GRAPH-ENHANCED EXPLORATION FOR GOAL-ORIENTED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Goal-oriented Reinforcement Learning (GoRL) is a promising approach for scaling up RL techniques on sparse reward environments requiring long horizon planning. Recent works attempt to build suitable abstraction graph of the environment and enhance GoRL with classical graphical methods such as shortest path searching; however, these approaches mainly focus on either graph construction or agent exploitation, but leave the exploration lack of study. This paper proposes Graph-enhanced GoRL (G2RL), a new GoRL framework for effective exploration and efficient training based on the state-transition graph. We first introduce the *optimal goals* for exploration on the graph and then use them as supervised signals to train a differentiable goal generator in a hindsight manner. Furthermore, we define *relevant trajectories* of a state based on its graph neighborhood and show that giving high priority to these trajectories would lead to an efficient policy learning. In addition to the theoretical results regarding optimal goal generation, our empirical results on standard discrete and continuous control benchmarks show that leveraging the state-transition graph is beneficial for GoRL to learn an effective and informative exploration strategy and outperform the state-of-the-art methods.

1 INTRODUCTION

Goal-oriented Reinforcement Learning (GoRL) (Nasiriany et al., 2019; Eysenbach et al., 2019; Huang et al., 2019; Levy et al., 2017; Kulkarni et al., 2016a) allows RL methods to tackle complex tasks with long-term credit assignment and sparse reward. The basic idea of GoRL is to comprise a goal generator to generate intermediate goals (also called subgoals in previous literature (Zhang et al., 2020; Paul et al., 2019)) which decomposes the original complicate task into a series of intermediate easier subtasks guided by goals, and a policy learner aiming to reach those generated goals. However, the performance of GoRL significantly relies on the generation of effective goals, which still remains a key challenge.

One promising solution is to construct suitable abstraction graphs of the environment and generate goals by classical planning algorithms on graph. These abstraction graphs can either be built with the state-transitions based on the collected experience (Eysenbach et al., 2019) or the abstract structure of the environment based on the observation (Shang et al., 2019). However, almost all existing approaches (Huang et al., 2019; Laskin et al., 2020; Zhu et al., 2019) focus on either graph construction based on the collected experience or agent exploitation over the graph by adopting classical planning techniques on graph (e.g., Dijkstra's algorithm, A* search) (Laskin et al., 2020; Eysenbach et al., 2019; Huang et al., 2019). There has been limited literature pursuing more effective and informative exploration by strategical (especially differentiable) goal generation based on the abstraction graph, which is challenging as the graph built on either state-transitions or observations is inherently dynamic and sometimes even much large in scale. In summary, most existing methods still suffer from two aspects of difficulties. (1) The goal space is often large as the state space, leading to difficulty in effective exploration (Zhang et al., 2020) (2) The historical trajectories are collected by various goals, and random sampling would result in sample inefficiency for policy learning (Fang et al., 2019), because of low reachability of the generated goals.

In this paper, we propose Graph-enhanced GoRL (G2RL), a novel GoRL framework that, based on a dynamically built state-transition graph, (1) develops a differentiable goal generator with hindsight supervision for effective environment explorations, and (2) designs a new trajectory sampling algorithm with goal proximity for efficient policy learning, as illustrated in Figure 1(a).

Concretely, we construct the state-transition graph from the replay buffer, where the nodes are the states and edges are the transitions. We extract the graph directly from the experiences for discrete environment and apply K-bins discretization technique (Kotsiantis & Kanellopoulos, 2006) for continuous environment. Notably, as shown in Figure 1(b), the graph in G2RL can be regarded as an auxiliary structural information for goal generation and trajectory sampling, and thus the discretization does not change a deterministic environment into a stochastic one.

For effective exploration, we begin with revealing the connection between the out-degree of nodes and the status of exploration on the corresponding states. The intuition behind this is quite straightforward: if the out-degree of a node is small, it implies that its next states have not been much visited, namely this state itself is not well explored. In the light of it, we consider these not well explored nodes are valuable for exploration and define the set of these nodes as *graph boundary*. We then enforce the goal generated from the states in the boundary, as we theoretically show that the optimal goals are preserved within the graph boundary. For handling graph's dynamics and large scale, we divide the graph boundary into several groups according to their out-degrees (i.e., the status of exploration). In such a way, our method is scalable, as it only needs to build a fix number of groups. We further develop an attention mechanism to select an appropriate group, in which the state with the highest value is assigned as intermediate goal. We introduce *optimal goals* as the supervision signals to learn our differentiable goal generator. These optimal goals are the most valuable and reachable states defined based on the planning algorithm on the graph in the next episode, and thus our goal generator is optimized in a hindsight manner.

Furthermore, for efficient policy learning, almost all existing GoRL approaches (Paul et al., 2019; Eysenbach et al., 2019) utilize randomly sampled trajectories to update the policy, leading to inefficiency as there are various subtasks guided by different goals. Instead, we design a novel *relevance sampling* technique, where the basic intuition is when updating the value of a state, the trajectories containing it and its neighboring states are likely to share similar goals. Hence, we call these trajectories *relevant trajectories* and give high priority to sampling these trajectories in policy learning.

