Smart Start and HER for a Directed and Persistent Reinforcement Learning Exploration in Discrete Environment

Heba Alrakh[1], Muhammad Fahmi Miskon[2]
Center for Robotics and Industrial Automation
Fakulti Kejuruteraan Elektrik
Universiti Teknikal Malaysia Melaka
76100 Durian Tunggal, Melaka, Malaysia

Rozilawati Mohd Nor[3]
Center for Robotics and Industrial Automation
Fakulti Teknologi Kejuruteraan Elektrik dan Elektronik
Universiti Teknikal Malaysia Melaka
76100 Durian Tunggal, Melaka, Malaysia

***Abstract*—Reinforcement learning (RL) solves sequential decision making problems through trial and error, through experiences can be amassed to achieve goals and increase the accumulative rewards. Exploration-exploitation dilemma is a critical challenge in reinforcement learning, particularly environments with misleading or sparse rewards which have shown difficulties to construct a suitable exploration strategy. In this paper a framework for Smart Start (SS) and Hindsight experience replay (HER) is developed to improve the performance of SS and make the exploration more directed especially in the early episodes. The framework Smart Start and Hindsight experience replay (SS+HER) was studied in discrete maze environment with sparse rewards. The results reveal that the framework doubles the rewards at the early episodes and decreases the time of the agent to reach the goal.**

***Keywords*—Reinforcement learning; hindsight experience replay; smart start; limit search space; exploration-exploitation trade off**

I. Introduction

People learn through interacting with the environment around them from their childhood where children's walking or trying to play, which is considered the major source of learning. The same in reinforcement learning, the machine is trying to interact with environment to collect information then use it to discover the best possible performance [1].

Exploration means learning new knowledge by trying new actions that the agent did not select before which might lead to a better action selection in the future causing increase in the accumulative reward [2]. In contrast, exploitation is using the same actions that the agent tried in the past and was effective in producing rewards in order to maximize the immediate reward. Excessive exploration will be wasting of time and cause less immediate reward because the agent might spend most of the time doing irrelevant actions or less reward actions. On the other hand, more exploitation will cause suboptimal solution. So the balance between both of them is becoming a critical challenge and an essential matter in order to get better results [3]. This is called exploration and exploitation dilemma.

There are many exploration strategies to solve this problem [4] but most of them are depending on collecting more data. A relatively new strategy is focusing on reducing the search space such as Proximal Policy Optimization (PPO) [5] and Trust Region Policy Optimization (TRPO) [6]. Although there are number of new limiting search space techniques, there are no experiments combining two techniques together.

The objective of this paper is combining Smart Start and Hindsight Experience Replay (HER) in order to reach a more efficient exploration. Smart Start guides an agent to a state where it supposes to discover the newest information, which is named the Smart Start state, S_{ss}. Smart Start does not modify the functionality of RL algorithm in which it is utilized with, yet it adds more persistent and directed exploration to the algorithm. On top of Smart Start strategy, a conceptually simple framework (HER) is added which utilizes experience to improve exploration by splitting the main goal to sub goals to learn from the previous errors.

The rest of this paper is organized as follows: Section II gives the description of the related work. Section III contains the background of the research. In Section IV, Smart Start and HER is discussed in details. Section V presents the experiments. In Section VI, the results are displayed. Lastly, Section VII is conclusion.

II. Related Work

Balancing between exploration and exploitation is solved by [7] which depends on Stratonovich's value of information which consists of two steps. The first one generates the base line of agent performance by measuring the achievable return of a policy in where there is no information regarding the states, afterward offsets these costs with a term that evaluates the average penalties when the state-action information is bounded above by a prescribed amount. Though, the optimization of value of information shows a softmax random exploration. Obviously, it depends on factors where the value factors is decided by human. Also it does not cover the multistate case so the improvement of optimal average cost can be achieved per episode.

