Improving Option Learning with Hindsight Experience Replay

Gabriel Romio
gromio@edu.unisinos.br
Univesidade do Vale do Rio dos Sinos
São Leopoldo, RS, Brazil

Mateus Begnini Melchiades
mateusbme@edu.unisinos.br
Univesidade do Vale do Rio dos Sinos
São Leopoldo, RS, Brazil

Gabriel de Oliveira Ramos
gdoramos@unisinos.br
Univesidade do Vale do Rio dos Sinos
São Leopoldo, RS, Brazil

ABSTRACT

Algorithms such as Option-Critic (OC) [2] and Multi-updates Option Critic (MOC) [6] have introduced significant advancements in the discovery and autonomous learning of options. However, these methods still tend to underperform in multi-goal environments or those with sparse rewards. In this work, we propose the integration of Hindsight Experience Replay (HER) [1] into MOC to enhance performance in these scenarios. To achieve this, the algorithm selects new goals based on previously reached states. The rewards for already completed iterations are then recalculated, leveraging even unsuccessful trajectories as if the intended objective had been achieved. Our method, which we refer to as MOC-HER, successfully solved multi-goal environments with sparse rewards, where traditional Hierarchical Reinforcement Learning algorithms failed. Additionally, when testing our algorithm in the same environments with dense rewards, we observed significant improvements over the original MOC.

KEYWORDS

Reinforcement Learning, Sparse Rewards, Multi-Goal Environments, Options Framework, Temporal Abstraction

1 INTRODUCTION

The concept of Hierarchical Reinforcement Learning aims to structure complex tasks into more manageable sub-tasks [9]. In this context, the Options framework enables the reuse of learned skills between different problems, thus accelerating the training process [9]. These options are closed-loop policies capable of executing a complete action over an extended period of time. Examples of options include tasks like picking up an object, or traveling from one city to another, while primitive actions are more fundamental operations, such as muscle twitches or applying electrical current to a motor.

Recently, methods for automatically discovering and learning options have been developed to improve and accelerate training, with a focus on generating and diversifying options. The automatic search for options was enhanced with the Option-Critic framework [2], while Asynchronous Advantage Option-Critic (A2OC) [4] implemented the `deliberation cost` to improve and bring greater stability to the use of options. Proximal Policy Option-Critic (PPOC) [5] was one of the first to achieve good results in tasks with continuous action spaces. The Multi-updates Option Critic (MOC) framework [6] introduced simultaneous updates of various options, significantly improving performance in high-complexity environments. This method has also proven effective in preventing options with degenerate solutions. More recently, Dynamic Option Creation [7] proposes an approach for dynamically creating new options during the training process. However, high-complexity tasks with sparse rewards remain a significant challenge in the context of Hierarchical Reinforcement Learning. The Hindsight Experience Replay (HER) algorithm [1], by recalculating rewards from failed trajectories through a replay buffer, introduced significant improvements in handling sparse rewards in multi-goal environments.

Following this concept, we extend MOC with a HER-based function to handle intra-option learning within the MOC algorithm. This method aims to improve or enable the discovery of options in these environments in a straightforward approach. To achieve this, we designed a replay buffer to store information about each state transition from the original MOC, including the active option. These states are then re-evaluated with a new goal, selected by one of the states reached by the algorithm. The trajectory is reassessed as if the objective had been to reach that state from the beginning, thus the experience can be utilized even if the trajectory did not achieve its original goal. Each state stored in the HER buffer has its reward value recalculated based on the new goal. The HER buffer is subsequently integrated into the MOC training buffer, becoming part of the evaluation and learning phases of the algorithm. After each learning step, the HER buffer is cleared. We refer to this enhanced framework as Multi-Updates Option Critic with Hindsight Experience Replay (MOC-HER).

The main contribution of this work is the introduction of a novel approach designed to solve multi-goal environments with sparse rewards within the context of Hierarchical Reinforcement Learning. Experimental evaluation shows that the combination of MOC and HER methods leads to improved results. Specifically, our findings show that the incorporation of HER enabled the resolution of sparse reward environments that remain intractable with standard MOC, achieving a reduction of up to 96.5% in negative reward in tested cases. Additionally, even in environments with dense rewards, applying HER significantly improved both learning speed and final performance. This resulted in a decrease of up to 38.1% in negative reward in tested cases as compared to the original algorithm.

