Temporary Goals for Exploration

Anonymous Author(s)
Affiliation
Address
email

Abstract

Exploration has always been a crucial aspect of reinforcement learning. When facing long horizon sparse reward environments modern methods still struggle with effective exploration and generalize poorly. In the multi-goal reinforcement learning setting, out-of-distribution goals might appear similar to the achieved ones, but the agent can never accurately assess its ability to achieve them without attempting them. To enable faster exploration and improve generalization, we propose an exploration method that lets the agent temporarily pursue the most meaningful nearby goal. Through experiments in four multi-goal environments, including a 2D PointMaze, an AntMaze, and a foraging world, we show that our method can improve an agent's ability to estimate the achievability of out-of-distribution goals as well as its frontier exploration strategy.

1 Introduction

In online reinforcement learning, the agent collects its own training data via exploration. More often than not, effective exploration requires deviations from the agent's greedy policy, as the reward function alone does not guide the agent to obtain sufficient coverage of the state and policy space. This is particularly true in the multi-goal and multi-task reinforcement learning settings [22, 24], where the agent must achieve a wide range of different goals.

There exist many methods to tackle this problem such as parameter space noise [23], curiosity by self-supervised prediction [19], entropy gain [21], and hierarchical exploration [17, 18, 14]. But even these methods can fail to achieve sufficient coverage of the full goal/task space, which may result in goal misgeneralization [6] and overconfidence when pursuing goals that lie outside of the agent's achieved goal distribution. This is because none of these methods explicitly direct the agent toward goals that are never seen by the agent.

An intuitive way to resolve the overconfidence issue is to have the agent attempt goals that lie beyond its known goal distribution so that the agent gathers the data necessary to determine their achievability. However, the naive pursuit of out-of-distribution goals can waste valuable environment interactions. To avoid this, we note that an agent often has significant buffer room when it is pursuing in-distribution goals (e.g., it achieves a goal mid-way through an episode in an undiscounted task). In such cases, exploring nearby, out-of-distribution goals can quickly inform the agent as to their achievability and provide effective exploration into the unknown region, whilst not compromising the agent's ability to achieve its current task.

We leverage this observation to propose a new method that utilizes temporary goals throughout a single rollout to improve goal generalization and exploration efficiency. We show how our method can be integrated with existing multi-goal reinforcement learning methods, and demonstrate empirical improvements over baseline algorithms in four multi-goal environments.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

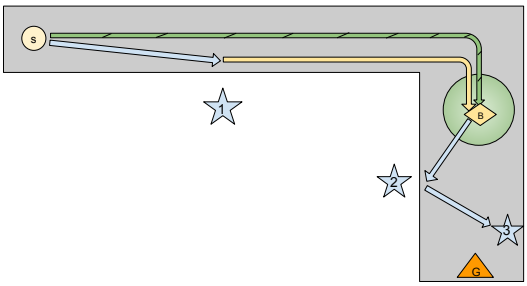


Figure 1: *A diagram of our algorithm in a 2d navigation environment. The grey region is the navigable area with the small yellow circle labeled with "s" representing the start and the yellow diamond (labeled "B") being the agent's behavioral goal. The environment's goal is the Triangle. The green dashed arrow represents the baseline MEGA algorithm [21], where the agent goes to its behavioral goal and explores randomly around it (the big green circle). Light blue and yellow arrows demonstrate a potential route an agent with temporary goals would take, with the temporary goals being the three blue stars labeled 1 to 3. Here the agent first pursues the temporary goal 1 for several steps before switching back to the behavioral goal (yellow path). It then continues to explore by pursuing temporary goals 2 and 3. Temporary goals here allow the agent to learn that the white region (lower left) is unachievable, and also to explore more effectively at the frontier.*

