Enhancing autonomy and efficiency in goal-conditioned reinforcement learning

Yang, Z.

Chapter 6

First Go, then Post-Explore: the Benefits of Post-Exploration in Intrinsic Motivation

Authors: Zhao Yang, Thomas Moerland, Mike Preuss, Aske Plaat

Published in 15th International Conference on Agents and Artificial Intelligence (ICAART), 22-24 February 2023, Lisbon, Portugal.

Abstract

Go-Explore achieved breakthrough performance on challenging reinforcement learning (RL) tasks with sparse rewards. The key insight of Go-Explore was that successful exploration requires an agent to first return to an interesting state ('Go'), and only then explore into unknown terrain ('Explore'). We refer to such exploration after a goal is reached as 'post-exploration'. In this paper, we present a clear ablation study of post-exploration in a general intrinsically motivated goal exploration process (IMGEP) framework, that the Go-Explore paper did not show. We study the isolated potential of post-exploration, by turning it on and off within the same algorithm under both tabular and deep RL settings on both discrete navigation and continuous control tasks. Experiments on a range of MiniGrid and Mujoco environments show that post-exploration indeed helps IMGEP agents reach more diverse states and boosts their performance. In short, our work suggests that RL researchers should consider using post-exploration in IMGEP when possible since it is effective, method-agnostic, and easy to implement.

6.1 Introduction

Go-Explore (Ecoffet et al., 2021) achieved breakthrough performance on challenging reinforcement learning (RL) tasks with sparse rewards, most notably achieving state-of-the-art, 'super-human' performance on Montezuma's Revenge, a long-standing challenge in the field. The key insight behind Go-Explore is that proper exploration should be structured in two phases: an agent should first attempt to get back to a previously visited, interesting state ('Go'), and only then explore into new, unknown terrain ('Explore'). Thereby, the agent gradually expands its knowledge base, an approach that is visualized in Figure 6.1. We propose to call such exploration after the agent reached a goal *post-exploration* (to contrast it with standard exploration).

There are actually two variants of Go-Explore in the original paper: one in which we directly reset the agent to an interesting goal (*restore-based* Go-Explore), and one in which the agent has to act to get back to the proposed goal (*policy-based* Go-Explore). In this work, we focus on the latter approach, which is technically part of the literature on intrinsic motivation, in particular intrinsically motivated goal exploration processes (IMGEP (Colas et al., n.d.)). Note that post-exploration does not require any changes to the IMGEP framework itself, and can therefore be easily integrated into other existing work in this direction.

While Go-Explore gave a strong indication of the potential of post-exploration, it did not structurally investigate the benefit of the approach. Go-Explore was compared to other baseline algorithms, but post-exploration itself was never switched on and off in the same algorithm. Thereby, the isolated performance gain of post-exploration remains unclear.

Therefore, the present paper performs an ablation study of post-exploration in both tabular and deep RL settings on both discrete and continuous tasks. Experiments in a range of MiniGrid and Mujoco tasks show that post-exploration provides a strong isolated benefit over standard IMGEP algorithms. As a smaller contribution, we also cast Go-Explore into the IMGEP framework, and show how it can be combined with hindsight experience replay (HER) (Andrychowicz et al., 2017) to make more efficient use of the observed data. In short, our work presents a systematic study of post-exploration and shows its effectiveness on both discrete navigation tasks and continuous control tasks.

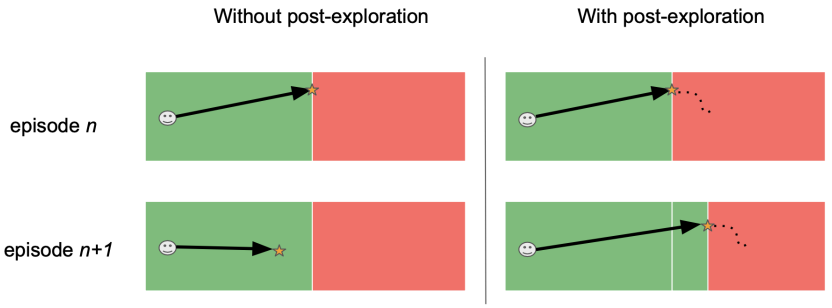

Figure 6.1: Conceptual illustration of post-exploration. Each box displays the entire state space, where green and red denote the (currently) explored and unexplored regions, respectively. **Left**: Goal-based exploration without post-exploration. The top graph shows the agent reaching a current goal, after which the episode is terminated (or the next goal in the green area is chosen). Therefore, in the next episode (bottom) the agent will again explore within the known (green) region), often leaving the unknown (red) area untouched. **Right**: Goal-based exploration with post-exploration. The top figure shows the agent reaching a particular goal, from which the agent now post-explores for several steps (dashed lines). Thereby, the known area (green) is pushed outwards. On the next episode, the agent may now also select a goal in the expanded region, gradually expanding its knowledge of the reachable state space.

