WHEN TO GO, AND WHEN TO EXPLORE: THE BENEFIT OF POST-EXPLORATION IN INTRINSIC MOTIVATION

Anonymous authors
Paper under double-blind review

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 systematic study of post-exploration, answering open questions that the Go-Explore paper did not answer yet. First, we study the isolated potential of post-exploration, by turning it on and off within the same algorithm. Subsequently, we introduce new methodology to adaptively decide *when* to post-explore and for *how long* to post-explore. Experiments on a range of MiniGrid environments show that post-exploration indeed boosts performance (with a bigger impact than tuning regular exploration parameters), and this effect is further enhanced by adaptively deciding when and for how long to post-explore. In short, our work identifies *adaptive* post-exploration as a promising direction for RL exploration research.

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 Fig.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., 2020). 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 and possible extensions of the approach. First of all, Go-Explore was compare 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. Second, it also remains unclear *when* we should post-explore (Go-Explore does this at every trial) and *for how long* we should post-explore (Go-Explore does this for a fixed number of steps).

Therefore, the present paper studies *adaptive* post-exploration. In particular, we make the decision to post-explore a function of the *novelty* of the reached goal, and the depth of post-exploration a function of the length of the goal-reaching part of the episode. Experiments in a range of MiniGrid tasks show that post-exploration provides a strong isolated benefit over standard IMGEP algorithms, which can be further enhanced by adaptively deciding when and for how long to post-explore. As

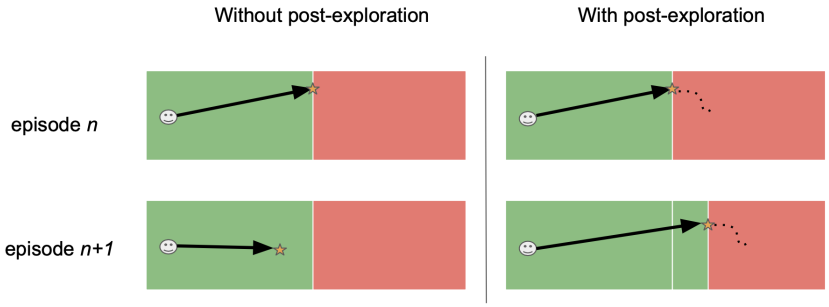

Figure 1: Conceptual illustration of post-exploration. Each box displays the entire state space, where green and red denote the (currently) explored and unexplored regions, respectively. **Left**: Goal-based exploration without post-exploration. The top graph shows the agent reaching a current goal, after which the episode is terminated (or a next goal in the green area is chosen). Therefore, at 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 a next episode, the agent may now also select a goal in the expanded region, gradually expanding its knowledge of the reachable state space.

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 identifies *adaptive* post-exploration as a promising direction for future RL exploration research.

2 Related Work

Go-Explore is a variant of a intrinsically motivated goal exploration processes (IMGEP) (surveyed by Colas et al. (2020)) which generally consists of three phases: 1) defining/learning a goal space, 2) sampling an interesting goal from this space, and 3) getting to the sampled goal based on information from previous episodes. Regarding the first step, Go-Explore (Ecoffet et al., 2021) uses a predefined goal space based on a downscaling of the state space. Other IMGEP approaches, like Goal-GAN (Florensa et al., 2018) use 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 use a predefined goal space, since our research questions are directed at the effects of post-exploration.

After a goal space is determined, we next need a sampling strategy to set the next goal. Example strategies to select the next goal include novelty (Ecoffet et al., 2021; Pitis et al., 2020), learning progress (Colas et al., 2019; 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 follow the last approach, since it 1) gave good performance and 2) puts pressure on the agent to adaptively decide when to post-explore.

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 which 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 et al. (2015), is also the 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 resets the episode or continues from the reached state. The main contribution of Go-Explore was the

Table 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), 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 during goal reaching**	**Post-exploration**
IMGEP	✓	✗
Restore-based Go-Explore	✗	✓
Policy-based Go-Explore + this paper	✓	✓

introduction of post-exploration, in which case we temporarily postpone selection of a new goal (Tab. 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.