2 BACKGROUND

2.1 Options

A Markov Decision Process (MDP) is traditionally based on the principle that an agent takes an action a at some state s, taking a fixed time t and affecting the next state s' and reward r at the subsequent time step $t + 1$. Therefore, each MDP consists of a set of states S, a set of actions A, an expected reward r_s^a for each state-action pair, a transition probability over states P, and a policy π. In contrast, Semi-Markov Decision Processes (SMDPs) [9] offer a framework to incorporate temporal abstraction within MDPs in Reinforcement Learning. Temporal abstraction involves considering actions or

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.

sequences of actions across multiple time steps, treating them as larger units that can be handled as single actions. Thus, SMDPs are a variation of MDPs designed to model discrete-event systems in continuous time, where actions have variable durations. In other words, SMDP models represent temporally extended courses of action. Each sub-policy representing one of these sets of actions, available to the agent during training, is called an option.

The concept of options [9] involves extending a set of primitive actions to include more elaborate courses of action. Then, an agent can consider to build a policy to select options instead of primitive actions. Each option consists of a policy $\pi_o : S \times A \rightarrow [0,1]$, a termination condition $\beta : S^+ \rightarrow [0,1]$, and an initiation set $I \subseteq S$. Thus, an option (I, π, β) is available at a state s_t only if that state belongs to I. Once an option is initiated, actions are chosen according to policy π until termination as sampled by β, at which point the agent can choose another option from those available for $s_t + 1$.

2.2 Discovering and learning options

The Option-Critic architecture [2] introduced a method of leveraging function approximation to autonomously discover and learn options from scratch. This framework builds on the policy gradient theorem to learn internal policies and termination conditions of options without the need for additional rewards or sub-goals. However, the Option-Critic architecture updates only the option active at any given time step. This limitation becomes particularly problematic in more complex situations such as environments with continuous action spaces. These cases often result in suboptimal solutions or degenerate options, where certain options are selected and updated at a significantly higher rate than others.

2.3 Multi-updates Option Critic

The Multi-updates Option Critic (MOC) algorithm [6] is based on intra-option learning and aims to update all relevant options for a given state without the need for additional estimators. Instead of updating only the chosen option at a given state using the current policy, MOC proposes updating all options that could potentially be selected within that state. This results in temporally extended options and prevents degenerate solutions, where certain options are chosen much more frequently than others.

The execution and selection process for options can be described as a *call-and-return* model. From a state s, the agent chooses an option o according to a policy over options μ. The action a is then determined by the corresponding intra-option policy $\pi_\zeta(a|s,o)$, parameterized by ζ. A termination function $\beta(s,o)$ defines the probability of ending the current option and selecting a new one. Along this process, the agent collects and evaluates experience about the environment, estimating a value function $V^\pi(s)$. Moreover, it provides an evaluation of the probability of each option being selected.

The MOC algorithm updates all relevant options simultaneously by adjusting the option value functions $Q_O(s, o; \theta)$, where θ represents the parameters to be adjusted. The learning step is performed using previously collected experience, which contains transition probabilities, selected actions, and their corresponding rewards. Options other than the sampled one are updated through an off-policy method using intra-option gradient rules, where the policy π_ζ is adjusted to improve the agent's performance based on past experiences. Additionally, the update process also takes into account the state-option occupation distribution, which reflects how frequently each state-action pair is visited by each option. The occupation distribution helps improve sampling efficiency and reduce the variance of estimates, as updates are weighted by the relevance of options within the observed states. The algorithm repeats this process of updating the option value functions and intra-option policies throughout training.

A potential issue with updating all options is the risk of reducing the diversity within the option set. To mitigate this, the MOC introduces a hyperparameter η, which controls the probability of updating all options. If η is sufficiently high, the algorithm will update all options. Otherwise, it will update only the sampled option.

2.4 Multi-Goal Reinforcement Learning

Multi-Goal Reinforcement Learning involves a set of continuous control tasks based on existing robotic hardware [8]. These tasks include sending an industrial robotic arm to a desired position or interacting with objects in the environment. Other tasks involve controlling an anthropomorphic robotic hand to move specific fingers or manipulate an object, such as a block or a pen. All of these environments feature a dynamic goal that changes after each episode, which is encoded in the observation space. Thus, the observation space contains, in addition to the current environment states, two new parameters: the `desired goal`, which represents the target destination that the agent needs to reach, and `achieved goal`, which represents the position the agent has reached at the current time step.