Another alternative solution, by applying Bayesian deep Q-networks (BDQN) is an efficient Thompson sampling based method in high dimensional RL problems. In [8]

Azizzadenesheli and Anandkumar studied the behaviour of BDQN and compared it to another method to solve exploration – exploitation trade off. Yet the problem is this method itself is difficult in implementing and time consuming and did not provide a sample efficiency guarantee.

On the other hand, Lin et al. [9] used demonstration data to guide the exploration of agents at the beginning of training, which can help agents learn faster. A demonstration data guided mechanism was proposed, which makes use of demonstration data to guide agents' actions in the training phase. But after running the experiment on Ant-v2 environment for random seed 3 and 4, the algorithm had experienced the unstable training problem at the beginning of training.

Also Colas et al. [10] tried to solve exploration – exploitation trade off in continuous environment especially in Continuous Mountain Car. By using "Goal Exploration Process - Policy Gradient" GEP-PG which contained of two stages: the first one was "Goal Exploration Processes" GEP which used a directed exploration of the continuous state action space for a specific environment. Then stored the results in the replay buffer of a deep RL algorithm, which processed them to perform sample efficient policy improvement.

Nair et al. [11] aimed to solve the exploration problem via imitation of a human expert. That combined demonstration-based imitation learning and reinforcement learning to solve exploration problems in robotic tasks. Also learn a policy from demonstrations and rewards, using demonstrations to make the RL problem easier. The main limitation of this work was small efficiency when solving tougher tasks.

III. BACKGROUND

A. Smart Start

Smart Start was developed for sparse or misleading rewards, in which the agent receives the rewards after achieving the goal which make the learning process harder [12]. Smart Start uses the previous information to find the region is expected to give the best information of the agent to solve the challenge and reach the goal. On the other hand, in normal learning the agent spends most of the time just re-exploring the states that have already visited.

Fig. 1 displays a normal RL contrasted with RL with Smart Start [13]. In normal RL, the agent starts in the first state s0 and continues its policy utilizing some exploration strategy, named πexplore, till the end of episode in a final state sterm. For Smart Start, in contrast, the agent firstly finds the Smart Start state Sss then obtains a policy πss leading the agent to Sss by utilizing past experiences. The policy πss is implemented till the agent is nearby Sss and consequently the agent implements the learned policy πexplore until finishing the episode in a terminal state sterm.

Here the functionality for discrete systems is considered in the environment. So it becomes essential to utilize the epsilon greedy or Upper Confidence Bound (UCB1) algorithm [14] for getting the Smart Start state and dynamic programming for guiding the agent to the Smart Start state.

1) Choosing smart start state: In choosing S_{ss}, select a reachable state. The state is called reachable when it has been visited no less than one time by the agent. Thus, the agent cannot determine the Smart Start state at the beginning of the learning process because it requires collecting more information about the surrounding environment at the beginning. The agent saves the visited states in a replay buffer (D) [15] which is used in many algorithms in Deep Reinforcement Learning, such as Deep Q-Network (DQN) [16], Deep Deterministic Policy Gradients (DDPG) and Hindsight Experience Replay (HER). The replay buffer has a specific capacity and uses many strategies for sampling the transition. Such as uniform sampling where each transition is sampled with equal probability or prioritized sampling where each transition is sampled with a high Temporal Difference error (TD) [17].

When selecting S_{ss}, the agent is searching for the optimal state in buffer *D* to begin exploring from. As a result, a state with a lower visitation density has a higher probability to be nearby unvisited states as a result has a high probability of leading to new information. In this project an easy approximation has been used and only taken into consideration the visitation density of discrete states. That can be verified simply by the visitation counts *C(s)* to every visited state. The following equation shows how to choose the smart start state [7]:

$$S_{ss} = \text{argmax}[\text{maxQ}(s,a) + c_{ss}\sqrt{\frac{\log \sum_{s \in S} C(s)}{C(s)}}] \forall s \in D \quad (1)$$

Where Q (s, a) is the action value function and D is the size of buffer. A constant css > 0 for varying the amount of exploitation and exploration and has to be selected suitably. Sometimes the agent only needs exploration especially at the start of the learning process to learn too much about the surrounding environment. A large value for css will produce more exploration. The css value may be reduced in the learning process to change from pure exploration to a suitable balance between exploration and exploitation.

