Comparing Deep Reinforcement Learning Approaches for Sparse Reward Settings with Discrete State-Action Spaces

Alp Şefik Çapanoğlu[1]*
[1]TU Delft

June 27, 2021

Abstract

One of the most challenging types of environments for a Deep Reinforcement Learning agent to learn in are those with sparse reward functions. There exist algorithms that are designed to perform well in settings with sparse rewards, but they are often applied to continuous state-action spaces, since economically relevant problems like robotic control and stock trading fall under this category. This means the continuous version overshadows the discrete state-action version of the sparse reward problem. Furthermore, research that focuses on sparse rewards is lacking in comparisons of algorithms dedicated to performing in this type of setting with other state-of-the-art Deep Reinforcement Learning algorithms. We devise an experimental setup to test a selection of algorithms from three state-of-the-art Deep Reinforcement Learning approaches; Hindsight Experience Replay, Maximum Entropy Reinforcement Learning and Distributional Reinforcement Learning. We show that as the cardinality of the state spaces in sparse reward settings increase, Hindsight Experience Replay approaches are superior in sample efficiency compared to the other two approaches studied.

1 Introduction

Deep Reinforcement Learning (DRL) is a field within Artificial Intelligence that excels at the making of sequential decisions, formulated such that an agent picks an action given a state. This is done through the non-linear approximation of a value function that takes the state and the action and outputs a value for this tuple. The agent chooses the action based on these assigned values. Deep Reinforcement Learning algorithms have been used in solving sequential decision-making tasks in very complex and very large search spaces. Examples to this include performing complex robotics tasks[1, 2] and beating humans in "hard" games such as Go[3], Dota2[4] and StarCraft[5]. This formulation is the most simple explanation of the function of a Deep Reinforcement Learning algorithm, and is precisely how Deep Q-Networks operate[6].

*Supervised by Greg Neustroev (g.neustroev@tudelft.nl) and Matthijs de Weerdt (m.m.deweerdt@tudelft.nl).

In the years since their inception, advancements in Deep Reinforcement Learning have taken the form of iterative improvements over their predecessors. There are also a number of approaches that have emerged due to this process, such as Maximum Entropy Reinforcement Learning and Distributional Reinforcement Learning.

There are multiple "open problems" in the field of Deep Reinforcement Learning, all of which make it difficult for a DRL algorithm to solve a problem. Properties that can make an environment—a formulation of a problem—difficult to learn for DRL agents include a high degree of randomness (stochasticity) in the environment, multi-modal reward functions (many local optima) and sparse reward functions (low information density). Naturally, a considerable amount of research focuses on improving algorithms or approaches so they perform well in these cases.

An example of an algorithm that is developed to address a difficult problem is Hindsight Experience Replay (HER). Hindsight Experience Replay addresses sparse rewards. Its functionality is discussed under Section 2. The paper introducing HER also introduces a problem with discrete state-action space and sparse reward, called bit-flipping. However, most applications of HER are focused on robotics applications where the state-action spaces of the problems are continuous. The only comparison done on bit-flipping in this paper is that of a Deep Q-Network (DQN) versus a DQN that uses HER[2]. This class of problem—a sparse-reward setting with a discrete state-action space—is understudied in terms of comparing state-of-the-art algorithms' performance on it.

One problem that can be formulated as a discrete state-action environment with sparse rewards is that of Quantum Circuit Optimization. Quantum Computing is a promising field of Computer Science, the use of which can provide up to exponential speedup to classical computing[7]. Recently, Deep Reinforcement Learning approaches have been applied to a number of Quantum Computing problems, such as Quantum Approximate Optimization Algorithms[8], Circuit Searching[9], Automated Quantum Programming[10] and Quantum Circuit Optimization[11].

Quantum Circuit Optimization (QCO) is an important problem as it allows the running of quantum algorithms to require less qubits by shortening their critical path or reducing the use of noisy operations. QCO can be formulated as a discrete state-action environment so that it can be solved by DRL agents, with circuits as states and equivalent gate transitions as actions[11]. Additionally, this formulation stands to benefit from a sparse binary reward. This would be done by designating the optimal circuit as the goal—the only state with a reward—and a starting state being created by taking actions away from the optimal circuit. This hypothetical formulation is a motivating example of how the understanding of the best performing algorithms in discrete state-action environments with sparse rewards stands to be useful in real-world applications.

