A Algorithms.

Algorithm 1 Training DHRL

1: sample $D_{lo}^{i} = (s_{t}, wp_{t}, a_{r}, r(s_{t+1}, wp_{t}), s_{t+1}), i \in B^{lo}$
2: relabel $wp_{t} \leftarrow \hat{w}p_{t} = ag_{t+10}$, to make $\hat{D}_{lo}^{i}$
3: update $Q_{\phi_{1}}^{lo}$ and $\pi_{\phi_{2}}^{lo}$ using $\hat{D}_{lo}^{i} \cup \hat{D}_{lo}^{i}$
4: update $Q_{\phi}^{lo}$ using $\hat{D}_{lo}^{i}$
5: if $t \mod d$ then
6: sample $D_{hi}^{i} = (s_{t}, g_{t}, s_{g_{t}}, r_{t}, s_{t+c_{h}}), i \in B^{hi}$
7: relabel $sg_{t} \leftarrow ag_{t+c_{h}}$, to make $\hat{D}_{hi}^{i}$
8: for $(s_{t}, g_{t}, sg_{t}, r_{t}, s_{t+c_{h}}) \in D_{hi}^{i}$ do
9: if $r(s_{t+c_{h}}, sg_{t}) = 0$ then
10: $r_{t} \leftarrow GradualPenalty$ if subgoal $\in L_{1}$
11: else if use GradualPenalty then
12: $r_{t} \leftarrow GradualPenalty(graph G, sg_{t}, Q_{graph}^{lo})$
13: else
14: $r_{t} \leftarrow penalty p_{1}$
15: end if
16: end for
17: update $Q_{\phi_{1}}^{hi}$ and $\pi_{\phi_{2}}^{hi}$ using $D_{hi}^{i}$
18: end if

Algorithm 2 Farthest Point Sampling Algorithm [2]

1: Input: set of states $\{s_{1}, s_{2}, \ldots s_{K}\}$, sampling number $k$, temporal distance function $Dist(\cdot \rightarrow \cdot)$
2: SelectedNode $= []$
3: DistList $= [\text{inf}, \text{inf}, \ldots \text{inf}]$
4: for $i = 1$ to $k$ do
5: FarthestNode $\leftarrow \arg\max(DistList)$
6: add FarthestNode to SelectedNode
7: DistFromFarthest $\leftarrow [\text{Dist(FarthestNode} \rightarrow s_{1}), \ldots, \text{Dist(FarthestNode} \rightarrow s_{K})]$
8: DistList $= \text{ElementwiseMin(DistFromFarthest, DistList)}$
9: end for
10: return SelectedNode

Algorithm 3 Planning with DHRL

1: while not done do
2: if $t \mod graph\_construct\_freq$ then
3: construct a graph $G(V, E):$ sample $V = \psi(s)$ where $s \in D^{lo}$ through FPS algorithm and get edge cost $E$ by Eq. (1)
4: end if
5: $sg_{t} = \pi^{hi}(s_{t}, g_{t})$
6: get $W: \{wp_{t,0} = \psi(s_{t}), wp_{t,1}, wp_{t,2}, \ldots, wp_{t,k-1}, wp_{t,k} = sg_{t}\}$
7: previous waypoint index $idp \leftarrow 0$; tracking waypoint index $idt \leftarrow 1$; tracking time $t_{tr} \leftarrow 0$
8: for $\tau = 1$ to $c_{h}$ do
9: get low-level action $a_{\tau}$ from $\pi^{lo}(a_{\tau} | s_{\tau}, wp_{t,idt})$
10: act $a_{\tau}$ in the environment and get $s_{\tau+1}$
11: $t_{tr} + = 1; t + = 1$
12: if agent achieve $wp_{t,idt}$ or $t_{tr} > Dist(wp_{t,idt} \rightarrow wp_{t,idt})$ then
13: $idp + = 1; idt + = 1; t_{tr} \leftarrow 0$
14: end if
15: end for
16: end while
Algorithm 4 Gradual Penalty

1: **Input:** graph $G(V, E)$, subgoal $s_{gt}, Q_{\text{graph}, \theta_2}^\text{lo}$, gradual penalty threshold $\zeta_1$, penalty $p_1$, penalty $p_2$
2: if $\min(v \in V, s_{gt}) < \zeta_1$ then
3: $r_t \leftarrow \text{penalty } p_1$ ⊂ subgoal ∈ $L_2$
4: else
5: $r_t \leftarrow \text{penalty } p_2$ ⊂ subgoal ∈ $L_3$
6: end if

Algorithm 5 Frontier-Based Goal-Shifting (FGS)