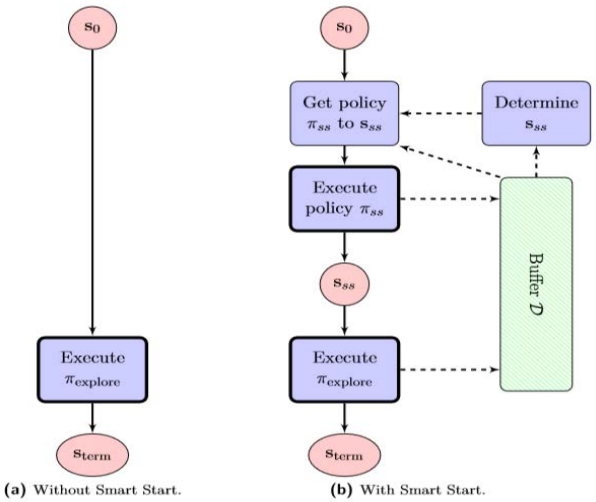

Fig. 1. Comparison between RL Episodes with and without Smart Start.

To explain the previous equation in details, suppose that there is a state with 10 visitation density then the value of $\sqrt{\frac{\log 10}{10}}$ will be 0.316. On the other hand, if a second state was with 20 visitation density then the value of the same part will be 0.255. Because of that the agent chooses the maximum value since it indicates to a less visitation density.

2) Leading the agent to smart start state: Leading the agent from the first state to S_{ss} is considered the second section of the Smart Start technique. As mentioned previously the Smart Start state has been visited before, consequently the trajectory to S_{ss} is known. The easiest method to reach the S_{ss} might be through replaying the trajectory. However this has clear consequences especially in stochastic environments because it has randomness behavior associated with it contrary to deterministic system [18]. Additionally, there is other issue which is the trajectory length as the trajectory might be totally a random path. After numerous iterations the trajectory to S_{ss} can involve numerous series random trajectories, leading to extremely long, complicated and time-consuming path to the Smart Start state. This is not a good option since it lessens the efficiency and the accumulative rewards.

The trajectory optimization method [19] can direct the agent to the Smart Start state. But it is needed to take into consideration the environment in the trajectory optimization with a view to prevent the agent going to regions with great penalties. This work will not consider the environment characteristics when leading the agent to the smart start state, because the main aim is finding a policy which causes to the shortest, most efficient and rapid track to S_{ss}.

This article focuses on discrete environments where a model based approach may be simply applied and almost has an optimum performance. The trajectory optimization may be done by dynamic programming which depends on Value Iteration to give the shortest and most reliable track to S_{ss}.

The agent keep counts the visitation states and learns a transition model as the Model Based Reinforcement Learning method where the transition model and reward function can be simply build. The agent is tracking the visitation counts of the total number of times an action *a* has been used in state *s* which symbolized through *C* (*s*, *a*) in equation 1, also the times number using action *a* in state *s* resulting a traversing to state *s0* which symbolized through *C* (*s*, *a*, *s0*). A rewards' sum for the reward function for each state-action pair is saved. Now the transition model and reward function can be built. Then transitions' reward is given to the S_{ss} in order to use this transitions, the whole transitions have a probability larger than zero for traversing to the Smart Start state receive a reward. That is because the agent is aiming to get into the area of the Smart Start but not exactly in the S_{ss} itself. An ideal policy to the S_{ss} may be obtained utilizing Value Iteration.