The contributions of this paper are as follows:

- We propose a graph structure enhanced goal generator built upon attention mechanism, which achieves effective exploration with optimal goals and can be learned with hindsight supervisions.
- We design a relevance sampling technique that leverages the state-transition graph information to select relevant experience sharing the similar directions (i.e., goals) for efficient policy learning.
- We theoretically show that optimal goals can be preserved in the boundary of the state-transition graph and empirically illustrate that our method is capable to capture these goals.

We benchmark our method on various tasks, including discrete control and planning tasks on grid worlds and challenging continuous control tasks, which are widely used in GoRL literature (Florensa et al., 2018; Nachum et al., 2018a;b; Savinov et al., 2018). Experimental results exhibit the superiority of G2RL on both asymptotic performance and sample efficiency compared with several state-of-the-art GoRL approaches, demonstrating the effectiveness of G2RL.

2 Graph Construction and Exploration of Goal-oriented RL

2.1 Goal-oriented Reinforcement Learning

Let $M = (S, A, T, P, R)$ be a Markov Decision Process (MDP) where S is the state space, A is the action space, $T \in \mathbb{Z}_+$ is the episode length, $P : S \times A \to \Delta(S)$ is the transition function which takes a state-action pair and returns a distribution over states, and $R : S \times A \to \Delta(\mathbb{R})$ is the reward distribution. The rewards from environments, called extrinsic, are usually sparse and thus, difficult to learn. To address this, following the basic idea of GoRL, we consider a framework comprising two main components: a goal generator aiming to generate goals $g \in G$ at each episode; and a policy learner aiming to maximize the intrinsic reward conditioned on generated goals, formulated as $R_g : S \times A \times G \to \Delta(\mathbb{R})$, and performs a primary action at every timestep. Following prior methods (Nachum et al., 2018a; Andrychowicz et al., 2017; Zhang et al., 2020), we enforce goal space G as a subspace of S. A policy $\pi : S \times G \to \Delta(A)$ prescribes a distribution over actions for each state and goal. We introduce π as a greedy deterministic policy. At each timestep, the agent samples an action $a \sim \pi(s, g)$ and receives a corresponding reward $r_g(s, a)$ that indicates whether or not the agent has reached the goal. The agent's task is to maximize its cumulative discounted future

reward. We use an off-policy algorithm to learn such a policy, as well as its associated goal-oriented Q-function and state value function:

$$Q^\pi(s,a,g) = \mathbb{E}\Big[\sum_{t=0}^{T-1} \gamma^t r_g(a_t,s_t) \mid s_t = s, a_t = a, \pi\Big], \; V^\pi(s,g) = \sum_a \pi(a|s,g) Q^\pi(s,a,g). \quad (1)$$

2.2 Graph Construction from Replay Buffer

As Figure 1(b) illustrates, we build a dynamic and directed graph on top of historical trajectories in the replay buffer at each timestep. We denote the state-transition graph at timestep t in episode e as $\mathcal{G}_t^e = (\mathcal{V}_t^e, \mathcal{E}_t^e)$ with relations $\mathcal{R}_t^e$, where the node space $\mathcal{V}_t^e$ represents state space $S_{\mathcal{G}_t^e}$, edge space $\mathcal{E}_t^e$ represents transitions $P_{\mathcal{G}_t^e}$, and relations $\mathcal{R}_t^e$ represents actions $A_{\mathcal{G}_t^e}$. $P_{\mathcal{G}_t^e}$ determines the existence of transition from s_t to s_{t+1}, and $A_{\mathcal{G}_t^e}$ shows the way to transit. For simplicity, we use $\mathcal{G}^e$ to denote $\mathcal{G}_0^e$. See Appendix A1 for an illustrated example of notations.

Note that it's non-trivial to construct an abstraction graph with continuous environments. Existing approaches either organize the graph based on sampled trajectories (Zhu et al., 2019; Huang et al., 2019), or introduce a parameterized distance network (Savinov et al., 2018; Eysenbach et al., 2019) to predict the distance of each pair of states and establish edges for those pairs whose distances are above the pre-defined threshold. In our implementation, we employ the K-bins discretization technique (Kotsiantis & Kanellopoulos, 2006) to discretize the state and action spaces, as it has been shown that animals explore their environments with the help of grid cells activated on particular intervals (O'Keefe & Dostrovsky, 1971; Banino et al., 2018; Colombo et al., 1998).* Hence, in this case, each node of the graph represents a bin of states/actions with similar locations/directions; and an edge exists when the number of transitions between pairs of states from two nodes reach a pre-defined threshold, and the corresponding relations show the actions of the transition. Formally, we can obtain discretized state and action spaces, denoted as $\widehat{S}$ and $\widehat{A}$. Notably, this discretization technique can consistently map a box to a fixed integer, and thus will not operate a deterministic MDP into a stochastic one. Without loss of generality, in the following parts, we mainly discuss in the context of discrete environments, as we have described how to extend it into continuous ones.

