Subgoal Discovery Using a Free Energy Paradigm and State Aggregations

Amirhossein Mesbah[1], **Reshad Hosseini**[1], **Seyed Pooya Shariatpanahi**[1], **Majid Nili Ahmadabadi**[1]

[1]School of ECE, College of Engineering, University of Tehran
amir.mesbah@ut.ac.ir, reshad.hosseini@ut.ac.ir, p.shariatpanahi@ut.ac.ir, mnili@ut.ac.ir

Abstract

Reinforcement learning (RL) plays a major role in solving complex sequential decision-making tasks. Hierarchical and goal-conditioned RL are promising methods for dealing with two major problems in RL, namely sample inefficiency and difficulties in reward shaping. These methods tackle the mentioned problems by decomposing a task into simpler subtasks and temporally abstracting a task in the action space. One of the key components for task decomposition of these methods is subgoal discovery. We can use the subgoal states to define hierarchies of actions and also use them in decomposing complex tasks. Under the assumption that subgoal states are more unpredictable, we propose a free energy paradigm to discover them. This is achieved by using free energy to select between two spaces, the main space and an aggregation space. The $model\ changes$ from neighboring states to a given state shows the unpredictability of a given state, and therefore it is used in this paper for subgoal discovery. Our empirical results on navigation tasks like grid-world environments show that our proposed method can be applied for subgoal discovery without prior knowledge of the task. Our proposed method is also robust to the stochasticity of environments.

Introduction

Reinforcement learning (RL) (Sutton and Barto 2018) is widely used in different aspects of our daily life from chatbots (Christiano et al. 2017) to autonomous driving (Kiran et al. 2021) and chip design (Mirhoseini et al. 2021). Classical RL algorithms generally suffer from being time-consuming, sample inefficient, and having difficulties in defining an appropriate reward function. Furthermore, the classical RL algorithms have difficulties in environments with long horizons, delayed rewards, and sparse rewards. Such environments are common in navigation, robotic manipulation, and many other tasks that we are dealing with daily.

Studies like Hierarchical Reinforcement Learning (HRL) (Hutsebaut-Buysse, Mets, and Latré 2022), Goal-Conditioned Reinforcement Learning (GCRL) (Liu, Zhu, and Zhang 2022), and using sub-spaces (Ghorbani et al. 2025) are some of the promising efforts that try to solve the aforementioned problems of classical RL algorithms. These works are trying to use a level of abstraction in the action space and the state space or they try to decompose tasks into simpler tasks and use the agent's experience to generalize solving long-horizon tasks.

We can see different examples of action abstraction in our daily lives. For example, instead of thinking about the performance of thousands of pieces of a car, we abstract the sequence of actions taking place into a high-level action like "speed up". Even more complex tasks like "turning right" can be decomposed into the sequence of lower-level actions "slowing down the speed", "changing the lane" and "turning the steer to the right". This process of abstraction can continue recursively. Also, we can decompose tasks like navigation and reaching a special goal state into reaching some defined subgoal states in a row to make the task easier.

Hierarchical learning which is inspired by learning earning in the human brain (Theves et al. 2021), uses abstraction in the action space and takes the power of this property to solve complex tasks faster. Combining hierarchical learning and reinforcement learning has been exploited in three major approaches named option framework (Sutton, Precup, and Singh 1998), feudal reinforcement learning (Dayan and Hinton 1992), and hierarchical abstract machines (Parr and Russell 1997). However, using hierarchical methods is hard in practice and it needs a good knowledge of the task for designing temporally extended actions. In most developed methods, especially the options framework, besides learning a hierarchy of policies the most challenging problem is option discovery, to find subgoal or bottleneck states to create options.

A similar approach to HRL is GCRL. In this setting an agent can have a high-level controller for finding appropriate subgoals in the environment. After detecting subgoals, a low-level policy can learn the sequence of subgoals and simultaneously decompose a complex task into simpler ones (Nachum et al. 2018; Gupta et al. 2020; Chane-Sane, Schmid, and Laptev 2021). Therefore, having a method to detect subgoal states in the environment can help with option discovery in HRL and for learning implicit or explicit subgoal-based policies in goal-conditioned settings.

As mentioned above, the other level of abstraction in an environment is state abstraction which leads to approaches like using sub-spaces or state aggregation and grouping (Daee, Mirian, and Ahmadabadi 2014; Hashemzadeh, Hosseini, and Ahmadabadi 2018, 2020; Li et al. 2023). Using sub-spaces and state aggregations can provide us with

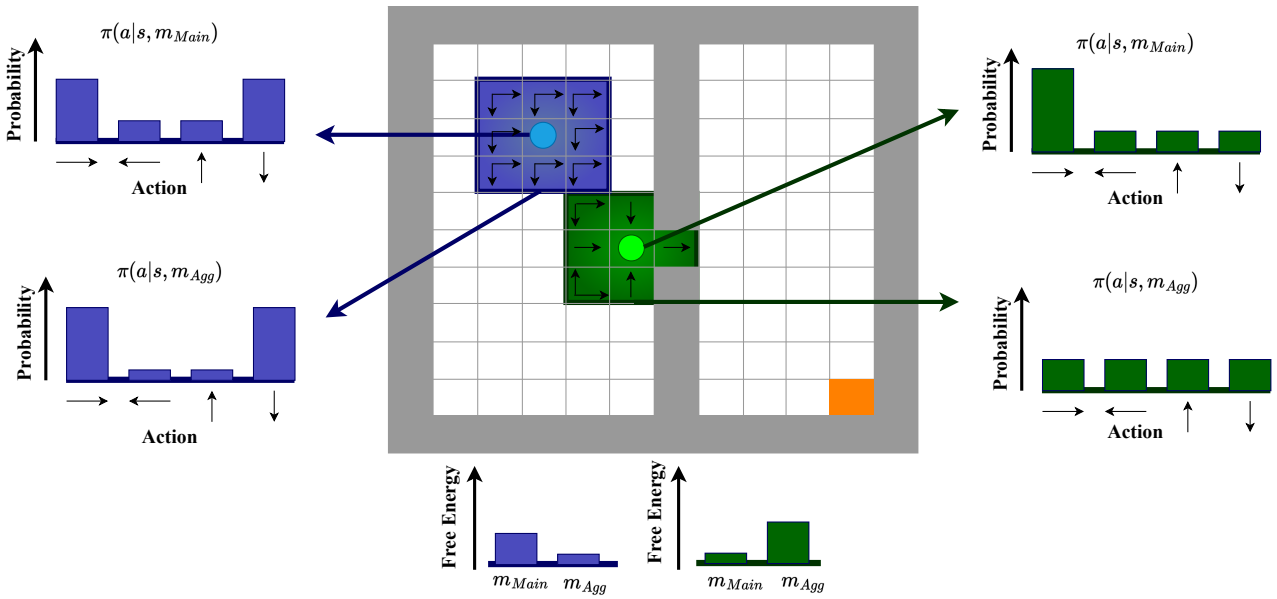

Figure 1: This figure illustrates a two-room environment with a doorway connecting them and a goal in the bottom right (orange). An agent starting from the top left can move in four directions. Each state in the main space corresponds to a 3x3 block of states in the aggregation space. The agent chooses between main space ($\pi(a|s, m_{Main})$) and aggregation space ($\pi(a|s, m_{Agg})$) policies based on their Free Energy (uncertainty measure) - selecting the space with a lower free energy. Near the bottleneck (doorway) states, the main space is preferred due to its lower free energy, while the aggregation space is chosen in states distant from doorways.

more samples from the environments and lead to a speed-up in exploration and learning in the initial episodes. We can lower the effect of sample inefficiency using sub-spaces which are many-to-one mapping from the main environment. In these kinds of tasks, selecting which states to aggregate and which sub-spaces to choose for learning in each step of episodes are the challenging questions. Furthermore, aggregating different states in state space can encounter the problem of perceptual aliasing (PA). This problem happens when the aggregated states have totally different policies and this causes the samples in aggregation spaces to become useless.