B. HER

Imagine that you want to cook any kind of food, and the first trial was bad because you did a mistake then the next time you will avoid that mistake to get a better result. This can simply explain the main idea behind HER which is letting the agent learn from all episodes even the episode was not successful for reaching the main goal *g*. Assuming that at the early episodes the agent cannot reach the final goal *g*, it is supposed that the state where the agent arrives is the virtual reward and the agent can get some reward instead of zero. Which can be seen in Fig. 2.

Suppose that there is an agent with a task *g*, every transition leads to the goal will give a reward 0 or 1. This is called a sparse reward environment where the agent gets the reward when reaching the goal only so that does not help in improving the actions next time. Because of that learning from the sparse reward is so difficult, but HER is solving this problem [20]. HER is taking into account the goal beside the state in the value function. So the transition will be saved in the replay buffer in the following format:

$(S_t \parallel g, a_t, r_t, S_{t+1} \parallel g)$

As mentioned above the agent must split the tasks so instead of storing the transitions regarding the goal *g* every time, the agent also stores the transitions regarding the new selected sub goals *g'* in order to decrease the sparsity. Because of that the agent can get more rewards which help in improving the next new actions which leading to lessen the time that the agent needs to reach the goal *g*. So the new transition will look like the next format:

$(S_t \parallel g', a_t, r', S_{t+1} \parallel g')$

It is not a good idea to tell the agent which sub goals to choose since it will cause a domain specific knowledge. There are four strategies to choose the sub goal. The first strategy uses final state in the episode. The second strategy uses random states that come from the same episode as the transition being replayed. The third strategy depends on choosing random states that come from the same episode. The last strategy chooses the sub goals in a random way. In this experiment, the second strategy is used mainly because it is considered the best one and has the maximum success rate [20].

Fig. 3 displays a diagram for normal reinforcement learning with (HER). In normal reinforcement learning, at every time step the agent receives a representative from the environment state $S_t \in S$ and a reward $R_t \in R$, on that basis selects an action a_t. After executing the action a_t, the agent receives again the modified s_{t+1} and r_{t+1}. The loop continues going on until the environment sends a terminal state, which finishes the episode. When HER is implemented, the same thing will happen except the experiences are stored in the replay buffer that is related to the goal. As a result, through each step the agent will get a batch of experiences, so the agent will store them in the replay buffer regarding the goal then it will choose a sub goal in order to use it.

The HER process can be described in steps as below:

1) Store tuple from the episodes using the goal *g*.
2) Select sub goals, *g'*, using one of the mentioned above strategy.
3) Store new tuples by replacing g to *g'*.

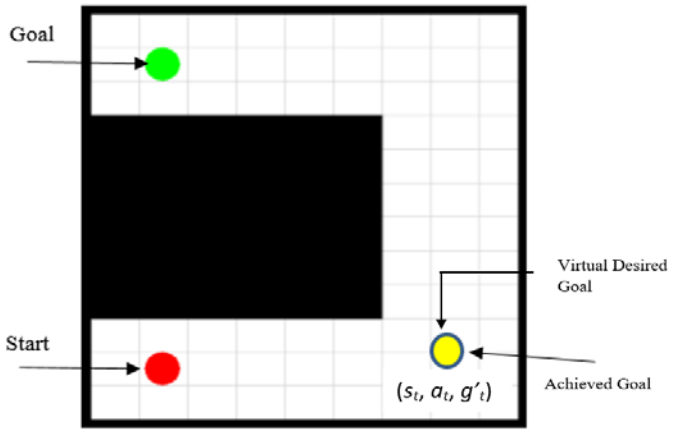

Fig. 2. Grid World Environment showing Virtual Goal in HER Technique.