The rewards can be computed as dense or sparse. In the dense case, the reward is calculated by the environment based on the Euclidean distance between the object and the goal. In the sparse case, the agent receives a reward of 0 when the object is at the target location and -1 in all other situations. In the latter case most RL algorithms do not perform effectively, because the reward function is extremely sparse and provides limited information. One way to improve performance is through the implementation of Hindsight Experience Replay (HER) [1].

The HER is an algorithm designed to re-examine failed trajectories by redefining the desired goal. After completing an episode $s_0, s_1, \ldots, s_T$, each transition $s_t \rightarrow s_{t+1}$ is stored in a replay buffer. During policy updates, alternative goals are introduced into this buffer, defined through a strategy $\mathbb{S}$ for sampling goals for replay, which can be `final`, `future`, or `episode`. The `final` strategy selects the new desired goal based on the achieved position from the last state of the episode. In the `future` strategy, the goal is randomly chosen from any state between the current and final states of the episode. Finally, the `episode` strategy selects a random state from the current episode as the desired goal, which may be either later or earlier than the current state. The rewards for this interaction are recalculated as if the goal had always been the state of the new goal selected. This approach enables the agent to gain valuable insights about how to reach this state, which can be leveraged by any off-policy Reinforcement Learning algorithm.

3 METHOD

In this work, we incorporated a HER buffer into the MOC framework to improve its performance in multi-goal environments, particularly those characterized by sparse rewards. As described in the algorithm presented in Section 3.1, our approach involves recording data for each state transition into a replay buffer throughout the agent's interaction with the environment. This data includes the obtained rewards and the selected options. Subsequently, new goals are sampled from states reached by the agent. For each stored transition in the buffer, the reward is recalculated based on the newly defined goal. This allows failed trajectories to be used in the learning process as though the intended goal had been achieved. When starting the evaluation and learning steps of the algorithm, the HER buffer is merged with the buffer containing the original iteration trajectories, so that all data is utilized for training. After these steps, the boot buffers are emptied. A detailed explanation of the HER implementation in MOC is provided in Section 3.2.

Algorithm 1 Multi-updates Option Critic with Hindsight Experience Replay

```
1: input: $\mathbb{S}$ (Strategy for sampling goals for replay, e.g. $\mathbb{S}(s_0, \dots, s_T)$); $r(s, a, g)$
   (Function to recalculate the rewards)
2: Set $s \leftarrow s_0$
3: Choose $o$ at $s$ according to $\mu_z(\cdot|s)$
4: Initialize empty HER buffer $R$
5: repeat
6:     Choose $a$ according to $\pi_\zeta(a|s, o)$
7:     Take action $a$ in $s$, observe $s'$, $r$
8:     Sample termination from $\beta_\nu(s', o)$
9:     if $o$ terminates in $s'$ then
10:        Sample $o'$ according to $\mu_z(\cdot|s')$
11:    else
12:        $o' = o$
13:    end if
14:    Define previous option $\bar{o} = o$
15:    1. Hindsight Experience Replay:
16:    Obtain the transition $(s, o, a, r, s')$ from the latest step
17:    $s'_{\text{goal}} \leftarrow \mathbb{S}$(current transition), where $s'_{\text{goal}}$ is the goal component of $s'$
18:    Recalculate $r_{\text{her}}$ based on the function $r(s, a, g)$ for the new goal $s'_{\text{goal}}$
19:    Store transition $(s, o, a, r_{\text{her}}, s')$ in HER buffer $R$
20:    2. Evaluation step:
21:    for each option $\tilde{o}$ in the option set $O$ do
22:        $\delta \leftarrow \mathbb{E}[U^\rho|s, \tilde{o}] - Q_\theta(s, \tilde{o})$ where $U^\rho$ is an importance sampling
   weighted target
23:        $\theta \leftarrow \theta + p_{\mu,\beta}(\tilde{o}|s, \bar{o})\alpha\delta\phi(s, \tilde{o})$
24:    end for
25:    Repeat the evaluation step for HER transition $(s, o, a, r_{\text{her}}, s')$
26:    3. Improvement step:
27:    for each option $\tilde{o}$ in the option set $O$ do
28:        $\zeta \leftarrow \zeta + p_{\mu,\beta}(\tilde{o}|s, \bar{o})\alpha_\zeta \frac{\partial \log \pi_\zeta(a|s,o)}{\partial \zeta} Q_\theta(s, o, a)$
29:    end for
30:    $\nu \leftarrow \nu - \alpha_\nu \frac{\partial \beta_\nu(s',o)}{\partial \nu}(Q_\theta(s', o) - V_\theta(s'))$
31:    $z \leftarrow z + \alpha_z \beta_\nu(s', o) \frac{\partial \mu_z(o'|s')}{\partial z} Q_\theta(s', o')$
32:    Repeat the improvement step for HER transition $(s, o, a, r_{\text{her}}, s')$
33: until $s'$ is a terminal state
```