Let an agent start from a start state $s_{\texttt{start}}$, aiming to a target state $s_{\texttt{target}}$. Then, $\mathcal{G}_t^e$ grows from a single node (i.e., $\mathcal{G}_0^0 = \{v_{\texttt{start}}\}$) and expands itself at each timestep within each episode, leading to the sequence $\{\{\mathcal{G}_t^1\}_{t=0}^{T-1}, \{\mathcal{G}_t^2\}_{t=0}^{T-1}, \ldots, \{\mathcal{G}_t^E\}_{t=0}^{T-1}\}$ where T denotes episode length and E is the number of episodes. As the replay buffer always maintains the past experiences from the previous episodes, we have that $\mathcal{G}_0^e = \mathcal{G}_{T-1}^{e-1}$ holds for $\forall e = 1, \ldots, E$. Given that $\mathcal{G}_t^e$ is always a connected directed graph, we describe the expansion behavior as consecutive expansion, which naturally corresponds to exploration process of the agent of approaching the $\mathcal{G}_{\texttt{whole}}$, where we define $\mathcal{G}_{\texttt{whole}}$ as the whole graph containing all possible transitions.

2.3 From Graph Exploration to Graph Boundary

In this subsection, we first provide our theoretical analysis of exploration on graphs which motivates us to use *certainty of state* to define *graph boundary*, and further show that the *goal optimality* can be preserved in the graph boundary.

Proposition 1. (Exploration Bound on Graph) *In the context of deterministic MDPs, we assume that the probability of out-degree of an arbitrary node in $\mathcal{G}_{\texttt{whole}}$ being less than or equal to d is larger than p (i.e., $P(\texttt{deg}(s) \le d) \ge p, \forall s \in S_{\mathcal{G}_{\texttt{whole}}}$). We can ensure that for any $t \ge 0, e \ge 1$, $P\big(|S_{\mathcal{G}_t^e}| \le \epsilon\big) \ge p^\delta$ holds, where $\epsilon = \sum_{t'=0}^{t} \min(e, d^{t'}) \cdot d^{t'} + \sum_{t'=t+1}^{T-1} \min(e-1, d^{t'}) \cdot d^{t'}$ and $\delta = \sum_{t'=0}^{t-1} \texttt{ind}(e > d^{t'}) + \sum_{t'=t}^{T-2} \texttt{ind}(e-1 > d^{t'})$, $\texttt{ind}(\cdot)$ is an indicator function.*

*For the environments with positional observations (e.g., Ant Maze in Figure A6(b)), we discretize the continuous state and action spaces, which converts the continuous space of each dimension into K bins. For environments with image inputs (e.g., VizDoom in Figure A6(e)), we follow the training setup of (Savinov et al., 2018) to train ResNet-18 (He et al., 2016) to obtain representations for these inputs, which can further be used to generate a graph given a pre-defined similarity threshold. As the generated graph is normally large-scale, we further apply K-bins discretization technique to reduce the scale of the graph. We provide further discussions about differences between our graph constructions and previous works in Appendix A4.1.

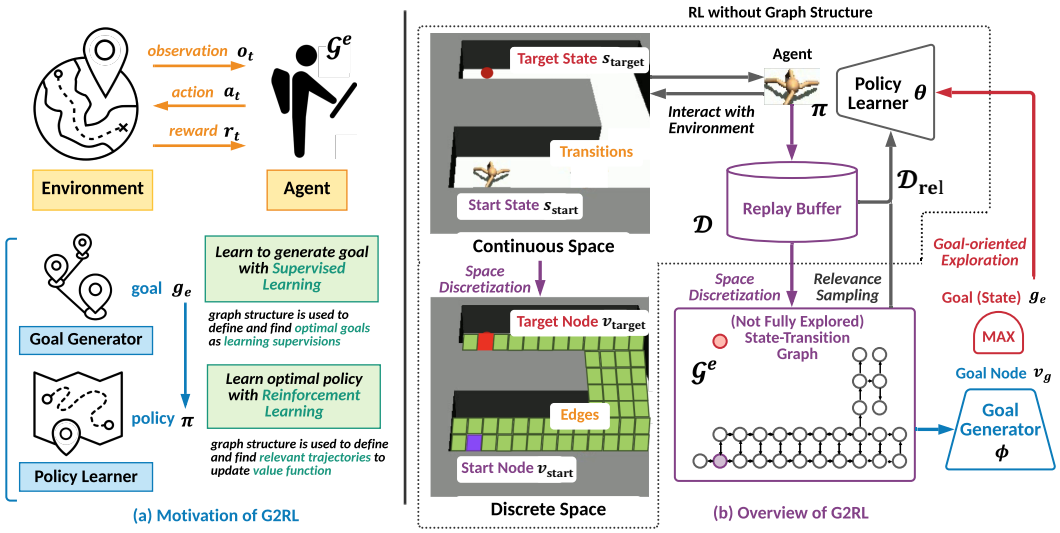


Figure 1: Illustrations of motivation (a) and overview (b) of G2RL. (a) The basic idea of G2RL is to leverage the state-transition graph to build a differentiable goal generator and an efficient policy learner, where the first one can be learned by supervised learning with hindsight supervisions, and the second one can be updated by reinforcement learning with relevance sampling. (b) G2RL first constructs the state-transition graph upon the replay buffer and then uses the graph to generate goal appropriate g_e and select relevant trajectories $\mathcal{D}_{\text{rel}}$. When encountering the continuous environments, we need to apply a space discretization technique mapping a bin of states into a node for graph structure.