6.2 Related Work

Go-Explore is a variant of intrinsically motivated goal exploration processes (IMGEP) (surveyed by Colas et al. (n.d.)) which generally consists of three phases: 1) defining/learning a goal space, 2) sampling an interesting goal from the goal space, and 3) getting to the sampled goal based on knowledge from previous episodes. Regarding the first step, Go-Explore (Ecoffet et al., 2021) pre-defines goals as down-scaled states, and the goal space is adaptively extended by adding new encountered states. Other IMGEP approaches, like Goal-GAN (Florensa et al., 2018) uses a generative adversarial network to learn a goal space of appropriate difficulty. As an alternative, AMIGo (Campero et al., 2021) trains a teacher to propose goals for a student to reach. In this work, we largely follow the ideas of Go-Explore, and pre-define the goal space as the visited state space, since our research questions are directed at the effects of post-exploration.

After a goal space is determined, we need a sampling strategy to set the next goals. Generally, desired next goals should be novel/diverse enough or lie on the border of the agent's knowledge boundary. Example strategies to select next goals include novelty (Ecoffet et al., 2021; Pitis et al., 2020), learning progress (Colas et al., 2019;

Table 6.1: Overview of moment of exploration in IMGEP papers. Most IMGEP approaches, like Goal-GAN (Florensa et al., 2018), CURIOUS (Colas et al., 2019), DIAYN (Pong et al., 2019), MEGA (Pitis et al., 2020) and AMIGo (Campero et al., 2021), only explore during goal reaching. Restore-based Go-Explore, which directly resets to a goal, only explore after the goal is reached. Our work, similar to policy-based Go-explore, explores both during goal-reaching and after goal reaching.

Approach	Exploration	Post-exploration
IMGEP	✓	✗
Restore-based Go-Explore	✗	✓
Policy-based Go-Explore + this paper	✓	✓

Portelas et al., 2020), diversity (Pong et al., 2019; Warde-Farley et al., 2019), and uniform sampling (Eysenbach et al., 2018). In this work, we use novelty (count-based) for sampling, since it has less tuneable hyper-parameters.

In the third IMGEP phase, we need to get back to the proposed goal. One variant of Go-Explore simply resets the agent to the desired goal, but this requires an environment that can be set to an arbitrary state. Instead, we follow the more generic 'policy-based Go-Explore' approach, which uses a goal-conditioned policy network to get back to a goal. The concept of goal-conditioning, which was introduced by (Schaul, Horgan, et al., 2015), is also a common approach in IMGEP approaches. A well-known addition to goal-based RL approaches is hindsight experience replay (Andrychowicz et al., 2017), which allows the agent to make more efficient use of the data through counterfactual goal relabelling. Although Go-Explore did not use hindsight in their work, we do include it as an extension in our approach.

After we manage to reach a goal, most IMGEP literature samples a new goal and either reset the episode or continue from the reached state. The main contribution of Go-Explore was the introduction of post-exploration, in which case we temporarily postpone the selection of a new goal (Table 6.1). While post-exploration is a new phenomenon in reinforcement learning literature, it is a common feature of most planning algorithms, where we for example select a particular node on the frontier, go there first, and then unfold the search tree into unknown terrain.

6.3 Background

We adopt a Markov Decision Processes (MDPs) formulation defined as the tuple $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ (Sutton & Barto, 2018). Here, $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions the agent can take, P specifies the transition function of the environment, and R

is the reward function. At timestep t, the agent observes state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$, after which the environment returns a next state $s_{t+1} \sim P(\cdot|s_t, a_t)$ and associated reward $r_t = R(s_t, a_t, s_{t+1})$. We act in the MDP according to a policy $\pi(a|s)$, which maps a state to a distribution over actions. When we unroll the policy and environment for T steps, define the cumulative reward (return) of the resulting trace as $\sum_{k=0}^{T} \gamma^k \cdot r_{t+k+1}$, where $\gamma \in [0, 1]$ denotes a discount factor. The goal in RL is to learn a policy that can maximize the expected return.