3.1 Algorithm

We implemented HER into MOC in a minimally intrusive approach to ensure all its benefits and improve its performance in sparse reward environments. A description of our implementation is provided in Algorithm 1, which extends the original MOC algorithm [6] with our HER integration. The algorithm outlines a simplified version of the process, focusing on a single episode to provide a clearer understanding of the overall approach. Our modifications to the original algorithm begin at line 1 and 4, where we introduce the initialization of the strategy $\mathbb{S}$ for sampling new goals for replay at each state along the trajectory $(s_0, \dots, s_T)$, the reward function $r(s, a, g)$, and the HER buffer R. Additionally, component 1 of Algorithm 1 (lines 15-19) includes the storage of transitions to new states in the replay buffer. This process includes the selection of a new goal, the calculation of the new reward based on this goal, and the storage of the transition with the updated values in the HER buffer (more details are provided in Section 3.2). This procedure is applied to each state transition within the episode. Finally, in lines 25 and 32, the evaluation and improvement steps are repeated for the corresponding transition from the HER buffer R, with the new goal and recalculated reward.

3.2 Implementing HER into MOC

In this section, we provide detailed information about the HER implementation. As previously mentioned, to improve MOC performance in multi-goal RL environments, we devise a replay buffer that is incremented after each time step. This buffer stores key information about the decisions and current state, which will be used for intra-option learning. This stored data includes the action performed by the agent, the option number that executed the action, the observed state, the resulting state, and the reward received following the action. To integrate the replay buffer into MOC learning in a minimally intrusive manner and preserve the algorithm's original structure, all state transitions are sequentially inserted into the HER buffer, maintaining the integrity of episodes.

At the end of each complete episode, the entire trajectory from the replay buffer is revisited. To define the new goal we use the strategy `future` from HER, adopted because it demonstrated the best performance in the original HER [1] and also in our preliminary experiments. In this strategy the goal is randomly selected from any state between the current and the final state of the actual episode. Furthermore, a new goal is chosen for each transition. The reward is then recalculated for each transition based on the new goal, utilizing the environment's reward function.

Finally, the revisited HER buffer is integrated into the buffer containing the experiences of the original MOC iterations. Once the desired batch size has been reached, the expanded buffer is used for intra-option learning during the improvement step. After this step, both buffers are cleared and the process is restarted throughout the training.

4 EXPERIMENTS

We now present empirical results to validate our method, aiming to show that MOC-HER: (1) handle multi-goal environments with sparse rewards that MOC is unable to solve, (2) improves MOC performance even in reward-dense environments, (3) maintains the quality of the options discovered by MOC, avoiding degenerate solutions.

4.1 Methodology

The goal of this work is to support the learning of options in environments with continuous action spaces and sparse rewards. To visualize the results of this implementation, we applied the MOC-HER algorithm to the FetchReach environment, available in Gymnasium Robotics [8], using 2, 4, 8 and 16 options.

