Dynamic Option Creation in Option-Critic Reinforcement Learning

Mateus B. Melchiades
Universidade do Vale do Rio dos Sinos
São Leopoldo, Brazil
mateusbme@edu.unisinos.br

Gabriel de O. Ramos
Universidade do Vale do Rio dos Sinos
São Leopoldo, Brazil
gdoramos@unisinos.br

Bruno C. da Silva
University of Massachusetts, Amherst
Amherst, United States of America
bsilva@cs.umass.edu

ABSTRACT

Reinforcement Learning (RL) is an increasingly popular technique in the field of machine learning due to its ability to learn by interacting directly with the environment. The options framework introduces the concept of temporal abstraction in MDPs by combining high level courses of action that may span over multiple time steps with primitive, single-step actions, which can greatly improve planning and learning speeds. Throughout the past two decades, there has been active interest in autonomous option discovery, as well as determining what characterizes a good option. The Option-Critic Architecture and its successors accomplished several improvements in autonomous option discovery. However, given the fact that in most problems the ideal number of options for learning an optimal policy is not evident, Option-Critic's reliance on a fixed set of options proves as a limitation. In the present work, we propose an algorithm for creating options dynamically in training time, using the Fast-Planning Option-Critic implementation as a base. The Dynamic Option Creation algorithm (DOC) analyzes the variance in episodic returns when selecting each option to determine whether the learning process would benefit from a new option. The variance in returns is expected to start high and decrease as the agent learns the environment, which may not happen if the current set of options cannot properly represent the desirable behavior. Our method manages to achieve similar cumulative per-episode reward in the four-rooms environment as FPOC adjusted to use the best number of options, with the added benefit of discovering such number automatically. The proposed method can also be adapted to other Option-Critic algorithms, solving a major limitation of the original architecture, which requires multiple runs with different parameters to determine the ideal number of options for the task.

KEYWORDS

Reinforcement Learning, Options, Dynamic, Creation, Option-Critic

1 INTRODUCTION

The Options Framework [22] introduces the concept of temporally abstracted actions in Reinforcement Learning, which has proven paramount in learning complex behavior patterns. Options can reduce the time to learn an optimal policy for a given environment by abstracting desired courses of action into sub-policies, known as intra-option policies, each representing some higher level action. For instance, where a traditional MDP would need to handle a robot's movement as a series of voltages to apply to each stepper motor controlling movement, options can learn a single extended action such as moving a leg or turning 180 degrees. Then, a policy over options would only have to learn how to best select this set of temporally extended actions to complete the task at hand. This method of choosing actions is comparable to human-like planning, where actions are not considered as a series of muscle twitches, but instead as higher order tasks like walking towards a place or picking up an object [22].

Despite its clear advantage, what characterizes a good option, as well as what should an option achieve, is still open for debate. While some works defend that discovered options should reach *bottleneck states*—i.e. hallways in a four-rooms environment—[15, 16, 18, 19], others suggest that options should instead focus on being transferable across different tasks [4, 5, 11]. Furthermore, most works in literature fail to acknowledge the extra planning cost inherent from having more options [23]. We approach the option discovery problem with the idea that options should complement each other, and that new options should be created if, and only if, the current set of options cannot cover the entire state space. Given an environment with a set of options that span between dynamic—learnable—and primitive—fixed—options, if no option is capable of providing consistent returns for a region in the state space, we can assume such region is not covered and a new option is necessary for handling it. An inconsistent set of returns can be represented by the variance in accumulated rewards over multiple episodes. When the agent is in the process of learning an environment, its accumulated return over time tends to vary drastically as it tries varied courses of action, but decreases as it learns what behaviors are effective. If we consider the high variability—variance—in returns as the agent's *uncertainty* towards an environment, we can interpret a continuously high variance as the agent struggling to learn some part of the environment.

In the present work, we introduce Dynamic Option Creation (DOC): an algorithm capable of automatically scaling the number of options dynamically by observing the variance in accumulated rewards over time. If the variance in accumulated rewards fails to decrease as training progresses, then a new option is automatically created and initialized using experience replay from steps where previous options are unlikely to be selected. We assess the effectiveness of our method in the classical four-rooms environment [22] and make a comparison against Fast-Planning Option-Critic [23]—the implementation used as base for ours. Our results show that our method achieves learning curves comparable to FPOC, but with the added benefit of scaling to the necessary number of options automatically. To the best of our knowledge, this is the first approach capable of creating options dynamically in Option-Critic. Given that traditional Option-Critic algorithms have to be executed multiple times, each with a different number of options to discover

Proc. of the Adaptive and Learning Agents Workshop (ALA 2025), Avalos, Aydeniz, Müller, Mohammedalamen (eds.), May 19 – 20, 2025, Detroit, Michigan, USA, ala-workshop.github.io. 2025.

the ideal value for the given task, our method is able to speed up training by only having to run once.

The main contributions of this work can be enumerated as:

1. A method for evaluating a candidate set of options using a metric based on each option's return variance as a proxy for epistemic uncertainty;
2. A mechanism to dynamically determine whether new options may be needed given the task at hand;
3. A method to initialize a new option's policy by actively selecting relevant experiences from a replay buffer;
4. An empirical evaluation of the proposed method demonstrating it can achieve equal (or better) performance than variants of the Option-Critic algorithm, *without requiring prior knowledge of the optimal number of options for a given set of tasks.*

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 details fundamental concepts. Section 4 introduces our Dynamic Option Creation approach. Section 5 presents the experimental evaluation. Concluding remarks and future directions are discussed in Section 6.

2 RELATED WORK

Many of the initial work in option discovery was primarily focused on autonomously discovering subgoals. One popular approach for determining good subgoals was through the concept of *bottleneck states*, which represent regions where the agent tends to visit frequently on successful paths but not on unsuccessful ones [15, 16, 18, 19]. The idea behind learning bottleneck states as options was that, by delegating the act of reaching important states to a separate policy, the main policy could focus on reaching the goal once it is there.