2 Preliminaries

We consider the multi-goal reinforcement learning setting. It can be described by a generalized Markov Decision Process $\mathcal{M} = \langle S, A, T, G, [p_{dg}] \rangle$, where S is the state space, A is the action, T is the transition function, G is the goal space and $[p_{dg}]$ is an optional desired goals distribution as introduced in [21]. We will demonstrate that our method can provide further benefit for state-of-the-art exploration algorithms in long-horizon sparse reward environments by integrating it into the Maximum Entropy Gain exploration strategy (MEGA) [21], which explores by setting a behavioral goal for the agent. In our continuous control examples we use the DDPG algorithm [15], which uses Q-learning for the critic [28] and deterministic policy gradients for the actor [26]. We also utilize Hindsight Experience Replay (HER) [2] to relabel the states stored in the replay buffer and then use them to train the actor and the critic.

Aside from integration with MEGA, our proposed method can be employed in multitask environments such as discrete navigation and resource collection environments that are not explicitly conditioned on a goal state. For these environments, we use deep Q-learning [16] as the baseline.

2.1 The Overconfidence and Ineffective Exploration Problem

Though algorithms such as MEGA [21] can solve long-horizon environments, they can fail at recognizing unachievable goals, especially when they are out-of-distribution. Such agents often sample their behavioral goals from their own buffer. While effective for discovering achievable goals, the agent never explicitly pursues any goal that is theoretically unachievable. As a result, the agent assumes that all nearby goals of an achieved goal are similarly achievable. This is also the case with exploration methods that rely on action space noise [15] or parameter space noise [23], as using noise to explore cannot inform the agent whether a state is unachievable or simply has not been explored.

Algorithms that focus on exploring at the frontier of the agent's capabilities [19, 7, 21] also face the issue of ineffective exploration at the frontier. All of the methods mentioned above essentially explore randomly after arriving at the frontier. This means it becomes exponentially unlikely for the agent to consistently move in any one direction, and therefore they don't explore as effectively as methods that provide temporally-extended exploration at the frontier.

We also observe that the frontier of achievable goals as determined by the agent is typically not optimal, and while en route to such a frontier there are many adjacent states that if pursued would provide the agent with more effective exploration.

3 Methods

We propose a method that periodically checks if there are meaningful goals nearby and then potentially pursues them for N steps. The method is flexible and can serve as an add-on module for many modern reinforcement learning algorithms. The temporary goal swapping serves 3 purposes: (1) it allows the agent to have a more accurate estimate of the achievability of out-of-distribution goals, (2) it helps to identify and pursue potentially more informative frontier goals and, (3) it provides temporally-extended exploration at the frontier.

Our method has variations that can be utilized in different ways depending on the environment and the base algorithm. When the environment is conditioned on a goal and has clearly defined step sizes, then Algorithm 1 can be implemented, and the function TEMPORARY_GOAL_SWAP can be run periodically to find potential goals that will provide better exploration.

Our method first samples C goals $\{g_i\}_{i=1}^{C}$ approximately N steps away. We generate each of these goals g_i according to the Equation 1:

$$g_i = g_a + N\sigma\epsilon^{(i)}, \epsilon_k^{(i)} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(-1.5, 1.5)\ \forall\ k \in \{1, \ldots, D\} \tag{1}$$

First, we sample a random uniform vector $\epsilon^{(i)}$ with the same number of dimensions as the goal, where each element $\epsilon_k^{(i)}$ is independently sampled from a uniform distribution from -1.5 to 1.5. We compute a scalar step size $\sigma = \|g_a - g'_a\|_2$ defined as the Euclidean distance between the current achieved goal, g_a, and the goal achieved one time step earlier, g'_a. Then we take the product of the random vector $\epsilon^{(i)}$ with the number of steps N and the step size σ, and add this vector to our current achieved goal g_a to get a goal that is approximately N steps away.

Out of the C sampled goals, it chooses the one with the lowest density in the agent's past experience:

$$\hat{g} = \underset{g' \in \{g_i\}_{i=1}^{C}}{\arg\min}\ p_\theta(g') \tag{2}$$

The density $p_\theta(g)$ can be estimated with a kernel density estimator fitted to the experience buffer of achieved goals.