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. 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.,

$$Q^\pi(s, a) = \mathbb{E}_{\pi,P}\Big[\sum_{k=0}^{T} \gamma^k \cdot r_{t+k} | s_t = s, a_t = a\Big] \tag{1}$$

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 $\pi^*(s) = \arg\max_a Q^*(s, a)$. A well-known RL approach to solve this problem is *Q-learning* (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

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot \max_a \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t)] \tag{2}$$

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 (Watkins & Dayan, 1992).

4 METHODS

We will first describe how we cast Go-Explore into the general IMGEP framework (Sec. 4.1), and subsequently introduce our new methodology to adaptively decide when and for how long to post-explore (Sec. 4.2).

4.1 IMGEP

Our work is based on the IMGEP framework shown in Appendix A.4, Alg 2. 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. We therefore choose to study tabular RL problems, in which we define the goal space as the set of states we have observed so far. The

goal space G is initialized by executing a random policy for one episode, while new states in future episodes augment the set.

For goal sampling we take uniform samples from the goal space, as for example also used by Eysenbach et al. (2018). This approach empirically provided best performance, and also increases the impact of the decision when to post-explore. Note that we could also use other methods, both for defining the goal space and for sampling new goals (for example based on novelty or learning progress), but these choices allow us to focus on the benefit of post-exploration.

To get (back) to a selected goal, we train a tabular goal-conditioned Q-learning agent. The update rule of the value function is similar to the standard Q-learning update shown in Eq. 2, except that Q is conditioned on the selected goal g, i.e., $Q(s, a, g)$. 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

$$R_g(s, a, s') = \mathbb{1}_{s'=g}, \tag{3}$$

Note that different goal-conditioned rewards functions, different back-up strategies and/or the use of function approximation are of course possible as well.

4.2 Post-exploration

The general concept of post-exploration was already introduced in Figure 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 new methodology to adaptively decide *when* to post-explore and for *how long* to post explore.

When to post-explore As is directly visible from Figure 1, post-exploration is likely most beneficial when performed from a state at the border of our currently explored region. After all, when we perform post-exploration from a state in the middle of the known region, it will likely only waste resources (and not discover anything new), while post-exploration from the border does have much potential. We propose to use the *novelty* of the particular state as a measure of its location in the explored region: when a state is only visited a few times, then it is more likely to reside at the border of our explored region (although other measures, like a connectivity graph of the visited states, is also possible).

We therefore propose to make the probability to post-explore in a particular reached goal g a function of the number of times we have visited that particular goal. We therefore track the number of times we visited a particular goal $n(g)$, and specify the probability to post-explore (*when* to post-explore) as

$$p_{pe}(g) = \left(\frac{1}{n(g)}\right)^{\beta}, \beta \in [0, \infty). \tag{4}$$

Here, β is a temperature parameter that allow us to scale the amount of post-exploration. When β is 0, the agent will always post-explore after reaching the goal, while $\beta \to \infty$ ensures the agent will never post-explore.

How long to post-explore We will denote the number of steps an agent post-explores after reaching a goal as n_{pe}. While Go-Explore always explored for a fixed number of steps (n_{pe} fixed), we hypothesize that the duration of post-exploration may also influence performance. Intuitively, the longer the path the agent took to reach the given goal, the more time it may want to spend in post-exploration (since it is expensive to get back there). We therefore propose to let the agent post-explore for a percentage p_{pe} of the length of the whole trajectory:

$$n_{pe} = p_{pe} \cdot n_{ep}, \tag{5}$$

where n_{ep} is the length of the goal-reaching part of the episode, and p_{pe} is a hyperparameter of our algorithm.

Figure 2: FourRooms, LavaCrossing and LavaGap. The latter two environments are procedurally generated based on environment seeds.

Hindsight relabelling 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 relabelling* (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'). Pseudocode for hinsight relabelling is provided in Appendix A.4, Alg. 1. In particular, 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.