In recent years, however, there have been numerous option discovery algorithms that go beyond just bottlenecks. Machado et al. [12] introduce the concepts of *eigenpurpose* and *eigenbehavior* for describing reward functions and policies by combining representation learning with option discovery. The authors construct proto-value functions (PVFs) [14] from the MDP's transition matrix which represent the desire to reach some specific region in the state space. Their conclusion was that *eigenoptions*—the options discovered by their method—tend to incentivize exploration while not discovering only bottlenecks, which they argue can hinder exploration when naively added to the agent's action set. Subsequent work focus on expanding the concept of eigenoptions to stochastic environments [13] and deep reinforcement learning [9], respectively.

Other authors take advantage of the abstraction capabilities of options for learning skills that can be used in tasks similar to the one the option was trained on. In Konidaris and Barto [11], the authors propose learning options over two separate representations, one in problem-space that is Markov and particular for the current task, and other in agent-space that may not be Markov but is retained across tasks. Croonenborghs et al. [4] approach skill transfer by extending the options framework to the relational setting, which allows abstractions over object identities. More recently, Han and Tschiatschek [5] propose finding *abstract successor options*, which represents options through their successor features. Successor features can be described as the discounted sum of features of state-action pairs encountered when starting from some state and following a given policy [2].

The Option-Critic (OC) Architecture is derived from Actor-Critic [10] and uses the policy gradient theorem [10, 21] to derive new policies, thus addressing the scalability issues that arise with finding subgoals and their respective optimal policies in more complex problems [1]. One of the method's greatest improvements over previous work is that it can gradually learn the intra-option policies and the policy over options at the same time, incurring no slowdown when compared to the traditional RL learning process. In the context of OC, the intra-option policies, termination functions, and policy over options are part of the actor, while Q_U and A_O belong to the critic.

The Asynchronous Advantage Option-Critic (A2OC) [6] improves upon Option-Critic by addressing its tendency to often degenerate options into single-step actions by using the notion of deliberation cost. The introduction of deliberation cost encourages the agent to sustain an option for a longer period by penalizing frequent option switching, leading to more stable and specialized options. The proposed algorithm builds on top of the asynchronous advantage actor-critic (A3C) [17], which leverages parallel agent updates to stabilize learning in large-scale environments. Meanwhile, the Actor-Critic Termination-Critic (ACTC) [7] iterates upon Option-Critic and A2OC by focusing on the predictability and *compressibility* of options. The authors propose that the termination condition of options should minimize the entropy of the final state distribution, meaning that an option should ideally terminate in a small, predictable set of states, thus leading to simpler and more behaviorally meaningful abstractions.

Other notable variations of Option Critic are the Safe Option-Critic and the Attention Option-Critic. The Safe Option-Critic [8] is an extension of OC that focuses on safe behavior. The authors consider a behavior as safe when it avoids regions of state-space with high uncertainty in its outcomes. By defining controllable states as those with a lower variance in TD error, the Safe Option-Critic algorithm adds controllability to the optimization objective. As a result, the proposed method managed to identify and avoid areas in the state space with high variability, which, as expected, result in more predictable returns. Meanwhile, the Attention Option-Critic [3] uses an attention mechanism for learning options that focus on different aspects of the observation space. The algorithm learns an attention mechanism with the intent of maximizing the expected cumulative return of the agent while also maximizing the distance between the attentions of options. When compared to Option-Critic, the learned options are not only more diverse, but also do not degenerate over time, meaning that they are more evenly selected and last longer.

We base our method on the Fast Planning Option-Critic proposed by Wan and Sutton [23], which extends option discovery to the multi-task problem. In the authors' implementation, all tasks share the same state and action spaces, but each task has its own corresponding set of terminal states. Once the agent reaches a terminal state for the current task, both the episode and task are complete. The proposed method manages to accomplish faster planning by reducing the number of elementary operations in option-value iterations. The proposed algorithm, however, is limited in three major ways according to the authors. The first limitation is the fact that the algorithm treats only the tabular setting, which limits the complexity of problems that can be tackled. The second limitation is the fixed set of tasks, which the authors argue should be discovered by the agent. The last major limitation is the reliance on a human-specified number of options, which we tackle in our approach. Although some of the early work in option discovery supported a dynamic number of options, recent implementations rely on a user-defined number, which can hinder their quality if set too low and hurt planning speeds if set too high. To the best of our knowledge, this is the first work that focuses on implementing a dynamic option scaling algorithm to Option-Critic algorithms.

3 PRELIMINARIES

A Markov Decision Process, MDP for short, is the formalization of the sequential decision making problem, where taking an action in the current time step affects not only the agent's immediate reward, but also the subsequent state, and, consequently, future rewards as well [20]. We can mathematically define a finite MDP as a tuple $\mathcal{M} \doteq \langle \mathcal{S}, \mathcal{A}, \gamma, r, p \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action sets, respectively, γ the discount factor, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the reward function, and $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ the probability distribution over next states and rewards given that an action $a \in \mathcal{A}$ is taken at state $s \in \mathcal{S}$ [6]. Given a state $s \in \mathcal{S}$, the agent takes action $a \in \mathcal{A}$, receives a reward $r \in \mathbb{R}$, and transitions to the next state $s' \in \mathcal{S}$ with regards to the probability distribution $p \to [0, 1]$, represented by Equation 1:

$$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}. \quad (1)$$

When a policy π interacts with an MDP $\mathcal{M}$, we say that π induces a Markov process over the states, actions, and rewards of $\mathcal{M}$ over which we can expect a discounted return as defined by the value function $v_\pi(s)\ \forall s \in \mathcal{S}$.