The lowest density goal has the potential to provide more efficient exploration as demonstrated in MEGA [21]. To summarize, low-density goals are a natural objective to expand the support of the achieved goal distribution in all directions as fast as possible. By pursuing low-density goals, the agent increases the entropy of its achieved goal distribution, and will eventually cover the support of the desired distribution, thus allowing the agent to achieve the desired objective.

When the temporary goal method is used in conjunction with a value-based method like DDPG [15], we can also consider some form of Q-value cutoff as used in MEGA[21], where we reject the goals

Algorithm 1 Temporary Goal Swap (TGS)

function TEMPORARY_GOAL_SWAP (current_achieved_goal $g_a \in \mathbb{R}^D$, current_behavioral_goal $g_b \in \mathbb{R}^D$, number_of_candidates $C \in \mathbb{N}$, number_of_steps $N \in \mathbb{N}$, step_size $\sigma \in \mathbb{R}^+$):

Sample C candidates $\{g_i\}_{i=1}^{C}$, where $g_i = g_a + N\sigma\epsilon^{(i)}, \epsilon_k^{(i)} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(-1.5, 1.5)\ \forall\ k \in \{1, \ldots, D\}$
$\hat{g} = \arg\min_{g' \in \{g_i\}_{i=1}^{C}} p_\theta(g')$ *# Choose goal with lowest probability density in replay buffer*
if $p_\theta(\hat{g}) < p_\theta(g_b)$ **then**
Collect experience $(s_k, a_k, r_k(s_k, \hat{g}))$ for N steps using $a_k \sim \pi(\cdot|s_k, \hat{g})$
else return

Algorithm 2 Temporary Weight Swap (TWS)

function TEMPORARY_WEIGHT_SWAP (valid_minimum_weight $a \in \mathbb{R}$,valid_maximum_weight $b \in \mathbb{R}$):

Sample a random new weight $\hat{w}$, where $\hat{w}_i \overset{\text{i.i.d.}}{\sim} \mathcal{U}(a, b)\ \forall\ i \in \{1, \ldots, D\}$
Collect experience $(s_k, a_k, r_k(s_k, \hat{w}))$ for N steps using $a_k \sim \pi(\cdot|s_k, \hat{w})$

with Q-values below a certain threshold. This can prevent repeated attempts toward regions the agent has already learned to be unachievable.

After finding a new low-density goal we compare it with its current behavioral goal g_b and if the temporary goal $\hat{g}$ has a lower density then it will be pursued for the next N steps before switching back to the original behavioral goal. This process can be repeated every kN steps until the end of an episode. So in a reversible environment, the temporary goal swap would take a detour of at most $2kN$.

If the behavioral goal is achieved before the end of an episode, then the process can be repeated every N steps to provide temporally extended exploration at the frontier.

The temporary goal method can also be incorporated into multitask environments such as resource management and navigation tasks [1]. By temporarily altering the weight of the resources, the agent would explore different options and acquire resources that it would not have collected under normal conditions. This can be accomplished by Algorithm 2.

In these environments, by pursuing a different set of resources, the agent naturally comes up with a temporally extended course of action to explore. Therefore temporary goals can provide temporally extended exploration at the frontier for both navigation and resource management environments.

In previous work, temporally extended exploration at the frontier is achieved by using temporally extended ϵ-greedy [5], which picks a random action and repeats it for several steps. However by temporarily altering the goal the agent is able to gain experience actually pursuing out-of-distribution goals and therefore can have a better estimate of their achievability, in addition, by picking a low-density goal we steer the agent in the most promising direction. Furthermore, in environments that require resource management, temporally extended ϵ-greedy does not provide a temporally extended course of action. For example in foraging world [3] continuously moving left means the agent is collecting any random resources it happens to come by, which is practically equivalent to ϵ-greedy. In contrast, by altering the resources' values the agent would naturally conduct temporally extended exploration by pursuing a different set of resources.