1: **Input:** $s_t$, graph $G(V, E)$, goal $g$, $Q_{\text{graph}, \theta_2}^\text{lo}$, cut-off threshold $\zeta_2$
2: $\text{Dist}(s, g) := \log_\gamma (1 + (1 - \gamma)Q_{\text{graph}, \theta_2}^\text{lo}(s, \pi(s, g)|g))$
3: if $\min_v \in V (\text{Dist}(v \rightarrow g)) < \zeta_2$ then
4: $V_{\text{candidate}} \leftarrow V + \text{noise}$
5: $g_t \leftarrow \text{random.choice}(V_{\text{candidate}}, \text{weight} = -Q_{\text{graph}, \theta_2}^\text{lo}(s_t, \pi(s_t, V_{\text{candidate}})|V_{\text{candidate}}))$
6: end if
7: return $g_t$

Algorithm 6 Overview of DHRL

1: **Input:** initial random steps $\tau_{\text{randomwalk}}$, initial steps without planning $\tau_{w/o \text{graph}}$, total training step $\tau_{\text{total}}$, Env, low-level agent $Q_{\text{critic}, \theta_1}^\text{lo}, Q_{\text{graph}, \theta_2}$ and $\pi_{\phi_1}$, high-level agent $Q_{\theta_3}$ and $\pi_{\phi_2}$
2: $\text{Dist}(s, g) := \log_\gamma (1 + (1 - \gamma)Q_{\text{graph}, \theta_2}^\text{lo}(s, \pi(s, g)|g))$
3: for $\tau = 1$ to $\tau_{\text{total}}$ do
4: if Env.done then
5: Env.reset (episode step resets to 0)
6: if Use FGS then
7: $g \leftarrow \text{FGS}(G, g, Q_{\text{graph}, \theta_2}^\text{lo})$
8: end if
9: end if
10: if $\tau < \tau_{\text{randomwalk}}$ then
11: $a_t \leftarrow \text{random.uniform(high = action.high, low = action.low)}$ ⊂ random action
12: else if $\tau < \tau_{w/o \text{graph}}$ then
13: $a_t \leftarrow \text{vanilla } HRL(s_{gt} = \pi_{\phi_2}^\text{hi}(s_t, g) \text{ and } \pi_{\phi_1}^\text{lo}(s_t, s_{gt}))$ ⊂ act without planning
14: else
15: if Graph $G$ is not initialized then
16: Create a graph $G(V, E)$ using FPS algorithm ⊂ initialize graph
17: end if
18: if episode step(the step of the environment) $\% c_l = 0$ then
19: $s_{gt} \leftarrow \pi_{\phi_2}^\text{hi}(s_t, g)$ ⊂ get subgoal
20: $\{w_{p_{t,1}}, w_{p_{t,2}}, \cdots w_{p_{t,k}}\} \leftarrow \text{Dijkstra's algorithm}(s_t, s_{gt})$ ⊂ get waypoints
21: current waypoint index $n = 1$
22: end if
23: if achieved $w_{p_{t,n}}$ or tried more than $\text{Dist}(w_{p_{t,n-1}}, w_{p_{t,n}})$ to achieve $w_{p_{t,n}}$ then
24: current waypoint index += 1
25: end if
26: $a_t \leftarrow \pi_{\phi_1}^\text{lo}(s_t, w_{p_{t,n+1}})$ ⊂ get low-level action
27: end if
28: Env.step($a_t$)
29: Train low-level agent $Q_{\text{critic}, \theta_1}^\text{lo}, Q_{\text{graph}, \theta_2}^\text{lo}$ and $\pi_{\phi_1}^\text{lo}$, high-level agent $Q_{\theta_3}^\text{hi}$ and $\pi_{\phi_2}^\text{hi}$
30: if $\tau$ % graph update freq = 0 then
31: Update Graph $G(V, E)$ using FPS algorithm
32: end if
33: end for
B Proofs of Theorems.

Derivation of equation 1. If a given policy \( \pi_{lo} \) requires \( n \) steps to get from current \( s \) to a goal \( g \), the \( \gamma \)-discounted return is \( Q_{lo}(s, \pi(s, g)|g) = (-1) + (-1)\gamma + (-1)\gamma^2 \ldots (-1)\gamma^{n-1} = \frac{1 - \gamma^n}{1 - \gamma} \).