The options framework is a set of methods for working with temporal abstraction in RL and can be described as closed-loop policies that focus on taking high-level actions that span over an extended period of time [22]. For instance, when planning a vacation, a human would normally think of steps like searching for a destination, booking a flight, or checking hotel availability. However, as MDPs are only capable of working with discrete time steps, each of these actions would actually need to be represented as a series of muscle twitches. By instead learning high level skills—sequences of actions that can take a non-deterministic number of steps to complete—as separate policies, an agent would only have to iterate over the set of high-level tasks to plan ahead. These temporally extended courses of actions are known as *options.* The authors build on the theory of semi-Markov decision processes (SMDPs), a variation of the regular MDP that is capable of modeling continuous-time discrete-event systems—in other words, SMDPs allow a model to support temporally-extended courses of action. A fixed set of options O defines a discrete-time SMDP embedded within the original MDP, where the base system is an MDP with single-step transitions and the options define potentially larger SMDP-like transitions.

An option can be represented by the triple $\langle \mathcal{I}, \pi, \beta \rangle$, where $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ represents its internal policy, $\mathcal{I} \subseteq \mathcal{S}$ is the initiation set denoting in what states the option may be initiated, and $\beta : \mathcal{S}^+ \to [0, 1]$ is the option's termination condition, where $\mathcal{S}^+$ represents the set of states plus the terminal state, if the latter exists. After a state transition, the option can either terminate with probability $\beta(s_{t+1})$ or continue its execution by determining a_{t+1} from $\pi(s_{t+1}, \cdot)$, transitioning to s_{t+2}, possibly terminating with probability $\beta(s_{t+2})$, and so on. Once an option terminates, the agent chooses a new option from the set of available options for the current state, defined by O_s. It is convenient to extend O_s for every $s \in \mathcal{S}$ to also cover the primitive actions in the set $\mathcal{A}_s$, which can be represented as a special type of option that can be initiated anywhere, only lasts a single time step, and always chooses a: $O^a_s = \langle \mathcal{I} : \mathcal{S},\ \pi : a\ \forall s \in \mathcal{S},\ \beta : 1\ \forall s \in \mathcal{S} \rangle$. By extending the option set, we have $|O| = |\textit{dynamic options}| + |\mathcal{A}|$. Although the original options framework does not touch on the subject of automatic option discovery, the authors define the concept of *subgoals* by assigning a *terminal subgoal value* $g(s)$ for each state to indicate how desirable it is for the learned option to terminate at s.

The Fast Planning Option-Critic algorithm introduced in Section 2 presents a tabular approach to multi-task option discovery, where the objective becomes solving a finite set of episodic tasks $\mathcal{N}$, all of which share the same state and action spaces $\mathcal{S}$ and $\mathcal{A}$, respectively [23]. The authors propose using interest functions as a replacement for the option's initiation set, which allows an option's initiation to be stochastic and thus more selective. An option using interest functions is defined as the tuple $\langle i_o, \pi_o, \beta_o \rangle$, where $i_o \in \Gamma$ is the option's interest function, $\Gamma : \{f \mid f : \mathcal{S} \to [0, 1]\}$. At each state, its initiation set is determined by sampling the interest functions for every option initializable in the current state: $\Pr(o \in \Omega(s)) = i_o(s)$.

4 DYNAMIC OPTION CREATION

The proposed algorithm consists of two major parts: option evaluation via uncertainty, and option initialization with selective experience replay. The former, expanded in Section 4.1, uses the variance in an option's returns as a measure of uncertainty, where options that struggle to learn specific behavior tend to consistently yield high variances. The latter, explained in Section 4.2, occurs when uncertainty crosses a defined threshold, which triggers the creation of a new option using selective experience replay with a focus on states where existing options fared poorly—that is, experience is not applied if the policy over options is more likely to choose an existing option over the newly created option. Figure 1 presents a diagram of the Dynamic Option Creation algorithm acting over the Fast-Planning Option Critic [23].

4.1 Defining Option Uncertainty

The return of an option has variance. This variance is influenced by the stochasticity of the environment, the stochastic nature of the option's policy, and the ongoing updates to the policy during the learning process—which may cause it to produce different returns even within the same episode. The first component affecting return variance—environmental stochasticity—cannot be controlled by the agent and is determined solely by the dynamics of the MDP itself. We refer to this as *exogenous variance*, as it arises from factors outside the agent's control.

During the learning process, policies (such as a meta-policy or an option's policy) are typically initialized randomly, e.g., by initializing an actor network with random weights. This process

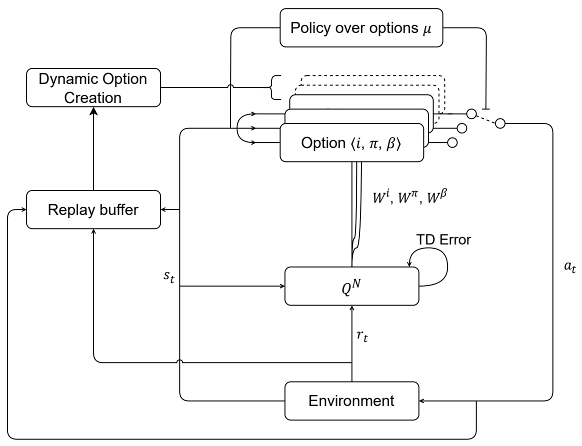


Figure 1: Overview of the agent's learning process with Dynamic Option Creation.

will temporarily result in increased return variance as the policy is still being updated frequently and significantly in the early stages of training. As the policy is updated towards convergence, the variance in returns will naturally decrease. For these reasons, we posit that variance in an option's return can serve as a proxy for epistemic uncertainty. Higher return variance often indicates that the agent has not yet sufficiently interacted with the system for an option's policy to converge to an effective behavior. As a result, returns tend to show higher variance early in training and when the option's policy is not yet reliable—both due to inherent policy stochasticity and to the frequent, impactful policy updates occurring when the option performs poorly and is far from convergence.