In this paper, we are interested in dealing with the question "How can we identify bottleneck states faster using state aggregations during an artificial RL agent's lifetime?". Using the state aggregations to detect bottlenecks or subgoals faster is the main motivation of our work. As illustrated in Figure 1, the aggregation of states in a bottleneck state causes an increase in the uncertainty of policy in the aggregation space and makes its policy close to a random policy. This is because of the PA problem and the difference in the policy of aggregated states. However, in states far from the bottleneck, we can use aggregation to improve the samples in the direction of the optimal policy. In bottleneck states, our considered method of aggregation does not help improve learning, and this transformation is not smooth in bottleneck states.

We present an algorithm to identify subgoal states by capturing this uncertainty using an information-theoretic concept called free energy. Our proposed algorithm can identify the bottleneck states faster than other methods which are mostly based on creating a graph of the environment. Also, there is no need to define the number of subgoals compared to methods with differentiable termination functions for the termination condition of options.

In summary, this paper provides the following contributions: (a) We propose a new bottleneck detection algorithm that identifies bottleneck states after living some episodes in the environment. (b) We demonstrate that our proposed method is empirically robust to the stochasticity of the environment and our method can identify subgoal states in the environment with up to 50% of stochasticity. (c) Our idea of model changes applies to both discrete and continuous state spaces considering some modifications. In the following sections, we review the literature and state our assumptions and preliminary findings. We then present our method and experimental results. Lastly, we discuss future research directions and conclusions.

Related Work

The studies on bottleneck or subgoal discovery can be categorized into two groups, GCRL and option discovery in HRL. In GCRL methods, a high-level controller tries to identify subgoals. The detected subgoals are used as the target of low-level policies (Chane-Sane, Schmid, and Laptev 2021; Nachum et al. 2018; Nair and Finn 2020) for learning decomposed tasks. These methods are implemented on the goal-conditioned Markov Decision Processes (MDPs) where we have the information about the goal state and our

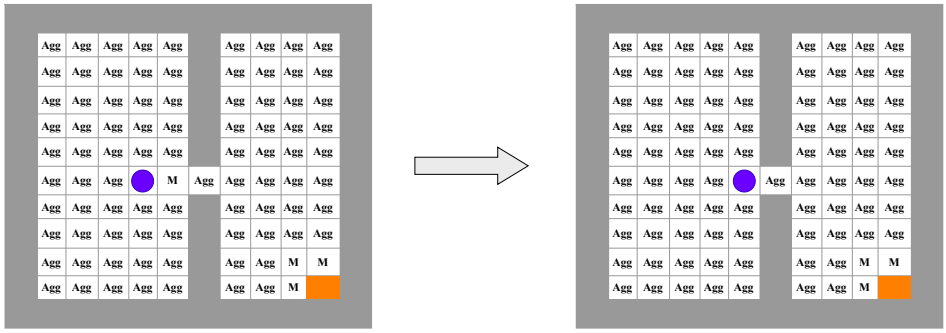

Figure 2: This figure demonstrates the agent's state space selection in a two-room environment. The blue circle represents the agent's current position, moving toward the goal (orange). States are assigned either $Main(M)$ or $Aggregation(Agg)$ space based on the minimum free energy. Near and at the doorway, the agent switches from Aggregation to Main space due to higher uncertainty in aggregated states at bottlenecks, illustrating the $model\ changes$ during navigation.

agent is trying to learn an optimal policy to reach the desired goal. In contrast to the assumption considered in these methods, we consider the vanilla MDP setting without providing any additional information on the goal state for our agent.

The other group of ideas for the subgoal detection problem are referred to as the option discovery problem in the HRL setting. This is one of the challenging problems in this framework alongside learning hierarchical policies (Hutsebaut-Buysse, Mets, and Latré 2022). It is important to note that option discovery is referred to as skill discovery in some cases. However, the definition and intuition of skill are different from options in the RL literature (Eysenbach et al. 2019).

Methods for creating new options must determine when to create an option, how to define its termination condition (skill discovery), how to expand its initiation set, and how to learn its policy. Consequently, option discovery methods can be categorized into three main groups: implicit option learning, gradient-based option discovery, and option learning approaches based on finding bottleneck states or subgoals.

Implicit option learning methods try to implicitly learn the options by augmenting the proposed semi-MDP into a new MDP by adding options to the state space, and actions of selecting options to the action space. In addition to this, a binary random variable β corresponding to each option is added to action space which is a termination condition with a value of True or False. This illustrates an option has to be terminated when its corresponding binary variable is True (Levy and Shimkin 2012). In (Daniel et al. 2016), they used probabilistic inference methods to infer the termination of an option in the augmented MDP setting. In this framework, the number of options has to be determined before the agent starts the learning process which may need domain knowledge in some environments, and it is hard to apply these methods in continuous domains.

Gradient-based option discovery methods try to define a policy gradient theorem in the options framework considering the option-state value and termination function (Bacon, Harb, and Precup 2017). In (Smith, van Hoof, and Pineau 2018), options are considered as latent variables and they are trained through policy gradient. Harb et al. (2018) used a deliberation cost for learning options in an option-critic framework. Using information-theoretic objectives to learn a diverse set of options was proposed in (Kamat and Precup 2020). In (Levy, Platt, and Saenko 2019; Fox et al. 2017; Riemer, Liu, and Tesauro 2018; Zhang and Whiteson 2019; Wan and Sutton 2022), the authors increased the levels of hierarchy to more than two by proposing new option-critic architectures. Furthermore, Khetarpal et al. (2020) proposed to learn the initial condition of an option in addition to each option's policy and termination condition. In general, gradient-based option discovery methods have shown very good performance, especially in continuous-state space environments. But, they have some drawbacks like being slow in finding options and requiring to fix the number of options before learning.

Option learning methods consider bottleneck states to be more informative and use them to design options automatically. These methods use information like trajectories and acquired rewards to find bottlenecks and design options. These methods usually construct a graph of the environment at the initial steps. They then use graph theoretic methods to find bottlenecks, the methods like min/max flow (Q-cut) (Menache, Mannor, and Shimkin 2002), minimizing the graph's cover time (Jinnai et al. 2019c, 2020, 2019b,a), graph clustering (Mannor et al. 2004), local graph partitioning (L-cut) (Simsek, Wolfe, and Barto 2005), betweenness (Şimşek and Barto 2008), strongly connected components (Kazemitabar and Beigy 2009), min degree and max distance (Zhu, Zhang, and Zhu 2022), and graph Laplacian, by using proto-value functions (Machado, Bellemare, and Bowling 2017; Mendonça, Ziviani, and Barreto 2019). Apart from graph theoretic methods, there are other methods that use heuristics and techniques for subgoal detection. (McGovern and Barto 2001) is one of the first works on option learning that incorporates the diverse density heuristic by collecting successful and unsuccessful trajectories and defining states with high frequency that always appear in successful trajectories. Some notable works for option learning use different approaches for option learning, like skill chaining to discover options in continuous state spaces (Konidaris and Barto 2009), action restriction (states

with unique action direction as subgoals) (Xiao, tong Li, and Shi 2014), calculating occurrence probability (Pateria et al. 2021) and access states with relative novelty for each state (Simsek and Barto 2004).