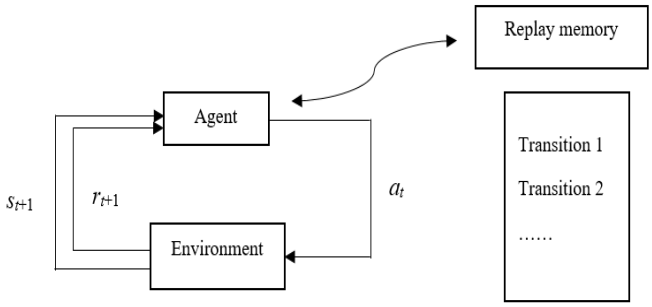

Fig. 3. The effect of Adding Hindsight Experience Replay.

IV. Smart Start and HER Algorithm

In this work both techniques are combined together in order to reach the more efficient exploration. Smart Start solves the challenges in a more efficient way. However, it is not a complete and independent exploration strategy. For this reason, Smart Start should be incorporated with other exploration strategies for increasing the accumulative rewards and decreasing the number of episodes. So a conceptually simple framework (HER) will be added which split the goal to improve exploration by saving the previous experiences.

Both Smart Start and HER was developed for sparse or misleading rewards. In Smart Start the agent receives the rewards after achieving the final goal which makes the learning process harder. So by adding HER the agent can receive some rewards before reaching the final goal that helping in decreasing the learning time.

The algorithm of how HER is implemented with Smart Start technique is provided in Algorithm 1. This algorithm can be utilized as a template for implementing HER with Smart Start framework.

Algorithm 1 Smart Start Framework with HER

1 : Initialize buffer D, and let AGENT starts from initial state $s0$
2 : **For** each episode **do**
3 : Sample a goal g and an initial state $s0$
4 : **For** t=0, T-1 **do**
5 : Sample an action a_t using the behavioral policy from A: $a_t \leftarrow \pi_b(s_t||g)$
6 : Execute the action a_t and observe a new state s_{t+1}
7 : **End for**
8 : **For** t=0, T-1 **do**
9 : $r_t := r(s_t, a_t, g)$
10 : Store the transition $(s_t||g, a_t, r_t, s_{t+1}||g)$ in D
11 : Sample a set of additional goals for replay G:=S (current episode)
12 : **For** $g' \in G$ **do**
13 : $r' := r(s_t, a_t, g')$
14 : Store the transition $(s_t||g, a_t, r_t, s_{t+1}||g)$ in D
15 : **End for**
16 : **End for**
// ---- Smart Start algorithm begins ---- //
17 : **if** $u \leq \eta$ and $|D| > 0$ **then** // Smart Start Algorithm
18 : // select S_{ss} utilizing upper confidence bound
19 : S_{ss} =argmaxa [max $Q(s,a) + c_{ss}\sqrt{\frac{\log|D|}{C(s)}}] \forall s \in D$
20 : // obtain policy using trajectory optimization
21 : π_{ss} = TRAJOPT (D, $s0$, S_{ss})
22 : // execute smart start policy
23 : **Repeat**
24 : Choose $a_t = \pi_{ss}(s_t)$
25 : Take action a_t and observe s_{t+1} and r_{t+1}
26 : Add (s_t , a_t , s_{t+1} , r_{t+1}) to D
27 : UPDATEAGENT (D)
28 : $t \leftarrow t+1$
29 : **Until** $d(s_t, S_{ss}) < \vartheta$, s_t is terminal or $t = T_{episode}$
// ---- Smart Start algorithm ends ---- //
30 : **End for**

From the algorithm, the buffer size D should be more than zero, to ensure there are states inside. Line 19 shows that the agent uses the buffer size instead of visitation count in equation 1 mainly because both of them are equal.

Normally, Smart Start is not used in each episode, it is utilized only in a specific ratio of the episodes. Now after implementing HER that leading to boost the accumulative rewards and enhance the performance of the agent in reaching the goal optimal solution as shown in Fig. 4.