This paper aims to explore the application of state-of-the-art DRL algorithms to problems with sparse reward and discrete state-action spaces. The metric to evaluate the chosen algorithms will be sample efficiency: the rate at which they learn with respect to training time. With these parameters, a question is constructed: "What state-of-the-art Deep Reinforcement Learning algorithm is the most sample efficient in sparse reward environments with discrete state-action spaces?".

The experimental process and reasoning around the chosen algorithms are detailed in Section 2. The resulting findings and each stage of experimentation can be found in Section 3. The results and experimental setup are evaluated through the lens of responsible research in Section 4. This is followed by the discussion of the conclusions and a lookout to future research in Section 5.

2 Method

In accordance with the bit-flipping environment that was chosen, we formulate an experimental setup to test different state-of-the-art DRL algorithms' performances.

To achieve this, first the chosen environment is discussed in Subsection 2.1 to identify the type of algorithms that should be chosen. Secondly, a set of algorithms are chosen based on the criteria identified. The process and reasoning behind this selection is detailed in Subsection 2.2. To conclude the Method Section, the the experimental setup is detailed in Subsection 2.3, in the name of reproducibility .

2.1 Environment

The bit-flipping environment consists of the state space $S = \{0, 1\}^n$ (all binary strings of length n) and the action space $A = \{0, 1, ..., n - 1\}$. In the action space, the index of the action corresponds to the bit that is flipped[2]. This means that by arbitrarily choosing the goal string and the starting string, random samples can be created. Training and validation sets can be created by randomly sampling out of the 2^n potential states.

This type of state space means that the available search space the agent has to explore before reaching a reward grows at an exponential rate. For a binary bit-string of length n, there are 2^n possible combinations. Per the formulation, reaching only one out of 2^n bit-strings awards the agent a non-negative reward. In this way, reaching a target is encoded into a maximization problem, and the global maximum in reward is achieved when the goal state is reached.

2.2 Algorithm Selection

The environment on which the chosen set of algorithms will be tested has a discrete state-action space, and a sparse reward function. Therefore a requirement is that the chosen algorithms be applicable to discrete state-action spaces. Another limiting requirement to address is that the chosen DRL algorithms should be available to the public under open-source licenses, because the implementation of these advanced algorithms would be very time consuming task and add a considerable potential for error.

The set of algorithms that are applicable to discrete state-action spaces is a small subset of the set of all DRL algorithms. The set of algorithms that have properties that hypothetically would make them perform well in a sparse-reward setting is even smaller. The approaches chosen out of this smaller subset were Proximal Policy Optimization (PPO)[12], Hindsight Experience Replay (HER)[2] and Quantile Regression Deep Q-Network (QR-DQN)[13]. The usage and the reasoning behind the selection of these algorithms are discussed below in their respective sections.

2.2.1 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is an on-policy, policy gradient approach. This means that in addition to learning the value of a state-action tuple, it also applies and optimizes a policy of how to pick from within those state-action tuples, instead of using a fixed policy like following the maximum estimated reward. This approach, in general, makes for algorithms that perform better than those that do not learn a policy (known as "off-policy algorithms"). Also, PPO includes novel changes to other algorithms within its class of policy gradient DRL algorithms. These changes improve on the sample inefficiency and lack of robustness other algorithms in that subclass. PPO takes a Maximal Entropy Reinforcement Learning approach, meaning an entropy (exploration) term is added to its optimization goal[12]. Due to its popularity and due to its entropy maximization term, it can be hypothesized that PPO's ability to optimize exploration will allow it to perform well in the sparse-reward setting at hand.

Apart from being a state-of-the art algorithm with good performance and exploration, a strong argument for choosing PPO can be made due to the fact that all papers mentioned so far that apply DRL to Quantum Computing problems use PPO exclusively[8, 9, 10, 11]. Hence, comparing the performance of PPO with other DRL algorithms becomes more relevant with regards to the motivation of applying this knowledge to the real world.