5 Experimental Setup

We test our work on three MiniGrid environments (Chevalier-Boisvert et al., 2018), visualized in Fig. 2. The `MiniGrid-FourRooms-v0` only has a single lay-out, while the `MiniGrid-LavaCrossingS11N5-v0` and `MiniGrid-LavaGapS7-v0` environments are procedurally generated. Results on these latter two environments are averaged over 10 seeds, i.e., 10 independently drawn instances of the task. For evaluation, we test the ability of the agent to reach every possible state in the state space. This checks to what extend the goal-conditioned value function 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. For all experiments, we set $\epsilon = 0.1$ (exploration during goal reaching), $\epsilon_{pe} = 1.0$ (full exploration during post-exploration), $\beta = 0$, and $p_{pe} = 0.5$, unless otherwise noted. Full details on hyperparameters can be found in Appendix. A.3.

6 Results

We split our results up into five research questions: 1) does post-exploration work in general, 2) to what extend should the agent also explore during goal reaching, 3) when should we post-explore, 4) for how long should we post-explore, and 5) is post-exploring also beneficial in continuing tasks, where we sample a next goal without resetting the agent to an initial state. Each research questions will be discussed below.

6.1 Does post-exploration work?

Fig. 3 compares an IMGEP agent with post-exploration to an IMGEP agent without exploration in the three test 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).

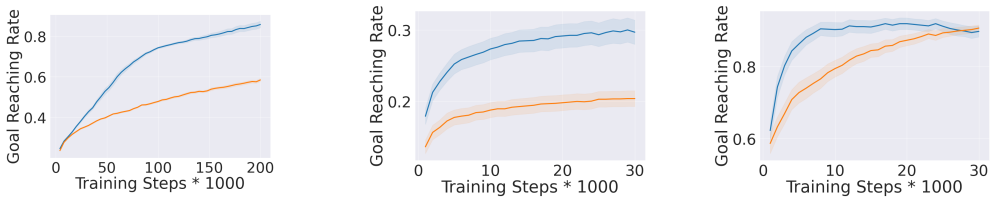

Figure 3: IMGEP agents with (blue) and without (orange) post-exploration. **Left**: Performance on FourRooms, with results average over three different values of ϵ $(0, 0.1, 0.3)$. **Middle**: Performance on LavaCrossing, averaged over 10 different environment seeds. **Right**: Performance in LavaGap, averaged over 10 different environment seeds. Overall, agents with post-exploration outperform agents without post-exploration.

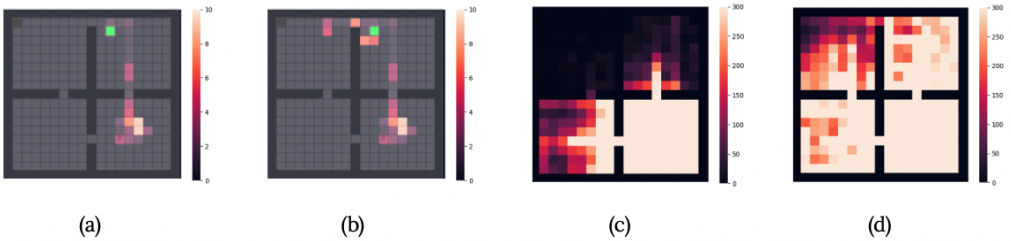

Figure 4: Comparison of characteristic traces (a,b) and coverage (c,d) for agents without post-exploration (a,c) and with post-exploration (b,d). Colour bars indicate the number of visitations, green square indicates the selected goal in a particular episode. (a): Standard exploration towards goal without 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.

A graphical illustration of the effect of post-exploration is provided in Fig. 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 room. This graph therefore gives a practical illustration of our intuition from Fig. 1.

To further illustrate this effect, Fig. 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 Fig. 3.

6.2 When should the agent explore?

Exploration and post-exploration play different roles in the IMGEP approach. Standard exploration (during goal reaching) may help improve our ability to reach a particular goal, while post-exploration especially helps to improve our ability to reach novel terrain. We therefore investigate whether both types of exploration should be jointly present, and to what extend they contribute to overall performance in our test environments.