The proposition implies that given fixed episode number e and length t, then its status of exploration $|S_{\mathcal{G}_t^e}|$ is largely influenced by the graph property (e.g., out-degree of nodes), which motivates us to leverage the state-transition graph information in goal generation for effective exploration. In the light of this, we further introduce *certainty of state* to associate the status of exploration on a state with its out-degree in the state-transition graph. Specifically, we can approximate the certainty of state $\texttt{cert}(s) \approx \texttt{deg}(s)$, where $\texttt{deg}(s)$ denotes the out-degree of s.[†] For continuous environments, we define the certainty of node by the average certainty of its states (i.e., their average number of taken actions), namely $\texttt{cert}(v) = \texttt{average}_{s \in v}(\texttt{cert}(s))$.

We introduce the boundary of $\mathcal{G}_t^e$, denoted as $\partial\mathcal{G}_t^e$, as a set of states, at least one of whose candidate actions is not readily taken (i.e., $\{s|\texttt{cert}(s) < |A|\}$) for discrete environments, and similarly, as a set of nodes, at least one of discretized actions of whose states is not readily taken (i.e., $\{v|\texttt{cert}(v) < |\widehat{A}|\}$) in continuous environments. Another intuitive way to define the boundary is using state/node visitation counts; however, we can not say that a frequently visited state is well explored if the same action is taken at each visit (see Appendix A4.2 for details). We implement this way as an ablation.

Notably, this definition is different from the landmark or frontier in (Huang et al., 2019; Yamauchi, 1997) which is constructed by farthest point sampling (Arthur & Vassilvitskii, 2006) or frontier detection (Keidar & Kaminka, 2012) approaches regardless of the exploration status of states.

Prior works (Huang et al., 2019; Eysenbach et al., 2019; Laskin et al., 2020) mainly generate goals by first drawing the shortest path through applying Dijkstra's algorithm or A* search and then assigning those intermediate states as goals. Their intuitions are straightforward that if we can access the whole state-transition graph $\mathcal{G}_{\texttt{whole}}$, we can obtain the optimal path/solution by these shortest path planning algorithms. We use $\mathcal{P}_{\texttt{whole}}$ to denote this path. However, such a case seldom happens and $\mathcal{G}_{\texttt{whole}}$ is kind of conceptual. Instead, we here introduce the definition of *optimal goal* based on dynamic graph at the beginning timestep in an arbitrary episode e (i.e., $\mathcal{G}^e$). We begin with extending the concept of *optimal path/solution* as the shortest path connecting the start state and the reachable state with the highest value. For consistency, we always assign the target state with the highest value. We use $\mathcal{P}_e$ to denote the path in $\mathcal{G}^e$, and introduce the definition of the optimal goal as follows.

[†]Notably, this certainty of state can be served as a local measurement to show the extent of exploration on state in discrete environments, which is actually proportioned to the global measurement (i.e., the number of visited states) in deterministic environments (see Proposition 3 in Appendix A4.2 for details). Also, when applied to environments with obstacles, as unable to know whether a state is near the wall, thus the certainty of the state always represents the number of readily taken actions of s, whose maximum value is $|A|$ (See Appendix A4.2 for detailed discussions).

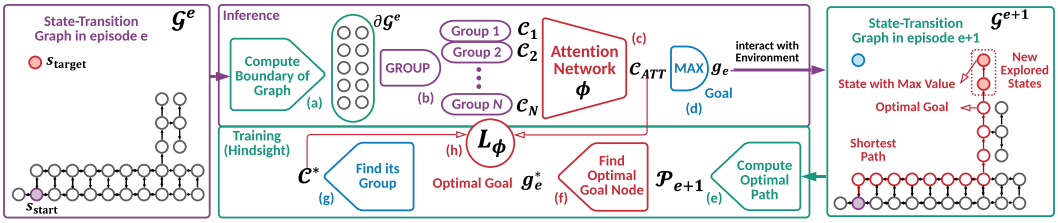


Figure 2: Illustrations of inference and training steps of our goal generator. During inference, we first constraint the candidate set of goal generation within the boundary of graph $\partial\mathcal{G}^e$ with theoretical guarantee (a), and then divide the nodes into several groups (b). A differentiable attention network is applied to select one group $\mathcal{C}_{\texttt{ATT}}$ (c), where the node with the highest value is assigned as the goal g_e (d). Our goal generator is trained in a hindsight fashion. Specifically, we first find the optimal path by planning on $\mathcal{G}^{e+1}$ (e), and then obtain the optimal goal node g_e^* as the most valuable and reachable node in the path $\mathcal{P}_{e+1}$ (f). The supervision signal is the optimal group $\mathcal{C}^*$ containing g_e^* (g). Note that L_ϕ with group supervisions instead goal supervisions can significantly eliminate instability brought from potentially inaccurate value estimation (h).