Thus, the temporal distance between \( s \) to \( g (= n) \) is derived from \( \gamma^n - 1 = (1 - \gamma)Q_{lo}(s, \pi(s, g)|g) \) as

\[
n = \log_{\gamma} (1 + (1 - \gamma)Q_{lo}(s, \pi(s, g)|g)).
\]

Definition B.1. \( W_G(s_t, s_g) = (wp_{t,0}, wp_{t,1}, \ldots, wp_{t,h}) \) is a sequence of waypoint obtained by the graph search algorithm and \( w(W_G, \tau) = wp_{t,i} \in W_G(s_t, s_g) \) is the waypoint that is given to low-level policy at \( \tau \).

Given the transition distribution of the environment \( \mathcal{T}(s_{t+1}|s_t, a_t) \), the transition data \( (s_t, g_t, s_g, r(s_{t+1}, g_t), \bar{s}_{t+1}, \bar{a}_{t+1}, \bar{s}_{t+1}, \bar{a}_{t+1}) \) from the high-level policy’s replay buffer has been obtained as

\[
s_{t+1} = \prod_{i=t}^{t+1} \mathcal{T}(s_{t+i}|s_t, a_t),
\]

where \( \bar{s}_{t+1} \) and \( \tau \) are the previous low-level policy and graph respectively. Also, by using a different graph \( G \) and an optimal policy \( \pi^{lo*} \), we get a new transition data \( (s_t, g_t, s_g, r(s_{t+1}, g_t), \bar{s}_{t+1}, \bar{a}_{t+1}, \bar{s}_{t+1}, \bar{a}_{t+1}) \), where

\[
s_{t+1} = \prod_{i=t}^{t+1} \mathcal{T}(s_{t+i}|s_t, a_t),
\]

For given \( s_t \) and \( s_{t+1} \), we define the off-policy error rate, which is the normalized distance error with respect to the total traversal distance according to the change of \( \pi^{lo} \) and \( G \) to \( \pi^{lo*} \) and \( G \), as

\[
\rho(G) = \frac{\text{Dist}(\psi(s_{t+1}) \to \psi(s_{t+1}))}{\text{Dist}(\psi(s_t) \to \psi(s_{t+1}))}.
\]

Lemma B.2. Suppose that \( \text{Dist}(. \to .) \) in Eq. (1) is Lipschitz continuous. Then, there exists a constant \( L > 0 \) such that \( \forall x, y \in \text{max} (\text{Dist}(x \to y), \text{Dist}(y \to x)) \leq L ||x - y|| \), where \( ||.|| \) is the Euclidean norm, since \( \text{Dist}(x \to y) = 0 \). Then, any graph is Lipschitz continuous in the Euclidean norm, whose existence is trivial, is an \( \epsilon \)-resolution graph w.r.t the Euclidean norm.

Proof of Theorem 4.2.

Proof. Let \( C^{x \to y} \) be one of the shortest paths from \( s \) to \( g \) and \( T \) be the distance of \( C^{x \to y} \). Also let \( p_1 \in C^{x \to y} \) be a point that \( \text{Dist}(\psi(s) \to p_1) = c_1 - \epsilon \). Then, \( \exists wp_1 \in \mathcal{V} \) s.t. \( \text{max}(\text{Dist}(p_1 \to wp_1), \text{Dist}(wp_1 \to p_1)) < \epsilon \), because \( G \) is an \( \epsilon \)-resolution graph. Since \( \text{Dist}(\cdot \to \cdot) \) is a temporal distance, it satisfies the triangular inequality and then, \( \text{Dist}(\psi(s) \to wp_1) < \text{Dist}(\psi(s) \to p_1) + \text{Dist}(p_1 \to wp_1) < (c_1 - \epsilon) + \epsilon = c_1 \) and \( \text{Dist}(wp_1 \to g) < \epsilon + (T - c_1 + \epsilon) = T - c_1 + 2\epsilon \).

Repeating the above procedure, let \( p_i + 1 \in C^{x \to y} \) be a point that \( \text{Dist}(wp_{i-1} \to p_{i+1}) = c_i - \epsilon \). Then, \( \exists wp_{i+1} \in \mathcal{V} \) s.t. \( \text{max}(\text{Dist}(p_{i+1} \to wp_{i+1}), \text{Dist}(wp_{i+1} \to p_{i+1})) < \epsilon \). Then, \( \text{Dist}(wp_i \to wp_{i+1}) < c_i \) and \( \text{Dist}(wp_{i+1} \to g) < T - (i + 1)c_i + 2(i + 1)\epsilon \). Consequently, the agent after \( T \) time-step will be closer than the \( \lfloor T/c_i \rfloor \)th waypoint from \( g \). The remaining distance is less than

\[
T - \lfloor T/c_i \rfloor c_i + 2\lfloor T/c_i \rfloor \epsilon.
\]