Graph theoretic and trajectory-based algorithms have some shortcomings in finding bottlenecks. Constructing graphs on the environment using trajectory can be time-consuming and inefficient. Also creating a graph may have problems in the environment with a high rate of stochasticity and this may cause to construction of an inaccurate graph of the environment. Rafati and Noelle (2019) tried to overcome these shortcomings by finding subgoals using unsupervised anomaly detection with the k-means algorithm on experience memory. Ramesh, Tomar, and Ravindran (2019) used successor representations to identify successor options. Manoharan, Ramesh, and Ravindran (2021) used an autoencoder to detect the subgoal of options.

Using sub-spaces has not been studied for subgoal detection, but we shortly review them here, because our method is based on the idea of using sub-spaces. In (Hashemzadeh, Hosseini, and Ahmadabadi 2018), the role of incorporating sub-spaces in the learning process, especially in the initial episodes was studied. Subspaces have the PA problem, to deal with this problem a clustering approach is used in (Hashemzadeh, Hosseini, and Ahmadabadi 2020). A powerful and general framework on how to integrate the decision of the subspaces and the main space was proposed in (Ghorbani et al. 2025) by incorporating a free energy paradigm. In this work, we use the free energy in our proposed framework for bottleneck discovery. Furthermore, our way of defining the subspaces is different from these previous works.

Background

Reinforcement Learning

RL is a framework for solving sequential decision-making tasks in a trial-error manner. Environments in this framework are modeled with a MDP defined by a tuple $< S, A, R, P, \gamma >$ where S is the state space, A is the action space, and R is the expected reward. P specifies the dynamics of the environment and γ is the discount factor. Assume in a MDP at time step t, the agent in state $s_t \in S$ commits action $a_t \in A$ on the environment, the next state of the agent in the environment $s_{t+1} \in S$ is determined by the transition probability $P_a(s, s^{'}) = Pr(s_{t+1} = s^{'} | s_t = s, a_t = a)$, and the agent receives the instant reward r_{t+1}, whose its expectation is equal to $R_a(s, s^{'})$. In most of RL problems, the agent does not know the model of the environment, and it needs to interact with the environment to learn a policy $\pi : S \times A \rightarrow [0, 1]$ during its lifetime in the environment and it is trying to find an estimate to an optimal policy π^*. An optimal policy in each state is a policy that maximizes the cumulative expected reward given by

$$G_t = \sum_{k=t}^{\infty} \gamma^k r_{t+k+1}.$$

In this work, we consider model-free algorithms where the agent does not directly learn the model of the environment, and instead, it calculates an estimate to the action-state value function $Q_t(s, a)$:

$$\begin{aligned} Q_t(s, a) &= \mathbb{E}_\pi[G_t | s, a] \\ &= \sum_{s^{'}, r} P_a(s, s^{'})(R_a(s, s^{'}) + \gamma \sum_{a^{'}} \pi(a^{'} | s^{'}) Q(s^{'}, a^{'})), \end{aligned}$$

using algorithms like single step SARSA (Sutton and Barto 2018) with the following update rule given that in time step t we are in state s and we take action a using our policy:

$$Q_{t+1}(s, a) = Q_t(s, a) + \lambda(r_{t+1} + \gamma Q_t(s^{'}, a^{'}) - Q_t(s, a)),$$

where λ is the learning rate, γ is the discount factor of the environment, $s^{'}$ is the next state upon taking the action, r_{t+1} is the instant reward, and $a^{'}$ is the next action that is chosen based on our policy in the next state. SARSA is an on-policy algorithm that chooses $a^{'}$ with respect to its policy in the next state $s^{'}$.

Free Energy

The free energy paradigm is related to the laws of thermodynamics that explain why energy flows in certain directions. We can use this paradigm for describing a system or making predictions. The free energy concept that we use in decision-making and neuroscience was introduced by (Friston, Kilner, and Harrison 2006) and then developed by (Ortega et al. 2015) for rationally bounded decision-making. From Friston, Kilner, and Harrison (2006) perspective, the human brain is trying to minimize a variable called free energy which is the same as minimizing a surprise function or maximizing evidence of the model of the environment. To achieve this goal, we need to have a good model of the environment.

If we model the observations of the environment using a random variable O, and the hidden state of the environment with the random variable S (we do not have access to the real state of the environment), our brain constructs a generative model defined as $P(O, S)$. We can calculate surprise with $-\log(P(O))$. As the probability of our observation O becomes high (near 1), the surprise of this observation becomes less. Considering a dummy distribution $Q(s)$ we can have:

$$-\log P(O) = -\log \sum_{s \in S} P(O, s) = -\log \sum_{s \in S} Q(s) \frac{P(O, s)}{Q(s)}.$$

The considered surprise function is convex, so we can use Jensens's inequality:

$$f(wx + (1 - w)y) \leq wf(x) + (1 - w)f(y),$$

to calculate an approximate upper bound for the surprise function. So we have:

$$\begin{aligned} -\log \sum_{s \in S} Q(s) \frac{P(O, s)}{Q(s)} &\leq -\sum_{s \in S} Q(s) \log \frac{P(O, s)}{Q(s)} \\ &= \sum_{s \in S} Q(s) \log \frac{Q(s)}{P(O, s)}. \end{aligned}$$

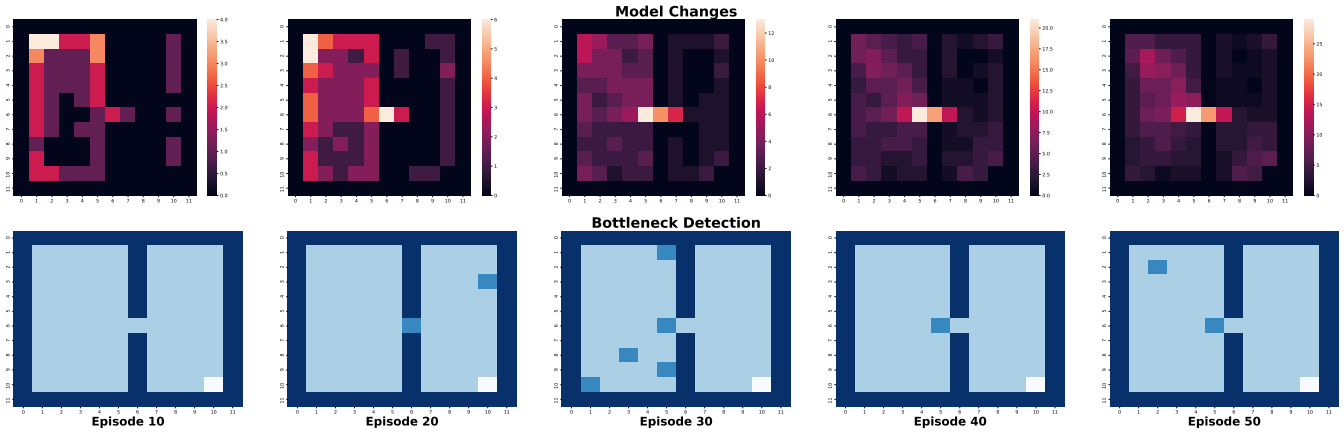

Figure 3: This figure tracks the incremental process of model changes from episodes 10 to 50 in a two-room environment, where the agent starts from the upper left corner and aims to reach a white goal state in the bottom right. The top row shows the number of model changes in transitioning from one state to another, while the bottom row shows identified bottlenecks considering the model changes. By episode 40, the agent accurately identifies the doorway as a bottleneck, and by episode 50, it also recognizes key states along the optimal path from the top-left start to the bottom-right goal.

This upper bound for the surprise function is called variational free energy F, which can be expanded as below:

$$\begin{aligned} F &= \sum_s Q(s) \log \frac{Q(s)}{P(s)P(O \mid s)} \\ &= KL(Q(s)\|P(s)) - \sum_s Q(s) \log P(O \mid s), \end{aligned} \tag{1}$$

where KL measures the KL divergence between Q and P. The first term of (1), named complexity, shows how much the approximation of the posterior (any arbitrary dummy distribution $Q(s)$) deviates from the prior $P(s)$. The second term, named accuracy, shows how likely are the states s given a specific outcome O. We will use this concept to determine which space is predictable and to detect an increase in uncertainty.