Define the state-action value $Q^\pi(s, a)$ as the *expected* cumulative reward under some policy π when we start from state s and action a, i.e.,

$$Q^\pi(s_t, a_t) = \mathbb{E}_{\pi,P}\Big[\sum_{k=0}^{T} \gamma^k \cdot r_{t+k} | s_t = s, a_t = a\Big] \tag{6.1}$$

Our goal is to find the *optimal* state-action value function $Q^*(s, a) = max_\pi Q^\pi(s, a)$, from which we may at each state derive the optimal policy $\pi^*(s) = argmax_a Q^*(s, a)$. A well-known RL approach to solve this problem is *Q-learning* (C. J. Watkins & Dayan, 1992). Tabular Q-learning maintains a table of Q-value estimates $\hat{Q}(s, a)$, collects transition tuples (s_t, a_t, r_t, s_{t+1}), and subsequently updates the tabular estimates according to

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot \max_a \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t)] \tag{6.2}$$

where $\alpha \in [0, 1]$ denotes a learning rate. Under a policy that is greedy in the limit with infinite exploration (GLIE) this algorithm converges on the optimal value function (C. J. Watkins & Dayan, 1992).

In deep RL, state-action value function $Q(s, a)$ is approximated by a parameterized neural network, denoted by $Q_\phi(s, a)$. While dealing with continuous action space, taking *max* operation over state-action values like in Equation (6.2) is intractable, we could learn a parameterized deterministic policy μ_θ for the agent instead. A deterministic policy maps a state to a deterministic action rather than a distribution over all actions. Then we can approximate $\max_a Q_\phi(s, a)$ with $Q_\phi(s, \mu_\theta(s))$.

$$L(\phi, D) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim D}\Big(r_t + \gamma \cdot \hat{Q}(s_{t+1}, \mu(s_{t+1})) - \hat{Q}(s_t, a_t)\Big)^2 \tag{6.3}$$

Deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015) is an actor-critic method that parameterizes the deterministic policy ('actor') and the state-action value function ('critic') as neural networks. The critic is updated by minimizing the mean squared TD error $L(\phi, D)$ in Equation (6.3), where D is a set of collected transitions. The actor is updated by solving the Equation (6.4).

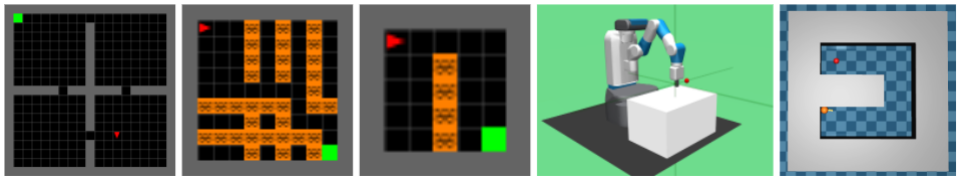

Figure 6.2: FourRooms, LavaCrossing, LavaGap, FetchReach, and PointUMaze. Room layouts of two lava environments are procedurally generated.

$$\max_{\theta} \mathbb{E}_{s_t \sim D} \Big[Q_\phi(s_t, \mu_\theta(s_t)) \Big] \tag{6.4}$$

6.4 Methods

We will first describe how we cast Go-Explore into the general IMGEP framework (Section 6.4.1), and subsequently introduce how we do post-exploration (Section 6.4.2).

6.4.1 IMGEP

Our work is based on the IMGEP framework shown in Algorithm 3. Since this paper attempts to study the fundamental benefit of post-exploration, we try to simplify the problem setting as much as possible, to avoid interference with other issues (such as goal space representation learning, goal sampling strategy design, reward function design, etc). We, therefore, define the goal space as the set of states we have observed so far. For continuous tasks, we discretize the whole state space into bins, so the goal space for continuous control tasks is the set of bins we have observed so far. Firstly, a bin will be sampled, then we further uniformly sample a goal from that bin. The goal space G is initialized by executing a random policy for one episode, while new states/bins in future episodes augment the set.