The FetchReach is a multi-goal environment that requires an industrial robot to reach a pre-defined target location. The environment features a continuous action space with four possible actions, each ranging from -1 to 1. The observation space contains 16 dimensions, with 3 values representing the achieved goal and 3 more representing the desired goal, all of which can take values in the range $-\infty$ to ∞. In sparse reward mode, the reward is 0 when the robot reaches the goal, i.e., when the Euclidean distance between the robot's achieved goal and the desired goal position is less than 0.05. At all other positions, the reward is -1. This calculation, formulated by the Gymnasium Robotics environments [3], is expressed in Equation (1).

$$r_{sparse} = \begin{cases} 0, & \text{if} \quad \|\text{achieved\_goal} - \text{desired\_goal}\|_2 < 0.05 \\ -1, & \text{otherwise} \end{cases} \tag{1}$$

In dense reward mode, the reward is discounted based on the Euclidean distance between the robot's achieved goal and the desired goal, according to Equation (2). This formulation also follows the framework of the Gymnasium Robotics environments.

$$r_{dense} = - \|\text{achieved\_goal} - \text{desired\_goal}\|_2 \tag{2}$$

An episode consists of 50 time steps, with the goal location changing after each episode. Moreover, an episode never ends until the predefined time steps are completed, so the algorithm must not only guide the robot to the target position but also keep it there. The experiments were performed for 10 random seeds, where the mean obtained between them was recorded. It is important to emphasize that the reward values were calculated based only on the original trajectories, ensuring that HER does not lead to a misleading evaluation of the results.

To measure and compare the performance improvement of MOC-HER over MOC, we utilize the relative improvement (RI) metric. This metric quantifies the percentage reduction in negative reward achieved by our algorithm in each experiment, as detailed in Equation (3).

$$\text{RI} = \frac{r_{\text{MOC-HER}} - r_{\text{MOC}}}{|r_{\text{MOC}}|} \times 100\% \tag{3}$$

Because of the simultaneous update of multiple options, the MOC prevents low-quality options or those that are ignored after a training period. Thus, we also aim to demonstrate that all options are utilized throughout the interactions, so that the introduction of HER does not compromise this characteristic of the MOC. To ensure this, we generate and present the usage rate of each option in these trajectories.

4.2 Numerical results

Our first experiment consists of comparing the results of MOC-HER whith those obtained with the standard MOC in FetchReach with sparse rewards, as in Equation (1). This is shown in Figure 1. An iteration corresponds to the set of episodes that constitute each training batch used during learning. Each iteration contains 40 episodes, with 50 time steps per episode, totaling 2000 time steps per iteration. The shaded areas represent the standard deviation. The insights demonstrate a situation where the MOC algorithm, regardless of the number of options, failed to identify a policy capable of solving the environment during the analyzed iterations. Conversely, MOC-HER found a solution quickly. It is noted that the number of time steps to resolve the environment increases according to the number of options. In addition, when the number of options exceeds 8, the approach becomes increasingly unstable. The obtained results demonstrate the benefits of implementing the HER buffer and the potential advantages of trajectory reward recalculation.

Table 1: Rewards (and standard deviation) after 100 iterations in the FetchReach environment with sparse rewards, and relative improvement (RI) of MOC-HER compared to MOC

Algorithm	*2 options*	*4 options*	*8 options*	*16 options*
MOC-HER	**-1.71 (0.26)**	-1.93 (0.389)	-2.77 (1.2)	-5.38 (7.8)
MOC	-49.9 (0.094)	-49.88 (0.147)	-49.82 (0.179)	-49.88 (0.109)
RI	96.57%	96.13%	94.44%	89.21%

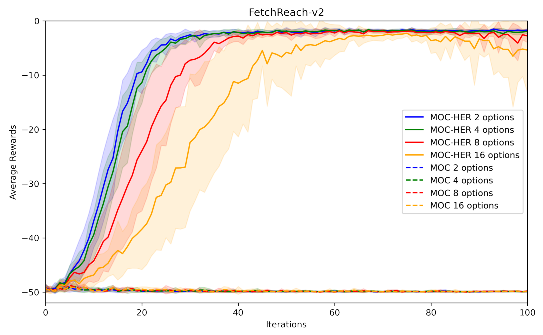

Figure 1: Results obtained for 2, 4, 8 and 16 options in the FetchReach environment with sparse rewards

Table 1 compares the results obtained by both algorithms at the end of each experiment with sparse rewards. It presents the numerical values, as well as the standard deviation, of the average rewards from the last iteration for each variation in the number of options. The last row reports the relative improvement of MOC-HER's results compared to MOC, calculated according to Equation (3). As the standard MOC failed to solve the environment, its rewards remained near the minimum possible value of -50. MOC-HER showed the best performance in the experiment with 2 options, surpassing MOC by 96.57% in terms of the RI metric. When the number of options is increased, its performance gradually decreased. With a high number of options, such as 16, there is a significant loss in quality and greater variation in the results, demonstrating the negative effects of having an excessive number of options.