We propose a novel approach to assess the quality of an option by observing its variance over a specified time period. After n episodes, the training process is interrupted and an evaluation process is executed for m episodes, where the agent always acts greedily. Let $\mathcal{P} = 0, 1, 2, \ldots, m$ be the set of all evaluation episodes. We can define $\mathcal{V}_o \doteq \{\sigma^2_{R^o_p} \mid O_p = o,\ p \in \mathcal{P}\}$ as the variance in accumulated rewards for option o, where R^o_p is the accumulated reward for every step in episode p where the agent chose option o. If we repeat the evaluation process enough times, we can build a set of variances denominated $\varsigma_o \doteq \left\{\mathcal{V}^1_o, \mathcal{V}^2_o, \mathcal{V}^3_o, \ldots\right\}$ containing the variances of multiple evaluation processes. By applying a definite integration over ς_o, we can observe the increase in variance over time. Figure 2a uses FPOC to demonstrate the variance of a single dynamic option in a 13×13 four-rooms gridworld [22], where every empty space represents a possible goal for any given episode (further details about the environment are provided in Section 5).

As we can see, the variance is highest at the first half of training, where the agent is still learning the environment. The definite integral represents the area between the x axis and the variance line and behaves the same way as the variance, except it provides a much clearer representation of growth over time. When we compare the integral with a linear function, we can see that some stages of learning are below the line, while others are above it. If we define the slope of this linear function as a threshold, we can use it to determine whether the variance is continuously higher than desired. We can determine the integral's slope for a given window of data by first applying a linear regression to the values inside it, which gives us a linear function in the form of $y = w_1 x + w_0$. We can then derivate the result relative to x and obtain w_1, which represents the slope in the regression. When applying this logic to a small enough window, the loss in precision is marginal, as shown in Figures 2b and 2c.

We can define the definite integral of ς_o by using the trapezoidal rule and then applying linear regression to its result, thus obtaining a linear function:

$$\operatorname{lsreg}\left(\sum_{k=1}^{|\varsigma_o|} \frac{\varsigma_{o_{k-1}} + \varsigma_{o_k}}{2}\right) \approx w_1 x + w_0 \tag{2}$$

where $\operatorname{lsreg}(x)$ is the linear regression over x.

Definition 1. *The uncertainty factor ε_o for the current option is defined by the derivative obtained from the regression in Equation 2, multiplied by a discount factor:*

$$\varepsilon_o \doteq (\, e\,|O|\,)^{-\frac{2}{f}} \left[\frac{d}{dx} \quad w_1 x + w_0\right], \tag{3}$$

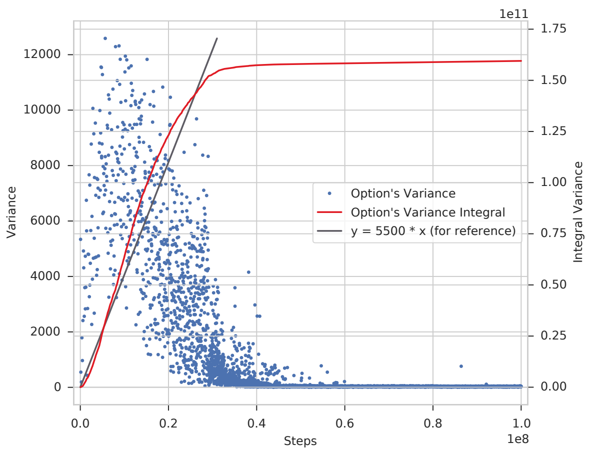
where the discount factor $(\, e\,|O|\,)^{-\frac{2}{f}}$ is used to prevent options from being created too frequently, being f the number of times the algorithm has calculated the uncertainty factor but did not create a new option, and $|O|$ the number of existing options.

The uncertainty factor is the value compared against the threshold for deciding whether a new option is necessary. If the factor is above the threshold, it means that the variance's integral is continuously high.

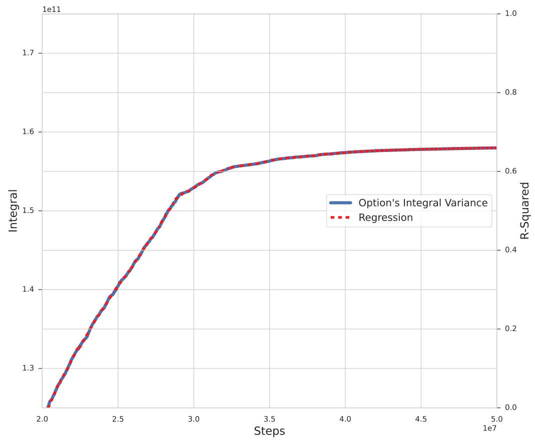
4.2 Initializing New Options

Once the algorithm has decided that creating a new option will benefit the learning process, it must initialize the new option in a way that complements existing ones. In other words, the new option must focus on where previous options struggle. We initialize a new option o^+ with pessimistic (negative) values for the option-value function q_{o^+} with the goal of discouraging exploration. Given that we want the new option to focus solely on states where previous options fail, exploration can lead o^+ to *copy* behavior already learned by other options. A pessimistic initialization, therefore, helps in giving more attention to state spaces whose return values are worse, leading to a more focused option. We also set the interest function for o^+ as $i_{o^+}(s) = -\sum_{o}^{O \setminus o^+} i_o(s)$, for all $s \in \mathcal{S}$. Intuitively, this ensures that the new option has *(1)* greater interest in states where the existing options have low interest—i.e., states where the existing options are likely to perform poorly; or *(2)* conversely, lower interest in states where existing options have high interest—i.e., states where they are likely to perform well.

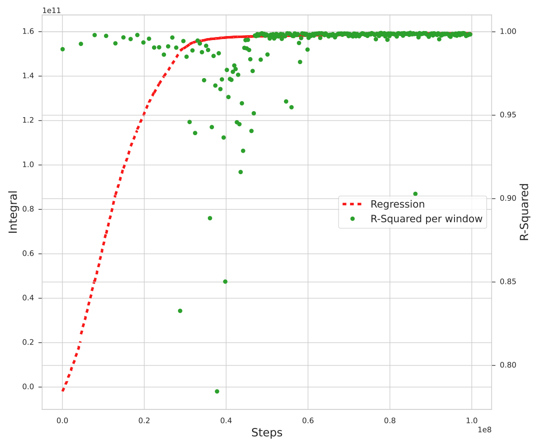
After conditioning o^+ to give higher emphasis towards state spaces where previous options perform poorly—pessimistic initialization of q_{o^+}—or are not covered by them—conditioning of i_{o^+}—we apply experience replay to the new option before resuming training.