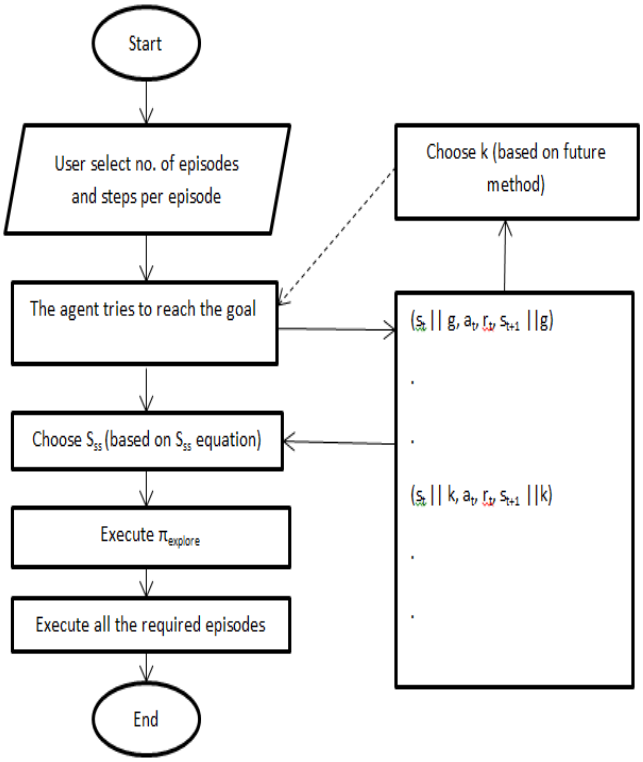

Fig. 4. Flowchart of Smart Start with HER.

V. Experiment

The experiment is done using python 19.2.3 on a grid world environment which is shown in Fig. 2. The initial state is represented by a red circle, and the goal state is represented by a green circle. The agent cannot pass through walls. Once the agent tries to pass a wall the state does not change. The episode is finished once the agent reaches either the state of goal or the limit of steps per episode which is 1000. The agent takes a reward only when the agent reaches the goal state as shown below:

R_{t+1} = {1 if $s_{t+1} = s_{goal}$

{0 if otherwise

The number of steps that the agent takes to reach the goal in this experiment is computed also the average reward per episode. The experiment is carried out on the Easy grid world environments [13]. Firstly, the experiment is carried out with Smart Start framework only then HER is added to the framework to compare between them.

The agent should utilize Smart Start each episode. So, in this experiment $\eta = 1$ will be used. Smart Start does not used in the first episode since there is no information has been stored yet. Through the experiment the value function will be zero. This makes the Smart Start parameter c_{ss} irrelevant in this experiment and can be set to an arbitrary positive value, a value of $c_{ss} = 0.1$ is utilized in this experiment.

VI. Results

Reinforcement learning agents learn from the reward which is given to the agent by the environment. There are certain suggestions in sparse rewards environment like ours. For an algorithm like Q-Learning this means the value function is zero until the goal has been reached for the first time. Also the number of steps that the agent takes to reach the goal for the first time is consequently a vital characteristic of the exploration strategy. But this problem was solved by adding HER which reduce the sparsity in the environment by choosing sub goals and give the agent rewards when reaching there.

To study the result of adding HER on the maze grid world environments, Fig. 5 and Fig. 6 display the average reward per episode and the number of steps required to achieve the goal using Smart Start alone and Smart Start with HER, respectively.

Fig. 5 shows the reward is doubled in SS+HER comparing to SS alone in the episodes which are less than 100. After that both SS alone and (SS+HER) give the same performance. So HER helps SS to be more directed and persistent in the beginning which is leading to more rewards and achieving the goal with a less time.

Fig. 6 shows the number of steps per episode for (SS+HER) is less than number of steps per episode for SS alone, the difference between them was 30% in the beginning then is reduced to reach 5% after the episode 100. When the number of steps is less in all the episodes as a result the time is needed in (SS+HER) is less than the time for SS alone. So (SS+HER) is faster than SS alone, as a result (SS+HER) reduces the learning time.

