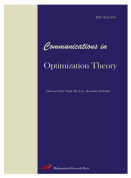
Commun. Optim. Theory 2024 (2024) 1 https://doi.org/10.23952/cot.2024.1

MATHRES Communications in Optimization Theory
Available online at http://cot.mathres.org



REINFORCEMENT LEARNING WITH GOAL-DISTANCE GRADIENT

KAI JIANG[1,*], XIAOLONG QIN[2]

[1]Institute of Fundamental and Frontier Sciences,
University of Electronic Science and Technology of China, Chengdu, China
[2]National Yunlin University of Science and Technology, Douliou, Taiwan

Abstract. Reinforcement learning typically utilizes environmental feedback rewards to train agents. However, rewards in real-world environments are often sparse, and in some cases, certain environments offer no rewards. The majority of current methods struggle to achieve satisfactory performance in environments with sparse or nonexistent rewards. While shaped rewards effectively address sparse reward tasks, their applicability is restricted to specific problems, and learning is also prone to local optima. We introduce a model-free method that addresses the issue of sparse rewards in a general environment without relying on environmental rewards. Our approach employs the minimum state transitions as a distance metric to substitute the environmental reward. Additionally, it introduces a goal-distance gradient to facilitate policy improvement. Building upon the characteristics of our method, we also present a bridge point planning strategy aimed at enhancing exploration efficiency and tackling more complex tasks. Experimental results demonstrate that our method outperforms previous approaches in addressing challenges such as sparse rewards and local optima in complex environments.

Keywords. Bridge point planning; Distance metric; Goal-distance gradient; Sparse rewards.

2020 Mathematics Subject Classification. 68T07.

1. INTRODUCTION

Reinforcement learning (RL) is widely employed to train agents, including robots, in performing tasks through feedback rewards within an environment. For example, training an agent to play FPS (First-person shooting) and Card-Based games [1, 2], to defeat a champion at the game of Go [3], to overtake human scores in 49 Atari games [4], as well as learning control manipulator arm to screw a cap onto a bottle [5] and build blocks [6]. Among the above tasks, perhaps the most central idea of RL is value functions $V(s)$ that represent overall feedback rewards in any state [7]. The agent is trained to optimize the value function, which encapsulates knowledge of rewards, enabling it to learn how to perform a single task. However, in diverse environments, agents often need to tackle multi-goal tasks where rewards are frequently sparse,

*Corresponding author.
E-mail address: jiagron@qq.com (K. Jiang).
Received December 7, 2022; Accepted September 1, 2023.

such as addressing a variety of cooperative multi-agent problems [8]. How can we devise a method capable of effectively handling multi-goal tasks within environments characterized by sparse or nonexistent rewards?

We approach these challenges from a human-like perspective. In reality, when confronted with intricate tasks, humans set goals and continually work towards them. Accordingly, we expect the agent to establish and attain novel goals throughout its self-supervised learning process. To guarantee the agent's ability to engage with the environment and acquire the knowledge necessary for goal achievement during training, selecting a universal distance function becomes imperative. General value functions $V_g(s)$ [9] represents the rewards of any state s in achieving a potential given goal g. In the general RL, the value function is represented by a function approximator $V(s,\theta)$, such as a neural network with parameters θ. The function approximator learns the value associated with the observed state by leveraging the structural information within the state space and extends this knowledge to unobserved values. Goals are typically defined within the agent's achievable scope, resulting in the goal space often sharing a similar structure with the state space. Therefore, the idea of value function approximation can be extended to both states s and goals g by using the general value function approximator $V(s,g,\theta)$ [10]. If the value function is related to the distance, such as the reward is to use the negative Mahalanobis distance in the latent space [6], the smaller the distance, the bigger the reward. We can also employ a generalized function approximation, denoted as $D(s,g,\theta)$, to represent the distance function.

To address tasks within environments featuring sparse or absent rewards, we explore the possibility of expressing rewards in terms of distances applicable across diverse environments. However, training a function capable of precisely assessing the distance between states directly from the raw signals provided by the environment proves to be a formidable challenge. For example, in many visual tasks, the interpretations of the value function indicated by pixel-wise Euclidean distance do not align with the interpretations of the value function in real-world states [11, 10]. Therefore, we propose a solution to this challenge by employing the distance function $D(s,g)$, which represents the minimal number of transition steps between state s and goal g. In each task, the agent is required to navigate through continuous action selection until it reaches a new state and achieves the goal. Consequently, the distance function is proven to be effective in training agents to successfully complete tasks in environments characterized by sparse or absent rewards.