2.2.2 Hindsight Experience Replay

Hindsight Experience Replay (HER) is more of a behavioural change than an algorithm; it is called an *implicit curriculum* by its designers[2]. This term is used to convey the fact that HER needs to be used conjunction with some off-policy algorithm: it is not an algorithm itself. For the purpose of this paper, the off-policy algorithm to be used alongside HER needs to be applicable to discrete state-action spaces. This is, as mentioned before, a critical limitation. Due to this limitation, Vanilla Deep Q-Networks (DQN)[6] were chosen. DQN is the first formulation of a Deep Reinforcement Learning agent and is easy to reason about due to its relative simplicity.

Put briefly, Hindsight Experience Replay (HER) is a way in which off-policy algorithms can learn more from their training runs. It achieves this by creating virtual goals that are exactly the destination reached by an unsuccessful training run by the algorithm, awarding those training runs where the algorithm actually did not reach the end goal. A truncated pseudocode representation of HER is given in Algorithm 1. Here, A refers to the off-policy DRL algorithm that is using HER. R is the replay buffer, where the experiences of the agent in training are stored to be used in optimizing the algorithm at the end of the episode. S is the strategy used for sampling virtual goals. s is a state, a is an action and g is a goal. r is the reward function, and takes a state, an action and a goal as input, providing a reward as output.

Algorithm‘1 demonstrates how aside from training regularly and learning from the steps it took in that training episode, the states visited by the off-policy agent are sampled from and attributed rewards as if they were goals. The steps sampled are added to the replay buffered and are potentially used from within the buffer when optimizing the algorithm. This formulation allows the algorithm to still learn how to reach certain states that the agent encounters, even if they aren't an actual goal. This way it has rewards to move to

Algorithm 1: Hindsight Experience Replay

Initialize A, R

for *episode = 1,...,N* **do**
  Sample a goal g and an initial state s_0
  Sample & execute actions for each *timestep* $\in [1, T]$ using A
  for t = *1,...,T* **do**
    reward$_t = r(s_t, a_t, g)$
    Store transition (s_t, g, reward$_t$, s_{t+1}) in R
    $G = S(episode)$
    foreach *g' in G* **do**
      reward' $= r(s_t, a_t, g')$
      Store transition (s_t, g', reward', s_{t+1}) in R
    end
  end
  for i = *1,...,M* **do**
    Sample minibatch B from R
    Perform one step of optimization on A using B
  end
end

when searching for the sparse rewards, and presumably escape local optima and search for the final reward. This hypothesis of HER escaping local optima with help from virtual rewards is a good avenue for future research and outside the scope of this paper.

2.2.3 Quantile Regression DQN

One of the newer fields of Reinforcement Learning is that of Distributional Reinforcement Learning. In this paradigm, the learning of the reward function is approached differently. Instead of the agent learning estimations for the state-action tuples, it learns to fit the value function to a distribution using probabilistic techniques.

Quantile Regression is a method of splitting the distribution of the value function over n quantiles. Quantile Regression Deep Q-Networks is an approach that builds on top of Deep Q-Networks[6]. Instead of the the neural network in DQN learning estimated magnitudes of the value function, the number of output nodes is changed to n to regress towards the correct positions of each quantile. The quantiles are regressed to fit the surface of the value function as optimally as possible given the n quantile resolution[13].

One additional emergent benefit of using Quantile Regression DQN (QR-DQN) is that it is off-policy. The concern of learning a policy to act on estimated values of the value functions is no longer needed since the whole distribution is being estimated. Due to the lack of a constraint in terms of on-policy learning, Hindsight Experience Replay can be used alongside QR-DQN to improve sample efficiency.

2.3 Testing Setup

All algorithms that were chosen are implemented as they are found in Stable Baselines 3 V1.0[14]. As of V1.0, which was used in this paper, QR-DQN is found in the *contrib* repository released alongside V1.0. Unless the value for a parameter is explicitly mentioned, the default parameters as implemented were used when executing the algorithms. Optuna was used for performing sanity checks for some of the values chosen[15]. A potential future avenue of research is applying exhaustive hyperparameter optimization to the algorithms.