(a) Option variance (blue) and its integral (red) in a sample environment, compared to a linear slope (gray).



(b) Option's integral variance (blue) and its linear regression (red) with a window size of 50 points



(c) Linear regression (red) and the R^2 for each window (green)

Figure 2: Obtaining the slope from the variance's definite integration.

Let us define the replay buffer B as an array that stores the history of the previous 50,000 episodes. Each element in the buffer represents a full episode, and is comprised of two values: the episode's history, and the episodic return. The former is a list of tuples $\langle S, r, S', \perp \rangle$ representing each step in the elapsed episode, where $\perp = 1$ if the step is terminal and 0 otherwise. The latter, formalized as $G_{1:\tau}$, represents the sum of all returns gained in the current episode, starting from the initial step $t = 1$ until termination $t = \tau$ if $\perp_\tau = 1$. The replay buffer can be formalized as:

$$
\begin{aligned}
B &\doteq \left\{ \left\{ \{S_E^1, S_E^2, S_E^3, \ldots, S_E^t\}, G_{1:t} \right\}^E \;\middle|\; E \in \mathcal{E} \right\} \\
S_E^i &\doteq \langle S_i, r_i, S_{i+1}, \perp_i \rangle,
\end{aligned}
\tag{4}
$$

where E is one episode in the set of all experience replay episodes $\mathcal{E}$, each containing one or more steps S_E, and $G_E \doteq \sum_{i \in E} r_i$ is the sum of rewards in each episode E.

For each step recorded in the buffer, we use the same option selection technique used in training to check whether the algorithm would select the newly created option. In other words, for every step in an experience replay episode, we run the option selection algorithm as if in training; if the option with highest likelihood of being selected is o^+, then we apply the experience step, otherwise, we skip to the next step. This process avoids training the new option with behavior that is already handled by previous options. Instead, we shift the new intra-option policy towards covering state spaces less visited by previous options.

FPOC uses the concept of interest functions to reduce the number of options being considered at any given step, thus helping achieve its goal of faster planning. Once the initiation set is determined by sampling the options' interest functions, the best option is selected with regards to its Q value [23]. We retain the same option selection process used by FPOC in DOC, as our main focus is in creating options. Algorithm 1 shows the complete algorithm for DOC.

5 EXPERIMENTAL EVALUATION

The main goal of Dynamic Option Creation is to provide comparable learning capabilities to its base algorithm—FPOC, in this paper—while reducing the need for retraining by autonomously determining the right number of options for the given environment. With this in mind, we devised a benchmark to compare DOC against the base FPOC algorithm [23] in the four rooms environment [22]. Although simple, we chose this environment as it is the most common benchmark for assessing tabular option discovery methods. More specifically, we compare both algorithms to assess the following hypotheses:

(1) DOC reaches similar maximum accumulated reward and learning speed when compared to FPOC;
(2) DOC creates new options only when necessary, that is, all options have a policy that represents meaningful behavior, with no noticeable overlap between them.

The four rooms environment, as the name implies, is composed of four interconnected rooms in a square grid, where each room has a 1×1 gap connecting it to its neighbor. Any tile that is not a wall is a place where the goal can spawn at the beginning of an episode. When an episode starts, the agent and goal are randomly placed in the environment. When the agent reaches the goal state, the episode

Algorithm 1: Dynamic Option Creation.

```
Input: Replay buffer size $z$, option creation threshold $L$,
       option adjust rate $n$, option evaluation episodes $m$,
       maximum training episodes $ep_{max}$, option consider
       rate $\tau$
Initialize $ep,\ t,\ \perp,\ n_{elapsed},\ f \leftarrow 0$;
Initialize $\varsigma_o, B \leftarrow \{\emptyset\}$;
while $ep < ep_{max}$ do
    if $\perp = 1$ then
        $n_{elapsed} \leftarrow n_{elapsed} + 1$;
        $B \leftarrow \{B \cup \{\langle S_i, r_i, S_{i+1}, \perp_i\rangle\ \forall i \in 1{:}t\},\ G_{1:t}\}$, drop
          first element if $|B| > z$ ;
        $t \leftarrow 0$;
    end
    if $n_{elapsed} = n$ then
        Run evaluation for $m$ episodes, store variance in $\varsigma_o$;
        if $|\varsigma_o| = \tau$ then
            foreach $o \in O$ do
                $\varepsilon_o \leftarrow (\ e\,|O|\,)^{-\frac{2}{f}} \left[\frac{d}{dx} w_1 x + w_0\right]$;
                if $\varepsilon_o > L$ then
                    Create new option $o^+$;
                    foreach $S_E^i = \langle S_i, r_i, S_{i+1}, \perp_i\rangle \in B$ do
                        $o \leftarrow$ best option for $S_i$;
                        if $o = o^+$ then
                            Apply experience step $S_E^i$ in $o^+$;
                        end
                    end
                    $f \leftarrow 0$;
                    Jump to line 29;
                else
                    $f \leftarrow f + 1$;
                end
            end
        end
        $n_{elapsed} \leftarrow 0$;
    end
    $S_t, A_t, r_t, \perp, S'_t \leftarrow$ Run regular training step for $t$;
    $t \leftarrow t + 1$;
end
```

terminates, and a new episode is randomly generated. The agent has four primitive actions available: *up*, *down*, *left*, *right*. When the agent passes through a *door*—any tile that divides two rooms—it receives a reward of −20; for any other action that does not terminate the episode, the agent receives a reward of −1. The agent receives a reward of 0 when it reaches the goal state. This subtle variation in dynamics from the traditional four rooms environment has the intention of further penalizing the agent for taking the longer route towards the goal, increasing the need for more than one option.