In this paper, our primary contribution lies in presenting a comprehensive approach capable of addressing diverse tasks within environments marked by sparse or nonexistent rewards. To achieve impactful training outcomes, we adapt the conventional framework of standard RL algorithms. We deviate from the action-value function and instead introduce a gradient-based method to enhance policies using distance information. Despite the typical application of *Reward shaping* [12, 13] rendering learning susceptible to local optimization, our approach empowers agents to attain optimal objectives. Moreover, our technique integrates the bridge point theory from SoRB [14], enhancing our method's ability to acquire valuable experiences even in unforeseen tasks.

While there exist numerous RL methods for solving problems within environments, current approaches typically rely on substantial data and specific task-oriented training. However, when

the environment changes, the previously acquired models may become obsolete. The true advancement of RL lies in its application to real-world environments rather than confining it to simulations. To achieve this, our agents must be trained to analyze problems rather than merely relying on past experiences. Our approach introduces a concept for agents to engage in problem analysis, enabling them to break down complex tasks into simpler components.

2. Background

2.1. **Reinforcement Learning.** Reinforcement Learning (RL) involves the interaction of a controlling agent with the environment to maximize rewards. We formulate this problem as a *Markov decision process* (MDP), where in the MDP is defined by a set of *state space* $\mathscr{S}$, a set of *action space* $\mathscr{A}$, a reward function $r : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$, an *initial state distribution* characterized by density $p(s_1)$, and transition probabilities $p(s_{t+1}|s_t, a_t)$. The agent's actions are governed by a policy π_θ parameterized by θ. The policy's objective is to determine the parameters θ that maximize the expected cumulative future rewards from the initial state. This objective is denoted as the performance objective $J(\pi) = \mathbb{E}[R^\gamma|\pi]$, where R represents the cumulative reward and γ is the discount factor. The anticipated total of future rewards is referred to as the *return*: $R_t^\gamma = \Sigma_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k)$, where $0 < \gamma < 1$. We can express the performance objective as an expectation:

$$\begin{aligned} J(\pi_\theta) &= \int_S \rho^\pi(s) \int_A \pi_\theta(s,a) r(s,a) da ds \\ &= \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[r(s,a)]. \end{aligned} \tag{2.1}$$

Value functions are defined to be the expected sum of future rewards: $V^\pi(s) = \mathbb{E}[R^\gamma|S = s; \pi]$ and $Q^\pi(s,a) = \mathbb{E}[R^\gamma|S = s, A = a; \pi]$. These satisfies the following equation called the Bellman equation,

$$\begin{aligned} V(s_t) &= \mathbb{E}_{s\sim\rho(\cdot|s,a), a\sim\pi_\theta}[r(s_t,a_t) + \gamma V(s_{t+1})], \\ Q(s_t,a_t) &= \mathbb{E}_{s\sim\rho(\cdot|s,a), a\sim\pi_\theta}[r(s_t,a_t) + \gamma \underset{a\in\mathscr{A}}{max} Q(s_{t+1},a_{t+1})]. \end{aligned} \tag{2.2}$$

The actor-critic is a widely used architecture based on the policy gradient theorem [7, 15, 16]. The actor-critic consists of two eponymous components. An actor adjusts the parameters θ of the stochastic policy $\pi_\theta(s)$. Instead of the unknown true action-value function $Q^\pi(s,a)$, an action-value function $Q^\omega(s,a)$ is used, with parameter vector ω. A critic estimates the action-value function $Q^\omega(s,a) \approx Q^\pi(s,a)$ using an appropriate policy evaluation algorithm such as temporal-difference learning.

2.2. **Deep Deterministic Policy Gradient.** Deep Deterministic Policy Gradients (DDPG) [17] is an off-policy actor-critic algorithm designed for handling continuous action spaces. DDPG comprises two key components: the *actor* and the *critic*. The actor primarily assumes the role of generating a deterministic policy denoted as μ, while the critic's role revolves around approximating the action-value function Q to facilitate the actor's policy learning process. In contrast to conventional stochastic policy gradients, DDPG employs a deterministic policy gradient. A widely utilized technique, generalised policy iteration [7], underpins the majority of model-free RL algorithms. Temporal-difference learning [15, 18, 19] and Monte-Carlo evaluation are commonly employed to estimate the action-value function $Q^\mu(s,a)$. The policy improvement process involves a greedy maximization of the estimated action-value function, denoted as $\mu^{k+1} = \underset{a}{argmax} Q^{\mu^k}(s,a)$.