Fig. 5, left, shows experiments on the FourRooms environments with (blue) and without (orange) post-exploration, where we also vary the amount of exploration (ϵ of an ϵ-greedy policy) during goal reaching. $\epsilon = 0$ implies there is no exploration at all during goal reaching, while $\epsilon = 1$ implies full exploration (and therefore no exploitation at all during goal reaching).

In general, we see that post-exploration has a much stronger effect on performance than the amount of exploration during goal reaching (the quantity that is typically tuned in (goal-based) RL experiments). Only when we set $\epsilon = 1.0$ (full exploration during goal reaching) do we lose performance, which of course makes sense, since the agent then hardly manages to get back to the selected goal. Similar results on other environments are shown in Appendix. A.

6.3 When should the agent post-explore?

We next investigate when to actually post-explore, i.e., whether it is beneficial to make the probability of post-exploration dependent on the novelty of a reached goal (Eq. 4). Results of these experiments on the FourRooms environment are shown in Fig. 5, right. For clarity, $\beta = 0$ means that the agent will always post-explore, while a higher value of β will decay the probability of post-exploration with more visits, an effect that becomes stronger the further β increases.

We see that selective post-exploration ($\beta = 0.01$ and $\beta = 0.05$) slightly outperforms full post-exploration ($\beta = 0$). This selective post-exploration decides to not post-explore when it is probably not beneficial, and therefore saves it samples for a next episode. However, we should clearly not overdo it, as a faster decrease in post-exploration probability ($\beta = 1$) clearly hurts performance. Note that the curve of this agent becomes almost flat near the end, probably because the agent does not post-explore anymore at all. Additional results on other environments are available in Appendix. A.

6.4 How long should the agent post-explore?

We next investigate whether the duration of post-exploration should be a fixed number (like in Go-Explore), or should be a function of the duration of goal-reaching (Eq. 5). In the former case we report a fixed n_{pe} (regardless of the length of the trajectory), while in the latter case we report the used proportion p_{pe}.

Fig. 6, left and middle, show the results of these experiments on FourRooms for different values of n_{pe} and p_{pe}. The left part of the graph shows the learning curves, in which we see that $n_{pe} = 20$ and $p_{pe} = 0.8$ achieve best performance, suggesting that both methods are on par. However, the middle of Fig. 6, shows the total number of post-exploration steps (top) and total number of relabelling steps (bottom) in each setting. This graph shows that $p_{pe} = 0.8$ uses roughly as many post-exploration steps as $n_{pe} = 15$, and way less than $n_{pe} = 20$. Therefore, the proportional post-exploration method (p_{pe}) does seem to be more effective at utilizing the post-exploration data.

6.5 Does post-exploration work in continuing environments?

Similar to Go-Explore, we investigated post-exploration primarily in the episodic setting (where we reset an agent after a goal is reached, or after post-exploration has completed). However, many specifications of goal-based RL assume a continuing task, in which there are no full resets, but the

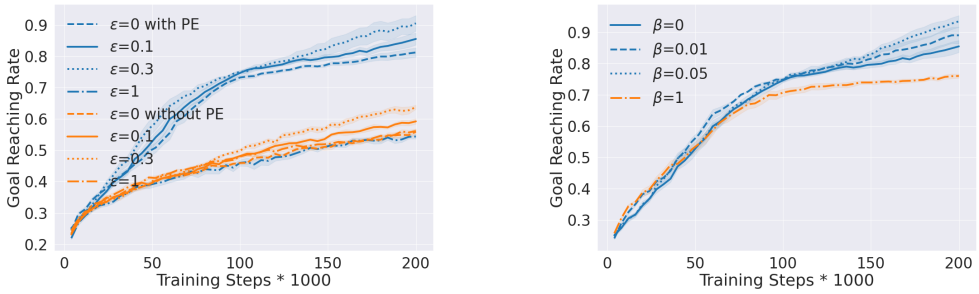

Figure 5: **Left**: Performance of agents with different exploration factors ϵ with and without Post-Exploration (PE) in the FourRooms environment. Adding slight exploration ($\epsilon = (0.1, 0.3)$) can further improve the performance. Adding fully random exploration will lead to similar results with ones without Post-Exploration. **Right**: Performance of agents with different β in the FourRooms environment. $\beta = 0$ is always Post-Exploration. Selective Post-Exploration ($\beta = (0.01, 0.05)$) works better than full Post-Exploration. Too aggressive β will hurt the performance.