In both Fig. 5 and Fig. 6, the graph is becoming flat after the learning time. As in the early episodes, the agent is just collecting information. After that, the agent uses this information to reach the goal with the maximum reward and minimum number of steps.

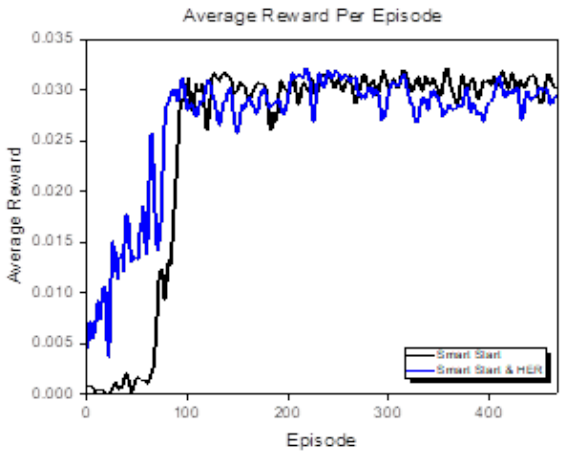

Fig. 5. The Average Reward of Smart Start alone and Smart Start with HER.

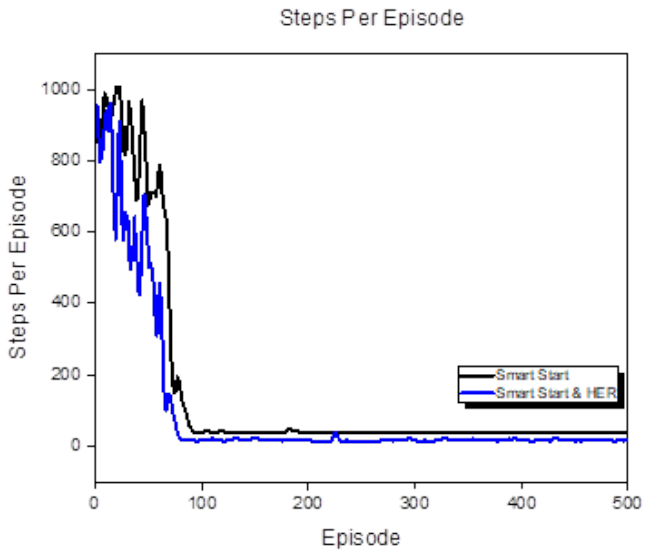

Fig. 6. Steps Per Episode of both Smart Start alone and Smart Start with HER.

VII. Conclusion

The Smart Start framework with HER was assessed by using the easy grid world environment in this experiment. The experiment considered the exploration performance of Smart Start in combination with a limit search space technique which is HER. The performance of exploration was determined as the average number of steps it took the agent to attain the goal state.

It has shown that Smart Start and HER together can enhance the exploration on discrete grid world environments. This clearly leads to efficient performance of the whole learning. The Smart Start and HER can simply be combined with several exploration strategies and reinforcement learning algorithms. That making Smart Start and HER a promising and attracting exploration basis of reinforcement learning challenges. The Smart Start technique was developed for environments with misleading or sparse rewards. This article assessed the performance of Smart Start and HER in discrete environments. For the future works, it is still an open area for other environments not only restricted to discrete environments with misleading or sparse rewards. This directly provides a rise to remarkable guidelines for future work in the same field in other environments such as continuous.

Acknowledgment

This research was supported by Ministry of Higher Education Malaysia Grant under project FRGS/2018/FTKEE-CERIA/F00384 and Center for Robotics and Industrial Automation (CeRIA), Faculty of Electrical Engineering (FKE), Universiti Teknikal Malaysia Melaka (UTeM).

References

[1] R. S. Sutton and A. G. Barto, Reinforcement Learning : An Introduction. London: MIT press, 2015.