For goal sampling, we use a count-based novelty sampling strategy which aims to sample goals that are less visited more often and more visited less often from the goal space. We therefore track the number of times $n(g)$ we visited a particular goal g, and specify the probability of sampling it $p(g)$ as

$$p(g)_{g \in G} = \frac{(\frac{1}{n(g)})^\tau}{\sum_{g \in G} (\frac{1}{n(g)})^\tau}. \tag{6.5}$$

Algorithm 3 IMGEP with post-exploration and hindsight (blue part is post-exploration)

Initialize: Goal-conditioned RL agent RL, environment `Env`, episode memory M, goal space G, goal sampling probability $p(g)$, goal-conditioned reward function $R_g(s, a, s')$.
while training budget left **do**
  $g \sim p(g)$ {Sample a goal g from the goal space G according to the goal sampling probability $p(g)$.}
  $s =$ `Env`.reset() {Reset the environment.}
  Execute a roll-out using RL and collect transitions T along the way.
  Update G and $p(g)$ using T. {Extend G by adding new encountered states from T.}
  Store collected transitions T to M.
  if g is reached **then**
    Execute a fixed number of steps using the random policy and collect transitions T_r along the way.
    Update G and $p(g)$ using transitions T_r.
    Store collected transitions T_r to M.
  end if
  Sample a batch B from M and hindsight relabel according to the reward function R_g.
  Update RL agent RL using batch B.
end while

Here, τ is a temperature parameter that allows us to scale goal sampling. When τ is 0, goals will be sampled uniformly from the goal space, while larger τ will emphasize less visited goals more.

Note that we could also use other methods, both for defining the goal space and for sampling new goals (for example based on curiosity or learning progress).

To get (back) to a selected goal, we train a goal-conditioned policy, for tabular settings we use Q-learning, and for deep RL settings we use DDPG. Agents are trained on the goal-conditioned reward function R_g, which is a one-hot indicator that fires when the agent manages to reach the specified goal:

$$R_g(s, a, s') = 1_{s'=g}. \tag{6.6}$$

Note that different goal-conditioned rewards functions, and different update methods are of course possible as well.

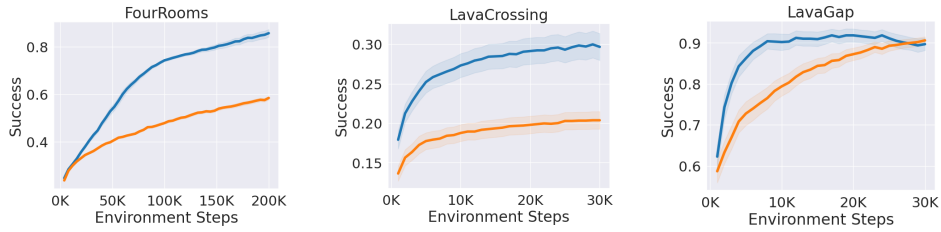

Figure 6.3: IMGEP agents with (blue) and without (orange) post-exploration. **Left**: Performance on FourRooms, with results averaging over three different values of ϵ $(0, 0.1, 0.3)$. **Middle**: Performance on LavaCrossing, averaged over 10 different environment seeds. **Right**: Performance in LavaGap, averaged over 10 different environment seeds. Overall, agents with post-exploration outperform agents without post-exploration.

6.4.2 Post-exploration

The general concept of post-exploration was already introduced in Figure 6.1. The main benefit of this approach is that it increases our chance to step into new, unknown terrain. Based on this intuition, we will now discuss when and how to post-explore, as well as how we utilize data collected during the post-exploration.

As is directly visible from Figure 6.1, post-exploration is aiming to expand the agent's knowledge boundary further when necessary. Since the less visited goals are sampled more often, when a selected goal is reached by the agent, that means the agent's knowledge boundary (the ability to reach goals) is already near the selected goal, then we decide to expand the boundary by doing post-exploration. In this work, we simply let the agent post-explore every time the selected goal is reached. More constraints can be applied here, such as only doing post-exploration when the reached goal is lying on the border of the unknown area or it is the bottleneck of a graph of the visited states, etc. We leave the design of smarter strategies for future work.