The free energy in decision-making when the agent is rationally bounded was studied by Ortega et al. (2015). They derived a free energy formulation for single-step and sequential decision-making problems. In this formulation, the authors combined a linear and a non-linear cost function for calculating the free energy. The linear cost function is related to the expected utility and the non-linear cost function is equal to the KL-divergence of prior and posterior choice probabilities as a measure of the information cost. The expected utility term is the same as the accuracy term in (1). The advantage of this formulation was that it has a closed-form optimal solution.

Aggregating different perspectives in RL using Free Energy

Aggregating different states, i.e. subspaces, has shown a notable enhancement in the performance of agents. In addition to speeding up learning, especially at the initial steps, using subspaces can lead to enhancing sample efficiency as well as lowering regret (Ghorbani et al. 2025; Hashemzadeh, Hosseini, and Ahmadabadi 2018, 2020). However, defining subspaces opens up different research questions like "which states to aggregate as a subspace", "How to cope with PA problem in subspaces" and "which subspaces to choose for learning at each step".

In the most recent work, Ghorbani et al. (2025) introduced a free energy based approach for choosing the best defined subspace for learning at each step. They have supposed a utility function based on the Thompson sampling policy (Russo et al. 2018), as the negative informational surprise:

$$U(a, s, m) = \log \pi_{TS}(a \mid s, m). \tag{2}$$

This utility gives information about the optimality of action a at state s and subspace m.

To utilize subspaces for learning and avoiding the PA problem, the following constraint is used to ensure the utility of the main space is close to the utility of subspaces for any target policy π:

$$\mathbb{E}_{\pi(a|s,m)}[U(a, s, m)] - \mathbb{E}_{\pi(a|s,m)}[U(a, s, m_{Main})] < K_1, \tag{3}$$

where m is any considered subspace and m_{Main} is the main space. Also, to lower the effect of inaccurate uncertainty estimation, another constraint is used to limit the policy by an arbitrary behavioral policy π_B:

$$D_{KL}(\pi(a|s, m)||\pi_B(a|s, m)) < K_2. \tag{4}$$

Considering the defined utility function and mentioned constraints, the problem of learning the optimal policy utilizing different subspaces changes into the following optimization problem which is a free energy minimization, similar to (1):

$$\pi^*(a|s, m) = \arg \min_{\pi(a|,s,m)} F(s, m, \pi(a|s, m)), \tag{5}$$

where the free energy for any target policy $\pi(a|s, m)$ and for

each state s and space m is given by

$$F(s,m,\pi(a|s,m)) = \mathbb{E}_{\pi(a|s,m)}[\frac{1}{\alpha}\log\frac{\pi(a|s,m)}{\pi_B(a|s,m)} + \frac{1}{\beta}\log\frac{\pi(a|s,m)}{\pi_{TS}(a|s,m_{Main})} - \log\pi_{TS}(a|s,m)]. \quad (6)$$

There is a closed-form solution for π^* in (5), that is given by

$$\pi^*(a|s,m) = \frac{1}{z(s,m)}\pi_B(a|s,m)e^{\alpha\hat{U}(a,s,m)}, \quad z(s,m) = \sum_a \pi_B(a|s,m)e^{\alpha\hat{U}(a,s,m)}, \quad (7)$$

wherein

$$\hat{U}(a,s,m) = U(a,s,m) - \frac{1}{\beta}(U(a,s,m) - U(a,s,m_{Main})). \quad (8)$$

Subgoal discovery using State Aggregations and Free Energy paradigm

In this paper, we consider the concept of bottleneck to define options in the environment. States like doorways in multi-room environments can be seen as subgoal states, so identifying such states can help us to decompose the task of navigation from one room to the other or it can be beneficial for autonomous option discovery in the options framework.

To detect such states, we assume our agent lives in an environment with a defined state space and we call this space $Main\ Space$:

$$m_{Main} : \phi_{Main}(s) = s, s \in S,$$

where ϕ_{Main} is an identity function. We update the Q-values of the main space with the update rule of our learning algorithm, SARSA:

$$Q_{Main}(s,a) = Q_{Main}(s,a) + \lambda(r + \gamma Q_{Main}(s^{'},a^{'}) - Q_{Main}(s,a)), \quad (9)$$

where λ is the learning rate, and r is the instant reward that the agent gets from the environment. Also, $(s^{'},a^{'})$ indicates the next state-action pair.

In addition to the main space, we assume our agent has access to its physical neighbor states and their action-state values. So we can define an $Aggregation\ Space$ (m_{Agg}):

$$m_{Agg} : \phi_{Agg}(s) = \{s' | s' \in S\ \&\ d(s,s') < L\}, s \in S,$$

where d is a distance metric like Euclidean or Manhattan distance. Also, L is the maximum distance of the neighborhood of the current state. We don't have any learning in the aggregation space and the Q-values of this space are calculated using a weighted average on aggregated states' Q-value in the main space:

$$Q_{Agg}(s,a) = \frac{1}{\sum_{s'\in\phi_{Agg}(s)} n_{Main}(s',a)} \times \sum_{s'\in\phi_{Agg}(s)} n_{Main}(s^{'},a) \times Q_{Main}(s^{'},a). \quad (10)$$

In this equation, $n_{Main}(s,a)$ gives the frequency of samples of action a in state s in the main space (m_{Main}).

Following (Ghorbani et al. 2025), the best space between the main and the aggregation space is the one that minimizes the free energy given by (5), mathematically speaking:

$$m^*(s) = \arg\min_m F(s,m,\pi^*(a|s,m)), \quad (11)$$

where π^* can be calculated by (7) for each space.

Thompson sampling policy is equal to

$$\pi_{TS}(a_i \mid s,m) = P(\bigcap_{j\neq i}\{Q_m(s,a_i) > Q_m(s,a_j)\}), \quad (12)$$

where $Q_m(s,a)$ is the belief distribution for the value of the action a in state s and space m. Calculating the exact belief distribution for each state-action value in each space is computationally complex and hard to achieve. Thus, we can apply approximation methods by calculating an upper and lower bound for these distributions (Audibert, Munos, and Szepesvári 2009), by considering a confidence interval of $1-\nu$:

$$P(\hat{Q}_m(s,a) - \mu < Q_m(s,a) < \hat{Q}_m(s,a) + \mu) \geq 1 - \nu, \quad (13)$$

where μ is given by

$$\mu = \overline{std}(s,a,m)\sqrt{\frac{2\log\frac{3}{\nu}}{n_m(s,a)}} + \frac{3\log\frac{3}{\nu}}{n_m(s,a)}, \quad (14)$$

and std is computed by

$$\overline{std}(s,a,m) = \sqrt{\frac{n_m(s,a)\sum_t \tilde{Q}_{t,m}(s,a)^2 - \left(\sum_t \tilde{Q}_{t,m}(s,a)\right)^2}{n_m(s,a)\times(n_m(s,a)-1)}}. \quad (15)$$

In this equation, $\tilde{Q}_{t,m}$ is the value of action a in state s and space m at the step t of the agent's lifetime.

We now have the ingredients to implement our idea of bottleneck discovery as explained in the introduction (see Figs 1,2). We defined model changes as a measure of the irregularity of a state. If by entering a state from its neighboring states, there is a change between the best spaces, we count up the value of the model change of that particular state, that is

$$MC(s) = \begin{cases} MC(s)+1 & m^*(s_{t-1}) \neq m^*(s_t), \\ MC(s) & \text{otherwise.} \end{cases} \quad (16)$$

In (16), $m^*(s_t)$ is the free energy model of the environment for the current state of the agent and $m^*(s_{t-1})$ is the free energy model of the previous state.

