A ALGORITHM

Algorithm 1 FLR with HER

Sample the batch of transition $\{(s_t, a_t, s_{t+1}, sg_i)\}_{|B|}$ in replay buffer $\mathcal{B}$
Sample an additional goals $\phi(s_{t+F})$ from **current episode** with a probability of p
$sg_i \leftarrow \phi(s_{t+F})$
if $\|\phi(s_{t+1}) - sg_i\| < \delta$ **then**
    $r_t \leftarrow \max_{sg:\varepsilon \leq \|sg_i - sg\|} \alpha V^l_{\pi_l}(s_{t+1}, sg)$
else
    $r_t \leftarrow -1$
end if
Update Q_θ using the transition $(s_t, a_t, r_t, s_{t+1}, sg, sg')$

Algorithm 2 DHRL (Lee et al., 2022)

Input: initial random steps $\tau_{\text{randomwalk}}$, initial steps without planning $\tau_{\text{w/o graph}}$, total training step τ_{total}, Env, low-level agent $Q^{\text{lo}}_{\text{critic},\theta_1}$, $Q^{\text{lo}}_{\text{graph},\theta_2}$ and $\pi^{\text{lo}}_{\phi_1}$, high-level agent $Q^{\text{hi}}_{\theta_3}$ and $\pi^{\text{hi}}_{\phi_2}$
$Dist(s, g) := \log_\gamma (1 + (1 - \gamma) Q^{\text{lo}}_{\text{graph},\theta_2}(s, \pi(s, g)|g))$
for $\tau = 1$ **to** τ_{total} **do**
    if Env.done **then**
        Env.reset (episode step resets to 0)
    end if
    if $\tau < \tau_{\text{randomwalk}}$ **then**
        $a_t \leftarrow$ random.uniform(high = action.high, low = action.low) ◁ initial random rollout
    else if $\tau < \tau_{\text{w/o graph}}$ **then**
        $a_t \leftarrow$ vanilla $HRL(sg_t = \pi^{\text{hi}}_{\phi_2}(s_t, g)$ and $\pi^{\text{lo}}_{\phi_1}(s_t, sg_t))$ ◁ act without planning
    else
        if Graph $\mathbf{G}$ is not initialized **then**
            Create a graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$ using FPS algorithm ◁ initialize graph
        end if
        if episode step(the step of the environment) $\%$ c_l = 0 **then**
            $sg_t \leftarrow \pi^{\text{hi}}_{\phi_2}(s_t, g)$ ◁ get subgoal
            $\{wp_{t,1}, wp_{t,2}, \cdots wp_{t,k}\} \leftarrow$ `Dijkstra'salgorithm`(s_t, sg_t) ◁ get waypoints
            current waypoint index $n = 1$
        end if
        if achieved $wp_{t,n}$ or tried more than $Dist(wp_{t,n-1}, wp_{t,n})$ to achieve $wp_{t,n}$ **then**
            current waypoint index $+= 1$
        end if
        $a_t \leftarrow \pi^{\text{lo}}_{\phi_1}(s_t, wp_{t,n+1})$ ◁ get low-level action
    end if
    Env.step(a_t)
    Train low-level agent $Q^{\text{lo}}_{\text{critic},\theta_1}$, $Q^{\text{lo}}_{\text{graph},\theta_2}$ and $\pi^{\text{lo}}_{\phi_1}$, high-level agent $Q^{\text{hi}}_{\theta_3}$ and $\pi^{\text{hi}}_{\phi_2}$ ◁ FLR is utilized when training low-level agents.
    if τ $\%$ graph update freq = 0 **then**
        Update Graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$ using farthest point sampling (FPS) (Arthur & Vassilvitskii, 2007) algorithm
    end if
end for