We assume that post-exploration could lead the agent to step into new naive areas, thus it is important to design an effective way to do post-exploration. In Go-Explore (Ecoffet et al., 2021), they mentioned random post-exploration is a simple and effective way. So in this work, we also use the random policy during the post-exploration. The agent will take a fixed number of random actions for post-exploration.

The agent in principle only observes a non-zero reward when it successfully reaches the goal, which may not happen at every attempt. We may improve sample efficiency (make more efficient use of each observed trajectory) through *hindsight experience relabelling* (HER) (Andrychowicz et al., 2017). With hindsight, we imagine states in our observed trajectory were the goals we actually attempted to reach, which allows us

to extract additional information from them ('if my goal had been to get to this state, then the previous action sequence would have been a successful attempt'). For tabular settings, we choose to always relabel 50% of the entire trajectory (goal reaching plus post-exploration), i.e., half of the states in each trajectory are imagined as if they were the goal of that episode. We choose to always relabel the full post-exploration part, because it likely contains the most interesting information, and randomly sample the remaining relabelling budget from the part of the episode before the goal was reached. For deep RL settings, we use the 'future' relabel strategy from the original HER work (Andrychowicz et al., 2017).

6.5 Experiments

6.5.1 Experimental Setup

We test our work on three MiniGrid environments (Chevalier-Boisvert et al., 2018) (tabular settings), and two Mujoco environments (Kanagawa, 2021; Plappert et al., 2018) (with using the function approximation), visualized in Figure 6.2. Results on two lava environments are averaged over 10 seeds, i.e., 10 independently drawn instances of the task. For evaluation, not like in Go-Explore where the agent needs to reach a specific final goal (the state with the highest score), we instead test the ability of the agent to reach every possible state in the whole state space. This checks to what extent the goal-conditioned agent is able to reach a given goal, when we execute the greedy policy (turn exploration off). All our results report the total number of environment steps on the x-axis. Therefore, since an episode with post-exploration takes longer, it will also contribute more steps to the x-axis (i.e., we report performance against the total number of unique environment calls). Curves display mean performance over time, including the standard error over 5 repetitions for each experiment (10 repetitions for DRL experiments).

6.5.2 Hyper-parameter Settings

Tabular Q-learning

All hyper-parameters we used for tabular Q-learning in this work are shown in Table 6.2. Learning rate α and discount factor γ are for Q-learning update (Equation (6.2)). τ is the temperature for goal sampling. (Equation (6.5)). ϵ is the exploration factor in ϵ-greedy policy. We average results over 3 different values of ϵ. p_{pe}

is the percentage of the whole trajectory that the agent will post-explore. $p_{pe} = 0.5$ means the agent will post-explore $0.5 * l$ steps after the given goal is reached, l is the length of the trajectory the agent takes to reach the given goal.

Parameters	Values
learning rate	0.1
discount factor	0.99
τ	0
ϵ	0, 0.1, 0.3
p_{pe}	0.5

Table 6.2: All hyper-parameters we used for tabular Q-learning experiments.

DDPG

All hyper-parameters we used for DDPG agent are shown in Table 6.3. The learning rate α is for both the policy and the critic update. τ is the temperature for goal sampling. (Equation (6.5)). Random ϵ is the probability of taking a random action and noise ϵ is the probability of adding noise to selected actions. n_{pe} is the number of steps that the agent will post-explore. n_{bins} is the number of bins we discretize the whole goal space on each dimension. Batch size is the size of the batch sampled for training every time. Replay k is the portion of data we will relabel. If $k = 4$, that means we will relabel 4/5 of the whole sampled training data. The replay strategy we used here is 'future'.[1]

Parameters	FetchReach	PointUMaze
learning rate	0.001	0.001
discount factor	0.98	0.98
τ	0	0.01
random ϵ	0.01	0.1
noise ϵ	0.01	0.1
n_{pe}	30	50
n_{bins}	20	100
batch size	16	2
replay k	4	4
replay strategy	future	future

Table 6.3: All hyper-parameters we used for DDPG agents.

[1]For the remaining parameters of DDPG agents, we use the default settings from https://github.com/TianhongDai/hindsight-experience-replay.