2.3. **Hindsight Experience Replay.** Dealing with tasks that encompass multiple distinct goals and sparse rewards has consistently posed a significant challenge in the field of RL. Addressing the issue of sparse rewards typically involves the implementation of an additional informative reward function, guiding the agent towards achieving the desired goals, e.g. $r_g(s,g) = -\|s-g\|_2$. While shaped rewards can indeed address certain issues, applying them to more complex problems remains challenging. From an intuitive perspective, tasks with multiple goals necessitate a larger number of training samples and more efficient use of samples compared to single-goal tasks. Hindsight Experience Replay (HER) [20] introduces a technique for effectively learning from samples in a sparse reward environment. HER not only enhances sample efficiency but also enables learning from sparse reward signals. The approach is founded on training universal policies [21], which accept both the current state and the goal state as inputs. In any trajectory $s_0, s_1, s_2, ..., s_t, ..., g$, the core concept of HER is to store the transition (s_t, a_t, s_{t+1}, g^*) in the replay buffer. Here, g^* is associated not only with the original goal g but also with a subset of other goals.

3. Goal Distance Gradient

To implement the utilization of distance instead of rewards in RL, the following two points must be taken into account. First, in order to replace rewards with distance, the reward function $V(s)$ and action-value function $Q(s,a)$ need to be substituted with the distance function $D(s,g)$. How the distance function should be defined and estimated requires consideration. Can the previous method for evaluating the action-value function also estimate the distance function? Secondly, in the absence of an action-value function, how can we leverage the distance function to enhance the policy? The distance function $D(s,g)$ might not offer an effective gradient for policy improvement. In the subsequent sections, we delve into the estimation of the distance function in Section 3.1, and present the Goal Distance Gradient method in Section 3.2.

3.1. **Estimate the Distance Function by TD.** The distance function $D(s,g)$ is employed to represent the minimal number of transitions required from state s to reach the goal g. In contrast to the value function $V(s)$, the distance function exhibits clear directionality, specifically $s \to g$. However, in reality, the value function $V(s;g')$ also encodes a goal g' that can yield the maximum cumulative reward upon achievement. If we set the immediate reward obtained at each step to 1, $D(s,g)$ and $V(s,g)$ become equivalent. This implies that each step signifies a transfer, reflecting how many transfers have been executed from s to g or how much reward has been accumulated. In such scenarios, we can employ temporal-difference learning techniques for estimating the distance function, such as Sarsa update [7], utilized by the critic in the on-policy deterministic actor-critic algorithm,

$$\delta_t = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{3.1}$$

and Q-learning update is used by critic to estimate the action-value function in the off-policy deterministic actor-critic algorithm,

$$\delta_t = r_t + \gamma Q(s_{t+1}, \mu(s_{t+1})) - Q(s_t, a_t). \tag{3.2}$$

Therefore, we only need to replace the reward r_t with the distance d_t, and then we can evaluate the distance function without considering off-policy or on-policy. We can define d_t as the

number of transfers of $s_t \to s_{t+1}$,

$$\delta_t = d_t + D(s_{t+1}, g) - D(s_t, g). \tag{3.3}$$

However, a significant issue remains unresolved: d_t always maintains a positive value. As iterations continue, there's a risk that the distance function could gradually inflate, consistently veering away from accurate estimation. To rectify this, it becomes crucial to establish a constant reference transition value, serving as the distance baseline between identical states. We denote s_g is the state after reaching the goal g, and the number of transfers d_{s_g} is 0,

$$\begin{aligned} D(s_t, g) &= d_t + D(s_{t+1}, g) d_t, \\ D(s_g, g) &= 0. \end{aligned} \tag{3.4}$$

The count of transfers is registered as 0 solely when the agent remains within its current state without any transitions. In essence, this implies that the distance between identical states s is stipulated as $D(s,s) = 0$. Consequently, the challenge of guiding the agent to proximity with the goal g starting from the initial state s is analogous to minimizing the function $D(s,g)$ to approach 0.

3.2. **Gradients of Distance Policies.** The Policy Gradient algorithm is commonly applied in continuous action spaces to enhance policies through global maximization at each step. In deterministic policy algorithms, an uncomplicated and effective approach to policy improvement involves leveraging the gradient of the action-value function $Q(s,a)$, as opposed to pursuing global maximization. For every state s, the policy parameters θ are updated using the negative gradient $\bigtriangledown_\theta Q(s, \mu_\theta(s))$, resulting in $\mu_\theta(s)$ producing the action a^* that maximizes $Q(s,a^*)$. Nonetheless, the distance function $D(s,g)$ is unable to furnish a gradient for updating the parameters θ of μ_θ in a manner analogous to the action-value function $Q(s,a)$. Consequently, we introduce a novel policy enhancement approach grounded in the gradient of goal distance.