Definition 1. (Optimal Goal) *For any graph $\mathcal{G}^e$, we define the optimal goal g_e^* in a hindsight manner, where we generate the optimal path $\mathcal{P}_{e+1}$ based on the graph in the next episode $\mathcal{G}^{e+1}$ and assign the state that is included both in $\mathcal{G}^e$ and $\mathcal{P}_{e+1}$ and with the highest value as g_e^*. Formally, we have*

$$g_e^* := \arg\max_s V^\pi(s, g_{e+1}),\ \forall s \in \mathcal{G}^e \cap \mathcal{P}_{e+1}, \tag{2}$$

where g_{e+1} is the generated goal in episode $e+1$, whose computation is later introduced in Eq. (4).

An illustrated example is provided in Figure 2. In seeking for g_e^*, we need to first obtain $\mathcal{P}_{e+1}$ by applying the shortest path planning algorithm on $\mathcal{G}^{e+1}$, and then rank all states in the path according to their value in a descending order, and successively search for the first state in overlapping part of $\mathcal{P}_{e+1}$ and $\mathcal{G}^e$ to assign as the goal. Similarly, in the continuous environments, the optimal goal node, denoted as v_g^* in episode e, is defined as the most valuable and reachable node in the shortest path $\mathcal{P}_{e+1}$ where the value of a node is defined as the maximum value of states in it.

Based on the definition above, to reduce the searching space, it is natural to further ask what kind of states in the episode e have a chance to become g_e^*. We answer this question by introducing the following theorem.

Theorem 1. (Goal Optimality Preserved in Graph Boundary) *In the context of deterministic MDPs, assume that if there exists a path connecting state s and goal g whose length is shorter than any other paths connecting another state s' and the goal g, then $V^\pi(s, g) > V^\pi(s', g)$ holds for $\forall s, s' \in S, g \in G$. Then, we have that g_e^* is always included in the boundary of the graph $\mathcal{G}^e$ (i.e., $g_e^* \in \partial\mathcal{G}^e$) holds for any episode e.*

This theorem implies that the candidate set for each goal generation can be limited to the boundary, which can reduce the goal space with optimality preserved. However, learning to generate an appropriate goal is still non-trivial concerning a dynamic and large-scale state-transition graph $\mathcal{G}^e$.

2.4 Scaling Up GoRL to Dynamic and Large Graphs by Group Division

To scale up GoRL to dynamic and large-scale state-transition graphs, one intuitive solution is to divide $\partial\mathcal{G}^e$ (i.e., the whole candidate goal space) into a fix number of candidate groups, denoted as $\mathcal{C}_1, \ldots, \mathcal{C}_N$, where N is the number of groups. Our group segmentation should follow the principle that $\cup_{n=1}^N \mathcal{C}_n = \partial\mathcal{G}^e$ and $\mathcal{C}_m \cap \mathcal{C}_n = \emptyset$ hold for $\forall m \neq n; m, n = 1, 2, \ldots, N$. To ensure all groups non-empty, we include the last visited state in the previous episode, denoted as $s_{\texttt{last}}$, into all groups $\mathcal{C}_n, n = 1, 2, \ldots, N$ during the initialization. We choose $s_{\texttt{last}}$ here, as $s_{\texttt{last}}$ is often both close to the target state and of high uncertainty. We then extending these groups by adding states within the graph boundary in the following two different ways.

- *Neighbor States* is to use the local neighborhood of $s_{\texttt{last}}$ within graph boundary, where $\mathcal{C}_n := \{s | s \in \mathcal{N}^{n-1}(s_{\texttt{last}}) \cap \partial\mathcal{G}^e\}$ for $n = 1, 2, \ldots, N-1$, and $\mathcal{C}_N := \partial\mathcal{G}^e - \cup_{n=1}^{N-1}\mathcal{C}_n$. $\mathcal{N}^{n-1}(s_{\texttt{last}})$ means $(n-1)$-hop neighbors of $s_{\texttt{last}}$. The intuitive motivation behind this is to keep the learning procedure stable by gradually adjusting the goal to explore the surrounding environment.
- *Uncertain States* is to utilize the certainty information to guide goal generation, where $\mathcal{C}_n := \{s | s \in S_{d=|A|-n} \cap \partial\mathcal{G}^e\}$ for $n = 1, 2, \ldots, N-1$, and $\mathcal{C}_N := \partial\mathcal{G}^e - \cup_{n=1}^{N-1}\mathcal{C}_n$. $S_{d=|A|-n}$ denotes set of states whose certainty equals to $|A| - n$. The intuitive motivation behind this is to eliminate the uncertainty in the graph through exploration.

Let $d_{\partial\mathcal{G}^e}$ denote the maximum out-degree in $\partial\mathcal{G}^e$, and $|S_{\partial\mathcal{G}^e}|$ denote the number of states in $\partial\mathcal{G}^e$. The complexity to construct $\mathcal{C}_1, \ldots, \mathcal{C}_N$ following the first perspective is $O(d_{\partial\mathcal{G}^e}^{N-1})$, and following the second one is $O(|S_{\partial\mathcal{G}^e}|)$. The detailed complexity analysis is available in Appendix A4.5. The above two approaches can be easily extended into continuous environments, where $\mathcal{C}_n := \{v | v \in \mathcal{N}^{n-1}(v_{\texttt{last}}) \cap \partial\mathcal{G}^e\}$ for *Neighbor States* and $\mathcal{C}_n := \{v | v \in \widehat{S}_{d=|\widehat{A}|-n} \cap \partial\mathcal{G}^e\}$ for *Uncertain States*, $v_{\texttt{last}}$ is the node including $s_{\texttt{last}}$.