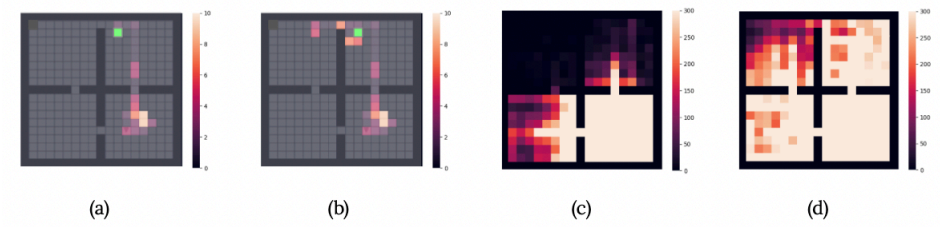

Figure 6.4: Comparison of characteristic traces (a,b) and coverage (c,d) for agents without post-exploration (a,c) and with post-exploration (b,d). Colour bars indicate the number of visitations, green square indicates the selected goal in a particular episode. (a): Standard exploration towards a goal without post-exploration. (b): With post-exploration, the agent manages to reach the next room. (c): Coverage after 200k training steps without post-exploration. (d): Coverage after 200k training steps with post-exploration. The boundary of the coverage with post-exploration clearly lies further ahead.

6.5.3 Results

Figure 6.3 compares an IMGEP agent with post-exploration to an IMGEP agent without exploration in the three MiniGrid environments. On all three environments, the agent with post-exploration outperforms the baseline agent significantly (on its average ability to reach an arbitrary goal in the state space).

A graphical illustration of the effect of post-exploration is provided in Figure 6.4. Part a shows a characteristic trace for an agent attempting to reach the green square *without* post-exploration, while b shows a trace for an agent *with* post-exploration. In a, we see that the agent is able to reach the goal square, but exploration subsequently stops, and the agent did not learn anything new. In part b, the agent with post-exploration subsequently manages to enter the next new room. This graph, therefore, gives a practical illustration of our intuition from Figure 6.1.

To further illustrate this effect, in Figure 6.4, c and d show a visitation heat map for the agent after 200k training steps, without post-exploration (c) and with post-exploration (d). The agent with post-exploration (c) has primarily explored goals around the start region (in the bottom-right chamber). The two rooms next to the starting room have also been visited, but the agent barely managed to get into the top-left room. In contrast, the agent with post-exploration (d) has extensively visited the first three rooms, while the coverage boundary has also been pushed into the final room already. This effect translates to the increased goal-reaching performance of post-exploration visible in Figure 6.3. We think such a principle holds in other environments as well.

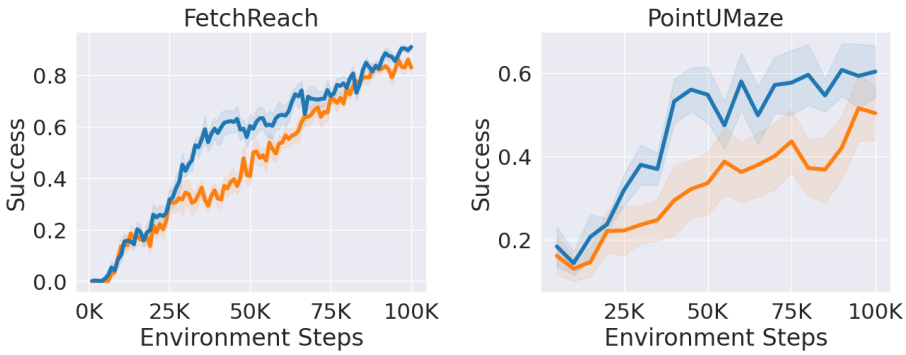

Figure 6.5: IMGEP agents with (blue) and without (orange) post-exploration. **Left**: Performance on FetchReach, with results averaging over 10 different random seeds. **Right**: Performance on PointUMaze, averaged over 10 different random seeds. Overall, agents with post-exploration outperform agents without post-exploration.

