episodes for which the learned policy is evaluated which is set to 100 for all the approaches. Tables 1, 2, 3, 4, 5, 6, 7, 8, and 9 show all the hyper-parameters for all the approaches and domains.

[1] https://www.gymlibrary.dev/environments/classic_control/mountain_car/

[2] https://github.com/lweitkamp/option-critic-pytorch

[3] https://github.com/AAIR-lab/CAT-RL.git

[4] https://github.com/DLR-RM/stable-baselines3

Hyper-parameters	COPlanLearn	Option-Critic	CAT+RL
Threshold (t_{succ})	0.9	0.9	0.9
n_{check}	100	100	100
n_{eval}	100	100	100
Cap (k)	10	10	10
Exploration rate (ϵ)	1.0	–	1.0
Minimum exploration rate	0.05	–	0.05
Exploration decay	0.9996	–	0.9996
Discount factor (γ)	0.99	0.99	0.99
Number of episodes	5000	5000	5000
Maximum episode length	500	500	500
Maximum option length	500	–	–
Number of predefined options	–	8	–

Table 1: Parameters used in Maze World.

Hyper-parameters	COPlanLearn	Option-Critic	CAT+RL
Threshold (t_{succ})	0.9	0.9	0.9
n_{check}	100	100	100
n_{eval}	100	100	100
Cap (k)	10	10	10
Exploration rate (ϵ)	1.0	–	1.0
Minimum exploration rate	0.05	–	0.05
Exploration decay	0.9996	–	0.9996
Discount factor (γ)	0.99	0.99	0.99
Number of episodes	5000	5000	5000
Maximum episode length	500	500	500
Maximum option length	500	–	–
Number of predefined options	–	8	–

Table 2: Parameters used in Maze World (less to more cluttered environment).

Hyper-parameters	COPlanLearn	Option-Critic	CAT+RL
Threshold (t_{succ})	0.9	0.9	0.9
n_{check}	100	100	100
n_{eval}	100	100	100
Cap (k)	10	10	10
Exploration rate (ϵ)	1.0	–	1.0
Minimum exploration rate	0.05	–	0.05
Exploration decay (small environment)	0.991	–	0.991
Exploration decay (large environment)	0.9996	–	0.9996
Discount factor (γ)	0.99	0.99	0.99
Number of episodes	5000	5000	5000
Maximum episode length (small environment)	75	75	75
Maximum episode length (large environment)	500	500	500
Maximum option length	500	–	–
Number of predefined options	–	8	–

Table 3: Parameters used in Maze World (small to large environment).

Hyper-parameters	COPlanLearn	Option-Critic	CAT+RL
Threshold (t_{succ})	0.9	0.9	0.9
n_{check}	100	100	100
n_{eval}	100	100	100
Cap (k)	10	10	10
Exploration rate (ϵ)	1.0	–	1.0
Minimum exploration rate	0.05	–	0.05
Exploration decay	0.9996	–	0.9996
Discount factor (γ)	0.99	0.99	0.99
Number of episodes	5000	5000	5000
Maximum horizon	600	600	600
Maximum option length	600	–	–
Number of predefined options	–	8	–

Table 4: Parameters used in Four Rooms World.

Hyperparameters	COPlanLearn	Option-Critic	CAT+RL
Threshold (t_{succ})	0.9	0.9	0.9
n_{check}	100	100	100
n_{eval}	100	100	100
Cap (k)	10	10	10
Exploration rate (ϵ)	1.0	–	1.0
Minimum exploration rate	0.05	–	0.05
Exploration decay	0.9996	–	0.9996
Discount factor (γ)	0.99	0.99	0.99
Number of episodes	5000	5000	5000
Maximum horizon	600	600	600
Maximum option length	600	–	–
Number of predefined options	–	8	–

Table 5: Parameters used in Four Rooms World (less to more cluttered environment).

Hyperparameters	COPlanLearn	Option-Critic	CAT+RL
Threshold (t_{succ})	0.9	0.9	0.9
n_{check}	100	100	100
n_{eval}	100	100	100
Cap (k)	10	10	10
Exploration rate (ϵ)	1.0	–	1.0
Minimum exploration rate	0.05	–	0.05
Exploration decay for small environment	0.991	–	0.991
Exploration decay for large environment	0.9996	–	0.9996
Discount factor (γ)	0.99	0.99	0.99
Number of episodes	5000	5000	5000
Maximum episode length (small environment)	100	100	100
Maximum episode length (large environment)	600	600	600
Maximum option length	600	–	–
Number of predefined options	–	8	–

Table 6: Parameters used in Four Rooms World (small to large environment).

Hyperparameters	COPlanLearn	Option-Critic	CAT+RL
Threshold (t_{succ})	0.9	0.9	0.9
n_{check}	100	100	100
n_{eval}	100	100	100
Cap (k)	10	10	10
Exploration rate (ϵ)	1.0	–	1.0
Minimum exploration rate	0.05	–	0.05
Exploration decay	0.9991	–	0.9991
Discount factor (γ)	0.99	0.99	0.99
Number of episodes	5000	5000	5000
Maximum episode length	300	300	300
Maximum option length	300	–	–
Number of predefined options	–	8	–

Table 7: Parameters used in Office World.

Hyperparameters	COPlanLearn	Option-Critic	CAT+RL
Threshold (t_{succ})	0.9	0.9	0.9
n_{check}	100	100	100
n_{eval}	100	100	100
Cap (k)	10	10	10
Exploration rate (ϵ)	1.0	–	1.0
Minimum exploration rate	0.05	–	0.05
Exploration decay	0.992	–	0.992
Discount factor (γ)	0.99	0.99	0.99
Number of episodes	7000	7000	7000
Maximum episode length	1500	1500	1500
Maximum option length	1500	–	–
Number of predefined options	–	8	–

Table 8: Parameters used in Taxi World.

Hyper-parameters	COPlanLearn	DQN	CAT+RL
Threshold (t_{succ})	0.9	0.9	0.9
n_{check}	100	–	100
n_{eval}	100	–	100
Cap (k)	10	–	10
Exploration rate (ϵ)	1.0	1.0	1.0
Minimum exploration rate	0.05	0.05	0.05
Exploration decay	0.9996	0.99	0.9996
Discount factor (γ)	0.99	0.99	0.99
Number of episodes	5000	5000	5000
Maximum episode length	200	200	200
Maximum option length	200	–	–
Number of predefined options	–	–	–

Table 9: Parameters used in Mountain Car.