Figure 5 shows the flow of our proposed method. In each step of the episode, we first estimate the Thompson sampling policy by approximating the intervals of state-action values for each space, which is computed by (12). After this step, we calculate the free energy of each space to decide which space has minimum free energy, computed by (5). If the free energy model of the current state is different than the previous state, we will count for model changes in the current state, as expressed by (16). For the aim of bottleneck detection, we apply Otsu's thresholding (Otsu 1979) to determine

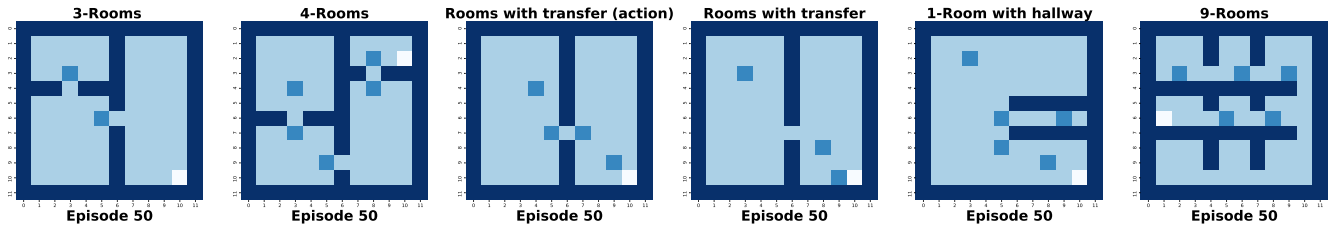

Figure 4: This figure shows detected bottlenecks (in light blue) across six different environments at episode 50, using a state aggregation distance of L=2 for aggregation. In each environment, the agent starts from the upper left corner and aims to reach a white goal state in the bottom right. In the 1-room with hallway environment, bottlenecks appear along the hallway and near the goal due to early exploration patterns and low model changes throughout this environment. While these hallway bottlenecks could be useful for defining "leaving hallway" options, increasing L could prevent their detection in case they are undesired.

the states with a higher count of model changes considering model changes as a matrix of all states. This thresholding algorithm is variance-aware and it clusters states into two groups of zeros and ones. The output of this thresholding method is a matrix with values of zero or one for each state. By piecewise multiplying the output of Otsu's thresholding algorithm and model changes matrices, and finally applying a non-maximum suppression on the result matrix, we can identify bottleneck states.

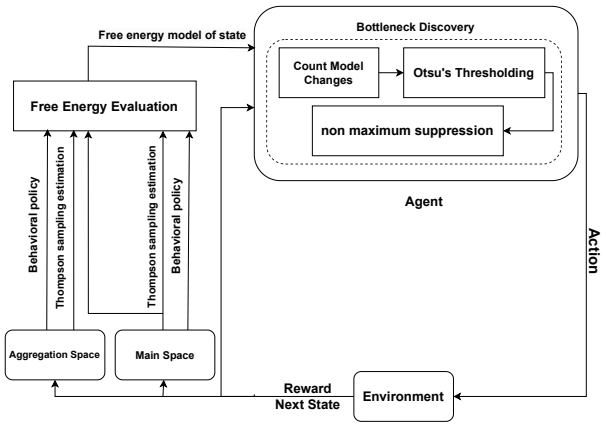

Figure 5: This diagram illustrates the proposed method. The system evaluates free energy in both aggregation and main spaces using an approximate estimation of Thompson sampling and behavioral policy. The bottleneck discovery module tracks model changes between states, applies Otsu's thresholding, and uses non-maximum suppression to identify bottleneck states. The agent interacts with the environment, receiving rewards and next states while updating its state space model based on free energy evaluation.

Algorithm 1 specifies the pseudocode of our proposed method. Also, the computational complexity of calculating free energy models for each space is $O(|S||A|^2)$, where $|S|$ denotes the cardinality of the state space and $|A|$ is the cardinality of the action space.

The idea of model changes can be used in environments with continuous state space with some modifications. Considering an infinite number of states in these kinds of environments, for calculating Q-values in aggregation space, we can sample from the neighborhood of the current state by considering a maximum distance and a distance measure. Also, to calculate the Q-value in the aggregation space, we need sample counts of each action in each neighbor state. We can use the experience of the agent which is saved in the replay buffer for this purpose.

Algorithm 1: Our Proposed Algorithm

for Each Episode **do**
  while $done \neq True$ **do**
    next state, reward, done = environment.step(action)
    calculate free energy model of state, using (5)
    if $m^*(state) \neq m^*(previous\ state)$ **then**
      Count for model change in state, using (16)
    end if
  end while
  Apply Otsu's thresholding on model changes matrices
  Non maximum suppression on $model\ \ changes \times Otsu's\ thresholding\ output$
end for
return bottleneck states

Algorithm 2: Model Changes in Continuous State Space

for Each Episode **do**
  while $done \neq True$ **do**
    next state, reward, done = environment.step(action)
    Sample from neighbor states with a distance of L
    Calculate Q-values for aggregation space using replay buffer
    Calculate Thompson sampling by considering the frequency of actions in the replay buffer
    Calculate free energy model of state, using (5)
    if $m^*(state) \neq m^*(previous\ state)$ **then**
      Count for model change in state, using (16)
    end if
  end while
end for
return Model changes for each state

In continuous state spaces, it is possible to calculate an approximation of a belief distribution by applying dropout before each weighting layer of the Q-network. Thus, we can calculate the Thompson sampling policy using (17), where N is the number of networks estimating Q-values for action a_i in state s and n_s is the number of times that action a_i has been selected in state s (Ghorbani et al. 2025; Gal and Ghahramani 2016):

$$\pi_{TS}(a_i|s) = \frac{n_s(a_i, s)}{N}. \tag{17}$$

Algorithm 2 provides the modification of our developed algorithm for model changes in the continuous domain.

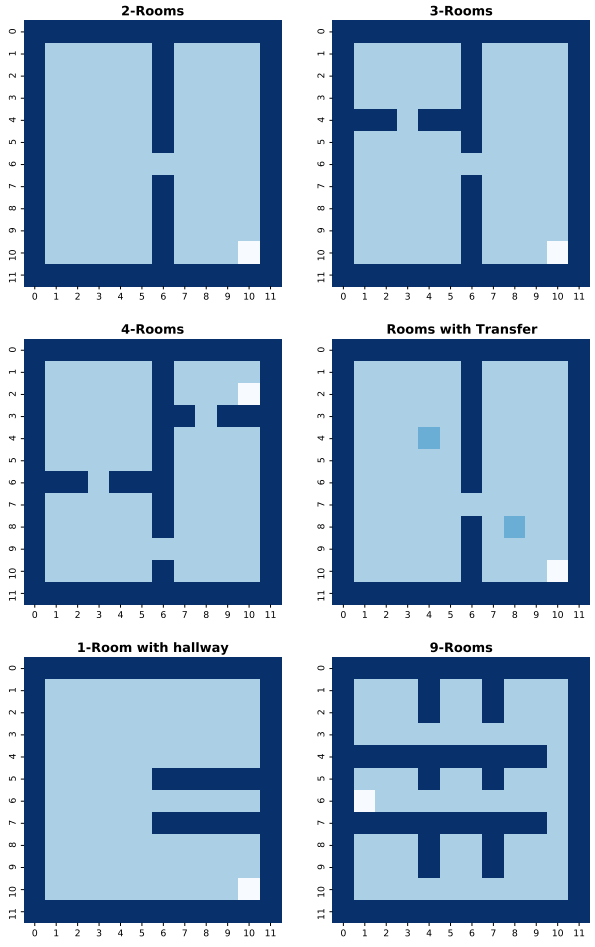

Figure 6: This figure shows six different grid-world environments used in the experiments: 2-rooms, 3-rooms, 4-rooms, rooms with a transfer state (in two variants), 1-room with a hallway, and 9-rooms. In each environment, the white cell represents the goal state. The rooms with transfer state environment has two versions: one with a special transfer action that is activated in state (4, 4), and another where stepping into state (4,4) automatically teleports the agent to state (8,8).