For the environment, the implementation of the bit-flipping environment is used from Stable Baselines 3[14]. The parameter *continuous* is set to False. Additionally, the "MultiInputPolicy" needs to be specified as the *policy* parameter for all agents so they work with the bit-flipping environment, as it extends the Goal Environment (gym.GoalEnv) class from OpenAI Gym[16]. The lack of environments that extend Goal Environment limited the environments that could be tested for this study.

The main question to be answered is which algorithm would perform best in terms of sample efficiency during training. For this reason training and validation sets were not separated as only a training set was necessary; the gathered data were rewards achieved by the agents during training. The training samples were created from within the state set of the bit-flipping problem by randomly sampling a starting state and a goal state.

3 Results

The first experiments performed were in order to rank the five following approaches:

1. Proximal Policy Optimization
2. Deep Q-Network
3. Deep Q-Network with Hindsight Experience Replay
4. Quantile Regression Deep Q-Network
5. Quantile Regression Deep Q-Network with Hindsight Experience Replay

In running these algorithms as they were implemented in Stable Baselines, the following graphs in Figure 1 were created. These graphs were plotted without implementing a moving average, so the data exhibits the noisy performance of the DRL algorithms. This noise hides the fine details of how fast the different algorithms learn, but with the exception of Quantile Regression DQN without Hindsight Experience Replay, all of the algorithms seem to perform well. Due to this result QR-DQN by itself was ruled out, but a further test was performed on DQN with and without HER by increasing the bit-string length from 7 to 10.

As seen in Figure 2, increasing the bit-string length by 3 caused the DQN implementation without HER to not learn anything meaningful. The reward it gathered never surpassed -6.5, whereas the implementation that used HER showed the rapid jump from a low average to a higher one, with maximums exceeding -2. This was enough evidence to disregard DQN without HER. The remaining algorithms would be PPO, DQN and QR-DQN both with HER.

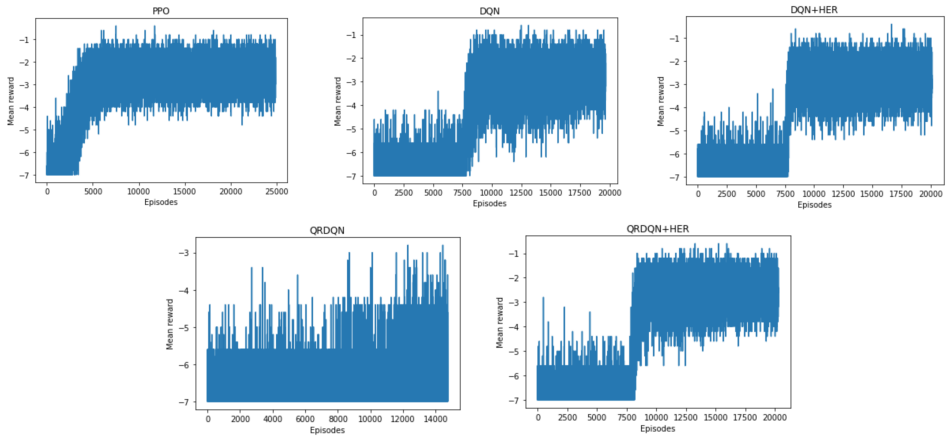

Figure 1: Graphs of reward reached by the tested algorithms on the bit-flipping environment with 7 bits, dependent on number of training epochs (on the x-axis). *First row from left to right:* PPO, DQN, DQN with HER. *Second row from left to right:* QR-DQN, QR-DQN with HER.

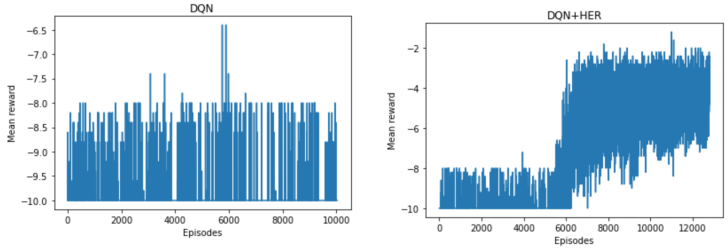

Figure 2: Graphs of reward reached by Deep Q-Network on the bit-flipping environment with 10 bits, with and without the use of Hindsight Experience Replay (in that order from left to right), dependent on training epochs (on the x-axis).