The temporary nature of the goals provides an upper limit on the amount of potentially wasted environment steps and also allows the method to be integrated into a continued learning setting [25]. The addition of temporary goals allows the agent to explore nearby states of interest while still following the main strategy, thus allowing it to perform satisfactorily on the main goal while further exploring the environment.

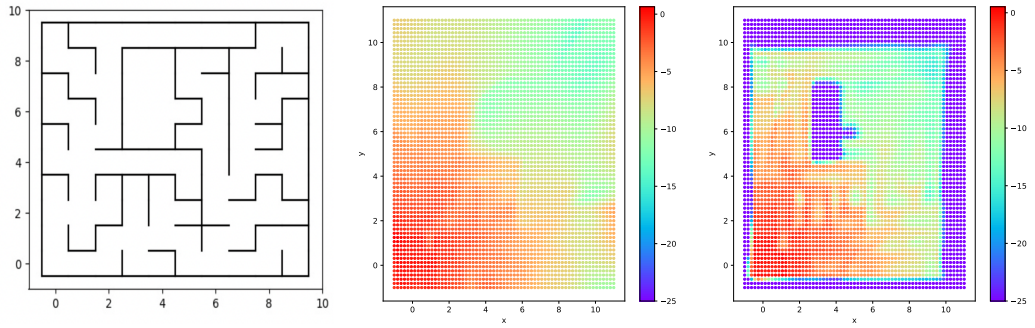
4 Experiments

We compare the proposed method against a baseline reinforcement learning agent that utilizes maximum entropy gain for exploration [21] in three continuous navigation environments. First a 2D PointMaze [27] with an unachievable region. Second, we propose a new environment, MOAT, to demonstrate the issue caused by the lack of temporally extended exploration at the frontier. Improved exploration is also shown in a third, larger-scale AntMaze experiment. Finally, we demonstrate the exploration improvement our method provides in resource management and navigation tasks against a DQN [16] baseline in a modified foraging world environment originally proposed in [3].

4.1 Temporary Goals Provide Better Estimate of Out-of-distribution States' Achievability

To demonstrate that the introduction of temporary goals can help achieve better classification of unachievable goals, we compare the baseline method MEGA [21] and MEGA combined with temporary goals in a maze environment. We blocked off a small portion in the middle from the rest of the maze which will not be achievable by the agent. The maze is also surrounded by walls on all sides, the outside of which is also unachievable. The environment is continuous and the agent can move in any direction. The agent starts at the bottom left corner.

After training both agents for the same amount of steps, they learned to solve the environment at the same rate (Appendix, Figure 7). We can then examine the Q-values of the states in the map. Figure 2b shows that the baseline agent overestimates its ability: inside the inaccessible region, the Q-values do not differ significantly when compared to the surrounding regions. Similarly, the agent is also confident in its ability to achieve goals that lie beyond the boundary of the map.



(a) Maze Environment (b) MEGA (c) MEGA + Temp Goals (*Ours*)

Figure 2: (**a**) A top-down view of the modified PointMaze [27] environment with a unreachable area. The agent observes $s = (x, y)$ coordinate in the maze and is given goal location $g = (g_x, g_y)$. (**b-c**) Value function $V(s, g)$ with fixed state on the bottom left corner and varying goal locations for an agent trained with (**b**) MEGA [21], compared to an agent trained with MEGA with Temporary Goals (*Ours*) (**c**). The unreachable area in the maze is accurately captured with our method.