3 Graph Enhancement for Goal-oriented RL

3.1 Differentiable Goal Generation with Hindsight Graph Supervisions

As the ideal agent is expected to generate appropriate goals that vary at the different learning stages, we propose a differentiable goal generation model that first employs an attention mechanism to select an appropriate one over these groups in term of the current situation and then assigns the state with the highest value in the selected group as the goal.

Concretely, for each group, we first encode a representation vector to represent its feature (e.g., the layer number of neighbors for *Neighbor States*, the certainty of nodes for *Uncertain States*). These representation vectors are then fed into a self-attention mechanism (Vaswani et al., 2017) to select the appropriate one over N groups. We use $\texttt{ATT}_\phi$ to denote self-attention function parameterized by ϕ and provide the detailed descriptions in Appendix A2.2. The output of $\texttt{ATT}_\phi$ is then fed to a multi-layer perceptron (MLP) with the ReLU activation function. The output of MLP is in the dimension $\mathbb{R}^{N\times 1}$, where the n-th element corresponds the logit value for $\mathcal{C}_n$. Then the selected group, denoted as $\mathcal{C}_{\texttt{ATT}}$, can be generated according to

$$\mathcal{C}_{\texttt{ATT}} = \arg\max_{\mathcal{C}_n} \texttt{MLP}(\texttt{ATT}_\phi(\mathcal{C}_1, \ldots, \mathcal{C}_N)), \quad (3)$$

Algorithm 1: G2RL

Initialize buffer $\mathcal{D} = \{s_{\texttt{start}}\}$, $\mathcal{D}_{\texttt{rel}} = \emptyset$
Initialize state-transition graph $\mathcal{G} = \{s_{\texttt{start}}\}$
for *epsiode number* $e = 1, 2, \ldots, E$ **do**
  Select an appropriate group using Eq. (3)
  Generate goal g_e using Eq. (4)
  Compute optimal goal g^*_{e-1} using Eq. (2)
  Update parameter ϕ using Eq. (5)
  for *timestep* $t = 0, 1, 2, \ldots, T-1$ **do**
    Receive observation s_t from environment
    $a_t \leftarrow \epsilon$-greedy policy based on $Q^\pi(s_t, a, g_e)$
    Take action a_t, receive reward r_t and next state s_{t+1}
    Append $(s_t, a_t, r_t, s_{t+1}, g_e)$ to $\mathcal{D}$
    Relabel rewards r_g with g_e
    Append (s_t, a_t, s_{t+1}) to $\mathcal{G}$ if $(s_t, a_t, s_{t+1}) \notin \mathcal{G}$
    if t *mod update_interval* $== 0$ **then**
      Update $\mathcal{D}_{\texttt{rel}}$ by sampling from $\mathcal{D}$
      Update parameter θ using Eq. (6)
    end
  end
end

where $\mathcal{C}_{\texttt{ATT}}$ is selected by the attention score. Next, we select the state with the highest value in $\mathcal{C}_{\texttt{ATT}}$ as the goal, which can be formulated as

$$g_e = \arg\max_s V^\pi(s, g_{e-1}), \ \forall s \in \mathcal{C}_{\texttt{ATT}}. \quad (4)$$

When applying our goal generator in continuous environments, as shown in Figure 1, the output of Eq. (4) is a goal node consisting of a bin of states. We further assign the state with the highest value within the bin of states as the goal.

As we are able to obtain the goal at episode e (i.e., g_e) by Eq. (4), we can further calculate the value function at episode e (i.e., $V^\pi(s, g_e)$), which is the key component to generate the optimal goal at episode $e-1$ (i.e., g^*_{e-1}) according to Eq. (2). Instead of directly using the optimal goal defined in Definition 1 as the supervision signals, we find the *optimal group* $\mathcal{C}^*$ that contains the optimal goal g^*_{e-1}, as the hindsight supervision on the group selection. This supervision signal enables us to update the attention network in Eq. (3) via a standard supervised learning algorithm, where the objective function can be formulated as

$$L_\phi = \mathbb{E}_{(s,a,g,s',r)\sim\mathcal{D}} \left[(\mathcal{C}^* - \mathcal{C}_{\texttt{ATT}_\phi})^2 + \alpha \cdot \|\phi\|_2^2 \right], \quad (5)$$

where $\|\phi\|_2^2$ is the L_2 regularizer and α is the corresponding hyper-parameter.

Note that L_ϕ, which updates goal generation under the supervision of the group instead of the goal, can significantly eliminate instability brought from potentially inaccurate value estimation, as our group division does not depend on any result from policy learning.