At this point, the decision was made to use a moving window in order to observe the average trends more meaningfully. The rapid cut-like jump observed in many of the graphs in Figures 1 and 2 was unexpected, but the observed peak in the gradient is later revealed to be caused by the noise.

Proceeding to increase the bit-string size further to 13, the results in Figure 3 were reached. The algorithms' epoch sizes in terms of steps were not tweaked, but all were trained for 5×10^4 steps with a learning rate $\alpha = 3 \times 10^{-4}$. The PPO implementation was unable to learn anything in this time, whereas both DQN and Quantile Regression DQN learned to consistently acquire a reward of -6 after about one fifth of the training. One drawback is that even if better than PPO, both algorithms seem to converge for a long time on the optimum with -6 reward. It could also be that the moving average influences these numbers and that in reality a very noisy behaviour around an average of -6 is being displayed.

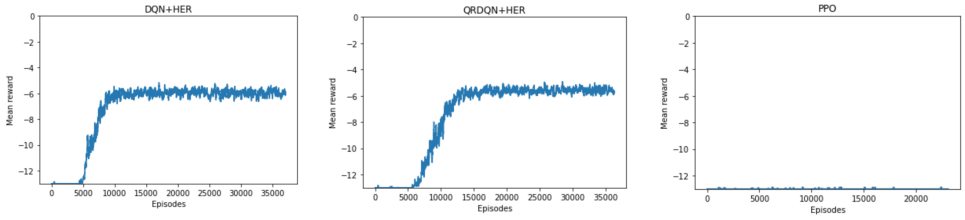

Figure 3: Graphs of reward reached by DQN and QR-DQN (both with HER) and PPO (in that order from left to right) on the bit-flipping environment with 13 bits, dependent on training epochs (on the x-axis).

Seeing these results, the entropy coefficient (the multiplier of the entropy term in the optimization target) was tested from the range $[0.0005, 0.3]$ using Optuna for 10 and 13 bit-string sizes. For the 10-bit version, an entropy coefficient 0.07 performed best, but no value for the entropy coefficient made PPO perform any better than a -13 to -12 range.

Similarly, QR-DQN's parameter for number of quantiles were tested over the range $[1, 50]$. The upper limit was placed at 50 because the example implementation of QR-DQN in Stable Baselines 3 documentation on the CartPole-v1 environment uses 50 quantiles[14]. No significant change in performance was observed within this range of quntiles.

4 Responsible Research

The approach chosen of elimination by comparison makes the reproducibility of the results ethically significant, since detailed analysis of the data is not provided. The decision was made early on to not implement new environments for testing, even though it would have diversified the results and strengthened the conclusions. This is because the scope of the project would make it impractical to verify these environments in detail, and could hinder replication studies either due to errors in implementation or due to unavailability.

To aid in the reproducibility of the findings, all code used was acquired from open source repositories, and the versions used were shared in the Testing Setup section. Also, while running the algorithms and the environment, their parameters were tweaked minimally in order to avoid possible human error in the reporting or re-implementation. All the parameter space searches were also reported, whether or not they produced any relevant results. With these steps taken, the setup can be replicated exactly, and the same steps of discovery can be recreated.

5 Discussion

During the experiments, the main question of which algorithms perform better in terms of sample efficiency was answered, although it took multiple steps of experimentation. The reason it became necessary for the secondary and tertiary rounds of experimentation was because the results were not as anticipated; algorithms either performed very close to each other or did not learn at all.

The conclusions reached after the experiments and the limitations of those conclusions are discussed in Subsection 5.1, and a future outlook is presented in Subsection 5.3.