We define deterministic policy $a = \pi(s;\theta)$ and a deterministic model $s' = f(s,a)$. The form of the deterministic Bellman equation for the action-value function is $Q(s,a) = r(s,a) + \gamma Q(f(s,a), \mu(f(s,a)))$. So, the relationship between action-value function $Q(s,a)$ and value function $V(s)$ is as follows:

$$\begin{aligned} Q(s,a) &= r(s,a) + \gamma V(s') \\ &= r(s,a) + \gamma V(f(s,a)). \end{aligned} \tag{3.5}$$

Use value function $V(s)$ instead of action-value function $Q(s,a)$ for policy improvement:

$$\begin{aligned} \mu(s) &= \underset{a \in A}{argmax} Q(s,a) \\ &= \underset{a \in A}{argmax} (r(s,a) + \gamma V(f(s,a))) \\ &\approx \underset{a \in A}{argmax} V(f(s,a)). \end{aligned} \tag{3.6}$$

While the value function itself cannot provide a gradient for policy improvement, the policy is indirectly improved through its relationship with the action-value function. The form of the Distance Bellman equation is $D(s,g) = d + D'(s',g)$, where d refers to the distance or number of transitions. The agent's goal is to obtain a policy that minimizes the distance between the

next state and the goal. Reference Equation 3.5 and 3.6, there are the following formulas:

$$\begin{aligned} D(s,g) &= d + D(s',g) \\ &= d + D(f(s,a),g) \\ &= d + D(f(s,\mu(s,g)),g) \end{aligned} \tag{3.7}$$

and

$$\begin{aligned} \mu(s,g) &= \arg\min_{a\in A} D(s',g) \\ &= \arg\min_{a\in A} D(f(s,a),g) \end{aligned} \tag{3.8}$$

So, the policy improvement method is to use the gradient $\bigtriangledown_\theta D(f(s,\mu_\theta(s,g)),g)$ to minimize the distance function $D(s,g)$. The improved policy $\mu^*(s,g)$ can output an action a that makes the distance $D(s',g)$ from next state s' to the goal g sufficiently small. When $D(s',g)$ is close enough to 0, agent can achieve the goal g.

3.3. **Algorithms.** We summarize the reinforcement learning algorithm for the Goal Distance Gradient (GDG) in Algorithm 1. Our main idea is to enable the agent to perceive the distance of the entire environment, which involves estimating the number of transitions required to reach different states. By setting a random goal, we estimate the number of agent transitions from different states to the goal using the temporal-difference method. We then train the agent using a distance function to move in the direction of decreasing the number of transitions, thereby guiding it to reach the goal. At the start of each episode, we assess whether to discover a bridge point that can link the starting point to the goal, thus shortening the distance between them, based on a predefined probability. Then, we employ a straightforward exploration policy to gather sufficient samples for training, which are stored in the replay buffer. Ultimately, we train our policy model using samples randomly drawn from the replay buffer and refine the model's parameters through fine-tuning. We outline the procedure for identifying bridge points in Algorithm 2. In numerous complex tasks, acquiring valuable experiences for learning can be challenging due to inadequate exploration. To ensure comprehensive exploration and gather a richer set of valuable experiences, we must not confine ourselves to the experiences we have already acquired. Hence, it is imperative to break free from this conundrum of thought and actively seek novel insights that remain undiscovered. Even after the agent has acquired a policy to reach the goal from the starting point, we mustn't relinquish the pursuit of potential improvements in our policy. Leveraging the estimated distance function, we explore whether there exist states that could potentially reduce the distance from the starting point to the goal beyond what is currently known. The existence of such a distinct state indicates the presence of superior policies for the agent to acquire. This state can be designated as a provisional goal, leading us to accumulate experiences that can enhance the policy further. The bridge point serves as a representation of this provisional goal, effectively connecting the starting point and the ultimate goal. In Section 4.2, we elaborate extensively on the application of the bridge point within our method.