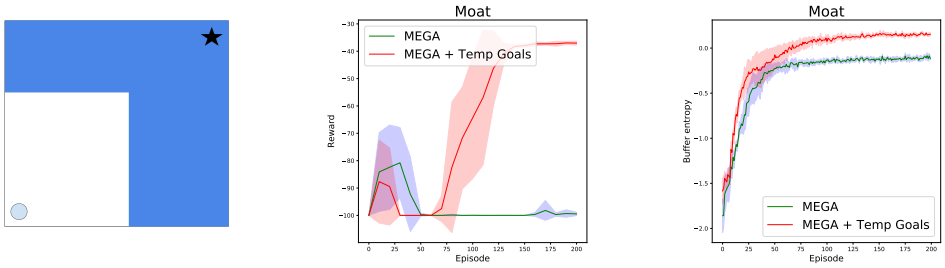
In contrast, when temporary goals are introduced, the agent pursues goals inside the blocked off region and is able to achieve more appropriate Q-values, as shown in Figure 2c. The agent also learns about the boundary of the map and the low Q-values outside of the boundary indicate that the agent is aware that those goals are not achievable.

4.2 Temporary Goals Provide Temporally Extended Exploration at the Frontier

To demonstrate the benefits of temporally extended goals at the frontier we conduct experiments in a novel MOAT environment, which captures a failure case of MEGA, and AntMaze environment.

Moat We created the environment MOAT to illustrate the importance of temporally extended exploration at the frontier. The MOAT is a 2D continuous navigation environment shown in Figure 3a. The blue circle at the bottom left is where the agent starts and the black star is the goal. The blue patch represents the moat and slows the agent's movement speed down by a factor of 10.

While algorithms such as MEGA and "first return, then explore" [7] can provide a temporally extended course of action for the agent to return to the frontier and explore from there, once at the frontier the agent performs random actions to explore further. Therefore in the MOAT environment, the agent



(a) MOAT environment (b) Test rewards (5 seeds) (c) Agent buffer entropy (5 seeds)

Figure 3: (**a**) A top-down view of the MOAT environment. The agent observes $s = (x, y)$ coordinate in the MOAT and is given goal location $g = (g_x, g_y)$. (**b**) Test time reward achieve by MEGA [21], compared to an agent trained with MEGA combined with Temporary Goals (*Ours*). (**c**) The entropy of the agent's replay buffer while attempting to navigate the environment. The shaded region is the standard deviation over 5 seeds.

cannot get far past the boundary of the moat since while performing ϵ-greedy exploration at the frontier the agent moves slowly and does not deviate far past its current state.

Here we compare the performance of MEGA and MEGA with the addition of temporary goals. In Figure 3b the green line represents the test time reward of MEGA without modification, and the shaded region is the standard deviation over 5 seeds.

In this environment, the baseline agent does not make any progress past the frontier of the moat, and therefore cannot solve the environment given a significant amount of time as shown in Figure 3b. The introduction of temporary goals that are a few steps away from the agent's current state allows the agent to explore in one direction consistently, which allows it to make progress in the moat and eventually solve the environment.

The temporary goals also introduce variation in the path the agent takes to get to the frontier, further increasing the entropy of the achieved goal buffer and the effectiveness of exploration. As shown in Figure 3c the agent with temporary goals has higher entropy in its achieved goal buffer, indicating that it has visited a more diverse set of states.

AntMaze These benefits also apply to the more complicated AntMaze [17, 27], where a simulated ant is tasked to navigate to different locations in a ⊐-shaped corridor. The version of AntMaze we used is 16 times larger by area than the one in [8] and takes around 500 steps to complete.

Our baseline MEGA agent performs random actions after reaching the frontier, even when the agent has the basic skills to navigate toward a nearby location. Additionally, since the state and action spaces in this environment are higher dimensional, random actions are not very effective at moving the ant a significant distance. Here the temporary goals continue to provide guidance for the ant towards a useful direction after it has reached its behavioral goal, resulting in faster learning progress, as illustrated in Figure 4.