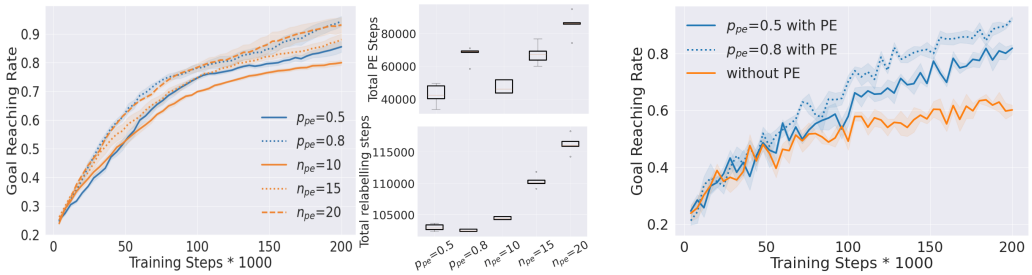

Figure 6: **Left**: Two different ways for Post-Exploration in FourRooms Environment. Left: performance of agents with different Post-Exploration methods. Blues are post-explore with percentage of the whole trajectory (p_{pe}) while oranges are post-explore certain steps (n_{pe}). **Middle**: the top sub figure is total steps that agent post-explores with different Post-Exploration methods; the bottom sub figure is the total steps of hindsight relabelling. Although we keep relabelling steps and Post-Exploration steps similar for two methods, the percentage one performs better. For example, $p_{pe} = 0.5$ and $n_{pe} = 10$. **Right**: Performance of agents with and without post-exploration in the non-episodic continuing FourRooms environment. Agents with post-exploration still outperform agents without post-exploration, likely because the next goal directs agents back into known territory.

agent simply attempts to reach the next sampled goal from the current state. This does give the standard agent (without post-exploration) the ability to make additional steps after reaching a goal, but it will do so in the direction of an already set goal (which may direct it back into known territory, instead of into new terrain). Fig. 6, right, shows results of these experiments. Again, post-exploration outperforms the agent without post-exploration, although the effect is less pronounced compared to the episodic setting. This confirms our hypothesis that setting a next goal is indeed not the same as post-exploration, since the agent will likely just move back into known territory.

7 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 IMGEP framework. Experiments in several MiniGrid environments show that post-exploration is beneficial, and may even have a stronger effect on performance than tuning exploration during goal reaching (which is usually tuned in RL experiments). According to our intuition, agent with post-exploration gradually push the boundaries of their known region outwards, which allows them to reach a greater diversity of goals. Moreover, we find that *adaptive* post-exploration, where we adjust when and for how long we post-explore based on previous data, may further enhance this benefit. Finally, the benefit of post-exploration is retained on continuing tasks, which shows that simply setting a next goal is not the same as proper post-exploration.

The current paper studied post-exploration in the tabular setting, to better understand its basic properties. In future work, post-exploration may be scaled up to higher-dimensional IMGEP problems which require function approximation, a learned goal space, and more complex goal sampling strategies. Moreover, our current implementation uses random post-exploration, which turned out to already work reasonably well. Another interesting direction for future work is to post-explore in a smarter way, for example by trying to set goals outside the known area, which was not possible in the tabular setting, but would be possible when we use function approximation. Altogether, post-exploration seems a promising direction for future RL exploration research.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Andres Campero, Roberta Raileanu, Heinrich Kuttler, Joshua B. Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. In *International Conference on Learning Representations*, 2021.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pp. 1331–1340. PMLR, 2019.

Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Intrinsically motivated goal-conditioned reinforcement learning: a short survey. *arXiv preprint arXiv:2012.09830*, 2020.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning diverse skills without a reward function. 2018.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1515–1528. PMLR, 10–15 Jul 2018.

