Factored State Abstraction for Option Learning

by

Marwa Abdulhai

S.B. Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 2021

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2021

Author .....
the Department of Electrical Engineering and Computer Science
August 17, 2021

Certified by .....
Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics, MIT
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Factored State Abstraction for Option Learning

by

Marwa Abdulhai

Submitted to the Department of Electrical Engineering and Computer Science
on August 17, 2021, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Hierarchical reinforcement learning has focused on discovering temporally extended actions (options) to provide efficient solutions for long-horizon decision-making problems with sparse rewards. One promising approach that learns these options end-to-end in this setting is the option-critic (OC) framework. However, there are several practical limitations of this method, including the lack of diversity between the learned sub-policies and sample inefficiency. This thesis shows that the OC framework does not decompose problems into smaller and largely independent components, but instead increases the problem complexity with each option by considering the entire state space during learning. To address this issue, we introduce state abstracted option-critic (SOC), a new framework that considers both temporal and state abstraction to effectively reduce the problem complexity in sparse reward settings. Our contribution includes learning a factored state space to enable each option to map to a sub-section of the state space. We test our method against hierarchical, non-hierarchical, and state abstraction baselines to demonstrate better sample efficiency and higher overall performance in both image and large vector-state representations under sparse reward settings.

Thesis Supervisor: Jonathan P. How
Title: R. C. Maclaurin Professor of Aeronautics and Astronautics, MIT

Acknowledgments

Working in the Aerospace Controls Laboratory and in collaboration with MIT-IBM as a SuperUROP in 2019-2020 and later as a Masters Student in 2020-2021 has been an incredible learning experience. I would like to thank Professor How, Dong-Ki, Matt, Miao, and Gerry for their guidance from the inception of choosing my project to (hopefully) publishing my first, first author paper. My research was funded by IBM and Samsung (as part of the MIT-IBM Watson AI Lab initiative), with computational support received through Amazon Web Services.

I am incredibly grateful to Dong-Ki who was my mentor throughout this entire process – teaching me how to hone my skills as a researcher, having multiple meetings with me to frame my contributions and method, and providing much needed critical feedback to help me grow. Dong-Ki's inspiring work ethic and faith in me as a researcher propelled me to apply to pursue a Ph.D. I am also incredibly grateful to Professor How for his mentorship from the early days of taking 6.141 under his instruction in 2018 to eventually advising me on my MEng thesis. Professor How & Dong-Ki's guidance in helping me with my first steps will always be a part of my journey as a researcher.

I would also like to thank my family – my father for believing in me throughout my undergraduate, Masters, and further graduate study aspirations, my mother for having the utmost faith in my capabilities, and my sisters Safa & Rumaisa for providing much needed distractions along the way. I would also like to thank my dear friend & roommate Ayşe and our cats Medina & Eevee for helping me de-stress with food, smiles, and meows after every meeting. Finally, I would like to thank God – the Most Gracious, the Most Merciful – for giving me the chance to come this far and meet so many wonderful people at MIT. I am truly grateful for everyone who has taught me all that I know, and I am excited for the experiences awaiting in my next journey of more graduate school at UC Berkeley!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Hierarchical reinforcement learning (HRL) provides a principled framework for decomposing problems into natural hierarchical structures [11]. By factoring a complex task into simpler sub-tasks, HRL offers a benefit over non-HRL approaches in solving problems with long horizons and delayed rewards. Extensive research in HRL has focused on discovering temporally extended actions (also referred to as options) that can be viewed as specialized skills to improve learning and planning efficiency [28, 32]. These works include the option-critic (OC) method that simultaneously discovers and learns option policies and termination conditions without requiring domain-specific knowledge [2].

However, the OC framework often shows empirical weaknesses, including lack of option diversity, short option duration, and large sample complexity required in learning options [19]. The key insight of this thesis is that OC suffers from these problems due to considering the entire state space in option learning and thus failing to reduce problem complexity as intended. For example, consider a maze domain in Fig. 1-1. The objective in this domain is to pick up the key and unlock the door to reach the green goal. A non-HRL learning problem in this setting can be viewed as a search over policy space $|\mathcal{A}|^{|\mathcal{S}|}$, where $|\mathcal{A}|$ and $|\mathcal{S}|$ denote the size of action space and state space, respectively. The OC framework considers the entire state space in

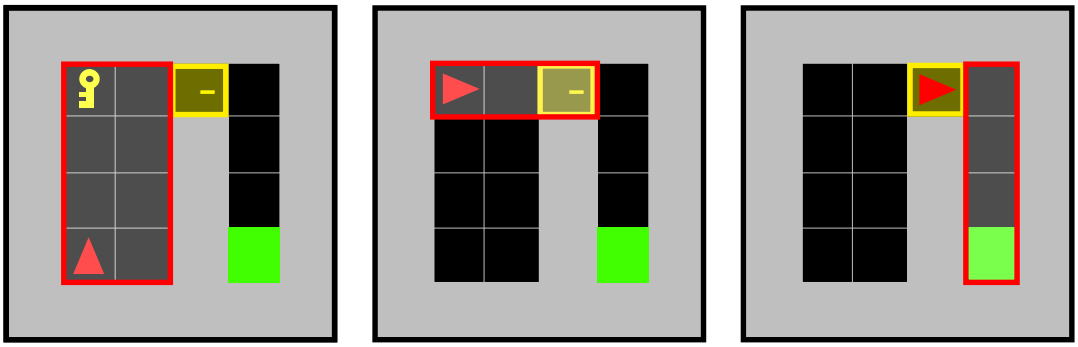


Figure 1-1: Decomposition in this maze consists of 3 sub-tasks; getting the key, opening the door, and going to the green goal. The agent should only focus on parts of the state pertaining to the sub-task as highlighted in red. Equipped with both temporal and state abstraction, the method proposed in this thesis (SOC) achieves a significant reduction in problem size.

learning options, resulting in problem size of $(|\Omega||\mathcal{A}|)^{|\mathcal{S}|}$ where $|\Omega|$ denotes the number of options. Because $|\Omega| \geq 1$, learning options can in fact increase problem complexity.

To address this issue, I propose considering both temporal abstraction (in the form of high-level skills) and state abstraction (in the form of symbolic representations) in learning options to fully benefit from hierarchical learning. Considering the maze example again, the agent does not need to consider the states related to going to the goal when trying to get the key (see Sub-Task 1 in Fig. 1-1). Similarly, the agent needs to only consider states that are relevant to solving the other sub-tasks. Hence, this state abstraction leads to the desired reduction in problem complexity, and an agent can mutually benefit from having both state abstraction and temporal abstraction in hierarchical learning.

1.2 Contribution

With this insight, this thesis introduces a new option learning framework: State-Abstracted Option-Critic (SOC). My main contribution is extending the option-critic framework to additionally consider state abstraction. By learning a factored state space (i.e. grouping similar states into smaller components), I achieve a significant reduction in the search over policy space, with the following benefits:

- **Improved Sample Efficiency**: SOC provides options to select learned factored components. This allows effective decomposition of problem size when scaling up to large state and action spaces, leading to faster learning.

- **Coordination between Options**: Unlike OC where each option policy is learned independently, SOC allows options to share information with one another. This leads to flexible and compositional relationships, functionalities that lead to learning diverse options.

My evaluation shows that SOC outperforms the HRL baseline option-critic [2], non-HRL baseline A2C [25], and state abstraction baseline A2C-RIMs [14] in both sample efficiency and converged performance.

Chapter 2

Preliminaries

2.1 Markov Decision Process

An agent's interaction in the environment can be represented by a Markov Decision Process (MDP) [30]. Specifically, a MDP is defined as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$; $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ is the state transition probability function, $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. An agent executes an action at each timestep t according to its stochastic policy $a_t \sim \pi_\theta(a_t|s_t)$ parameterized by θ, where $s_t \in \mathcal{S}$. An action a_t yields a transition from the current state s_t to the next state $s_{t+1} \in \mathcal{S}$ with probability $P(s_{t+1}|s_t, a_t)$. An agent then obtains a reward according to the reward function $r_t = \mathcal{R}(s_t, a_t)$. An agent's goal is to maximize its expected return $\mathbb{E}\left[\sum_t \gamma^t r_t | s_0\right]$.

2.2 Options

A Markovian option $\omega \in \Omega$ consists of a triple $(I_\omega, \pi_\omega, \beta_\omega)$ in which $I_\omega \in \mathcal{S}$ is an initiation set, π_ω is an intra-option policy and $\beta_\omega : \mathcal{S} \to [0, 1]$ is a option termination function [32]. Similar to option discovery approaches [9, 24, 34], all options are available in each state. MDPs with options become SMDPs [29] with an associated optimal value function over options $V^*_\Omega(s_t)$ and option-value function $Q^*_\Omega(s_t, \omega_t)$, where ω_t denotes the selected option at timestep t.

2.3 Option-Critic Framework

Bacon et al. [2] introduces a framework to learn a series of actions represented by an option ω. An option ω_t is selected according to a policy over options $\pi_\Omega(\omega_t|s_t)$. The intra-option policy π_ω selects primitive action a_t until termination of the option (as determined by β_ω), which triggers a repetition of this procedure. With the intra-option policy $\pi_{\omega,\theta}$ parameterized by θ and the termination function $\beta_{\omega,\vartheta}$ parameterized by ϑ, the option-value function can be written as:

$$Q_\Omega(s_t, \omega_t) = \sum_{a_t} \pi_{\omega,\theta}(a_t|s_t) Q_U(s_t, \omega_t, a_t), \tag{2.1}$$

where $Q_U : \mathcal{S} \times \Omega \times \mathcal{A} \to \mathcal{R}$ is the option-value function upon arrival:

$$Q_U(s_t, \omega_t, a_t) = r_t + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) U(s_{t+1}, \omega_t). \tag{2.2}$$

The value of executing ω_t upon entering state s_{t+1} is given by:

$$U(s_{t+1}, \omega_t) = (1 - \beta_{\omega,\vartheta}(s_{t+1})) Q_\Omega(s_{t+1}, \omega_t) + \beta_{\omega,\vartheta}(s_{t+1}) V_\Omega(s_{t+1}). \tag{2.3}$$

Chapter 3

State-Abstracted Option-Critic

This sections shows the importance of state abstraction in significantly reducing the problem size per option, an optimization encompassing both state and temporal abstraction, and the policy gradient for intra-option policy learning.