4. EXPERIMENTS

Our experiments were devised to tackle and provide insights into the following inquiries: 1) Is the goal-distance gradient effective in enhancing policy performance. 2) How does the GDG

Algorithm 1 Goal Distance Gradient(GDG)

Initialize critic network $D(s,g)$, actor $\mu(s,g)$ and $f(s,a)$ with weights θ^D, θ^μ, and θ^f
Initialize target network D' with weight $\theta^{D'} \leftarrow \theta^D$
Initialize replay buffer R and soft replace rate τ
Initialize bridge point search exploit rate ε
for episode = 1,M **do**
  Sample a goal g and an initial state $s0$
  if $random(0,1) < \varepsilon$ **then**
    Search a bridge point
  end if
  for $t = 0, T-1$ **do**
    Get an action $a_t = \mu(s_t, g) +$ noise
    Execute a_t and observe a new state s_{t+1}
    Store the transition$(s_t, a_t, s_{t+1}, g, d)$ in R
  end for
  for $i = 0, T-1$ **do**
    Sample a random minibatch from R_i
    Set $x_i = d_i + D'(s_{i+1}, g_i)$ and $y_i = f(s_i, a_i)$
    Update network by minimizing the loss:

$$L^D = \frac{1}{N}\sum_i (x_i - D(s_i, g_i))^2$$

$$L^f = \frac{1}{N}\sum_i (y_i - s_{i+1})^2$$

    Update the actor policy:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_{\theta^\mu} D(f(s, \mu(s,g)), g)|_{s_i, g_i}$$

    Update the target networks:

$$\theta^{D'} \leftarrow \tau\theta^D + (1-\tau)\theta^{D'}$$

  end for
end for

method, incorporating bridge planning, perform in intricate tasks. 3) To what extent does the presence of local optima impact the performance of GDG in single-goal tasks.

4.1. **Can goal-distance gradients improve policy?** DDPG (Deep Deterministic Policy Gradient) [17] stands as a classic algorithm within the realm of deterministic policy reinforcement learning techniques. At its core, DDPG aims to determine an action that, upon execution, can lead to a state yielding the highest possible reward. The crux of our algorithm lies in identifying an action that facilitates the swiftest attainment of the goal. In scenarios where each step incurs a reward of -1 and reaching the goal yields a reward of 0, the value function within DDPG

Algorithm 2 Search a bridge point

```
while Bridge point not found do
    bridge ← Search()
    d_sg = D(start, goal)
    d_sb = D(start, bridge)
    d_bg = D(bridge, goal)
    if d_sg > d_sb + d_bg then
        Stop searching
    else if Number of searches exceeded then
        Stop searching
    else
        Continue searching
    end if
end while
```

corresponds in essence to the distance function in our approach, both carrying a similar interpretation. If we consider the combination of the start state s and the goal state g as a single state, then we have $D(s,g) = -V(s\|g)$. In this context, the only distinction between our approach and DDPG is the method for finding an optimal action.

To eliminate the influence of external factors, we have chosen to work within the environment of the 7-DOF Fetch robotics arm [20], where the distance between the starting point and the destination can be accurately computed. The states of the initial and goal points are represented using coordinates in Euclidean space, allowing for direct calculation of the Euclidean distance.

In this experiment, we consider the distance from the binary representation of the state s to the goal state g as the form $d(s,g) = \|s-g\|_2$. Therefore, we can directly use the expressions $D(s,g) = d(s,g)$ and $V(s\|g) = -d(s,g)$ to represent the distance function in our method and the value function in DDPG. Hence, we can bypass the distance evaluation step and proceed directly to the policy improvement phase. In theory, our approach should yield performance comparable to that of DDPG in this experiment. Fig.1 clearly demonstrates that both methods can easily complete this task, with consistent convergence effects and final results between the two approaches. This indicates the feasibility and effectiveness of our method.

4.2. **Application of bridge point in GDG.** These experiments will demonstrate the importance of accurate distance estimates for the success of our method. In their work, Eysenbach et al. [14] define $d_{sg}(s,g)$ as the expected number of steps required to reach state s from state g under the optimal policy. However, it's worth noting that this method may not be suitable for all environments. In real-world environments, it's often not possible to estimate the optimal distance from state s to state g in advance. Therefore, we attempt to utilize our distance function $D(s,g)$ instead of $d_{sg}(s,g)$ in SoRB to estimate the distance from s to g ahead of time. So as to help search bridge point to establish the connection between the start s and the goal g to better complete the task.