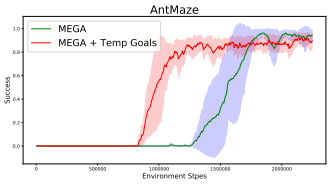


Figure 4: AntMaze test time success (3 seeds)

4.3 Temporary Goals Improve Exploration in Resource Management Tasks

To demonstrate exploration improvement of our method in a resource management and navigation task, we will measure the performance of DQN [16] as a baseline in a modified foraging world environment proposed in [3] and compare that against DQN with temporary goals.

The foraging world environment is a 2D grid-based environment, where the agent can move in all four directions by one grid at a time. The setting we used includes three types of fruits and two types of nutrients. Each type of fruit has a different combination of nutrients. A nutrient's value depends on the amount and type of nutrients the agent has collected.

We adopt the Scenario 1 setting in [3] where nutrient 1 is worth 1 point until the agent collects more than 10 of nutrient 1, then it will have a -1 value. Nutrient 2 is worth -1 point unless the agent has between 5 and 15 nutrient 2s, then nutrient 2 is worth 5 points. This is captured in the desirability function $d_i(x_i)$ which depends on the amount of nutrient x_i for each nutrient i:

$$d_1(x_1) = \begin{cases} +1 & x_1 \leq 10 \\ -1 & x_1 > 10 \end{cases}$$

$$d_2(x_2) = \begin{cases} -1 & x_2 \leq 5 \\ +5 & 5 < x_2 < 25 \\ -1 & x_2 \geq 25 \end{cases}$$

The amount of nutrients (x_1, x_2) also decays as the agent makes a move. This is the default setting of foraging world, and for a more detailed description see [3]. The only modification we made is that we appended the nutrient values $(d_1(x_1), d_2(x_2))$ to the observation space so that the agent can see how much each type of nutrient is worth.

The optimal strategy here would be to collect 10 nutrient 1 and 25 nutrient 2s with an emphasis of collecting nutrient 2. The baseline algorithm does not do this. It mainly focuses on nutrient 1 and never learns to switch to nutrient 2. We hypothesize that this is due to the lack of a temporally extended exploration strategy. After initially learning to not collect any fruit with nutrient 2, the ϵ-greedy exploration strategy is not enough to test that hypothesis at different phases of exploration. Because performing a random move intermittently is very unlikely to result in the agent collecting a fruit containing nutrient 2 while having the correct amount of nutrient 2s, especially when the agent is only attempting to collect nutrient 1. Therefore the agent can only receive a total reward of around 20 as shown in Figure 5.

In this environment, the Temporary Weight Swap method (Algorithm 2) is employed by switching to a random nutrient weight $\hat{w}$ for $N = 10$ steps every $k = 100$ steps. We independently sample a new $\hat{w}$ uniformly where each $\hat{w}_i \overset{\text{i.i.d.}}{\sim} \mathcal{U}(-10, 10)\ \forall\ i \in \{1, \ldots, 2\}$, and set $d_i(x_i) = \hat{w}_i$. The environment is set to 300 steps per episode. The temporary goal method improves exploration by first providing a strong connection between the nutrient values and reward. This is because the agent can now learn how a different nutrient value would affect its reward. Temporary goals also provide a temporally extended exploration strategy. By modifying the nutrient values, the agent is given an incentive to try collecting different types of fruits even though they could be further away.

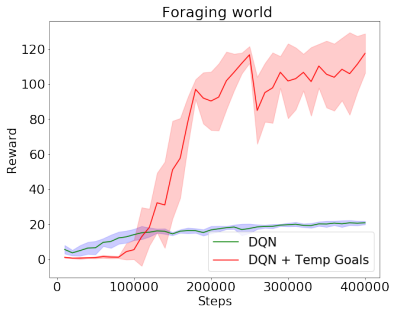


Figure 5: Foraging world test rewards with the default nutrient values (5 seeds).

Figure 5 demonstrates that the agent can overcome the "local optimum" strategy of only collecting fruits with nutrient 1, and thus is able to achieve a much greater reward.