3.1 Decomposition of Problem Size

I begin by formally motivating the need for state abstraction. As described in Fig. 1-1, the size of a learning problem exponentially increases as the number of states increases. I have showed that OC results in larger search problem size of $(|\Omega||\mathcal{A}|)^{|\mathcal{S}|}$ and does not provide the ability to scale as is expected from HRL methods (see Introduction). My insight is that options can still make this problem simpler to solve by considering both temporal and state abstraction components.

Corollary 1 *(Decomposition of State).* With state abstraction, problem size can be reduced such that:

$$(|\Omega||\mathcal{A}|)^{|\tilde{\mathcal{S}}_\omega|} \ll |\mathcal{A}|^{|\mathcal{S}|} \ll (|\Omega||\mathcal{A}|)^{|\mathcal{S}|}. \tag{3.1}$$

Here, $\tilde{\mathcal{S}}_\omega \in \mathcal{S}$ denotes the section of the state space where option ω operates, with $\tilde{\mathcal{S}}_\omega \ll \mathcal{S}$. I illustrate this idea further in Fig. 3-1 where each option maps to a section

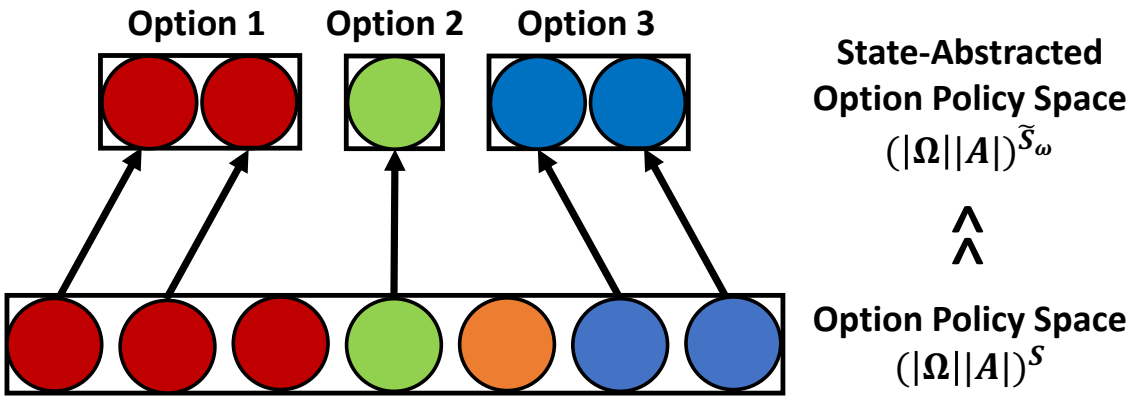


Figure 3-1: Having each option map to a smaller subset of the state space allows for the search over policy space to be significantly reduced compared to considering the entire state space for each option.

of the state space. Options decompose the learning problem in a way that makes finding a policy for a specific task easier, by only considering subset of the state space as shown in Corollary 1. This results in exponential decrease in the size of the effective state space for each option over size of the true state space across all factors.

To further elucidate the reduction in problem size, I apply this formulation to the example in Fig. 1-1, where $|\mathcal{A}| = 4$ and $|\mathcal{S}| = 16$. With temporal abstraction and $|\Omega| = 3$, I get a problem size of $(3 \times 4)^1 6 = 1.85^{17}$. With both temporal and state abstraction, I obtain a much smaller problem size of 6.58^4. This number is determined by considering the smaller state size for each option as follows (from option 1 to option 3): $4^8 + 4^3 + 4^4$.

Although a combination of temporal and state abstraction yields a significant reduction in the problem size, there is a theoretical limit to the amount of state abstraction that will maintain the equality described in Corollary 3.1 of ensuring the search over policy space is minimized. As adding more options in the SOC framework results in further decomposition of the state space, I bound the number of options to maintain a minimized search over policy space based on the following condition:

Corollary 2 *(A Limit on Number of Options)* An option ω can be split into 2 options ω' and ω'' with smaller state spaces maintaining $|\pi_{\omega,\theta}| \geq |\pi_{\omega',\theta}| + |\pi_{\omega'',\theta}|$ such

that:

$$|\pi_{\omega,\theta}| = |\mathcal{A}|^{|\tilde{\mathcal{S}}_\omega|} \geq |\mathcal{A}|^{|\tilde{\mathcal{S}}_{\omega'}|} + |\mathcal{A}|^{|\tilde{\mathcal{S}}_{\omega''}|} \tag{3.2}$$

In other words, adding an additional option is only helpful when considering if the sum of the sub-policies created from splitting the option into two is smaller than the single option on its own, as shown in Corollary 2.

3.1.1 Problem Statement with State Abstraction

I now formally define the decomposition of problem size shown in Corollary 1 in the optimization, where the discounted return expected over all the trajectories starting at an initial factored state $\tilde{s}_0$ and option ω_0:

$$\rho(\Omega, \theta, \vartheta, \tilde{\mu}, \tilde{s}_0, \omega_0) = \mathbb{E}_{\Omega,\theta,\vartheta,\tilde{\mu}}\Big[\sum_{t=0}^{\infty} \gamma^t r_t | \tilde{s}_0, \omega_0\Big], \tag{3.3}$$

where $\tilde{\mu}$ is the Markov abstraction i.e. $\tilde{\mu} : \mathcal{S} \rightarrow \tilde{\mathcal{S}}_\omega$ that maps states from one state space $\mathcal{S}$ to states from a different state space $\tilde{\mathcal{S}}_\omega$ with $|\tilde{\mathcal{S}}_\omega| \ll |\mathcal{S}|$. Following van Seijen et al. [33], I re-define the trajectory $(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \ldots, s_H)$ as a difference sequence via abstraction $\tilde{\mu}$: $(\tilde{s}_t, a_t, r_t, \tilde{s}_{t+1}, a_{t+1}, r_{t+1}, \ldots, s_H)$ with $\tilde{s}_t = \tilde{\mu}(s_t)$ with $t = 1, \ldots, H$, where H represents the episodic horizon. For notation simplicity, I denote the option-specific state $\tilde{s}_{\omega_t}$ as $\tilde{s}_t$ throughout this thesis.

I define the abstraction performed in this approach as a mapping to a context-specific state space, where different states are described by different state components of $\mathcal{S}$. I will assume that this decomposition of the state space can be described as a factored MDP, which I will describe in the next subsection. These assumptions will be useful when considering how the hierarchical nature of options can capture the structure of the environment through abstraction.

Option Learning Gradients. Given a set of Markov options with stochastic intra-option policies differentiable in their parameters θ, I denote the gradient of the ex-

pected discounted return with respect to θ and initial condition $(\tilde{s}_0, \omega_0)$:

$$\sum_{s_t,\omega_t} \mu_\Omega(s_t, \omega_t | s_0, \omega_0) \sum a_t \frac{\partial \pi_{\omega,\theta}(a_t|\tilde{s}_t, \omega_t)}{\partial \theta} Q_U(s_t, \omega_t, a_t), \tag{3.4}$$

where μ_Ω denotes the discounted weighting of (s_t, ω_t) along trajectories originating from (s_0, ω_0). I expand the option-value function upon arrival with state abstraction as:

$$Q_U(s_t, \omega_t, a_t) = r_t + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) U(\tilde{\mu}(s_{t+1}), \omega_t). \tag{3.5}$$

where $U(\tilde{\mu}(s_{t+1}), \omega_t)$ equals to:

$$(1 - \beta_{\omega,\vartheta}(\tilde{s}_{t+1})) Q_\Omega(s_{t+1}, \omega_t) + \beta_{\omega,\vartheta}(\tilde{s}_{t+1}) V_\Omega(s_{t+1}). \tag{3.6}$$

Regarding the termination function, the gradient of the expected discounted return objective with respect to ϑ and the initial condition $(\tilde{s}_1, \omega_0)$ is:

$$\sum_{a_t} \pi_{\omega,\theta}(a_t|\tilde{s}_t, \omega_t) \sum_{s_{t+1}} \gamma P(s_{t+1}|s_t, a_t) \frac{\partial U(\tilde{\mu}(s_{t+1}), \omega_t)}{\partial \vartheta}. \tag{3.7}$$

3.1.2 State Abstraction Properties

In order to map each option ω_t to a subsection of the state, there are several required properties of the state space.

In particular, I assume that the state distribution of the environment is defined by a set of sub-MDPs. Factoring the state into a series of state variables known as the factored state space allows us to represent the full set of knowledge relevant to the underlying dynamics of the world at large.

Factored MDP. The structure of the environment can be modeled as a factored MDP, where each state $\mathcal{S}$ is factored into n variables: $\mathcal{S} = \{\mathcal{S}^1, \ldots, \mathcal{S}^n\}$ [15]. Each $\mathcal{S}^i$ will take on values in a finite domain. A state variable $s \in \mathcal{S}$ is an assignment of values to the set of state variables, i.e. $s = [s^1, \ldots, s^n]$ such that $\mathcal{S} \subseteq \mathcal{S}^1 \times \ldots \times \mathcal{S}^n$.

[16]. These variables are also assumed to be largely independent of each other with a small number of causal parents $\mathcal{P}_a^{\mathcal{S}^i}$ to consider at each timestep (i.e. $\mathcal{P}_{a_t}^{\mathcal{S}^i} \ll |\mathcal{S}|$). As a result, I can consider the following simplification of the environment's transition function:

$$P(s_{t+1}|s_t, a_t) \approx \prod_{i=1}^{n} P^i(s_{t+1}^i|\mathcal{P}_a^{\mathcal{S}^i}(s_t), a_t). \tag{3.8}$$

The definition of the factored MDP as a group of independent variables allows us to assume that any option mapping to a different subset of these variables will learn a unique policy.

Context-Specific Independence. Hence at any given time, only a small subset of this factored state space may be necessary for the agent to be aware of at any given moment. As per Boutilier et al. [5], I define this subset to be a context A context Z is a pair $(Z, \mathcal{Z})$ where $Z \subseteq S$ is some subset of state variables and $\mathcal{Z}$ is the space of possible joint assignments of state variables in the subset. A state s is in the context $(Z, \mathcal{Z})$ when its joint assignment of variables in Z is present in $\mathcal{Z}$. Two variables $X, Y \subseteq S \setminus Z$ are contextually independent under $(Z, \mathcal{Z})$ if $Pr(X|Y, Z = z) = Pr(X|Z = z)\ \forall z \in \mathcal{Z}$. This independence relation is referred to as context specific independence (CSI) [8].