In training, agents often struggle to navigate around obstacles and reach the goal. As depicted in Figure 2-a, they tend to move directly toward the goal and often collide with obstacles in

$\|$ denotes concatenation

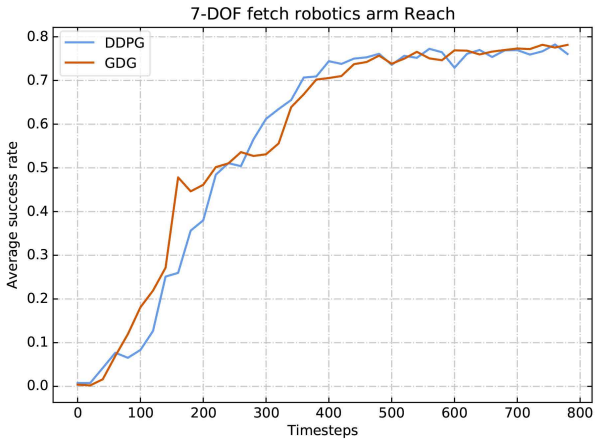


FIGURE 1. Success rates for GDG and DDPG in the 7-DOF fetch robotics arm to reach the goal.

environments like FourRooms. The challenge intensifies when the obstacles are wider, as the agent must deliberately avoid the apparent goal location before reaching it. This concept can be perplexing for agents as they need to understand why they should steer clear of the goal before closing in on it. Introducing the notion of a 'bridge point' denoted as p_b offers a solution. A bridge point serves as a connection between the start and the goal, resembling an intermediate checkpoint. Instead of the agent's primary objective being to reach the final goal, it now involves reaching the bridge point p_b before proceeding to the actual goal. This approach simplifies the task for the agent, breaking it down into two consecutive sub-tasks. Figure 2-b illustrates the discovery of a bridge point B that the agent can reach, while Figure 2-c demonstrates that B is also reachable from the goal. Figure 2-d showcases how, with the assistance of bridge point B, the agent can successfully complete the journey from the start to the goal. However, in more complex scenarios, additional bridge points may be required to establish a complete connection between the start and the ultimate goal.

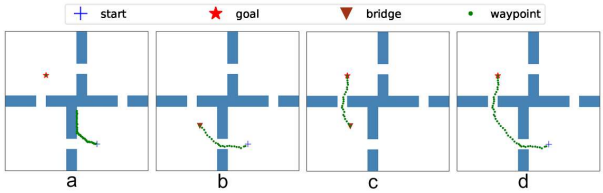


FIGURE 2. Bridge point establishes the connection between the start and the goal: (a)No bridge points connect the start and the goal, (b)Find a bridge point that can connect to the start, (c)Find a bridge point that can connect to the goal, (d)With the help of the bridge point, the path from the start to the goal was found.

4.3. **Performance comparison in complex environments.** Now, we are testing the performance of our method in a more complex environment, as illustrated in Figure 5. We have employed similar techniques to create an environment that simulates city streets, featuring more obstacles, winding paths, and a larger overall area. In this environment, in contrast to the simple FourRooms map, the maximum distance (in terms of steps) between the start and the goal

has increased from 120 to 240, and agents are required to navigate around more obstacles. We have observed that this task poses a significant challenge for the agent, requiring it to maneuver around multiple obstacles to reach the goal. In the FourRooms environment, the agent can typically reach the goal by bypassing a maximum of two obstacles. However, in the city environment, agents may need to bypass as many as nine obstacles to reach the farthest goal.

In this experiment, we intended to make a comparison with the SoRB algorithm [14]. However, the SoRB algorithm relies on distances obtained directly from the environment in advance, while our method learns the distance from the environment later in the process. As a result, a direct comparison between our method and SoRB in this experiment is not feasible. Hence, we conducted evaluations on five methods: Goal-Distance Gradient, Goal-Distance Gradient with Bridging Planning, Deep Deterministic Policy Gradients (DDPG) [17], Hindsight Experience Replay (HER) [20], and a stochastic method. We ensured that each method was assessed using the same goal sampling distribution for a fair comparison.

During the training of the aforementioned methods, each method was tested 200 times for every 20,000 training episodes, and the average success rate of the results was computed. As shown in Figure 3, it can be observed that the success rate of DDPG and HER remains relatively stable at around 0.2 in the later stages of training, showing limited further improvement. Although my method also experiences a plateau in success rate during late training, it generally maintains a level around 0.5. In contrast, the success rate of GDG with bridging planning exhibits fluctuation, ultimately reaching and sustaining around 0.8.