3.2 EFFICIENT POLICY LEARNING WITH RELEVANCE SAMPLING ON GRAPH

With a generated goal at the beginning of each episode e, we can build the key blocks of our method, i.e., a goal-oriented policy and its associated value function. We use an off-policy algorithm to learn such a policy π, as well as its associated goal-oriented Q-function. For example, we obtain a policy by acting greedily, *w.r.t.*, the Q-function as $Q^{\pi}(s,a,g) \leftarrow r_g(s,a) + \gamma \cdot \max_{a'} Q^{\pi}(s',a',g)$. A principal way for policy learning is to randomly sample past trajectories with goal relabeling technique (Andrychowicz et al., 2017) to update parameters. However, as stated in (Fang et al., 2019), random sampling over all historical trajectories would result in inefficiency of training, as not all past experiences are equally useful to policy learning. In seeking for an efficient policy learning strategy, one feasible solution is importance sampling technique; unfortunately, its estimation is often of unnecessarily high variance (Mahmood et al., 2014). One possible reason is that when the current policy diverges from the old policy too much, the accuracy decreases. This observation motivates TRPO (Schulman et al., 2015) to reposition the optimization with constraints of not going too far in changing the policy. In this light of this, we propose a novel *relevance sampling* technique based on a simple assumption: those trajectories containing the node or its neighbor nodes of the current states are likely to share the close goals and the similar goal-oriented policies. Formally, when we update Q-function of state s, we sample $\mathcal{D}_{\texttt{rel}} := \{\tau | \tau \cap (\mathcal{N}^1(s) \cup \{s\}) \neq \emptyset\}$ from replay buffer $\mathcal{D}$, where each sampled trajectory contains at least one state in the neighborhood of s (i.e., $\mathcal{N}^1(s) \cup \{s\}$). It can be easily extended into continuous environments by defining $\mathcal{D}_{\texttt{rel}} := \{\tau | \tau \cap (\mathcal{N}^1(v) \cup v) \neq \emptyset\}$ where v is the node including s. We also exam the effect of the scope of neighborhood in the experiment. The Q-network is learned by minimizing

$$L_\theta = \mathbb{E}_{(s,a,g,s',r)\sim\mathcal{D}_{\text{rel}}}\big[(r_g + \gamma \cdot \max_{a'} Q^{\pi_\theta}(s',a',g) - Q^{\pi_\theta}(s,a,g))^2 + \beta \cdot \|\theta\|_2^2\big], \tag{6}$$

where β is the weight of the regularization term.

We provide the overall G2RL algorithm in Algorithm 1 (See the discussions on novelties and limitations of the algorithm in Appendix A2.1), where the state-transition graph are applied to both the goal generation and policy learning phases. Besides, G2RL holds the property to converge to the unique optimum if Q-learning strategy is adopted, which is further discussed in Appendix A3.3.

4 EXPERIMENTS

4.1 COMPARATIVE EXPERIMENTS

We evaluate the performance of G2RL against the state-of-the-art RL algorithms including (1) *HER* (Andrychowicz et al., 2017), (2) *MLP* (Huang et al., 2019), (3) *GoalGAN* (Florensa et al., 2018), (4) *CHER* (Fang et al., 2019), (5) *SPTM* (Savinov et al., 2018), (6) *SoRB* (Eysenbach et al., 2019), (7) *HIRO* (Nachum et al., 2018b), (8) *HRAC* (Zhang et al., 2020), (9) *GTG* (Jiang et al., 2021) over discrete and continuous environments. Concretely, the discrete task includes Maze, and the continuous tasks contain Ant Maze, Low-Stochastic and High-Stochastic Ant Maze, FetchPush, FetchPush with Obstacle, VisDoom and Matterport3D (see Appendix A5.1 for detailed descriptions of these environments), where Ant Maze, VisDoom (Kempka et al., 2016) and Matterport3D (Chang et al., 2017) are based on Maze but more challenging with either high-dimensional state spaces or image input, Low-Stochastic and High-Stochastic Ant Mazes are stochastic versions of Ant Maze where the connectivity of mazes keeps changing with low and high probabilities, FetchPush and FetchPush with Obstacle are standard robotic manipulation environments based on OpenAI Gym (Brockman et al., 2016). Notably, these environments are widely used benchmarks in GoRL community (Florensa et al., 2017; Savinov et al., 2018; Ren et al., 2019; Zhang et al., 2020; Nachum et al., 2018a; Kempka et al., 2016; Chaplot et al., 2020a). For comprehensive evaluation for the performance of G2RL, we employ DQN (Mnih et al., 2013) on discrete tasks and DDPG (Lillicrap et al., 2015) on continuous tasks. For VizDoom and Matterport3D environments, we follow (Savinov et al., 2018; Chaplot et al., 2020a) and introduce ResNet-18 (He et al., 2016) and Mask-RCNN (He et al., 2017) to handle RGB images obtained from the environments. We provide detailed descriptions of baselines, evaluation environments, implementation details in Appendix A5.