Therefore when in a specific context, only parts of the state of the world are relevant. This can be considered as a naturally occurring notion of a task or sub-task that emerges from the structure of the environment and the definition of CSI, allowing us to take advantage of hierarchical reinforcement learning paradigms such as option-critic. An abstraction of the environment can decompose the environment into sub-tasks. Any single sub-task maps to a single or portion of a context specific abstraction that defines its own unique subset of the state space. It may possibly share a part of the state space with another task, which can map to any single or portion of a context specific abstraction as well.

Context Specific Abstract MDPs. Intuitively, a variable is relevant to an MDP if there is any possibility that its value at some timestep will have an eventual influence, directly or indirectly, on the value of the reward. However, in a given context it is possible to define a less restrictive notion of local relevance. A relevant-variables projection is a simple projective abstraction as in Baum et al. [3], Boutilier [4], Gardiol and Kaelbling [13] that selects a subset of relevant variables in a context $\sigma_{(Z,\mathcal{Z})}([s^1, s^2, \ldots, s^n]) = [s^{i_1}, s^{i_2}, \ldots, s^{i_r}]$ where the superscript i refers to state variables that are not dropped and the new abstract state contains $r \leq n$ variables.

Connection to HRL. It is well known that sample efficiency strongly depends on the size of the input space. As such, I reduce the size of the input space an agent's option must concern itself with such that $|Z| \leq |\tilde{\mathcal{S}}_\omega|$, allowing an option ω to be learned efficiently. The hierarchical nature of option-critic allows the agent to identify the causal independent mechanisms that govern the transition dynamics. The agent can simplify its understanding of the environment further in the context of the particular problem it is interested in by leveraging a notion of a option relevant state space. In this sense, an agent can fully incorporate its learning from other relevant options that share state variables while ignoring irrelevant details to the task at hand. Recent approaches have also pointed to abstraction as the core of the efficiency issue [36], noting that an abstraction of the state space could lift the agent to a higher-level state space with lower dimensions. This allow for efficient exploration and environment modeling, which are the core benefits of hierarchical reinforcement learning paradigms.

3.2 OC with Entropy Maximization

I re-formulate the option critic framework to learn the maximum-entropy intra-option policies and define soft option policy evaluation and soft option improvement. The new objective to maximize the discounted return and entropy over actions for states

expected over all trajectories:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{\tau}[r_t + \alpha_\theta \mathcal{H}(\pi_{\omega,\theta}(\cdot|s_t))] \tag{3.9}$$

where τ denotes each sampled trajectory, intra-option policy of ω is parametrized by θ, and α_θ represents the entropy constant for the intra-option level.

As done in option-critic, I will employ the the call-and-return option execution model, in which an agent picks option ω according to its policy over options π_Ω, then follows the intra-option policy $\pi_{\omega,\theta}$ until termination dictated by $\beta_{\omega,v}$, a process which is repeated for a number of steps.

I will use function approximators to learn the inter-Q function $Q_\psi(s_t, \omega_t)$, intra-Q function $Q_\phi(s_t, \omega_t, a_t)$, intra-option policies $\pi_{\omega,\theta}(s_t, a_t)$, and termination policies $\beta_{\omega,v}(s_t)$. This framework does not learn the inter-option policy and derive option ω from the inter-option Q function as in the option-critic paper. The entropy at the inter-option level is also ignored as the the policy over options is deterministic i.e. exploiting Q.

3.2.1 Definitions

In order to take the gradients with respect to ψ, θ, ϕ and v for inter-option, intra-option, and termination levels, I must re-define equations similar to those used in option critic by including entropy maximization in their objectives. I define the inter-option value function, given a state s_t and option ω_t:

$$Q_\psi(s_t, \omega_t) = \mathbb{E}_{a_t}[Q_\phi(s_t, \omega_t, a_t) - \alpha_\theta \log \pi_{\omega,\theta}(s_t, a_t) \tag{3.10}$$

where $Q_\phi : \mathcal{S} \times \Omega \times \mathcal{A} \to \mathbb{R}$ is the value of executing an action in the context of a state-option pair. I define the intra-option value function, given a state s_t, option ω_t, and action a_t:

$$Q_\phi(\omega_t, s_t, a_t) = r_t + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) U(\omega_t, s_{t+1}), \tag{3.11}$$

$$U(\omega_t, s_{t+1}) = (1 - \beta_{\omega,v}(s_{t+1}))Q_\psi(s_{t+1}, \omega_t) + \beta_{\omega,v}(s_{t+1})V_\psi(s_{t+1}), \tag{3.12}$$

$$V_\psi(s_{t+1}) = \max_{\omega_{t+1}} Q_\psi(s_{t+1}, \omega_{t+1}) \tag{3.13}$$

I use the max for the value function in the epsilon-greedy case as per the option critic paper.

3.2.2 Soft Option Learning

I formulate SOC to learn the intra-option policies $\pi_{\omega,\theta}(s_t, a_t)$, the termination policies $\beta_{\omega,v}(s_t)$, intra-Q function $Q_\phi(s_t, \omega_t, a_t)$ and inter-Q function $Q_\psi(s_t, \omega_t)$. This approach alternates between optimizing the networks with stochastic gradient descent as in SAC.

3.2.3 Soft Option Evaluation

The inter-option Q value function trained to minimize the squared residual and optimized as follows:

$$J_Q(\psi) = \mathbb{E}_{s_t,\omega_t \sim D}\Big[\frac{1}{2}\Big(Q_\psi(s_t, \omega_t) - \mathbb{E}_{a_t \sim \pi_{\omega,\theta}}\big[\bar{Q}_\phi(s_t, \omega_t, a_t) - \alpha_\theta \log \pi_{\omega,\theta}(s_t, a_t)\big]\Big)^2\Big], \tag{3.14}$$

where $\bar{Q}_\phi$ is the target intra-Q function.

The intra Q-function is trained to minimize the soft Bellman residual and optimized as follows:

$$J_Q(\phi) = \mathbb{E}_{s_t,\omega_t,a_t,r_t,s_{t+1} \sim D}\Big[\frac{1}{2}\Big(Q_\phi(s_t, \omega_t, a_t) - \big(r_t + \gamma\Big[(1 - \beta_{\omega,v}(s_{t+1}))\bar{Q}_\psi(s_{t+1}, \omega_t) + \beta_{\omega,v}(s_{t+1})\mathbb{E}_{w_{t+1} \sim \pi_\Omega}\big[\bar{Q_\psi}(s_{t+1}, \omega_{t+1})\big]\Big]\big)\Big)^2\Big] \tag{3.15}$$

where $\bar{Q}_\psi$ is the target inter-Q function. This approach then samples the beta probability from the current policy. The next-option used in the target inter-Q function

come from the current inter-option policy.

3.2.4 Soft Policy Improvement

The policy improvement step involved updating policy in direction that maximizes the rewards, and is done via the soft intra-Q function calculating in the policy evaluation step. After updating the policy towards the exponential of the soft Q-function, it is projected to the space of acceptable policies using the information projection defined in terms of Kullback-Leibler (KL) divergence. So overall the policy improvement step. The intra-option policy represents the probability density of option given state for option ω, optimized as follows:

$$\pi_{\omega,\theta}^{new} = \arg\min_{\pi_{\omega,\theta}\sim\Pi} D_{KL}\Big[\pi(.|s_t)_{\omega,\theta}|\frac{\exp(\frac{1}{\alpha}Q^{\pi_{\omega,\theta}^{old}}(s_t,.))}{Z^{\pi_{\omega,\theta}^{old}}(s_t)}\Big] \tag{3.16}$$

$$\begin{aligned} J_{\pi_{\omega,\theta}}(\theta) &= \mathbb{E}(s_t,\omega_t)\sim D, a_t\sim\pi_{\omega,\theta}\Big[\log_{\pi_{\omega},\theta}(s_t,a_t) - Q_\phi(s_t,\omega_t,a_t)\Big] \\ \widetilde{a}_t &= \tanh(\mu_{\omega,\theta}(s_t) + \sigma_{\omega,\theta}(s_t)\odot\epsilon), \epsilon\sim\mathcal{N}(0,1) \end{aligned} \tag{3.17}$$

I apply the reparameterization trick at the intra-option policy level in which a sample from $\pi_{\omega,\theta}$ is drawn by computing a deterministic function of state, policy parameters, and independent noise. Similar to SAC, I use a squashed Gaussian policy to ensure the actions are bounded to a finite range.

The termination policy objective is defined as follows:

$$\begin{aligned} J_\beta(\omega,v) &= \sum_{\omega_t,s_{t+1}\sim D}\beta_{\omega_t,v}(s_{t+1})A_\psi(s_{t+1},\omega_t) \\ A_\psi(s_{t+1},\omega_t) &= Q_\psi(s_{t+1},\omega_t) - V_\psi(s_{t+1}) \\ &= Q_\psi(s_{t+1},\omega_t) - \max_{\omega_t} Q_\psi(s_{t+1},\omega_t) \end{aligned} \tag{3.18}$$

3.2.5 Algorithm

This approach learns two Q-functions for the the intra-Q function $Q_{\phi_1}(s_t, \omega_t, a_t)$ & $Q_{\phi_2}(s_t, \omega_t, a_t)$, and two Q functions for the inter-Q function $Q_{\psi_1}(s_t, \omega_t)$ & $Q_{\psi_2}(s_t, \omega_t)$. Similar to SAC, this helps mitigate positive bias in the policy improvement step that is known to degrade performance of the value based method.

The Q-functions for intra-option policy are learned with MSBE minimization, by regressing to a single shared target $\bar{Q_\phi}$. The shared target is computed using two target Q-networks, and the target Q-networks are obtained by polyak averaging the Q-network parameters over the course of training. I use this double-Q trick for both intra-Q functions and inter-Q functions.

3.2.6 Modifications for SOC in Discrete Action Space

In the discrete action space, all objectives hold with only changes of the term $\pi_{\omega,\theta}(s_t, a_t)$ to output a probability instead of a density. I represent n_a as the number of actions.