Experimental Results

To test the performance of our algorithm, we designed two sets of environments with discrete and continuous state spaces. Figure 6 and Figure 7 indicate our gird-world environments with discrete and continuous state space, respectively. The coordinates of the agent in each step are considered as states. The agent can do four actions navigating up, down, left, and right in both kinds of environments. If the agent's action leads to collision with walls, the agent will remain in the same state. Each action can fail with a probability of p, in this case the agent goes to any of the neighboring states with the equal probability. We use the Euclidean distance with a maximum distance of $L = 2$ for the aggregation space in all of the environments.

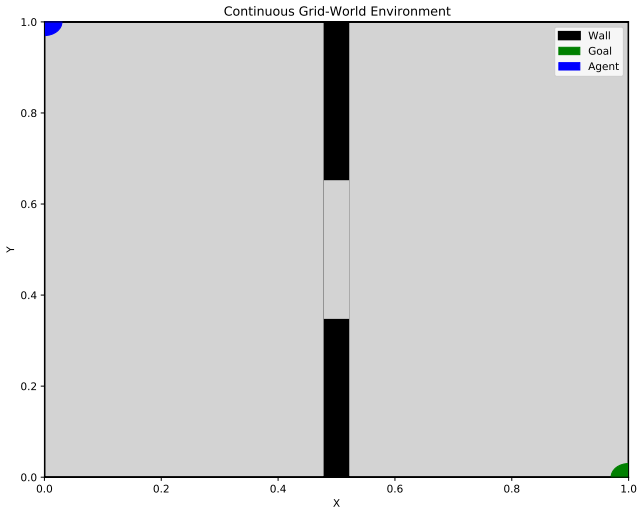

Figure 7: Environment with continuous state space used in our experiments. The goal state in this environment is specified with a green color at the right-bottom corner. The agent can start learning from different corners of the environment except for the goal state.

The agent receives a reward of -1 for taking each step in the environment. If the agent reaches the goal state it will get $+10$ as a reward and if it takes an action that leads to collision with walls it will get a reward of -10. In all of the environments, the agent can start randomly from a state at the corners of the environment, and the episode terminates if the agent reaches the goal state or if it reaches the maximum steps, defined for each environment.

As shown in Figure 6, we consider 6 environments with discrete states, where the environment with a transfer state has two versions. In the first version, we consider an additional transfer action which will be fired just in the state (4, 4) and it moves the agent to the state (8, 8). This action, similar to other actions, has a probability of p of failing, in which case the agent transitions to a random neighboring state. In the other version, the state (4, 4) acts like a teleport that transfers the agent to the state (8, 8). The maximum step for 2-rooms, 3-rooms, and rooms with the transfer is 100 steps, and because of the complexity of tasks in 4-rooms

and 9-rooms environments, the maximum step is considered 500 in these environments. Also, the 1-room with a hallway environment is a tricky environment, so the number of maximum steps is 150 in this environment. All of the environments contain bottleneck states, because of their design which is room-to-room navigation task they have. It is clear that bottleneck states are the doorway states (in the multi-room environment) and the neighbor states of the transfer state in the environments with the transfer state.

In the environment with the continuous state space, each action results in a displacement of 0.1 units in the corresponding coordinates of the agent's current state. For instance, the right action would move the agent from $(0, 0)$ to $(0, 0.1)$. After a maximum of 300 steps, if the agent could not reach the goal state, the episode is terminated. The reward function is the same as that of the discrete environments and there is no transfer state.

Results in Discrete State Spaces

In our implementations, we consider the agent learns and interacts with the environment using the SARSA algorithm that has an epsilon greedy policy as its behavioral policy. The discount factor is chosen to be equal to 0.9, and the learning rate is equal to 0.99 with a decaying rate of 0.001. Also, the epsilon is 0.3 and it decays exponentially with a rate of 0.3. For parameters α and β in equations (6), (7), and (8), we choose $\alpha = 4$ and $\beta = 7$, that are the same parameters used in the implementation of (Ghorbani et al. 2025). Similar to (Ghorbani et al. 2025), we also observed that our method is not sensitive to the choice of these two parameters. All these parameters are the same for the results in all discrete-state environments and all results are an average of 10 runs. In addition, the default probability of failing an action p is set equal to 33% for all environments. We initialized the agent at the top-left corner of the environment in all experiments. However, our method's ability to identify bottleneck states is not dependent on the initial state, as long as the starting and goal states are in different rooms.

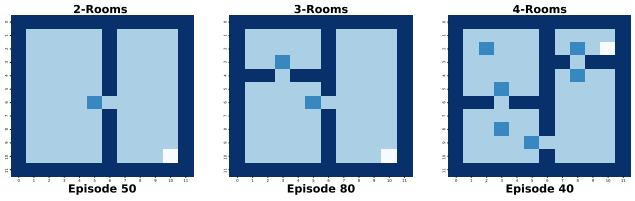

Figure 8: Performance of our proposed bottleneck detection within the high probability of action failure ($p = 50\%$) in episode 50 of 2-rooms, episode 80 of 3-rooms, and episode 40 of 4-rooms environments. Identified bottlenecks (light blue states) in all of these environments are on the optimal path to reaching the goal (white state).

Figure 3 shows the results of the model change counts and our bottleneck discovery algorithm for different number of episodes for the 2-room discrete environment. As we can see in the initial episodes we have rare model changes in different states. This is because of the exploration of the agent at the first steps. However, our agent can successfully detect the doorway bottleneck after some episodes.

Figure 4 shows the identified subgoal/bottleneck states in different grid-world environments after 50 episodes. We can see that our agent can detect states like doorways and states around the transfer states. Also in the tricky 9-rooms environment, the agent was able to find some of the doorways that are needed to reach the final goal. In the 1-room with a hallway environment, the identified bottleneck states appear reasonable from a task decomposition perspective. They can be used to define tasks like "entering the hallway" or "leaving the hallway". However, careful tuning of the algorithm's parameters for this environment can lead to the discovery of more specific subgoals that contribute to reaching the final goal.

Because methods based on environment graphs struggle in highly stochastic environments, we tested our algorithms' ability to handle high levels of stochasticity. As shown in Figure 8, our algorithm successfully identified bottleneck states in environments with a high probability of action failure $p = 50\%$.

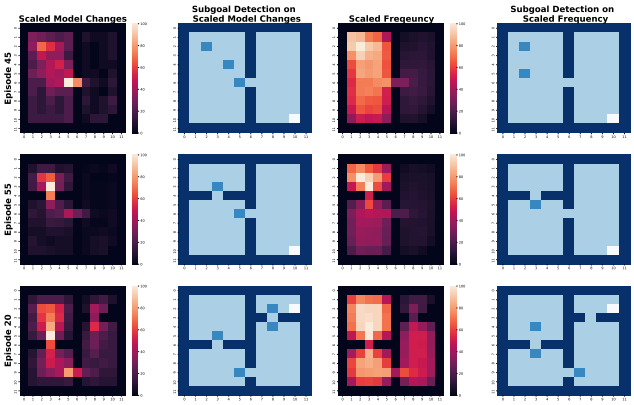

Figure 9: Performance of our proposed bottleneck detection algorithm in comparison with frequency-based algorithms in different environments. Columns one and two (from left) show the scaled model changes and the resulting bottlenecks. Columns three and four (from left) show the scaled state frequency and the detected bottlenecks. We scaled model changes and state frequencies for better comparison.