Comparisons between agents with post-exploration and ones without on Mujoco environments are shown in Figure 6.5 and we use the DDPG agent with HER as the baseline (orange lines in Figure 6.5) to compare with. With post-exploration, agents (blue lines in Figure 6.5) outperform baseline agents in both FetchReach and PointUMaze environments. We observed that at the beginning and the end, two agents (with post-exploration and without) tend to have similar performances. We interpret the phenomenon as follows: the agent is not able to reach any selected goals in the beginning so there are no post-exploration and no benefits; then gradually the agent starts being able to reach selected goals and post-exploration comes in thus further improving the performance; in the end, when the whole state space is been entirely discovered post-exploration will not add benefits anymore. We think this phenomenon happens in environments where the baseline agent (without post-exploration) can also fully discover the whole state space over time, however, it would less happen in environments where there are many bottlenecks so that the baseline agent can hardly fully discover the entire state space.

By adding post-exploration, the agent is more likely to step into new unknown areas and thus can further improve the state coverage. Ideally, we want the agent's visitation of the state space to be as uniform as possible. More specifically, the higher entropy indicates that the visitation of the whole state space is more uniform. The

entropy is computed using the equation below:

$$-\sum_{i=1}^{N} p(x_i)\log(p(x_i)) \tag{6.7}$$

where the whole state space is discretized into N bins and $p(x_i) = n(x_i)/\sum_{i=0}^{N} n(x_i)$, $n(x_i)$ is the number of visitations of the i^{th} bin. Figure 6.6 shows the entropy on agents' visitation of the state space during the training. With post-exploration (blue lines), the entropy always lies above the one without (orange lines). Therefore, the principle (Figure 6.4) that we saw in the discrete navigation tasks is still true in continuous control tasks.

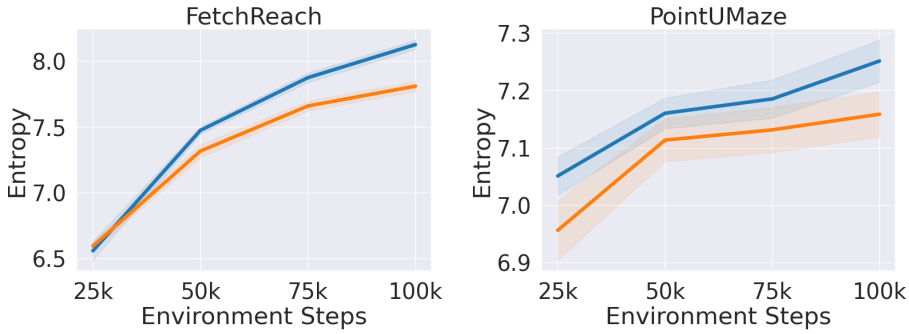

Figure 6.6: The entropy of the visitation on the state space for the agent with (blue) and without (orange) post-exploration. **Left**: Entropy on FetchReach, with results averaging over 10 different random seeds. **Right**: Entropy on PointUMaze, averaged over 10 different random seeds. Adding post-exploration can increase the entropy of the agent's visitation on the state space.

6.6 Conclusion and Future Work

An intrinsically motivated agent not only needs to set interesting goals and be able to reach them but should also decide whether to continue exploration from the reached goal ('post-exploration'). In this work, we systematically investigated the benefit of post-exploration in the general IMGEP framework under different RL settings and tasks. Experiments in several MiniGrid and Mujoco environments show that post-exploration is not only beneficial in navigation tasks under tabular settings but also can be scaled up to more complex control tasks with neural networks involved. Ac-

cording to our results, agents with post-exploration gradually push the boundaries of their known region outwards, which allows them to reach a greater diversity of goals. Moreover, we realize that 'post-exploration' is a very general idea and is easy to be plugged into any IMGEP method. Researchers should put more attention to this idea and consider using it when possible.

The current paper studied post-exploration in a simplified IMGEP framework, to better understand its basic properties. In the future, it would be interesting to plug post-exploration into other existing IMGEP methods directly to show its benefits. Moreover, our current implementation uses random post-exploration, which turned out to already work reasonably well. So an interesting direction for future work is to post-explore in a smarter way, like using macro actions or options. For example, in tasks where we need to control a more complex agent such as an ant or a humanoid robot, then random actions will never help the agent stand up properly and it is even harder to lead the agent step into new areas. Another promising future direction is to investigate the 'adaptive' post-exploration. Intuitively, post-exploration will be most likely useful when it starts from a state that is new or important enough and the agent should post-explore more if the reached area is more naive, etc. In short, the agent should adaptively decide when and for how long to post-explore based on its own knowledge boundary or properties of reached goals. Altogether, post-exploration seems a promising direction for future RL exploration research.