Eq. (6) for the inter-option Q value function will now become:

$$J_Q(\psi) = \mathbb{E}_{s_t,\omega_t \sim D}\Big[\frac{1}{2}\Big(Q_\psi(s_t, \omega_t) - \sum_{i=1}^{n_a} \pi_{\omega,\theta}(s_t, a_i)\big[\bar{Q}_\phi(s_t, \omega_t, a_i) - \alpha_\theta \log \pi_{\omega,\theta}(s_t)\big]\Big)^2\Big], \tag{3.19}$$

where $\bar{Q}_\phi$ is the target intra-Q function.

I have also defined a new objective for the intra-option policy. There is no need for the re-parameterisation trick as the policy now outputs the exact action distribution, where the actions are sampled from the current policy. Eq.(9) changes to:

$$J_{\pi_{\omega,\theta}}(\theta) = \sum_{i=1}^{n_a} \pi_{\omega,\theta}(s_t, a_i)\Big[\log_{\pi_{\omega,\theta}}(s_t, a_i) - Q_\phi(s_t, \omega_t, a_i)\Big] \tag{3.20}$$

where $(s_t, \omega_t) \sim D$ and $a_i \sim \pi_{\omega,\theta}$

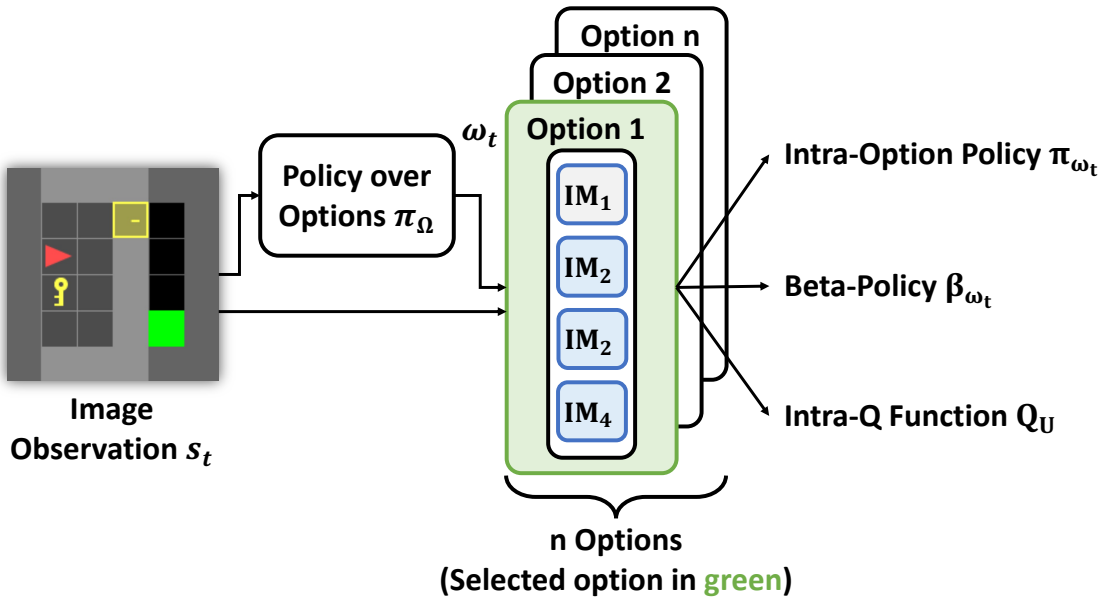


Figure 3-2: Architecture for intra-option policy of State Abstracted Option-Critic (SOC). This presents the overview of the SOC architecture, where policy over options π_Ω selects ω_t, continuing until the termination signal from β_{ω_t}.

3.3 Learning State Abstractions with Independent Mechanisms

This framework SOC considers an agent that operates with temporal abstraction through n options as well as state abstraction through decomposition of each option into a subset of independent mechanisms (IMs). I will now describe the architecture for the latter component, which was previously not a part of the OC framework. Finally, I will define the modifications to the learned option-value functions (i.e., inter-Q and intra-Q), intra-option policy, and the termination policy.

3.3.1 Overview

The algorithm first samples an option ω_t at timestep t from the inter-q function (denoted as Policy over Options in Fig. 3-2) with parameter ψ and internal represen-

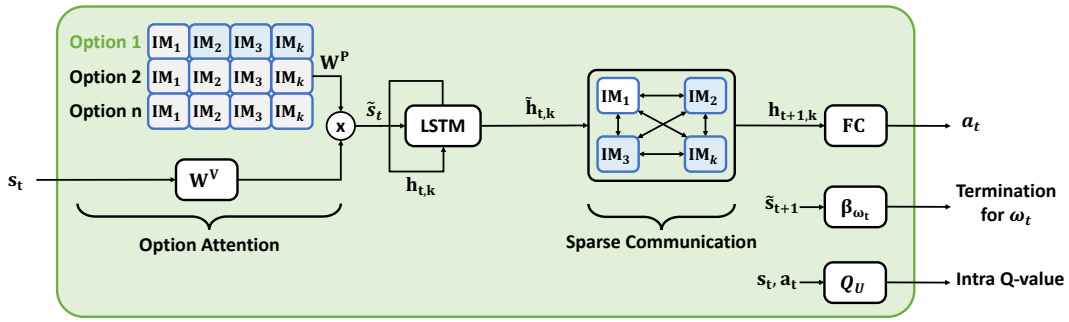


Figure 3-3: This figure presents the overview of the intra-option policy, with the Option Attention, Recurrent, Sparse Communication Layers and Fully Connected Components

tation h_ϕ. This option ω_t can terminate at any time through the beta-policy $\beta_{\omega_t,\vartheta}$ with parameter ϑ. The inter-q function and beta policy is where I primarily integrate temporal abstraction through the definition of n options.

The algorithm then samples an action a_t from the intra-option policy $\pi_{\omega,\theta}$, where it will compute the state abstraction specific to the option ω. I model the intra-option policy system into k recurrent independent mechanisms [14], where each component learns its function from the state space. Specifically, an independent mechanism (IM) at timestep t is defined as a vector-valued state $h_{t,k}$. I chose to model the intra-option policy as a group of independent mechanisms for the ability of IM's to specialize into sub-components allowing options to self organize into groupings of independent components (i.e. context) as described in the state abstraction properties. The intra-option Q function with parameter ϕ and internal representation h_ϕ is used to determine the intra-option policy gradient as described in the problem statement.

3.3.2 Intra-Option Learning

The intra-option policy consists of four components: the option attention, recurrent layer, sparse communication, and fully connected modules. These components allow options to map to a subset of k_A of k independent mechanisms (IMs) that may be

both shared and separate between them.

Option Attention Mechanism. The first layer of the intra-option policy is an option attention mechanism, to select which independent mechanisms are activated for any input option ω_t. Each option is passed through a look-up table W^P of size $n \times k$, which ensures parameters are kept separate between each option as shown in Fig. 3-3. This framework multiplies the observation s_t by the learned parameter W^V to latently learn the factored state space.

A dot product is done with the output of the look-up table selecting parts of the factored state space in interest of the option under consideration, as shown below:

$$\tilde{s} = \text{softmax}(W^P)W^V \tag{3.21}$$

A top-k operation is performed such that only k_A independent mechanisms components are selected from the available k independent mechanisms. This ensures that each option is operating under a reduced state space by operating over a subset of independent mechanisms. Note that I chose to have a single parameter W^P that serves as a look-up table to determine the mapping of option ω_t to a selection of independent mechanisms as opposed to having separate parameters as in [14]. This ensures a fixed context selection over time (i.e. a fixed mapping to a subset of the state space) by an option, because considering many different subsets by the same option would be the same as considering the entire state space, leading to a large search for policy space as previously done by OC.

Coordination between Options. The second component is an RNN that is shared across all options, which takes the input from the previous layer $\tilde{s}$ and the respective hidden states $h_{t,k}$ for all independent mechanisms. Each recurrent operation is done for each active independent mechanism separately. The output of the second layer $\tilde{h}_{t,k}$ is passed into a sparse communication module consisting of gating weights

$(K_{\text{comm}}, Q_{\text{comm}}, V_{\text{comm}})$. The output $h_{t+1,k}$ of this layer is as:

$$\text{softmax}\Big(\frac{\tilde{h}_{t,k}W^{Q_{\text{comm}}}(\tilde{h}_{t,k}W^{K_{\text{comm}}})^T}{\sqrt{d_e}}\Big)\tilde{h}_{t,k}W^{V_{\text{comm}}} + \tilde{h}_{t,k} \tag{3.22}$$

Although the weight and learning for each independent mechanism is separate, all activated independent mechanisms for any option ω_t is able to read from all other independent mechanisms as shown in Fig. 3-3 facilitating coordination in learning amongst the options. This formulation is presented in Eq. (3.22) similar to Goyal et al. [14]. I only update the top k_A selected independent mechanisms from the option selection module, but through this transfer of information, each active independent mechanism can update its internal state and have improved exploration through contextualization. There are many choices in what components to share across options. I explore these choices and their implications in Chapter 5. Finally, I have a separate fully connected layer for each option, providing the final output action. I have delineated this process in Fig. 3-3.

Factored State Representation. The beta policy uses the top-k attention scores from the Option Attention module as the factored state input $\tilde{s}$ into its network. This approach still enable policy over options Ω to reason over the full state space, and have empirically found that providing the factored state as input for the intra-option function Q_ϕ did not help during training.

The difference between previous state abstraction methods such as recurrent independent mechanisms (RIMs) [14] and my method is as follows: this approach consider both state and temporal abstraction, only feeding relevant parts of the state space to each option through the recurrent networks. I have chosen certain components to be shared and not shared between options, which I explore further in Chapter 5.