5.1 Conclusions

The original question of what algorithms perform best out of PPO, DQN with HER and QR-DQN was reached by sequentially eliminating alternatives. QR-DQN did not perform well by itself, as it has no mechanisms for maximising exploration, and the bit-flipping environment's reward function is too sparse to fit to a distribution. DQN by itself showed similar performance to QR-DQN, although the question of why it performed better than QR-DQN as seen in Figure 1 is unanswered.

In the third round of tests, increasing the bit-string length from 10 to 13 caused PPO to learn nothing of significance within the allocated training time, meaning its sample efficiency would functionally be zero. In short, this means for the research question as asked originally, the answer is that out of the elected set HER is the best state-of-the art DRL approach for discrete state-action spaces with sparse reward in terms of sample efficiency.

The fact that HER performed well in these tests was expected. What stands out more is how PPO declines from being one of the best candidates in the first round of tests to not learning anything in the third round. In Figure 1 PPO outperforms the rest of the candidates by being the most sample efficient out of all the algorithms. The reason for this is that the 7-bit version of the problem is small enough that PPO and DQN can explore the space until they find the singular reward and converge to it. PPO does this faster than the other algorithms, as it is explicitly maximising entropy, meaning it is exploring maximally before finding the goal state. However as the state space grows exponentially with base 2, the 13-bit version of the bit-flipping problem has 64 times more states to explore than the 7-bit version.

In both the 7-bit and the 13-bit versions PPO is exploring maximally but stochastically, and with a fixed number of steps until a training episode ends. The maximum step count is equal to the bit-string size, which grows linearly with bit-string length. This means that in addition to exploring a much larger space, the exploration episodes that PPO is performing become shorter in relation to the size of the search space as the bit-string length increases. So, the state space grows 64 times, but the maximum steps that the agent can explore grows approximately 1.85 times. In contrast, HER makes it so DQN and QR-DQN get intermediate rewards that they can "move out to" from their starting position. This means that even though they are demonstrably worst at exploring the space on their own, with HER they can explore on a metaphorical expanding perimeter and outperform PPO when the space gets larger.

Furthermore, the third round of experimentation reveals that DQN and QR-DQN (both with HER) to perform almost identically. There are small differences: maximum reward received by QR-DQN is higher and DQN is more sample efficient. The reason for the relative inefficiency in QR-DQN may be that it is optimizing many quotients to fit the arbitrary virtual rewards, which may be causing it to readjust and therefore converge slower. This is an untested hypothesis and a good avenue for future exploration.

The main reason why QR-DQN's advancements do not reflect to its performance how-

ever is that there is no meaningful reward function to fit a distribution over. Even if the entire space ends up populated with virtual rewards because of HER, understanding the distribution would not be advantageous. Even if there is information in this distribution, it would be unrelated to the essence of the problem, and related to the HER formulation.

5.2 Evaluation

There were two main limiting factors in this paper that affected the scope and rigor of the conclusions.

The first limiting factor was the time and resource constraint the tests were performed under. This meant that hyper-parameter optimizations or exhaustive training (training until PPO starts to learn in the 13-bit problem) were not feasible. Training over more episodes and optimizing relevant hyper-parameters would very likely have resulted in better results.

The more limiting shortcoming however was the lack of environments available for Hindsight Experience Replay formulations. This limited the number of sparse reward environments to one, and other properties such as the multi-modality of rewards could not even be explored. It is possible to formulate such environments, and one was attempted based on the NChain-V0 environment from OpenAI Gym[16]. This proved too time consuming, mostly due to the lack of documentation of the variety of environment development tools and classes with the exception of in-line comments. It was abandoned due to these difficulties, alongside the difficulty it would add to reproducing the findings associated with it.

The main drawback of the conclusions is the lack of quantitative metrics that are used for comparison. The main reason for this was because new and different explorations were prioritized over analysing metrics for the sake of developing a more complete understanding of the behaviour causing the data, most of which provided no significant findings. This can be improved upon in an analytic follow up study, where data that has been generated during this study can be reanalyzed with more concrete metrics.

5.3 Future Outlook

There are multiple directions that can follow this study. The first one proposed is that the findings reported are detailed and diversified. More algorithms from each segment of Maximal Entropy RL, Hindsight Experience Replay and Distributional RL could be analysed. More and better quantified metrics can be established and compared, and more environments can be tested that have discrete state-action spaces and sparse rewards.