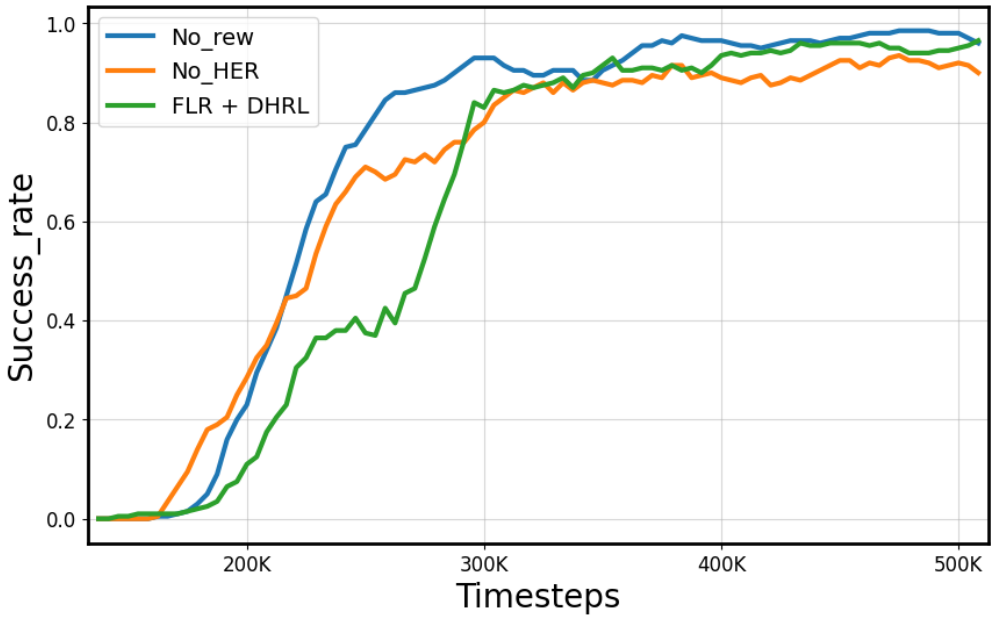

Figure 6: Comparision with oracle-based approach

B Additional Results

B.1 Compared With Oracle

In this section, we compare our FLR+DHRL approach with an Oracle-based DHRL method in the context of the specific problem domain we address. We introduce two baseline methods for comparison: 'No rew' and 'No HER'.

- No rew: This approach assigns a reward of -1 regardless of whether the agent reaches the goal or not, specifically penalizing the occurrence of falling states. This design allows for a strong penalty for falling states.
- No HER: The 'No HER' method does not employ hindsight experience replay for falling states in the entire trajectory. This strategy ensures that the low-level policy does not learn trajectories leading to falling states, naturally encouraging the agent to avoid falling.

As shown in Figure 6, our FLR+DHRL approach does not precisely match the performance of the oracle-based method but demonstrates nearly equivalent performance. We attribute this observation to the nature of FLR, which penalizes falling states more strongly but does not explicitly incorporate mechanisms to avoid learning from undesirable transition data, akin to the 'No HER' method.

This analysis suggests that while imposing penalties for irreversible states is crucial for addressing our specific problem, avoiding such states altogether may also serve as a viable strategy for problem resolution. The FLR+DHRL approach strikes a balance between penalizing falling states and encouraging the avoidance of undesirable transitions, contributing to its effectiveness in the considered problem domain.

B.2 Hyperparameters

We utilized the majority of hyperparameters exactly as employed in the previous baselines. We experimented with varying the number of landmarks, initial rollout, and the future step in HER, as detailed in Table 1, 2, and 3. We also report our hyperparameters in Table 4.

Table 1: Common hyperparameters setting of HGRL baselines

	PIG	DHRL	HIGL
RL algorithm	DDPG	TD3	TD3
high-level τ	-	0.005	0.005
π_h learning rate	-	0.0001	0.0001
$Q^h_{\pi_h}$ lr	-	0.001	0.001
γ^h	-	0.99	0.99
high-level train freq	-	10	10
low-level τ	0.01	0.005	0.005
π_l learning rate	0.0002	0.0001	0.0001
$Q^l_{\pi_l}$ learning rate	0.0002	0.001	0.001
γ^l	0.99	0.99	0.95
hidden layer	-	(256,256,256)	(300,300)
number of landmarks	400	300	20-150
number of novelty landmarks	-	-	20-50
batch size	128	1024	128
hindsight relabelling ratio	0.8	0.8	-
hindsight future steps	150-200	150	-

Table 2: Hyperparameters for DHRL

	DHRL
number of random rollout episodes	100-200
initial episodes without graph planning	75
gradual penalty transition rate	0.2
high-level train freq	10
target update freq	10
actor update freq	2

Table 3: Hyperparameters for PIG

	PIG
hidden layer for actors	(400,400,400,400)
hidden layer for critics	(400,400,400,400,400)

Table 4: Hyperparameters for FLR

	FLR
reward scale α	0.1
next subgoal distance ε	2
the number of sampling next subgoal	10