[2] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, “A Survey of Deep Reinforcement Learning in Video Games,” no. 61573353, pp. 1–13, 2019, [Online]. Available: http://arxiv.org/abs/1912.10944.

[3] L. Shani, Y. Efroni, and S. Mannor, “Exploration conscious reinforcement learning revisited,” 36th Int. Conf. Mach. Learn. ICML 2019, vol. 2019-June, pp. 9986–10012, 2019.

[4] A. D. Tijsma, M. M. Drugan, and M. A. Wiering, “Comparing exploration strategies for Q-learning in random stochastic mazes,” 2016 IEEE Symp. Ser. Comput. Intell. SSCI 2016, 2017, doi: 10.1109/SSCI.2016.7849366.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” pp. 1–12, 2017, [Online]. Available: http://arxiv.org/abs/1707.06347.

[6] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, “Trust region policy optimization,” 32nd Int. Conf. Mach. Learn. ICML 2015, vol. 3, pp. 1889–1897, 2015.

[7] I. J. Sledge and J. C. Principe, “Balancing exploration and exploitation in reinforcement learning using a value of information criterion,” ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc., pp. 2816–2820, 2017, doi: 10.1109/ICASSP.2017.7952670.

[8] K. Azizzadenesheli and A. Anandkumar, “Efficient exploration through Bayesian deep Q-networks,” 2018 Inf. Theory Appl. Work. ITA 2018, 2018, doi: 10.1109/ITA.2018.8503252.

[9] K. Lin et al., “Exploration-efficient Deep Reinforcement Learning with Demonstration Guidance for Robot Control,” 2020, [Online]. Available: http://arxiv.org/abs/2002.12089.

[10] C. Colas, O. Sigau, and P. Y. Oudeyer, “GEP-PG: Decoupling exploration and exploitation in deep reinforcement learning algorithms,” 35th Int. Conf. Mach. Learn. ICML 2018, vol. 3, pp. 1682–1691, 2018.

[11] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Overcoming Exploration in Reinforcement Learning with Demonstrations,” Proc. - IEEE Int. Conf. Robot. Autom., pp. 6292–6299, 2018, doi: 10.1109/ICRA.2018.8463162.

[12] J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter, “RUDDER: Return Decomposition for Delayed Rewards,” in 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019, no. NeurIPS, pp. 1–12.

[13] Bart Keulen, “Smart Start A Directed and Persistent Exploration Framework for Reinforcement Learning,” 2018.

[14] E. Hartog and H. Moreines, “New techniques in automatic flight control system design,” SAE Tech. Pap., vol. 3, pp. 397–422, 1961, doi: 10.4271/610369.

[15] S. Zhang and R. S. Sutton, “A Deeper Look at Experience Replay,” 2017, [Online]. Available: http://arxiv.org/abs/1712.01275.

[16] J. Fan, Z. Wang, Y. Xie, and Z. Yang, “A Theoretical Analysis of Deep Q-Learning,” vol. 120, no. 1995, pp. 1–4, 2019, [Online]. Available: http://arxiv.org/abs/1901.00137.

[17] M. Tokic, P. Ertle, G. Palm, D. Söffker, and H. Voos, “Robust exploration/exploitation trade-offs in safety-critical applications,” IFAC Proc. Vol., vol. 8, no. PART 1, pp. 660–665, 2012, doi: 10.3182/20120829-3-MX-2028.00160.

[18] V. Anagnostopoulou, “Stochastic and deterministic absorption in neutron-interference experiments,” vol. 36, no. 9, pp. 1–17, 1987.

[19] J. T. Betts, “Survey of numerical methods for trajectory optimization,” J. Guid. Control. Dyn., vol. 21, no. 2, pp. 193–207, 1998, doi: 10.2514/2.4231.

[20] M. Andrychowicz et al., “Hindsight experience replay,” Adv. Neural Inf. Process. Syst., vol. 2017-Decem, no. Nips, pp. 5049–5059, 2017.