The second possible direction would be to expand on this study by adding approaches, or algorithms that incorporate a number of these approaches, like Rainbow. Rainbow is an algorithm that incorporates components like an entropy term (Kullback-Leibler loss) and distribution fitting, benefiting both from the Maximum Entropy Reinforcement Learning and Distributional Reinforcement Learning approaches[17]. In this study the approaches were separated for the purpose of being able to reason about the individual approaches' strengths and shortcomings. Another noteworthy combination would be the joining of some discrete variation of the Soft Actor Critic (SAC) algorithm with HER[19]. SAC is an off-policy Maximum Entropy Reinforcement Learning Algorithm[18], meaning this combination would both benefit from the sample efficiency that HER provides and from efficient exploration. Testing combinations like this will expand on the findings of this paper, and has the potential

of providing superior algorithms for use in solving discrete state-action space and reward-sparse environments.

Finally, the conclusions arrived may be taken directly to the motivating example. A study may be undertaken by implementing the suggested alternative implementation of QCO with sparse rewards inspired by Fösel, et al.'s formulation. PPO, Fösel, et al.'s choice[11], and HER can then be compared on this environment. In this case, an intermediate or precursory study of different HER offsprings such as Soft HER[20], or the combination of HER with more advanced algorithms than DQN, such as Soft Actor Critic [19] is advisable to identify the best candidate under the umbrella of HER to compare with PPO.

References

[1] Peters, Jan, Sethu Vijayakumar, and Stefan Schaal. "Reinforcement learning for humanoid robotics." Proceedings of the third IEEE-RAS international conference on humanoid robots. 2003.

[2] Andrychowicz, Marcin, et al. "Hindsight experience replay." arXiv preprint arXiv:1707.01495 (2017).

[3] Silver, David, et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play." Science 362.6419 (2018): 1140-1144.

[4] Berner, Christopher, et al. "Dota 2 with large scale deep reinforcement learning." arXiv preprint arXiv:1912.06680 (2019). arXiv:1908.08054 (2019).

[5] Vinyals, Oriol, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning." Nature 575.7782 (2019): 350-354.

[6] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." nature 518.7540 (2015): 529-533.

[7] Shor, Peter W. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer." SIAM review 41.2 (1999): 303-332.

[8] Wauters, Matteo M., et al. "Reinforcement-learning-assisted quantum optimization." Physical Review Research 2.3 (2020): 033446.

[9] Kuo, En-Jui, Yao-Lung L. Fang, and Samuel Yen-Chi Chen. "Quantum Architecture Search via Deep Reinforcement Learning." arXiv preprint arXiv:2104.07715 (2021).

[10] McKiernan, Keri A., et al. "Automated quantum programming via reinforcement learning for combinatorial optimization." arXiv preprint arXiv:1908.08054 (2019).

[11] Fösel, Thomas, et al. "Quantum circuit optimization with deep reinforcement learning." arXiv preprint arXiv:2103.07585 (2021).

[12] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[13] Dabney, Will, et al. "Distributional reinforcement learning with quantile regression." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1. 2018.

[14] Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., & Dormann, N.. (2019). Stable Baselines3. https://github.com/DLR-RM/stable-baselines3.

[15] Akiba, Takuya, et al. "Optuna: A next-generation hyperparameter optimization framework." Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019.

[16] Brockman, Greg, et al. "Openai gym." arXiv preprint arXiv:1606.01540 (2016).

[17] Hessel, Matteo, et al. "Rainbow: Combining improvements in deep reinforcement learning." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1. 2018.

[18] Haarnoja, Tuomas, et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." International Conference on Machine Learning. PMLR, 2018.

[19] Prianto, Evan, et al. "Path Planning for Multi-Arm Manipulators Using Deep Reinforcement Learning: Soft Actor-Critic with Hindsight Experience Replay." Sensors 20.20 (2020): 5911.

[20] He, Qiwei, Liansheng Zhuang, and Houqiang Li. "Soft Hindsight Experience Replay." arXiv preprint arXiv:2002.02089 (2020).