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pp. 7750–7761. PMLR, 2020.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pp. 835–853. PMLR, 2020.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. In *International Conference on Learning Representations*, 2019.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

A APPENDIX

A.1 MORE RESULTS ON LAVACROSSING

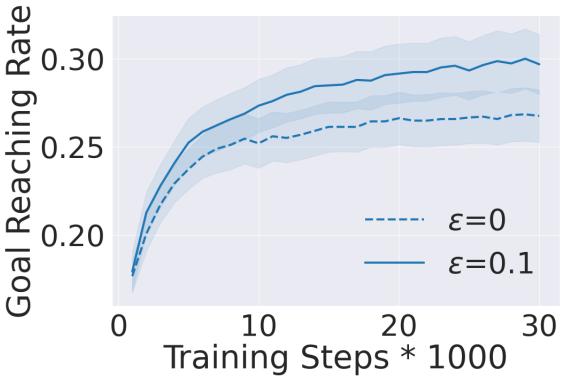

Figure 7: Performance of agents with Post-Exploration with different exploration factor ϵ in LavaCrossing. Adding exploration ($\epsilon = 0.1$) during goal reaching can improve the performance. Each experiment in LavaCrossing is averaged over 10 different environment seeds and 5 repetitions.

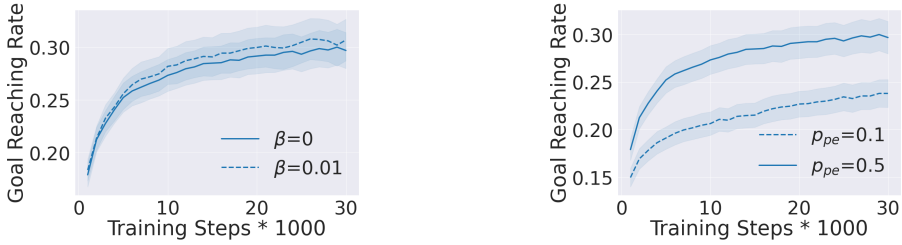

Figure 8: Left: Performance of agents with different β. The selective Post-Exploration ($\beta = 0.01$) slightly outperform full Post-Exploration ($\beta = 0$). Right: Performance of agents with different p_{pe}. More Post-Exploration steps lead to better performance.

A.2 MORE RESULTS ON LAVAGAP

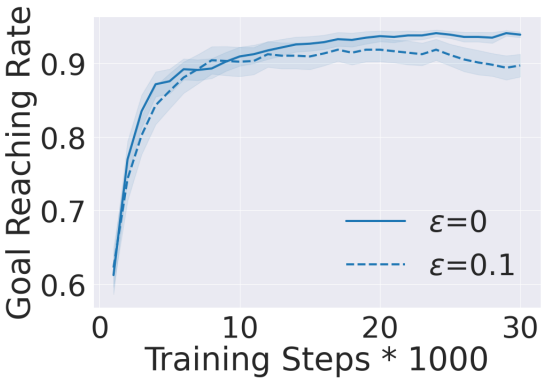

Figure 9: Performance of agents with different exploration factors ϵ in LavaGap. Adding exploration during goal reaching does not improve the overall performance. We think that's because the LavaGap environment is too small so that the exploration can barely bring the agent to new states. Each experiment in LavaCrossing is averaged over 10 different environment seeds and 5 repetitions.

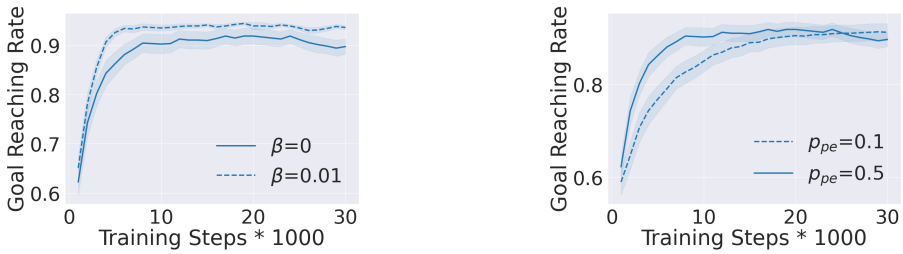

Figure 10: Left: Performance of agents with different β. Selective Post-Exploration ($\beta = 0.01$) outperforms full Post-Exploration ($\beta = 0$). Right: Performance of agents with different p_{pe}.