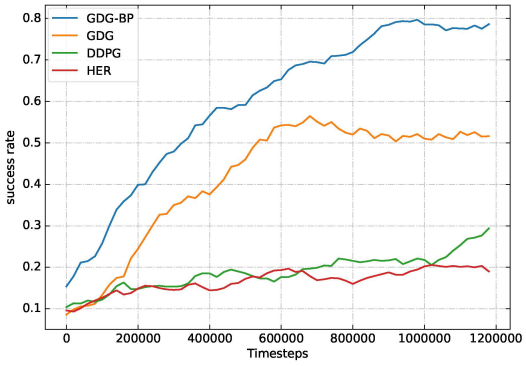


FIGURE 3. Comparison of our method with DDGP and HER in the city environment over time with an average success rate.

In this environment, we calculated the average success rates for each method across six different distance-based tasks. For each distance, we randomly generated 100 start and goal points, recording whether each attempt was successful. For each distance task, success was defined as reaching the goal within 500 steps, while failure was recorded otherwise. We conducted each experiment with five different random seeds. As depicted in Figure 4, we represent the average results of the five experiments as solid lines and use shaded areas to indicate the upper and lower bounds of these results. It is evident that GDG with bridging planning maintains a relatively high success rate at longer distances, whereas other methods are primarily effective for shorter distances.

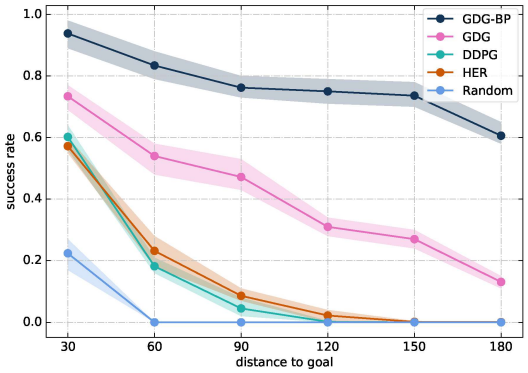


FIGURE 4. Learning curves: Testing is conducted every 5 timesteps. Each test employs the policy obtained from 100 task executions, with each task consisting of 50 distinct goals. We repeat this process five times using different random seeds and calculate the average success rate based on these repetitions.

Figure 5 below shows the navigation results of our method at different distances between the start and the goal in the city environment. From the agent's path, it almost chooses a shortest distance to complete the task and achieve the goal as soon as possible.

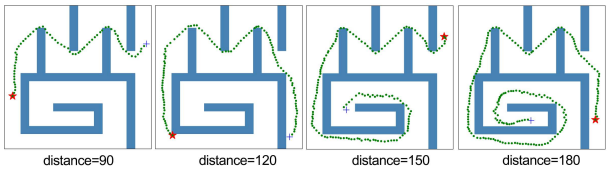


FIGURE 5. Performance of our method in the city environment at different distances, displayed from left to right: 90, 120, 150, and 180.

4.4. **Is GDG vulnerable to local optima if there is only one goal?** We employed a standard map with obstacles, as depicted in Figure 6, to assess the impact of local optima on both our method and the general algorithm. In this environment, a region close to the goal serves as a local optimal solution but cannot reach the actual desired goal. The optimal path, however, must extend far beyond this local optimal region and receive more negative feedback before reaching the intended goal. Using distance as an evaluation metric can potentially lead the agent into the local optimal region due to its proximity to the goal. It typically takes a maximum of 10 steps to reach the local optimal area from the start, whereas more than 80 steps are required to reach the desired goal. We evaluated four methods: DDPG [17] with the distance function as a negative reward, DDPG with sparse rewards, GDG, and GDG with bridging planning.

At the same exploration rate, our experiments demonstrate that GDG and GDG with bridging planning can successfully avoid getting trapped in the local optimal area and reach the goal, while the other two methods fail to complete the task. The performance of GDG and GDG with bridging planning remains consistent throughout this experiment. Analyzing the distance curve on the right in Figure 6 reveals that during the initial training phase, all methods except DDPG become stuck in the local optimal area. However, as training progresses, GDG and GDG with bridging planning actively seek ways to escape this trap and eventually discover the correct

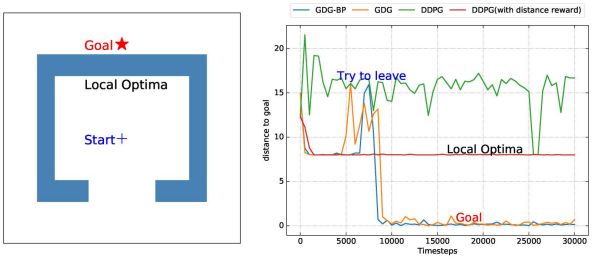


FIGURE 6. Left: Environment with a single goal. Right: Agent's final distance from the goal under four training methods.

path to the goal. DDPG with distance-based rewards also falls into the local optimal area after training but remains stuck there, unable to reach the target. In contrast, DDPG, which only receives rewards upon reaching the goal, continually explores, and its minimum distance from the goal remains within the local optimal area.