Although the primary objective of this approach was to improve performance in sparse environments, performance improvements were also observed in multi-goal environments with dense reward

Table 2: Rewards (and standard deviation) after 100 iterations in the FetchReach environment with dense rewards, and relative improvement (RI) of MOC-HER compared to MOC

Algorithm	*2 options*	*4 options*	*8 options*	*16 options*
MOC-HER	**-0.89 (0.088)**	-0.93 (0.11)	-1.05 (0.087)	-1.33 (0.105)
MOC	-1.44 (0.056)	-1.49 (0.054)	-1.61 (0.074)	-1.84 (0.253)
RI	38.19%	37.58%	34.78%	27.61%

systems. We used the same FetchReach environment, but with dense rewards, where the reward is calculated as shown in Equation (2). In this scenario, MOC-HER outperformed the original MOC across all tested numbers of options, as illustrated in Figure 2.

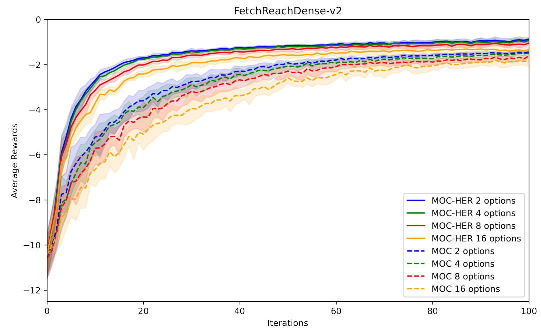

Figure 2: Results obtained for 2, 4, 8 and 16 options in the FetchReach environment with dense rewards

The numerical comparison between MOC and MOC-HER in the dense environment is presented in Table 2, which reports the mean reward of the last iteration along with the standard deviation. In this scenario, the improvements achieved by MOC-HER remain significant, ranging from 27.61% to 38.19%, as measured by the RI metric, with the best performance occurring when using the minimum number of options. Although performance decreases as the number of options increases for both algorithms, this decline was less evident and more gradual in the dense reward environment compared to the sparse reward case.

In this experiment the potential scalability of the algorithm is demonstrated, indicating that MOC-HER is not restricted to environments with sparse rewards or those with difficult solutions for traditional algorithms. Moreover, it also shows that HER can be effectively utilized to improve performance in environments with varying characteristics and reward structures.

Finally, we must ensure that the introduction of HER into MOC does not affect the quality of option discovery. Thus, we run the algorithm using 10 different seeds and recorded the mean and standard deviation of the usage rate for each option during training with 2 and 4 options. Figure 3 presents the results for the FetchReach environment with sparse rewards, while Figure 4 shows the results for the same environment but with dense rewards.

When comparing the options under the same conditions, we can observe that MOC-HER maintains a task split among options that is comparable to the original MOC. With 2 options, both algorithms

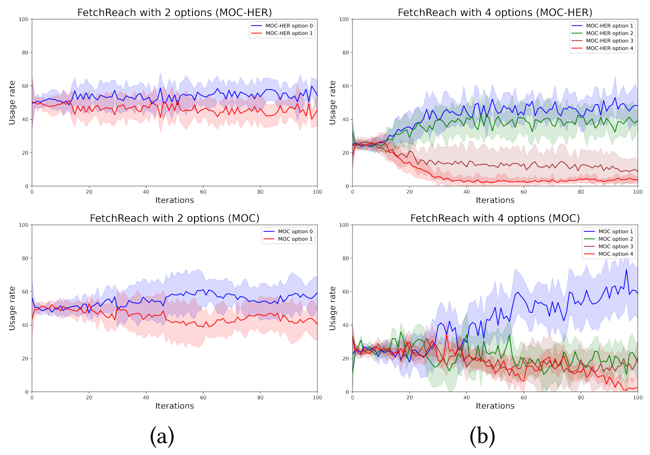

Figure 3: Percentage of usage of each option in the FetchReach environment with sparse rewards