5 Related work

Curiosity The temporary goals method is related to curiosity in non multi-goal reinforcement learning algorithms that maximized the entropy in its state space [11, 13]. These algorithms reward observing novel states, thus suffer from the "noisy-TV" problem [4]. In contrast, since the temporary goals method is focused on the goal space it does not have this issue. The agent employing temporary goals can also learn about its abilities, whereas the curiosity-driven agents cannot.

Goal Relabeling The temporary goals could be compared with goal relabeling [12], which Hindsight Experience Replay (HER) [2] is a form of. Both HER and temporary goals provide the agent with imagined experiences, which allow it to learn about other goals than the one it is pursuing. In navigation environments, temporary goals can be employed with HER simultaneously, as is demonstrated in our experiments. In our modified foraging world HER could help the agent form a connection between nutrient weights and its reward just like temporary goals, by relabeling the nutrient weights. However since HER does not explicitly guide the agent's movement, it alone is not able to help the agent discover the optimal strategy.

Hierarchical reinforcement learning The temporary goals method could behave similarly to a Hierarchical reinforcement learning agent [17, 18] that uses its low-level policy to intermittently pursue unconventional goals. Goal-conditioned hierarchical reinforcement learning algorithm HESS [14] does this by probabilistically utilizing an exploration strategy to pursue a novel subgoal for a predetermined amount of steps. But by directly conditioning the agent on those goals, temporary goals allow the agent to accurately assess the achievability of those goals. Furthermore, temporary goals guide the main policy to explore novel goals and thus forgoes the need for a separate explore policy.

Generative Methods The idea of generating goals for a multi-goal reinforcement learning agent has previously been used in Goal Generative Adversarial Network [8], which is a modified version of generative adversarial network [10]. It employs a goal generator to generate the agent's behavioral goals and a discriminator that is trained to determine if the goals are of appropriate difficulty. In our

method since we only need to generate goals that are nearby, there is no need to train two complex models to capture the intricacies of the environment dynamics. The density of the agent's past experience provides a fast and effective heuristic to generate nearby goals. Furthermore, due to the goal's temporary nature, the cost of an unachievable goal is much lower, while some of the temporary goals can potentially be unachievable, the agent will pursue a baseline line policy such as MEGA [21] where the goal is guaranteed to be achievable.

6 Conclusion

This work proposes a method to improve exploration and generalization in a variety of multi-goal reinforcement learning environments by injecting temporary goals that guide the agent to test the achievability of nearby out-of-distribution goals. This provides a temporally extended course of action for exploration even at the frontier of the agent's capability. We achieve these benefits by temporarily changing the agent's goal, which when combined with the entropy gain heuristic, guides the agent towards previously under-explored states. The strategy solves environments that baseline agents are unable to solve and discovers better strategies than the baseline algorithm in several long-horizon multi-goal environments.

This work has some limitations that should be addressed by future works. Firstly is the issue of safety [9]. The temporary goals method is primarily only applicable in environments that are reversible. In non-reversible environments, the temporary goals can put the agent in a state from which it can no longer achieve the main objective, thus nullifying the benefits. This might be addressed with some form of reversibility detection (something that humans do quite naturally). Second, our experiments are conducted in environments where the goal spaces are predefined and semantically meaningful. It would be interesting to see the performance of temporary goals on a latent goal space learned by the agent.

References

[1] A. Abels, D. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher. Dynamic weights in multi-objective deep reinforcement learning. In *International Conference on Machine Learning*, pages 11–20. PMLR, 2019.

[2] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[3] A. Barreto, D. Borsa, S. Hou, G. Comanici, E. Aygün, P. Hamel, D. Toyama, S. Mourad, D. Silver, D. Precup, et al. The option keyboard: Combining skills in reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[4] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[5] W. Dabney, G. Ostrovski, and A. Barreto. Temporally-extended ϵ-greedy exploration. *arXiv preprint arXiv:2006.01782*, 2020.