3.3.3 Implementation

To optimize the option-value functions, intra-option policy, and beta policy with both state abstraction and temporal abstraction as defined in Eq. (3.3), I choose to apply

Algorithm 1 State Abstraction Option-Critic with Entropy Maximization

Initialize inter-Q network parameters ψ_1, ψ_2
Initialize intra-Q network parameters ϕ_1, ϕ_2
Initialize intra-policy parameter θ
Initialize beta-policy parameter ϑ
Initialize constant α_θ
$D = []$ ▷ Replay buffer initialization
Initialize internal states h_{ψ_1}, h_{ψ_2}, h_{ϕ_1}, h_{ϕ_2}, $h_{t,k}$, h_ϑ
for each iteration **do** $s \leftarrow s_0$
  Choose ω_t sampled from π_Ω
  for each environment step **do**
    $(a_t, h_{t+1,k}) \sim \pi_{\omega_t,\theta}(s_t, h_\theta)$
    Take action a_t and observe state s_{t+1} and r_t
    Get $\tilde{s}_{t+1} \sim \pi_{\omega_t,\theta}(a_t, s_{t+1}, h_{t+1,k})$
    $D \leftarrow D \cup \{s_t, \omega_t, a_t, r_t, s_{t+1}, \tilde{s}_{t+1}, h_{\psi_1}, h_{\psi_2}, h_{\phi_1}, h_{\phi_2}, h_{t+1,k}, h_\vartheta\}$
    if $\beta_{\omega_t,\vartheta} = 1$ **then** ▷ terminates in $\tilde{s}_{t+1}$
      Choose ω_{t+1} according to π_Ω
    for each gradient step **do**
      $\psi_i \leftarrow \psi_i - \nabla_{\psi_i} \lambda_{\psi_i} J(\psi_i)$ for $i = 1, 2$
      $\bar{\psi}_i \leftarrow \tau\psi_i(1-\tau)\bar{\psi}_i$ for $i = 1, 2$
      $\theta \leftarrow \theta - \nabla_\theta \lambda_\theta J_{\omega,\theta}(\theta)$
      $\phi_i \leftarrow \phi_i - \nabla_{\phi_i} \lambda_{\phi_i} J(\phi_i)$ for $i = 1, 2$
      $\bar{\phi}_i \leftarrow \tau\phi_i(1-\tau)\bar{\phi}_i$ for $i = 1, 2$
      $\vartheta_i \leftarrow \vartheta_i - \nabla_{\vartheta_i} \lambda_{\vartheta_i} J(\vartheta_i)$ $=0$

entropy maximization [17]. I encode the intra-option and inter-option Q functions as LSTM networks, with beta-policy feeding in the factored state as input. The intra-option policy architecture is described in Fig. 3-3. Algorithm 1 describes the implementation of my approach.

Chapter 4

Related Works

4.1 Hierarchical RL

There have been two major approaches to HRL; the options framework to learn higher level skills [2, 27, 31, 32] and goal-conditioned hierarchies [20, 22, 23, 26] to learn higher level sub-goals. Goal-conditioned HRL models require a predefined representation of the environment and mapping of observation space to goal space, where as the OC facilitates the long timescale credit assignment by dividing the problem into pieces and learning higher level skills.

While OC provides temporal decomposition of the problem, other approaches such as Feudal Networks [35] decompose problems with respect to the state space. Feudal approaches use a manager to learn more abstract goals over a longer temporal scale and worker modules to perform more primitive actions with fine temporal resolution. The approach in this thesis (SOC) employs a combination of both visions, necessitating both temporal and state abstraction for effective decomposition. Although some approaches such as the MAXQ framework employ both, MAXQ [10] they involve learning recursively optimal policies that can be highly sub-optimal [11].

4.2 Decomposition with State Abstraction

My approach is related to prior work that considers decomposition of the learning problem. Mixture of Experts (MoE) [18] learn a context specific mixture of modules end to end, considering a number of experts, each a simple feed-forward neural network, and a trainable gating network which selects a sparse combination of experts to process each input. Although MoE on its own provides a solution to decomposition in parameter space, it does not address the issue of decomposition in time which is essential in RL frameworks. Another notable methods that perform state abstraction for the purposes of decomposition include Chitnis et al. [8], Konidaris et al. [21] for planning and Abel et al. [1] for life-long learning.

Chapter 5

Evaluation

This section demonstrates SOC's efficacy on a diverse suite of domains. The code is available at `https://github.com/abdulhaim/soft_option_critic`

5.1 Domain Settings

I demonstrate the performance of my approach compared to both HRL and non-HRL baselines with domains. These domains require various levels of both temporal and state abstraction, including state abstraction in both image and vector space, as well as temporal abstraction in domains in sparse reward settings. Lastly, I test on domains that are hypothesized to not require any temporal or state abstraction.

- **MiniGrid** [7]: A library of open-source grid-world domains in sparse reward settings where world is a grid of size $d \times d$, with each tile in the grid containing exactly zero or one object, with possible object types such as the wall, door, key, ball, box and goal indicated by different colors. The goal for the domain can vary between obtaining a key to matching similar colored objects. The agent receives a sparse reward of 1 when it successfully reaches a green goal tile, and zero for failure.

  The observation is a partially observable view of the grid-world environment

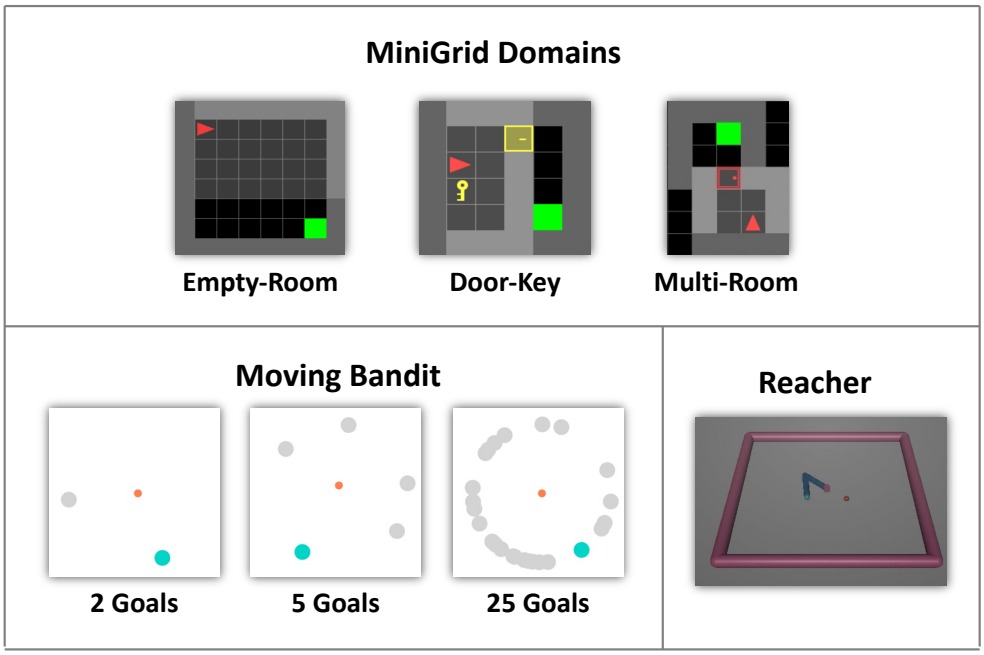


Figure 5-1: Visualization of MiniGrid domains [7], multiple goals scenarios of Moving Bandit [12], and Reacher [6]

using a compact and efficient encoding, with 3 input values per visible grid cell and including 7x7x3 values in total. For the Empty Room domain, an agent is randomly initialized at the start of each episode and must learn to navigate to the green goal location. For the MultiRoom domain, a randomly initialized agent must learn to go through the green door to the green agent. Lastly, I test on the Key Domain in MiniGrid as described in the motivation. For the latter two domains, at the start of every episode, the structure of the grid also changes. The implementation for this method can be found here: `https://github.com/maximecb/gym-minigrid`

- **Moving Bandit** [12]: This 2D sparse reward setting considers m marked positions in the environment that change at random at every episode, with 1 of the m positions being the correct goal. An agent receives a reward of 1 and terminates when it close to the correct goal position and 0 otherwise.

  I use the implementation of Moving Bandits found here, modified to terminate when the sparse reward of 1 is received as opposed to continuing till the max

Table 5.1: $\bar{V}$ and Area under the Curve (AUC) for algorithms solving MiniGrid Domains. Table shows mean and standard deviation computed with 10 random seeds. Best results in bold (computed by t-test with $p < 0.05$). Note that SOC has the highest AUC and $\bar{V}$ compared to a non-HRL method, a HRL method, and a state abstraction baseline.

Algorithm	Abstraction	
	Temporal	State
A2C [25]	✗	✗
OC [2]	✓	✗
A2C-RIM [14]	✗	✓
SOC (Ours)	✓	✓

timesteps. `https://github.com/maximilianigl/rl-msol`

- **Reacher** [6]: In this simulated MuJoCo task of OpenAI Gym environment, a robot arm consisting of 2 linkages with equal length must reach a random red target placed randomly at the beginning of each episode. The agent receives a reward signal of 1 when its euclidean distance to the target is within the denoted threshold, and 0 otherwise. To consider sparse reward settings, I have made a minor change to the reward signal in this domain. A tolerance is specified to create a terminal state when the end effector approximately reaches the target. The tolerance creates a circular area centered at the target, which is specified as 0.003.

5.2 Baselines

I compare to several baselines in order to provide context for the performance of SOC. The hyperparameters selected are noted below.

- **A2C** [25]: This on-policy method consists of no state or temporal abstraction. I use the implementation of A2C found here: `https://github.com/dido1998/Recurrent-Independent-Mechanisms`. For fair comparison, I implement both the actor and the critic as an LSTM, and set the number of workers to be 1.

Table 5.2: MiniGrid DoorKey & Multi-Room Hyperparameters

Hyperparameter	Value
Batch Size	100
Learning Rate	0.0005
Alpha	0.001
Options	3
IM	4
Top-K	3
Hidden Size	6
Value Size	32
Discount Factor	0.95

Table 5.3: MiniGrid EmptyRoom Hyper-parameters

Hyperparameter	Value
Batch Size	100
Learning Rate	0.0003
Alpha	0.005
Options	3
IM	4
Top-K	3
Hidden Size	6
Value Size	32
Discount Factor	0.95

- **OC** [2]: I consider an off-policy implementation of option-critic to demonstrate the performance of a hierarchical method considering only temporal abstraction. For fair comparison to SOC, I apply entropy maximization to the OC framework. I also implement the inter-q, intra-q, intra-pi, and beta-policy as an LSTM to maintain fair comparison with the SOC framework.

- **A2C-RIM** [14]. This approach considers A2C with recurrent independent mechanisms, a baseline that employs only state abstraction. I use the implementation of A2C-RIMs found here:
`https://github.com/dido1998/Recurrent-Independent-Mechanisms`

5.3 Hyperparameter Values

I report the hyperparameter values used for each of the methods in the experiments. I used consistent hyper-parameters between SOC, OC, A2C, and A2C-RIMs when they applied.