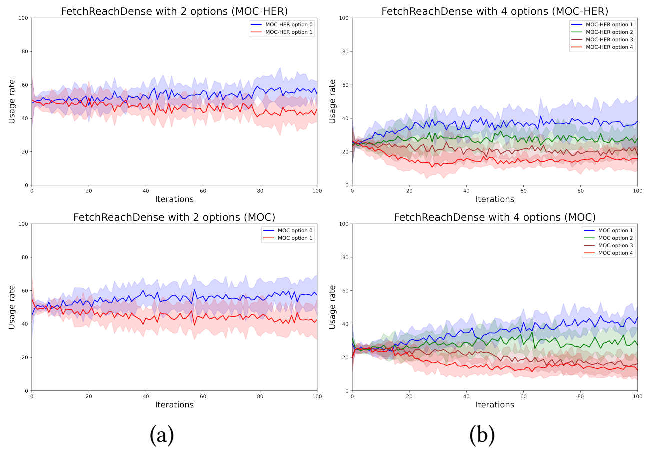

Figure 4: Percentage of usage of each option in the FetchReach environment with dense rewards

allocated each option to approximately half of the time steps in both sparse and dense reward settings, as shown in Figure 3a and Figure 4a. In the sparse reward environments, the standard MOC failed to converge, leading to greater instability in the usage rate of each option compared to the MOC-HER. In Figure 3b, when increasing to 4 options in the sparse reward environment, some options were selected significantly more than others. This result suggests that the number of options may be overestimated, and a smaller set of options might yield comparable performance. However, the algorithm assigned tasks to all options, ensuring that each continued to be periodically utilized. A similar behavior occurs in the original MOC algorithm. In the dense reward environment, as illustrated in Figure 4b, the usage rate of each option was more evenly distributed, suggesting that this reward structure facilitates a more balanced task allocation by the algorithm.

5 CONCLUSION

In the present work, we proposed including Hindsight Experience Replay in the MOC algorithm, aiming to improve its performance in environments with extremely sparse or ill-defined binary rewards. We observed a significant improvement in performance, allowing learning in contexts where the standard MOC algorithm has been unable to produce satisfactory results. Moreover, even in multi-goal environments with continuous rewards, MOC-HER was able to provide improvements over the original algorithm.

Despite initial positive results, additional tests are still needed to evaluate the scalability of the algorithm. Future work could explore the application of MOC-HER in additional experiments, particularly in more complex environments, such as other robotic tasks involving greater interaction with objects and scenarios. Variations in the size of the HER buffer could also be explored by limiting the proportion of stored samples used during training, in order to analyze how this impacts the learning process. A more thorough analysis of different goal-sampling strategies could yield deeper insights into their effects on performance. Moreover, incorporating HER into other temporal abstraction algorithms, such as PPOC [5], may provide valuable comparisons for evaluating the results obtained with the proposed approach.

Finally, integrating Dynamic Option Creation [7] into MOC-HER could offer significant advantages, permitting the evaluation and adaptation of the number of options within a set. This approach would not only optimize the selection of options by preventing overestimation but also enable the algorithm to dynamically identify the most suitable and efficient set of options.

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] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf

[2] 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

[3] Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. 2024. *Gymnasium Robotics*. http://github.com/Farama-Foundation/Gymnasium-Robotics

[4] 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

[5] Martin Klissarov, Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. Learnings Options End-to-End for Continuous Action Tasks. arXiv:1712.00004 [cs.LG] https://arxiv.org/abs/1712.00004

[6] Martin Klissarov and Doina Precup. 2021. Flexible Option Learning. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 4632–4646. https://proceedings.neurips.cc/paper_files/paper/2021/file/24cceab7ffc1118f5daaace13c670885-Paper.pdf

[7] Mateus Begnini Melchiades, Gabriel De Oliveira Ramos, and Bruno Castro Da Silva. 2025. Dynamic Option Creation in Option-Critic Reinforcement Learning. In *Proceedings of the 2025 International Conference on Autonomous Agents and Multiagent Systems* (Detroit, United States of America) *(AAMAS '24)*. International Foundation for Autonomous Agents and Multiagent Systems.

[8] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. 2018. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. arXiv:1802.09464 [cs.LG] https://arxiv.org/abs/1802.09464

[9] 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 (1999), 181–211. https://doi.org/10.1016/S0004-3702(99)00052-1