We compare FPOC and DOC using different parameters for number of options and threshold, respectively. Each experiment was

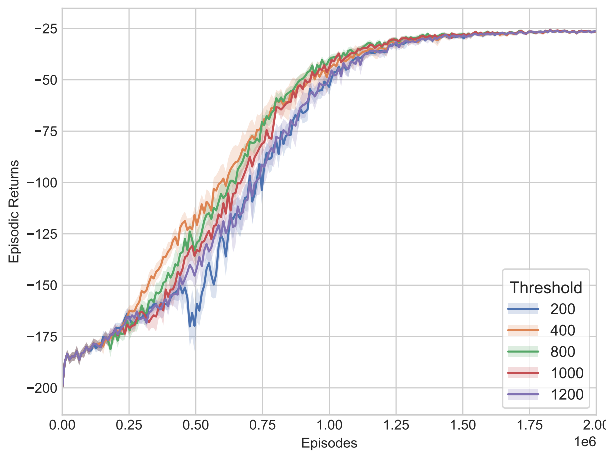


Figure 3: Average episodic return over time for Dynamic Option Creation with different threshold values.

executed 30 times with the same set of random seeds. All experiments consisted of 5,000,000 training episodes, although no method took more than 1,300,000 episodes to reach maximum return. After the training procedure was completed, we calculated the moving average of the mean accumulated reward for every 10,000 episodes, using a window size of 30 data points. We use the moving average as a way to reduce the variability in mean episodic returns, thus helping better visualize the learning behavior. In the evaluation process, we measure both the mean accumulated reward once the training process completes, as well as the average number of episodes each algorithm took to converge to the maximum return value observed in the experiments. Section 5.1 analyzes the sensitivity of the threshold parameter, describing how much it impacts learning. Meanwhile, Section 5.2 compares the accumulated reward over time and learned options for both FPOC and our method.

5.1 Threshold Sensitivity

The threshold is the most important parameter when configuring Dynamic Option Creation, given that it indicates how much variance over time is acceptable before initializing a new option. If the threshold is set too low, the algorithm may create options unnecessarily, while setting it too high can hinder learning performance. Figure 3 demonstrates the mean episodic returns using DOC with different threshold values.

As shown in Figure 3, the threshold parameter can have a perceptible impact on learning performance and in the number of created options. Regardless, the algorithm was able to eventually reach the maximum return for all threshold values. We cannot, however, ensure the same behavior will occur in more complex environments. The learning curves for threshold $L = 400$, $L = 800$, and $L = 1000$ present similar behavior, which can be attributed to the fact that both resulted in 2 options at the end of training for all 30 runs. Meanwhile, $L = 1200$ oscillated between 2 and 3 options depending on the seed, while $L = 200$ always resulted in 4 options.

Table 1: Average number of episodes until reaching a mean accumulated reward of -26 for FPOC and Dynamic Option Creation with different threshold values. Methods sorted by this average.

Algorithm	*Avg. Episodes to Convergence*	*Std Error*
DOC (L=800)	**1,078,092.86**	**9,209.72**
DOC (L=400)	1,101,214.29	23,248.26
DOC (L=1000)	1,107,321.43	9,569.50
DOC (L=200)	1,152,632.14	8,818.80
DOC (L=1200)	1,171,682.14	11,673.91
FPOC (4 Options)	1,185,590.00	9,176.59
FPOC (2 Options)	1,236,082.76	10,528.60
FPOC (1 Option)	1,288,789.29	7,668.41

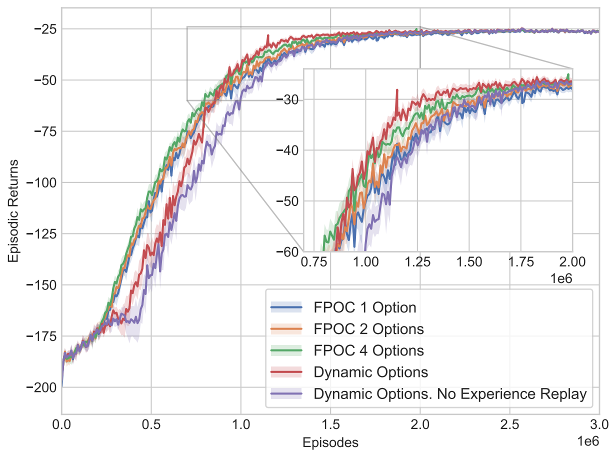


Figure 4: Evaluation curves for FPOC with 1, 2, and 4 options and Dynamic Option Creation.

Whenever a new option is created, the episodic returns tend to stagnate or decrease for some episodes as the new option gets introduced into the training process. In general, the sooner an option is introduced, the smaller the impact in learning performance is. As evidenced by the comparison above, however, increasing the threshold by six times did not have an impact on maximum accumulated reward. Table 1 shows how many episodes on average each threshold value took to reach a mean accumulated reward of -26 over 10,000 episodes, which is close to the upper bound reached by all algorithms in this experiment.

5.2 Numerical Results

Figure 4 shows the accumulated reward for FPOC with 1, 2, and 4 dynamic options alongside Dynamic Option Creation with and without experience replay using a threshold of 800, the best value found in Section 5.1. The shaded area around each line represents the variability in results between different runs of the same experiment.

Table 2: Mean (and standard deviation) episodic return over the last 1000 episodes for FPOC with 1, 2, and 4 options, compared to Dynamic Option Creation with and without experience replay.