Thus, if an agent follows the sequence of waypoints \( \{s, wp_1, wp_2, \ldots, g\} \), which is generated from a graph search algorithm over \( G \) and \( \pi^{lo*} \), the error rate over this path satisfies

\[
\rho(G) \leq \frac{T - \lfloor T/c_i \rfloor c_i + 2\lfloor T/c_i \rfloor \epsilon}{T} \leq \frac{T - (c_1 - 2\epsilon)(T/c_1)}{T} = \frac{2\epsilon}{c_1}.
\]

Thus the off-policy error rate \( \rho \) is equal or less than \( 2\epsilon/c_1 \) during \( T \). Since all path from \( s \) to \( g \) takes at least \( T \) time-steps, this upper-bound of error rate is also satisfied in all path from \( s \) to \( g \).
C Additional Results

Figure 10: Comparison with shallow RL (SAC) and vanilla HRL (HIRO). The completely failed baselines are occluded by others.
As shown in the table above, the wider the initial distribution, the easier it is for the agent to explore the map. In other words, the ‘fixed initial state distribution’ condition we experimented with in this paper is a more difficult condition than the ‘uniform initial state distribution’ that previous graph-guided RL algorithms utilize. Of course, ‘fixed initial state distribution’ requires less prior information about the state space. We further experimented with ours (DHRL) under various types of reset conditions as shown in Table 2. As expected, our algorithm shows faster exploration at the uniform reset point.
Table 3: Comparisons between our algorithm (DHRL) and baselines: The numbers next to the environment names are the time-steps for training the models. The results are averaged over 4 random seeds and smoothed equally. ‘-D’ and ‘-S’ mean dense reward and sparse reward respectively. We use NVIDIA RTX A5000.
Table 4: Hyperparameters for HRL: When evaluating the previous HRL algorithms, we used the same hyperparameters as used in their papers. We also tried various numbers of landmarks and $c_h$ which may affect the performance in long-horizon tasks.

<table>
<thead>
<tr>
<th></th>
<th>HIRO</th>
<th>HRAC</th>
<th>HIGL</th>
</tr>
</thead>
<tbody>
<tr>
<td>high-level $\tau$</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>$\pi^{hi}$ lr</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>$Q^{hi}$ lr</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>high-level $\gamma$</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>high-level train freq</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$c_h$</td>
<td>-</td>
<td>10-50</td>
<td>-</td>
</tr>
<tr>
<td>low-level $\tau$</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>$\pi^{lo}$ lr</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>$Q^{lo}$ lr</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>low-level $\gamma$</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>hidden layer</td>
<td>(300,300)</td>
<td>(300,300)</td>
<td>(300,300)</td>
</tr>
<tr>
<td>number of coverage landmarks</td>
<td>-</td>
<td>-</td>
<td>20-250</td>
</tr>
<tr>
<td>number of novelty landmarks</td>
<td>-</td>
<td>-</td>
<td>20-250</td>
</tr>
<tr>
<td>batch size</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
</tbody>
</table>

Table 5: Hyperparameters for SAC

<table>
<thead>
<tr>
<th></th>
<th>SAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>hidden layer</td>
<td>(256, 256, 256)</td>
</tr>
<tr>
<td>actor lr</td>
<td>0.0003</td>
</tr>
<tr>
<td>critic lr</td>
<td>0.0003</td>
</tr>
<tr>
<td>entropy coef</td>
<td>0.2</td>
</tr>
<tr>
<td>$\tau$</td>
<td>0.005</td>
</tr>
<tr>
<td>batch size</td>
<td>256</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Table 6: Hyperparameters for DHRL

<table>
<thead>
<tr>
<th></th>
<th>DHRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>hidden layer</td>
<td>(256, 256, 256)</td>
</tr>
<tr>
<td>initial episodes without graph planning</td>
<td>75</td>
</tr>
<tr>
<td>gradual penalty transition rate</td>
<td>0.2</td>
</tr>
<tr>
<td>high-level train freq</td>
<td>10</td>
</tr>
<tr>
<td>actor lr</td>
<td>0.0001</td>
</tr>
<tr>
<td>critic lr</td>
<td>0.001</td>
</tr>
<tr>
<td>$\tau$</td>
<td>0.005</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.99</td>
</tr>
<tr>
<td>number of landmarks</td>
<td>300-500</td>
</tr>
<tr>
<td>target update freq</td>
<td>10</td>
</tr>
<tr>
<td>actor update freq</td>
<td>2</td>
</tr>
</tbody>
</table>