A.3 Hyper-parameter Settings

All hyper-parameters we used in this work are shown in Tab. 2. Learning rate α and discount factor γ are for Q-learning update (Eq. 2). β is the temperature for selective Post-Exploration (Eq. 4). ϵ is the exploration factor in ϵ-greedy policy. p_{pe} is the percentage of the whole trajectory that the agent will Post-Explore. n_{pe} is the constant number of steps the agent will Post-Explore. Detailed explanation of each parameters can be found in corresponding sections.

Parameters	**Values**
learning rate	0.1
discount factor	0.99
β	0, 0.01, 0.05, 1
ϵ	0, 0.1, 0.3, 1
p_{pe}	0.1, 0.5, 0.8
n_{pe}	10, 15, 20

Table 2: All hyper-parameters we used in this work.

A.4 Algorithms

Alg. 2 is the IMGEP framework we use in this work, where the red part indicates the extension with post-exploration (without the red part, the algorithm simply reduces to standard IMGEP). Alg. 1 shows how we relabel the trajectory and perform update on the relabelled data.

Algorithm 1 Single hindsight relabelling

Initialize: Episode $M = \{g, s_0, a_0, r_0, s_1, a_1 ...\}$, goal-conditioned value function $Q(s, a, g)$, learning rate $\alpha \in \mathbb{R}^+$, total number of relabelled goals k.

repeat k times:
  $i \leftarrow$ set-index-to-label-as-goal() {avoid duplicates over repetitions}
  $g = s_i$ {set hindsight goal}
  for t in 0 to $(i-1)$:
    $r_t = \mathbb{1}_{s_t = g}$
    $\hat{Q}(s_t, a_t, g) \leftarrow \hat{Q}(s_t, a_t, g) + \alpha \cdot [r_t + \gamma \cdot \max_a \hat{Q}(s_{t+1}, a, g) - \hat{Q}(s_t, a_t, g)]$

Algorithm 2 IMGEP Q-learning with post-exploration and hindsight (red part is post-exploration)

Initialize: Goal-conditioned value function $\hat{Q}(s, a, g)$, environment `Env`, episode memory M, goal space G, the number of post-exploration steps n_{pe}, goal-conditioned reward function $R_g(s, a, s')$, the probability distribution of a goal g being post-explored $p_{pe}(g)$, learning rate $\alpha \in \mathbb{R}^+$, number of hindsight relabels k.

while training budget left **do**
  $g \sim G$ {sample a goal g from G}
  $s \sim p_0(s)$ {reset environment}
  $M \leftarrow \{\}$
  while s not terminal **and** $g \neq s$ **do**
    $a \leftarrow$ `e-greedy` $(\hat{Q}(s, a, g))$
    $s' \leftarrow$ `Env.step`(a) {simulate environment}
    $r \leftarrow R_g(s, a, s')$ {goal-conditioned reward}
    $\hat{Q}(s, a, g) \leftarrow \hat{Q}(s, a, g) + \alpha \cdot [r + \gamma \cdot \max_{a'} \hat{Q}(s', a', g) - \hat{Q}(s, a, g)]$
    M`.append`$(\langle s, a, r, s', g \rangle)$
    G`.update`(s') {Possibly augment goal space with new observation}
    $s \leftarrow s'$ {$\Delta \sim$ Uniform[0,1]}
  end while
  if g is reached **and** $\Delta \leq p_{pe}(g)$ **then**
    for 1 **to** n_{pe} **do**
      a = `RandomAction()` {Post-exploration has random action selection}
      s' = `Env.step`(a)
      $r = 0, g =$ `None`
      M`.append`$(\langle s, a, r, s', g \rangle)$
      $s \leftarrow s'$
    end for
  end if
  `Hindsight`$(M, \hat{Q}(s, a, g), \alpha, k)$ { Alg. 1 }
end while