Comparison with Experience-Based methods

Since there is no quantitative criterion to compare the performance of bottleneck discovery algorithms, we show the bottlenecks discovered by our method and an experienced-based method. Similar to our method, we apply non-maximum suppression on the resulting state visit counts of the experience-based method. Figure 9 shows the results for different environments with the action failure probability of 33%. Because of the high level of stochasticity in these environments, relying on the trajectory information of the agent can be misleading when determining subgoal states. Therefore in such settings, the performance of experience-based

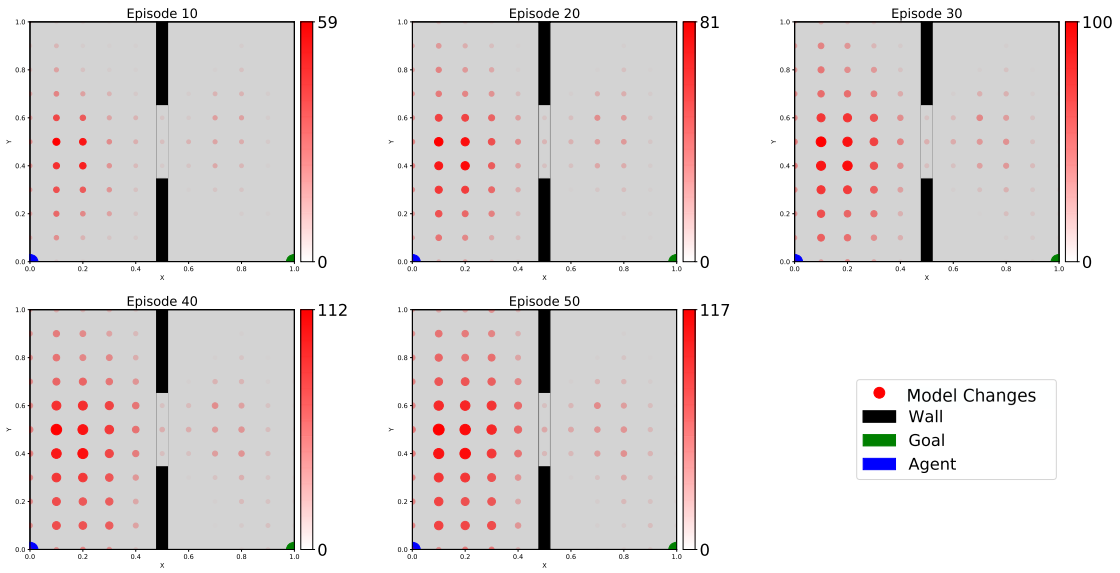

Figure 10: This figure shows the evolution of model changes in a continuous 2-room environment across episodes 10-50. The intensity of red dots represents the frequency of model changes, with brighter red indicating more changes. The agent starts from either the bottom-left or top-left corner of the left room and aims to reach the green goal in the bottom-right corner. The visualization shows an increasing concentration of model changes near the doorway connecting the two rooms, as indicated by the brighter red dots in that area.

methods substantially deteriorates. In contrast, our proposed method succeeds in identifying the correct subgoal states. This is because our method considers both the behavioral policy and the uncertainty in subspaces which is relatively robust to the stochasticity of the environment.

Model Changes in Continuous State Spaces

For the environment with continuous state space (Figure 7), we used DQN (Mnih et al. 2015) with a fully connected architecture to learn the task. The replay buffer has the capacity of 10000 samples. The target network's weights get updated every 5 episodes. The behavioral policy of the agent is epsilon-greedy and the parameters α and β are the same as our experiments in the discrete environments.

Figure 10 shows the result of Algorithm 2 for computing model changes in this environment. The results demonstrate that the phenomenon of model changes is not limited to the tabular settings and we have model changes when using function approximations, such as deep neural networks, for learning.

Conclusion

In this paper, we study the problem of subgoal discovery in different grid-room environments. We showed that our method can detect bottleneck states in different types of doorways and transfer states and it is robust to the noise and stochasticity of the environment. Our method does not need to save full information of trajectories in memory or to generate a graph from interactions of the agent in the environment, which can be misleading when the noise of the environment is considerably high. Our proposed method detects the bottleneck states in the environment without any supervision or predefined number.

There are several directions to expand this work in the future. One avenue is using model changes for bottleneck detection in environments with continuous states or actions, and environments with sparse rewards. Another direction of future work is searching for efficient ways to lower the time complexity of our algorithm. Making use of the discovered bottlenecks to learn reasonable and interpretable options in HRL or GCRL can be an interesting extension of the proposed method.

References

Audibert, J.-Y.; Munos, R.; and Szepesvári, C. 2009. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19): 1876–1902. Algorithmic Learning Theory.

Bacon, P.-L.; Harb, J.; and Precup, D. 2017. The option-critic architecture. In *AAAI conference on artificial intelligence*, volume 31.

Chane-Sane, E.; Schmid, C.; and Laptev, I. 2021. Goal-Conditioned Reinforcement Learning with Imagined Sub-

goals. In *International Conference on Machine Learning*, volume 139, 1430–1440. PMLR.

Christiano, P. F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Şimşek, O.; and Barto, A. 2008. Skill characterization based on betweenness. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.

Daee, P.; Mirian, M. S.; and Ahmadabadi, M. N. 2014. Reward Maximization Justifies the Transition from Sensory Selection at Childhood to Sensory Integration at Adulthood. *PLOS One*, 9(7): 1–13.

Daniel, C.; Van Hoof, H.; Peters, J.; and Neumann, G. 2016. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104: 337–357.

Dayan, P.; and Hinton, G. E. 1992. Feudal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann.

Eysenbach, B.; Gupta, A.; Ibarz, J.; and Levine, S. 2019. Diversity is All You Need: Learning Skills without a Reward Function. In *International Conference on Learning Representations*.

Fox, R.; Krishnan, S.; Stoica, I.; and Goldberg, K. 2017. Multi-Level Discovery of Deep Options. arXiv:1703.08294.

Friston, K.; Kilner, J.; and Harrison, L. 2006. A free energy principle for the brain. *Journal of Physiology-Paris*, 100(1-3): 70–87.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, volume 48, 1050–1059. PMLR.

Ghorbani, M.; Hosseini, R.; Shariatpanahi, S. P.; and Ahmadabadi, M. N. 2025. Learning from different perspectives for regret reduction in reinforcement learning: A free energy approach. *Neurocomputing*, 614: 128797.

Gupta, A.; Kumar, V.; Lynch, C.; Levine, S.; and Hausman, K. 2020. Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning. In *Conference on Robot Learning*, volume 100, 1025–1037. PMLR.

Harb, J.; Bacon, P.-L.; Klissarov, M.; and Precup, D. 2018. When waiting is not an option: Learning options with a deliberation cost. In *AAAI Conference on Artificial Intelligence*, volume 32.

Hashemzadeh, M.; Hosseini, R.; and Ahmadabadi, M. N. 2018. Exploiting generalization in the subspaces for faster model-based reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(6): 1635–1650.

Hashemzadeh, M.; Hosseini, R.; and Ahmadabadi, M. N. 2020. Clustering subspace generalization to obtain faster reinforcement learning. *Evolving Systems*, 11(1): 89–103.

Hutsebaut-Buysse, M.; Mets, K.; and Latré, S. 2022. Hierarchical Reinforcement Learning: A Survey and Open Research Challenges. *Machine Learning and Knowledge Extraction*, 4(1): 172–221.

Jinnai, Y.; Abel, D.; Hershkowitz, D.; Littman, M.; and Konidaris, G. 2019a. Finding Options that Minimize Planning Time. In *International Conference on Machine Learning*, volume 97, 3120–3129. PMLR.