The parameters for the MiniGrid Domain and the Moving Bandit & Reacher Domains are provided in Tables 5.2–5.5.

Table 5.4: Moving Bandit Domain

Hyperparameter	Value
Batch Size	100
Learning Rate	0.005
Alpha	0.005
Options	3
IM	4
Top-K	3
Hidden Size	6
Value Size	16
Discount Factor	0.95

Table 5.5: Reacher-v2

Hyperparameter	Value
Batch Size	100
Learning Rate	0.001
Alpha	0.001
Options	3
IM	6
Top-K	4
Hidden Size	6
Value Size	64
Discount Factor	0.95

Table 5.6: $\bar{V}$ and Area under the Curve (AUC) for algorithms solving MiniGrid Domains. The table shows mean and standard deviation computed with 10 random seeds. Best results in bold (computed via a t-test with $p < 0.05$). Note that SOC has the highest AUC and $\bar{V}$ compared to the non-HRL method A2C [25], the HRL method OC [2], and the state abstraction baseline A2C-RIM [14].

Algorithm	MiniGrid Empty		MiniGrid MultiRoom		MiniGrid KeyDoor	
	$\bar{V}$	AUC	$\bar{V}$	AUC	$\bar{V}$	AUC
A2C	0.71 ± 0.25	364 ± 54	$\mathbf{0.62 \pm 0.26}$	104 ± 27	0.04 ± 0.07	16 ± 10
OC	0.56 ± 0.36	312 ± 143	0.40 ± 0.15	108 ± 45	0.55 ± 0.44	252 ± 217
A2C-RIM	0.57 ± 0.40	283 ± 47	0.19 ± 0.23	52 ± 22	0.57 ± 0.40	12 ± 1
SOC	$\mathbf{0.92 \pm 0.12}$	$\mathbf{470 \pm 21}$	$\mathbf{0.75 \pm 0.06}$	$\mathbf{200 \pm 27}$	$\mathbf{0.87 \pm 0.17}$	$\mathbf{499 \pm 25}$

5.4 Results

Question 1. *When is state abstraction needed?*

To address this question, I compare performance between SOC, and HRL (OC), non-HRL (A2C), and state abstraction (A2C-RIMs) methods in the MiniGrid Domain. Table 5.6 shows show both final task-level performance ($\bar{V}$) (i.e., final episodic average reward measured at the end of the session) and area under the task-level learning curve (AUC). Higher values indicate better results for both metrics.

Overall, for all three scenarios of MiniGrid, I found SOC to have the highest AUC and $\bar{V}$ than the baseline methods. With a sparse reward function and dense state representation (image observation), MiniGrid provides the opportunity to test the

temporal and state abstraction capabilities of my method. Specifically, I observe that OC has a lower performance due to the inability of the options learned to diversify by considering the entire state space and the high termination probability of each option. A2C results in sub-optimal performance due to its failure in sparse reward settings. Finally, due to inefficient exploration and large training time required for A2C-RIM to converge, it is unable to match to the performance of SOC. I see a smaller difference between SOC and these baselines for the Empty domain, as there is a smaller amount of state abstraction that is required. For the Multi-Room domain which is on the second order of difficulty, there is a increasingly larger gap as the agent needs to consider only the states in one room when trying to exit that room and states of the other room when trying to reach the green goal. Lastly, I see the most abstraction required for the Key Door domain where the baselines are unable to match the performance of SOC. As described in Fig. 1-1, OC equipped with only temporal abstraction is unable to match the performance of SOC consisting of both temporal and state abstraction. I show the evaluation reward plots for the Empty, Multi-Room, and DoorKey Domains comparing SOC and the baselines.

Question 2. *What does it mean to have diverse options?* I visualize the behaviour of both options for temporal abstraction and independent mechanisms for state abstraction to further understand whether options are able to learn diverse sub-policies. I test this hypothesis in the DoorKey Domain, where Fig. 5-5a shows the option trajectory for the following sub-tasks: getting the key, opening the door, and going to the door. I find that each options is only activated for one sub-task. Fig. 5-5b shows the mapping between options and independent mechanisms, with each option mapping to a unique subset of independent mechanisms. To understand whether these independent mechanisms have mapped to different factors of the state (and hence have diverse parameters), Fig. 5-5c checks the correlation between options. I find low correlation between option 1 & option 2 and option 2 & option 3 but higher correlation between option 1 & option 3. I observe high correlation between option 1 & option 3 in getting the key and opening the door due to the shared states between them. By employing both temporal and state abstraction, SOC is able to learn diverse and

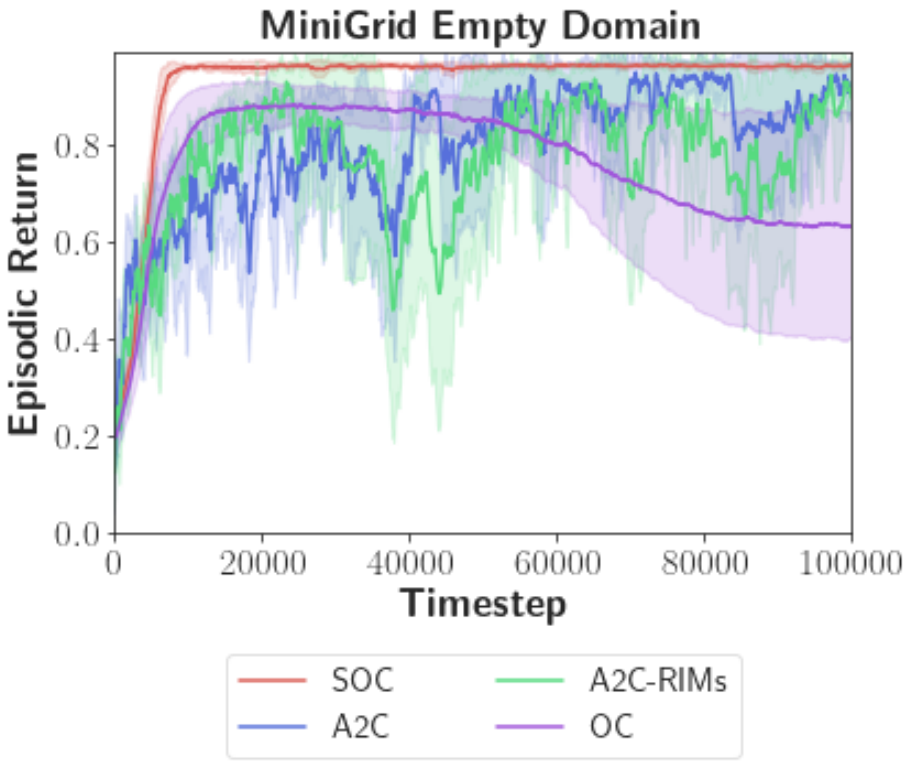


Figure 5-2: Performance of SOC vs. Baselines for MiniGrid-Empty-Random-6x6-v0

complementary option policies.

Question 3. *How does evaluation performance change as the need for state abstraction increases in a domain?*

This experiments begins with the simplest case of Moving Bandits, where the agent must learn the true goal amongst two goal locations as shown in Fig. 5-1. SOC is seen to converge at a faster rate than OC in Fig. 5-6. However in order to understand the full benefits of state abstraction, I observe the performance with increasing state abstraction in the Moving Bandits Domain. This experiments consider the performance determined by Area under the Curve (AUC) for an increasing number of spurious features in the observation. Namely, I add 3 & 23 additional goals to the 2 goal observation to test the capabilities of SOC when requiring increasing amounts of state abstraction. In this scenario, the agent must ignore the spurious goal locations