Algorithm	*Performance at Convergence*
DOC	-25.85 (± 1.72)
DOC (No Experience Replay)	-25.87 (± 1.74)
FPOC (1 Option)	-25.86 (± 1.73)
FPOC (2 Options)	-25.91 (± 1.73)
FPOC (4 Options)	-25.94 (± 1.73)

As we can see in Figure 4, DOC with experience replay, despite learning slower than FPOC at first, managed to achieve maximum reward before FPOC with any number of options. In fact, all DOC variations analyzed in Table 1 converged faster than FPOC. The sudden increase in learning speed can be attributed to the creation of a new option with experience replay, which helps condition it to a specific state space, instead of deferring this process to the regular training loop. Meanwhile, the slower learning speed at the beginning of training happens because, while DOC is undergoing the cost of determining how many options are necessary, FPOC already starts with knowledge about the number of options it will use. Table 2 compares the mean episodic returns achieved by all algorithms once the training completes.

5.3 Learned Policies

Figure 5 shows the learned intra-option policies for each dynamic option alongside their interest and termination functions for two different goal positions. In the figure, the goal positions are denoted in green. In both experiments, two options were obtained.

By observing the policies on the left, we can see that DOC created a second option that focuses on navigating towards the goal room, while keeping the first option responsible for reaching the goal from inside the room itself. The observed behavior can be considered meaningful due to the fact that both policies represent some well-defined course of action, thus confirming our second hypothesis. Furthermore, by looking at the interest functions of both options, we can see that the only overlap present in their initialization is at the top-left corner of the goal room, which also corroborates our second hypothesis.

When the goal is located at a door, we can see a slightly different, but consistent, result. Instead of splitting behavior between walking towards the goal room and moving to the goal from inside the room, each option covers a different half of the environment. We attribute this behavior to the fact that the goal is between two rooms, so we can consider both the top and bottom-left rooms as the goal room. When looking from this perspective, the option on the left focuses on navigating towards the goal, while the option on the right goes towards the goal room, which is the same behavior observed when the goal is inside a single room.

6 CONCLUSION

This work proposed a novel method for dynamically creating options in training time in option-critic algorithms. Our approach

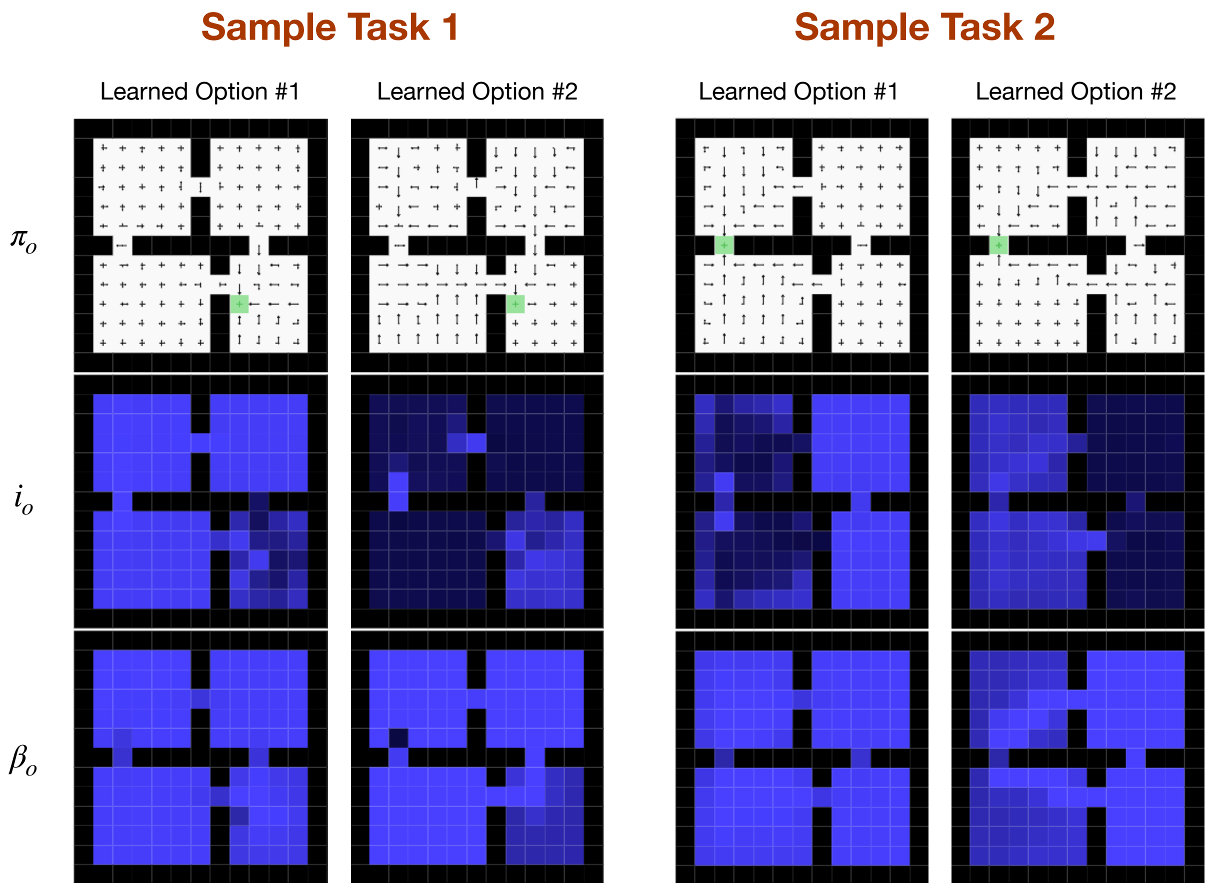


Figure 5: Policies learned by the Dynamic Option Creation algorithm in two sample tasks. Each task has a different goal position. Our algorithm learned two options in each of these tasks. The images show the intra-option policy π_o, the option's interest functions i_o, and the option's termination functions β_o.

uses the variance in episodic returns as an indication of uncertainty, which—assuming a deterministic environment—tends to decrease as the intra-option policy converges to its optimality. When compared to the Fast-Planning Option-Critic [23], our approach tends to learn options only as necessary without a significant increase in training steps needed for convergence. We consider learning options dynamically as a vastly superior approach, given that complex problems heavily benefit from more options, but estimating the best number for an environment is not possible most of the time. Dynamic Option Creation has the potential to reduce re-training efforts by guessing the correct number of options in the first try, thus making the Option-Critic architecture more robust.