Jinnai, Y.; Abel, D.; Park, J. W.; Hershkowitz, D. E.; Littman, M. L.; and Konidaris, G. 2019b. Skill Discovery with Well-Defined Objectives. In *International Conference on Learning Representations Workshop on Structure and Priors in Reinforcement Learning*.

Jinnai, Y.; Park, J. W.; Abel, D.; and Konidaris, G. 2019c. Discovering Options for Exploration by Minimizing Cover Time. In *International Conference on Machine Learning*, volume 97, 3130–3139. PMLR.

Jinnai, Y.; Park, J. W.; Machado, M. C.; and Konidaris, G. 2020. Exploration in Reinforcement Learning with Deep Covering Options. In *International Conference on Learning Representations*.

Kamat, A.; and Precup, D. 2020. Diversity-Enriched Option-Critic. arXiv:2011.02565.

Kazemitabar, S. J.; and Beigy, H. 2009. Using Strongly Connected Components as a Basis for Autonomous Skill Acquisition in Reinforcement Learning. In *Advances in Neural Networks – ISNN 2009*, 794–803. Springer Berlin Heidelberg.

Khetarpal, K.; Klissarov, M.; Chevalier-Boisvert, M.; Bacon, P.-L.; and Precup, D. 2020. Options of interest: Temporal abstraction with interest functions. In *AAAI Conference on Artificial Intelligence*, volume 34, 4444–4451.

Kiran, B. R.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A. A.; Yogamani, S.; and Pérez, P. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6): 4909–4926.

Konidaris, G.; and Barto, A. 2009. Skill Discovery in Continuous Reinforcement Learning Domains using Skill Chaining. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc.

Levy, A.; Platt, R.; and Saenko, K. 2019. Hierarchical Reinforcement Learning with Hindsight. In *International Conference on Learning Representations*.

Levy, K. Y.; and Shimkin, N. 2012. Unified Inter and Intra Options Learning Using Policy Gradient Methods. In Sanner, S.; and Hutter, M., eds., *Recent Advances in Reinforcement Learning*, 153–164. Springer Berlin Heidelberg.

Li, Z.; Zhu, D.; Hu, Y.; Xie, X.; Ma, L.; ZHENG, Y.; Song, Y.; Chen, Y.; and Zhao, J. 2023. Neural Episodic Control with State Abstraction. In *International Conference on Learning Representations*.

Liu, M.; Zhu, M.; and Zhang, W. 2022. Goal-Conditioned Reinforcement Learning: Problems and Solutions. In *International Joint Conference on Artificial Intelligence*, 5502–5511. IJCAI.

Machado, M. C.; Bellemare, M. G.; and Bowling, M. 2017. A Laplacian Framework for Option Discovery in Reinforcement Learning. In *International Conference on Machine Learning*, volume 70, 2295–2304. PMLR.

Mannor, S.; Menache, I.; Hoze, A.; and Klein, U. 2004. Dynamic abstraction in reinforcement learning via clustering. In *International Conference on Machine Learning*, 71. Association for Computing Machinery.

Manoharan, A.; Ramesh, R.; and Ravindran, B. 2021. Option Encoder: A Framework for Discovering a Policy Basis in Reinforcement Learning. In *Machine Learning and Knowledge Discovery in Databases*, 509–524. Springer International Publishing.

McGovern, A.; and Barto, A. G. 2001. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. In *International Conference on Machine Learning*, 361–368. Morgan Kaufmann Publishers Inc.

Menache, I.; Mannor, S.; and Shimkin, N. 2002. Q-Cut—Dynamic Discovery of Sub-goals in Reinforcement Learning. In *European Conference on Machine Learning*, 295–306. Springer Berlin Heidelberg.

Mendonça, M.; Ziviani, A.; and Barreto, A. 2019. Laplacian using Abstract State Transition Graphs: A Framework for Skill Acquisition. In *8th Brazilian Conference on Intelligent Systems*, 263–268. IEEE.

Mirhoseini, A.; Goldie, A.; Yazgan, M.; Jiang, J. W.; Songhori, E.; Wang, S.; Lee, Y.-J.; Johnson, E.; Pathak, O.; Nova, A.; Pak, J.; Tong, A.; Srinivasa, K.; Hang, W.; Tuncer, E.; Le, Q. V.; Laudon, J.; Ho, R.; Carpenter, R.; and Dean, J. 2021. A graph placement methodology for fast chipdesign. *Nature*, 594(7862): 207–212.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Nachum, O.; Gu, S. S.; Lee, H.; and Levine, S. 2018. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Nair, S.; and Finn, C. 2020. Hierarchical Foresight: Self-Supervised Learning of Long-Horizon Tasks via Visual Subgoal Generation. In *International Conference on Learning Representations*.

Ortega, P. A.; Braun, D. A.; Dyer, J.; Kim, K.-E.; and Tishby, N. 2015. Information-Theoretic Bounded Rationality. arXiv:1512.06789.

Otsu, N. 1979. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1): 62–66.

Parr, R.; and Russell, S. 1997. Reinforcement Learning with Hierarchies of Machines. In *Advances in Neural Information Processing Systems*, volume 10. MIT Press.

Pateria, S.; Subagdja, B.; Tan, A.-H.; and Quek, C. 2021. End-to-end hierarchical reinforcement learning with integrated subgoal discovery. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12): 7778–7790.

Rafati, J.; and Noelle, D. 2019. Unsupervised Methods For Subgoal Discovery During Intrinsic Motivation in Model-Free Hierarchical Reinforcement Learning. In *AAAI conference on artificial intelligence Workshop on Knowledge Extraction from Games*, 17–25.

Ramesh, R.; Tomar, M.; and Ravindran, B. 2019. Successor Options: An Option Discovery Framework for Reinforcement Learning. In *International Joint Conference on Artificial Intelligence*, 3304–3310. IJCAI.

Riemer, M.; Liu, M.; and Tesauro, G. 2018. Learning Abstract Options. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Russo, D. J.; Van Roy, B.; Kazerouni, A.; Osband, I.; Wen, Z.; et al. 2018. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1): 1–96.

Simsek, O.; and Barto, A. G. 2004. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *International conference on Machine learning*, 95. Association for Computing Machinery.

Simsek, O.; Wolfe, A. P.; and Barto, A. G. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *International conference on Machine learning*, 816–823. Association for Computing Machinery.

Smith, M.; van Hoof, H.; and Pineau, J. 2018. An Inference-Based Policy Gradient Method for Learning Options. In *International Conference on Machine Learning*, volume 80, 4703–4712. PMLR.

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

Sutton, R. S.; Precup, D.; and Singh, S. 1998. Between MOPs and Semi-MOP: Learning, Planning & Representing Knowledge at Multiple Temporal Scales. Technical report.

Theves, S.; Neville, D. A.; Fernández, G.; and Doeller, C. F. 2021. Learning and representation of hierarchical concepts in hippocampus and prefrontal cortex. *Journal of Neuroscience*, 41(36): 7675–7686.

Wan, Y.; and Sutton, R. S. 2022. Toward Discovering Options that Achieve Faster Planning. *arXiv preprint arXiv:2205.12515*.

Xiao, D.; tong Li, Y.; and Shi, C. 2014. Autonomic discovery of subgoals in hierarchical reinforcement learning. *The Journal of China Universities of Posts and Telecommunications*, 21(5): 94–104.

Zhang, S.; and Whiteson, S. 2019. DAC: The Double Actor-Critic Architecture for Learning Options. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zhu, X.; Zhang, R.; and Zhu, W. 2022. MDMD options discovery for accelerating exploration in sparse-reward domains. *Knowledge-Based Systems*, 241: 108151.