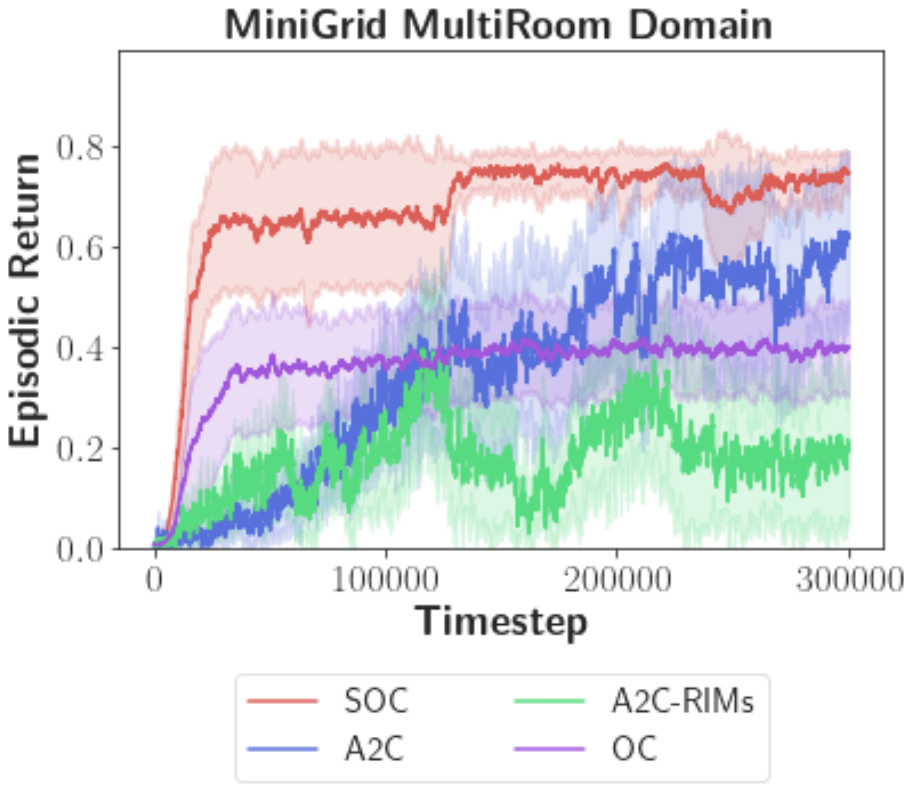


Figure 5-3: Performance of SOC vs. Baselines for MiniGrid-MultiRoom-N2-S4-v0

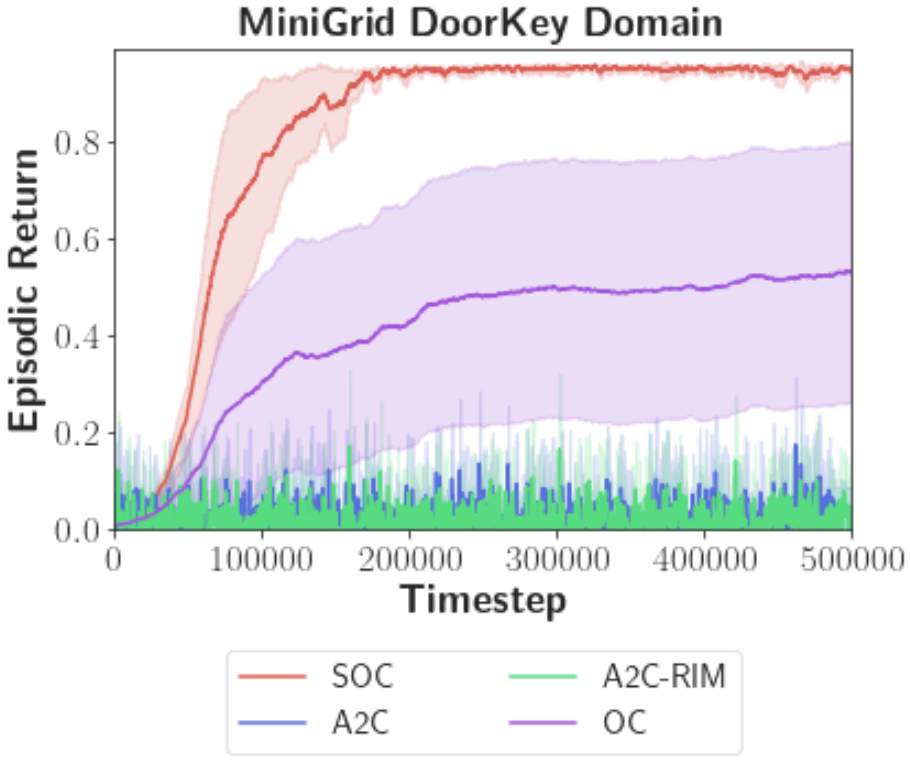
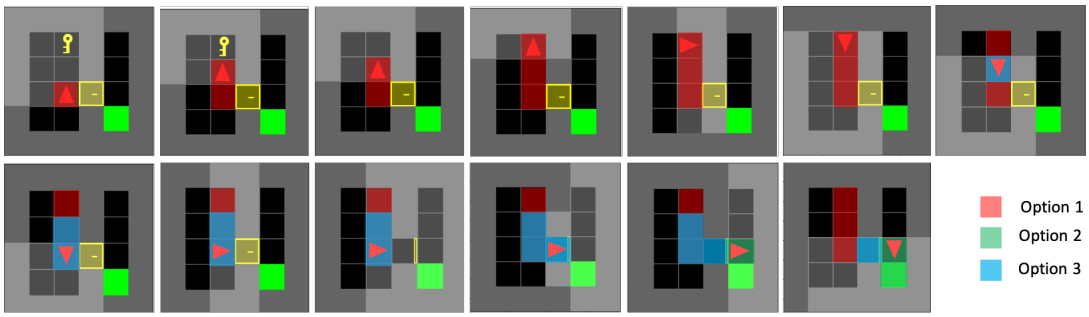
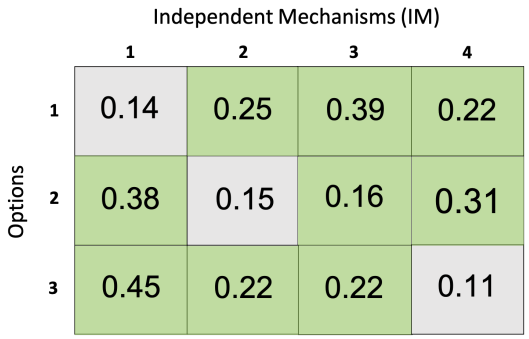


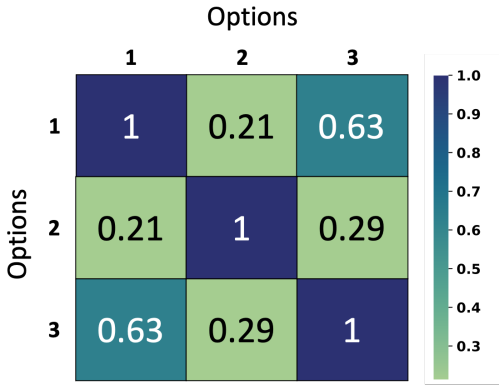
Figure 5-4: Performance of SOC vs. Baselines for MiniGrid-DoorKey-6x6-v0



(a) Option Trajectory



(b) Independent Mechanism Selection by Options



(c) Option Correlation

Figure 5-5: (a) Temporal abstraction is demonstrated by the use of 3 options for following tasks: option 1 for getting the key, option 2 for opening the door, and option 3 for going to the door. (b) Each option maps to a unique subset of independent mechanisms, corresponding to their unique functions in the domain. (c) There is low correlation between option 1 & option 2 and option 2 & option 3 but higher correlation between option 1 & option 2 corresponding to the shared states between them.

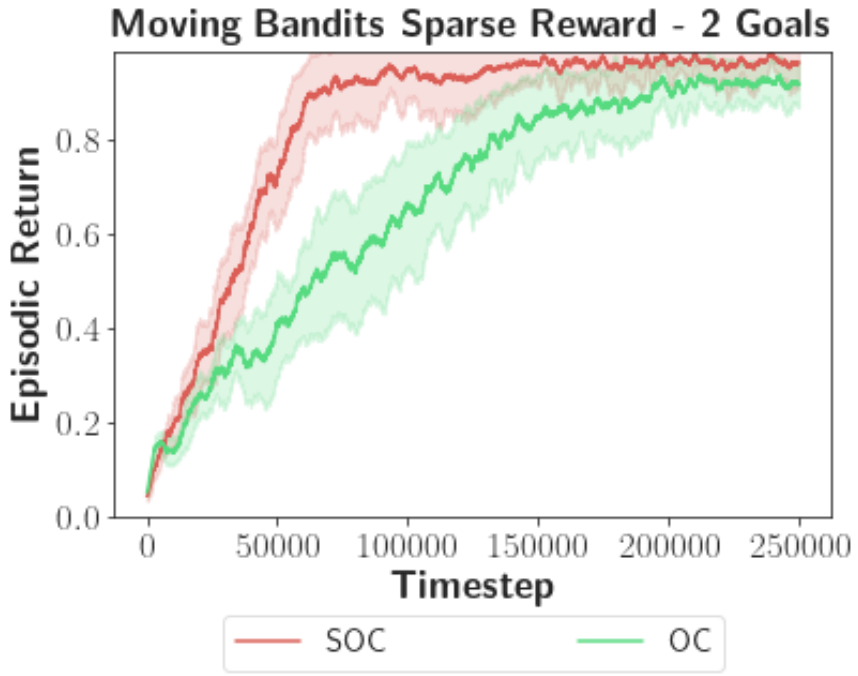


Figure 5-6: In Moving Bandits with 2 goal locations, there is only one true goal where the agent receives reward of 1. In this scenario, SOC converges faster than OC.

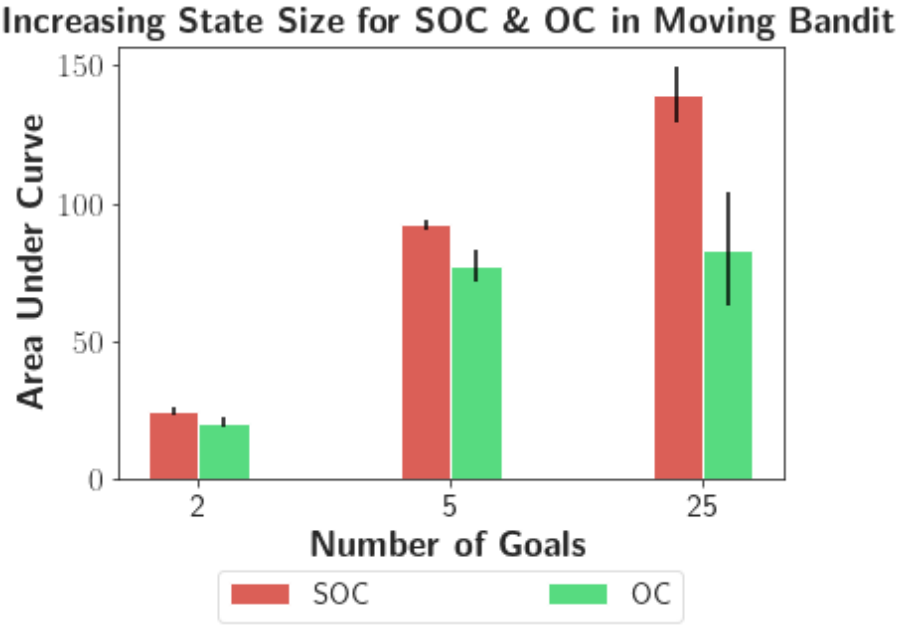


Figure 5-7: Area Under Curve (AUC) of SOC and OC in Moving Bandit domain in sparse reward settings. As the number of spurious features on the x-axis increases, the gap between the AUC performance between SOC and OC increases. This indicates a greater need for state abstraction. Note that there is see a different AUC across number of goals simply due to different max train iteration for each goal setting, with the focus on the increasing difference of AUC between the methods.