[6] L. L. Di Langosco, J. Koch, L. D. Sharkey, J. Pfau, and D. Krueger. Goal misgeneralization in deep reinforcement learning. In *International Conference on Machine Learning*, pages 12004–12019. PMLR, 2022.

[7] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[8] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

[9] J. Garcıa and F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[11] E. Hazan, S. Kakade, K. Singh, and A. Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691. PMLR, 2019.

[12] L. P. Kaelbling. Learning to achieve goals. In *IJCAI*, volume 2, pages 1094–8. Citeseer, 1993.

[13] L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[14] S. Li, J. Zhang, J. Wang, Y. Yu, and C. Zhang. Active hierarchical exploration with stable subgoal representation learning. *arXiv preprint arXiv:2105.14750*, 2021.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[17] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[18] O. Nachum, H. Tang, X. Lu, S. Gu, H. Lee, and S. Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019.

[19] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[20] S. Pitis, H. Chan, and S. Zhao. mrl: modular rl. `https://github.com/spitis/mrl`, 2020.

[21] S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7750–7761. PMLR, 2020.

[22] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[23] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

[24] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

[25] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018.

[26] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[27] A. Trott, S. Zheng, C. Xiong, and R. Socher. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

[28] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

A Implementation Details

We used the modular RL code base [20] for all of our continuous navigation experiments with the default hyper-parameters for all of the environments. The only modification is the introduction of temporary goals. The scoring of the temporary goals is based on the density module as detailed in [21].

For the foraging world environment, our experiment is based on the Option Keyboard code base [3] with default hyper-parameters in the default setup with 3 fruits and 2 nutrients. We ran scenario 1 as detailed in [3] and the only modification is that we made the weight/value of each nutrient part of the observation. When the weights are temporarily swapped, the weights in the observation are swapped to the same values as well. When implementing Algorithm 2 for this experiment, we chose a valid minimum weight of -10 and a valid maximum weight of 10.

All the experiments have been conducted with 5 different seeds, except for AntMaze where 3 seeds are used.

B Low-density Prioritization

The prioritization of low density goals as temporary goals provides meaningful benefits. We compare with using random nearby goals (i.e., setting number of candidates $C = 1$) as temporary goal:

$$\tilde{g} = g_a + N\sigma\epsilon, \epsilon_k \overset{\text{i.i.d.}}{\sim} \mathcal{U}(-1.5, 1.5)\ \forall\ k \in \{1, \ldots, D\}$$

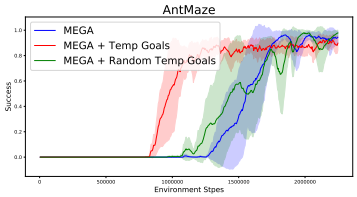


Figure 6: Antmaze test success (3 seeds)

In the AntMaze environment, using random nearby goals $\tilde{g}$ as temporary goals results in slower learning progress as indicated by the green line in 6. The random goals are drawn from the same distribution as detailed in Algorithm 1, but in this experiment, except that we always replace the goal with a random nearby goal.

C PointMaze learning progress

In PointMaze pursuing temporary goals does not appear to affect learning progress, as shown in Figure 7. We hypothesize the reason is that the temporary goals did not distract from the maze exploration due to our cutoff mechanism. And since the maze is fairly small, taking only 50 steps to complete, temporally extended exploration at the frontier is not as important.

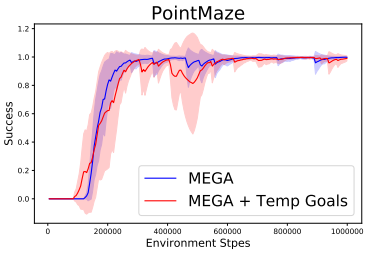


Figure 7: PointMaze test success with and without temporary goals. (5 seeds)