5. CONCLUSIONS

In this paper, we presented a general method for addressing sparse reward and non-reward tasks through distance evaluation. Additionally, we proposed a gradient algorithm based on goal-distance to answer how distance evaluation can enhance policy improvement. In scenarios where the actual distance cannot be precomputed, distance estimation assists bridge planning to enhance the exploration rate. Experimental results demonstrate that our method outperforms previous algorithms in complex sparse reward tasks and mitigates the impact of local optima on learning effectiveness. As future work, we aspire for agents to autonomously set their exploration goals in unfamiliar environments and tackle intricate problems, akin to identifying bridge points.

REFERENCES

[1] S. Song, J. Weng, H. Su, et al., Playing FPS games with environment-aware hierarchical reinforcement learning, Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), pp. 3475–3482, 2019.

[2] T. Liu, Z. Zheng, H. Li, et al., Playing card-based rts games with deep reinforcement learning, Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), pp. 4540–4546, 2019.

[3] D. Silver, A. Huang, C.J. Maddison, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (2016) 484-489.

[4] X. Guo, S. Singh, R. Lewis, et al., Deep learning for reward design to improve monte carlo tree search in atari games, arXiv preprint arXiv:1604.07095, 2016.

[5] S. Levine, C. Finn, T. Darrell, et al., End-to-end training of deep visuomotor policies, J. Mach. Learn. Res. 17 (2016) 1334–1373.

[6] A.V. Nair, V. Pong, M. Dalal, et al., Visual reinforcement learning with imagined goals, In Advances in Neural Information Processing Systems, pp. 9191–9200, 2018.

[7] R.S. Sutton, A.G. Barto, Introduction to Reinforcement Learning, The MIT Press, Cambridge, 1998.

[8] H. Fu, H. Tang, J. Hao, et al., Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces, Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), pp. 2329–2335, 2019.

[9] R.S. Sutton, J. Modayil, M. Delp, et al., Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction, In The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pp. 761–768, 2011.
[10] N. Ponomarenko, L. Jin, O. Ieremeiev, et al., Image database tid2013: Peculiarities, results and perspectives, Signal Processing: Image Communication, 30 (2015) 57–77.
[11] R. Zhang, P. Isola, A.A. Efros, et al., The unrea- sonable effectiveness of deep features as a perceptual metric, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 586–595, 2018.
[12] M.J. Mataric, Reward functions for accelerated learning, Machine Learning Proceedings, pp. 181–189, 1994.
[13] A.Y. Ng, D. Harada, S. Russell, Policy invariance under reward transfor- mations: Theory and application to reward shaping, ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning, pp. 278–287, 1999.
[14] B. Eysenbach, R. Salakhutdinov, S. Levine, Search on the replay buffer: Bridging planning and reinforcement learning, NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems, pp. 15246–15257, 2019.
[15] J. Peters, S. Schaal, Natural actor-critic, Neurocomputing, 71 (2008), 1180–1190.
[16] T. Degris, P.M. Pilarski, R.S. Sutton, Model-free reinforcement learning with continuous action in practice, 2012 American Control Conference (ACC), 2012.
[17] T. Lillicrap, J.J. Hunt, A. Pritzel, et al., Continuous control with deep reinforcement learning, MM '19: Proceedings of the 27th ACM International Conference on Multimedia, pp. 2637–2641, 2019.
[18] S. Bhatnagar, M. Ghavamzadeh, M. Lee, et al., Incremental natural actor-critic algorithms, NIPS'07: Proceedings of the 20th International Conference on Neural Information Processing Systems, pp. 105–112, 2007
[19] T. Degris, M. White, R.S. Sutton, Linear off-policy actor-critic, arXiv:1205.4839, 2012.
[20] M. Andrychowicz, F. Wolski, A. Ray, et al., Hindsight experience replay, NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 5055–5065, 2017.
[21] T. Schaul, D. Horgan, K. Gregor, et al., Universal value function approximators, Proceedings of the 32nd International Conference on Machine Learning, 37 (2015) 1312-1320.