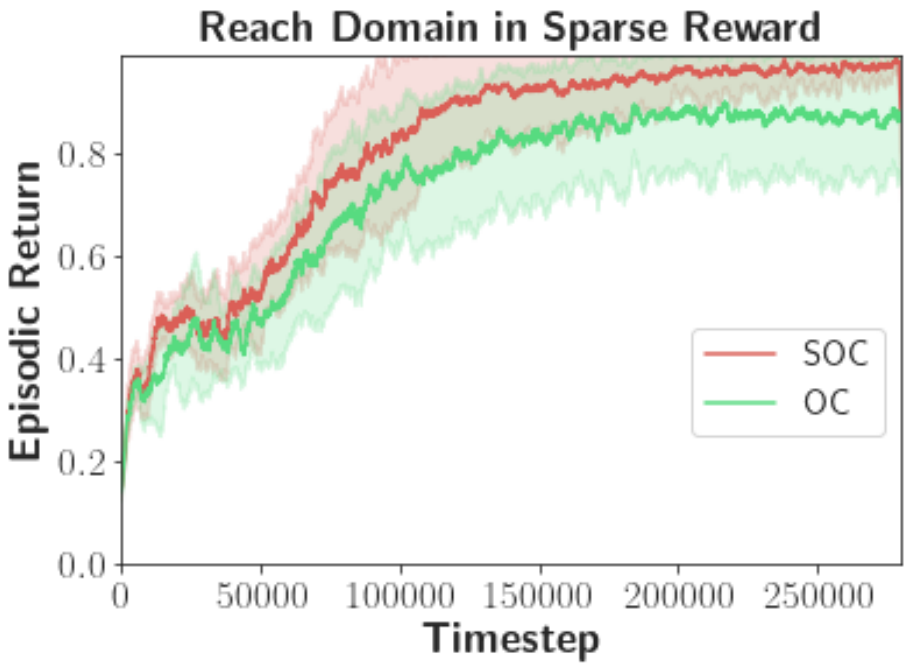


Figure 5-8: In domains with small or negligible state abstraction, the benefit of SOC is not as significant as shown in the Reacher domain, where the low-dimensional state representation does not require state abstraction.

and only learn to move to the correct goal location as indicated in its observation. As shown in Fig. 5-7, SOC performs significantly better than OC as the number of spurious goal locations it must ignore increases. I expect this behaviour due to the SOC's capability of both temporal and state abstraction.

Question 4. *When is state abstraction not required?*

The Reach Domain allows us to observe the effects of SOC when no state abstraction is required. This domain does not require state abstraction as the entire state space consists of relevant information to achieve the goal at hand. Specifically, the low-level representation of the state space as a 4-element state vector, with the first 2 elements containing the generalized positions and next 2 elements containing the generalized velocities of the two arm joints, are essential to reach the goal location. As expected in Fig. 5-8, the performance between SOC and OC is similar in this domain as the observation space does not contain any features that are useful for SOC to perform state abstraction.

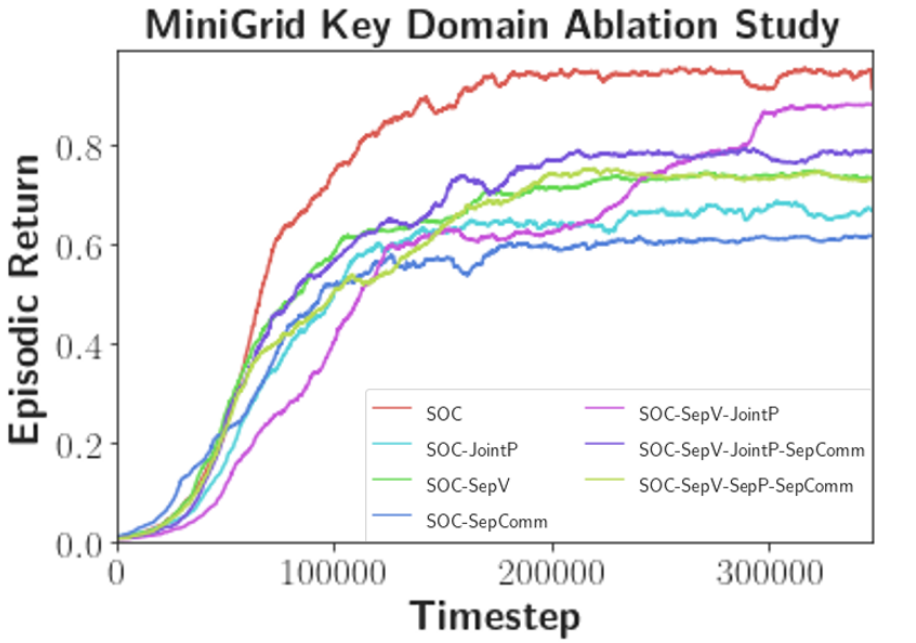


Figure 5-9: I perform an ablation study experimenting with various components that can be shared and not shared between option in the intra-option learning framework shown in Fig. 3-3. I find that too much sharing or too little sharing between options can lead to sub-optimal performance due to lack of coordination.

Question 5. *What is shared between option policies in the context of independent mechanisms structures?*

As described in Fig. 3-3, there are 4 components that make up the intra-option learning model. In order to investigate which are necessary to share between options, I perform an ablation study with a subset of the available choices shown in Fig. 5-9. Specifically, I study sharing of the look-up table W^P in the input attention mechanism between options (SOC JointP), learning a separate factored state parameter W^V between options (SOC SepV), learning separate parameters for ($W^{Q_{\text{comm}}}, W^{K_{\text{comm}}}$, $W^{V_{\text{comm}}}$) of the sparse communication layer for each option (SOC SepComm), and three other combinations of these three modules. I find the lowest performance for the method with a separate sparse communication module for each option. I hypothesize this is due to lack of coordination between each option in updating their active IM's and ensuring other non-active IM's are learning separate and complementary

parameters. Having a joint W^P parameter results in the second lowest performance. This effectively maps each option to the same set of IM's, leading to lack of diversity between option policies and only allowing for difference in the fully-connected layer of each option. Lastly, I observe the third lowest performance with a separate parameter for the factored state parameter W^V between options. Each option learning from a different representation can lead to similar option-policies and states for certain sub-goals unaccounted for.

Chapter 6

Conclusion

In this thesis, I have introduced SOC framework (SOC) for incorporating both temporal and state abstraction for effective decomposition of problem into simpler components. The key idea underlying the proposed formulation is to map each option to a reduced state space, effectively considering each option as a subset of independent mechanisms. I evaluated my method on several benchmarks with varying levels of required abstraction in sparse reward settings. The results indicate that SOC is able to decompose the problems at hand more effectively than standard state of the art HRL, non-HRL, and state-abstraction methods. Future work for this project would include experimenting on larger domains with more complicated requirements for state and temporal abstraction. I would also like to experiment with how my method would cope in transfer learning settings. I believe that this work can help provide the community with a theoretical foundation to build off for addressing the deficiencies in HRL methods.

Bibliography

[1] David Abel, Dilip Arumugam, Lucas Lehnert, and Michael Littman. State abstractions for lifelong reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 10–19. PMLR, 10–15 Jul 2018. URL `http://proceedings.mlr.press/v80/abel18a.html`.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. *CoRR*, abs/1609.05140, 2016. URL `http://arxiv.org/abs/1609.05140`.

[3] Jiri Baum, Ann E Nicholson, and Trevor I Dix. Proximity-based non-uniform abstractions for approximate planning. *Journal of Artificial Intelligence Research*, 43:477–522, 2012.

[4] Craig Boutilier. Correlated action effects in decision theoretic regression. In *UAI*, pages 30–37, 1997.

[5] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. In *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, UAI'96, page 115–123, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc. ISBN 155860412X.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL `http://arxiv.org/abs/1606.01540`.

[7] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. *GitHub repository*, 2018.

[8] Rohan Chitnis, Tom Silver, Beomjoon Kim, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Camps: Learning context-specific abstractions for efficient planning in factored mdps. *arXiv preprint arXiv:2007.13202*, 2020.

[9] Christian Daniel, Herke van Hoof, Jan Peters, and Gerhard Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104, 09 2016. doi: 10.1007/s10994-016-5580-x.

[10] Thomas Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *The Journal of Artificial Intelligence Research (JAIR)*, 13, 12 2000. doi: 10.1613/jair.639.

[11] Yannis Flet-Berliac. The promise of hierarchical reinforcement learning. *The Gradient*, 2019.

[12] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *CoRR*, abs/1710.09767, 2017. URL `http://arxiv.org/abs/1710.09767`.

[13] Natalia Gardiol and Leslie Kaelbling. Envelope-based planning in relational mdps. *Advances in Neural Information Processing Systems*, 16:783–790, 2003.

[14] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

[15] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19: 399–468, Oct 2003. ISSN 1076-9757. doi: 10.1613/jair.1000. URL `http://dx.doi.org/10.1613/jair.1000`.

[16] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[18] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[19] Anand Kamat and Doina Precup. Diversity-enriched option-critic. *arXiv preprint arXiv:2011.02565*, 2020.

[20] Dong-Ki Kim, Miao Liu, Shayegan Omidshafiei, Sebastian Lopez-Cot, Matthew Riemer, Golnaz Habibi, Gerald Tesauro, Sami Mourad, Murray Campbell, and Jonathan P. How. Learning hierarchical teaching policies for cooperative agents, 2020.

[21] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *J. Artif. Int. Res.*, page 215–289, January 2018.

[22] T. D. Kulkarni and K. R. Narasimhan. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Neural Information Processing Systems 2016*, 2016.

[23] Andrew Levy, Robert W. Platt, and Kate Saenko. Hierarchical actor-critic. *ArXiv*, abs/1712.00948, 2017.

[24] Daniel J. Mankowitz, Timothy A. Mann, and Shie Mannor. Adaptive skills adaptive partitions (asap). In *NIPS*, 2016.

[25] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016.

[26] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning, 2019.

[27] Shayegan Omidshafiei, Dong-Ki Kim, Jason Pazis, and Jonathan P. How. Crossmodal attentive skill learner. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 139–146, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[28] Doina Precup and Richard S. Sutton. *Temporal Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst, 2000. AAI9978540.

[29] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994. ISBN 0471619779.

[30] ML Puterman. Markov decision processes. 1994. *Jhon Wiley & Sons, New Jersey*, 1994.

[31] Matthew Riemer, Miao Liu, and Gerald Tesauro. Learning abstract options. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/cdf28f8b7d14ab02d12a2329d71e4079-Paper.pdf`.

[32] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(99)00052-1. URL `https://www.sciencedirect.com/science/article/pii/S0004370299000521`.

[33] Harm van Seijen, Shimon Whiteson, and Leon Kester. Efficient abstraction selection in reinforcement learning. *Computational Intelligence*, 30(4):657–699, 2014. doi: https://doi.org/10.1111/coin.12016. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/coin.12016`.

[34] Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, and koray kavukcuoglu. Strategic attentive writer for

learning macro-actions. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/c4492cbe90fbdbf88a5aec486aa81ed5-Paper.pdf.

[35] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *CoRR*, abs/1703.01161, 2017. URL http://arxiv.org/abs/1703.01161.

[36] Zheng Wen, Doina Precup, Morteza Ibrahimi, Andre Barreto, Benjamin Van Roy, and Satinder Singh. On efficiency in hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/file/4a5cfa9281924139db466a8a19291aff-Paper.pdf.