Despite the benefits mentioned above, our method has limitations that are not handled in the scope of the current work. The most noteworthy shortcoming is the reliance on a human-provided uncertainty threshold, which can severely hinder learning if set incorrectly, thus nullifying the benefit added by our algorithm. Future work should focus on determining such threshold based on parameters gathered from the training process or the environment itself. Another opportunity for improvement lies in adapting and testing the proposed algorithm with function approximation Reinforcement Learning, where we believe Dynamic Option Creation could bring the greatest benefit given such model's aptitude for learning good policies for complex environments. Lastly, future work should assess the viability of DOC in environments with high amounts of exogenous variance.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback. This research was partially supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (grants 443184/2023-2, 313845/2023-9, and 445238/2024-0).

REFERENCES

[1] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The Option-Critic Architecture. *Proceedings of the AAAI Conference on Artificial Intelligence* 31, 1 (Feb 2017). https://doi.org/10.1609/aaai.v31i1.10916

[2] Andre Barreto, Will Dabney, Remi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. 2017. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/350db081a661525235354dd3e19b8c05-Paper.pdf

[3] Raviteja Chunduru and Doina Precup. 2022. Attention Option-Critic. *arXiv preprint arXiv:2201.02628* (Jan 2022). http://arxiv.org/abs/2201.02628

[4] Tom Croonenborghs, Kurt Driessens, and Maurice Bruynooghe. 2008. *Learning Relational Options for Inductive Transfer in Relational Reinforcement Learning.* Lecture Notes in Computer Science, Vol. 4894. Springer Berlin Heidelberg, Berlin, Heidelberg, 88–97. https://doi.org/10.1007/978-3-540-78469-2_12

[5] Dongge Han and Sebastian Tschiatschek. 2022. Option Transfer and SMDP Abstraction with Successor Features. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence.* International Joint Conferences on Artificial Intelligence Organization, Vienna, Austria, 3036–3042. https://doi.org/10.24963/ijcai.2022/421

[6] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. 2018. When Waiting Is Not an Option: Learning Options With a Deliberation Cost. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (Apr 2018). https://doi.org/10.1609/aaai.v32i1.11831

[7] Anna Harutyunyan, Will Dabney, Diana Borsa, Nicolas Heess, Remi Munos, and Doina Precup. 2019. The Termination Critic. arXiv:1902.09996 [cs.AI] https://arxiv.org/abs/1902.09996

[8] Arushi Jain, Khimya Khetarpal, and Doina Precup. 2021. Safe Option-Critic: Learning Safety in the Option-Critic Architecture. *The Knowledge Engineering Review* 36 (2021), e4. https://doi.org/10.1017/S0269888921000035

[9] Martin Klissarov and Marlos C. Machado. 2023. Deep Laplacian-based Options for Temporally-Extended Exploration. *arXiv preprint arXiv:2301.11181* (Jan 2023). http://arxiv.org/abs/2301.11181

[10] Vijay R Konda and John N Tsitsiklis. 1999. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, Vol. 12. MIT Press.

[11] George Konidaris and Andrew Barto. 2007. Building Portable Options: Skill Transfer in Reinforcement Learning. In *Ijcai*, Vol. 7. 895–900.

[12] Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. 2017. A Laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th international conference on machine learning (Proceedings of machine learning research, Vol. 70).* PMLR, 2295–2304. https://proceedings.mlr.press/v70/machado17a.html

[13] Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. 2018. Eigenoption Discovery through the Deep Successor Representation. *arXiv preprint arXiv:1710.11089* (Feb 2018). http://arxiv.org/abs/1710.11089

[14] Sridhar Mahadevan and Mauro Maggioni. 2007. Proto-value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes. *Journal of Machine Learning Research* 8, 10 (2007).

[15] Amy McGovern and Andrew G Barto. 2001. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. *Computer Science Department Faculty Publication Series. 8.* (2001).

[16] Ishai Menache, Shie Mannor, and Nahum Shimkin. 2002. Q-Cut—Dynamic Discovery of Sub-goals in Reinforcement Learning. In *Machine Learning: ECML 2002*, Gerhard Goos, Juris Hartmanis, Jan Van Leeuwen, Tapio Elomaa, Heikki Mannila, and Hannu Toivonen (Eds.), Vol. 2430. Springer Berlin Heidelberg, Berlin, Heidelberg, 295–306. https://doi.org/10.1007/3-540-36755-1_25

[17] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16).* JMLR.org, 1928–1937.

[18] Özgür Şimşek and Andrew Barto. 2008. Skill characterization based on betweenness. In *Advances in neural information processing systems*, Vol. 21. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2008/file/934815ad542a4a7c5e8a2dfa04fea9f5-Paper.pdf

[19] Martin Stolle and Doina Precup. 2002. Learning Options in Reinforcement Learning. In *Abstraction, Reformulation, and Approximation*, G. Goos, J. Hartmanis, J. Van Leeuwen, Sven Koenig, and Robert C. Holte (Eds.). Vol. 2371. Springer Berlin Heidelberg, Berlin, Heidelberg, 212–223. https://doi.org/10.1007/3-540-45622-8_16

[20] Richard S. Sutton and Andrew Barto. 2018. *Reinforcement learning: an introduction* (second edition ed.). The MIT Press, Cambridge, Massachusetts London, England.

[21] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in neural information processing systems* 12 (1999).

[22] Richard S. Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112, 1–2 (Aug 1999), 181–211. https://doi.org/10.1016/S0004-3702(99)00052-1

[23] Yi Wan and Richard S. Sutton. 2022. Toward Discovering Options that Achieve Faster Planning. *arXiv* arXiv:2205.12515 (Sep 2022). http://arxiv.org/abs/2205.12515 arXiv:2205.12515 [cs].