The learning curves of G2RL and baselines across all tasks are plotted in Figure 3 (See the results of High-Stochastic AntMaze in Appendix A6.1). In the Maze with discrete state and action spaces, G2RL achieves comparable results with these state-of-art methods, while in other tasks G2RL consistently surpasses all baselines both in sample efficiency and asymptotic performance. We test the

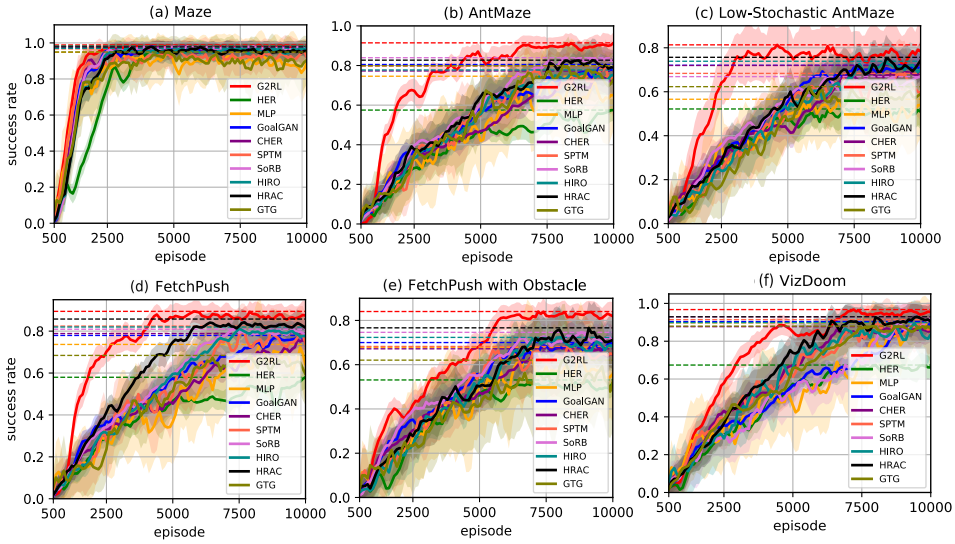


Figure 3: Comparison of learning curves of our model G2RL against baselines algorithms on on various environments average across 10 different random seeds. The solid curves depict the mean, the shaded areas indicate the standard deviation, and dashed horizontal lines show the asymptotic performance.

generalizability of G2RL to the unseen environments of Maze and Matterport3D in Appendix A6.2, whose results verify the consistent superiority of G2RL.

We further check whether G2RL is able to generate meaningful goals for exploration by visualizing distributions of goals generated by G2RL, SoRB and HER in Ant Maze. As shown in Figure A9 in Appendix A6.3, the goals generated by G2RL gradually move towards the target state. Those goals selected from the visited states are considered to be reachable during training. In comparison, the goal distribution of HER has been stuck around the initial state for many episodes. We also report the log files of G2RL and HER in Maze in Appendix A6.4. One can observe that compared to HER, G2RL can sample more states and take more actions in the same number of episodes.

4.2 Ablation Studies on Model Design and Experimental Setting

Impact of Relevance Sampling. We investigate the impact of using the local structure of the abstraction graph to select relevant trajectories for efficient policy learning. We build a variant to uniformly drawing trajectories from the replay buffer $\mathcal{D}$ denoted as *UNIFORM*. We use *RELEVANCE1*, *RELEVANCE3*, *RELEVANCE10* to denote our method, where $\mathcal{D}_{\texttt{rel}}$ with the neighborhood scope of 1, 3, 10 are used instead of $\mathcal{D}$ respectively. We evaluate these setting in AntMaze and results are reported in Figure 4(a). The results show that with the relevance sampling, the training of G2RL can be much more efficient. We investigate its broader impact by solely incorporating relevance sampling with other GoRL algorithms, and report results in Appendix A6.1..

Impact of Group Selection. We provide two strategies to divide the boundary of the abstraction graph into several groups, namely building groups from *Neighbor States* of last visited states or *Uncertain States* in the graph. We demonstrate the performance of G2RL using neither of them denoted as *NOGroup*. We show the performance of adopting the first strategy and set the number of groups as 3 denoted as *NEIGH3*. We then illustrate the performance of using the second strategy with the number of groups equal to 2, 3, 4 denoted as *UNCERT2, UNCERT3, UNCERT4*. Figure 4(c) shows that *UNCERT2, UNCERT3, UNCERT4* perform better than *NOGroup* in Ant Maze, which indicates the necessity of the group selection. In the main experiment, we adopt *UNCERT3*.

Impact of Discretization. *DISCRETE10, DISCRETE20, DISCRETE30* in Figure 4(d) denote the set of $K = 10, 20, 30$ in K-bins discretization. From results on the AntMaze, we find that for simple tasks, the choice of K is not critical. K is set as 20 in the main experiment.

Impact of Graph Boundary. For investigation of impact of graph boundary, we build three variants for graph boundary construction, namely *CERT* using the certainty of states, *VISIT5* and *VISIT10* using 5% and 10% most-infrequently visited states, *RAN* randomly sampling states whose sizes are the same as *CERT*, to build the graph boundary. As the group selection performs on the graph boundary, we divide the same number of groups for *VISIT5* and *VISIT10* according to the visitation counts of states, and randomly build the groups for *RAN*. We evaluate the performance of these methods in the Ant Maze. As results shown in Figure 4(e), we can observe that *CERT* outperforms *VISIT5* and *VISIT10*, and significantly surpasses *RAN*.

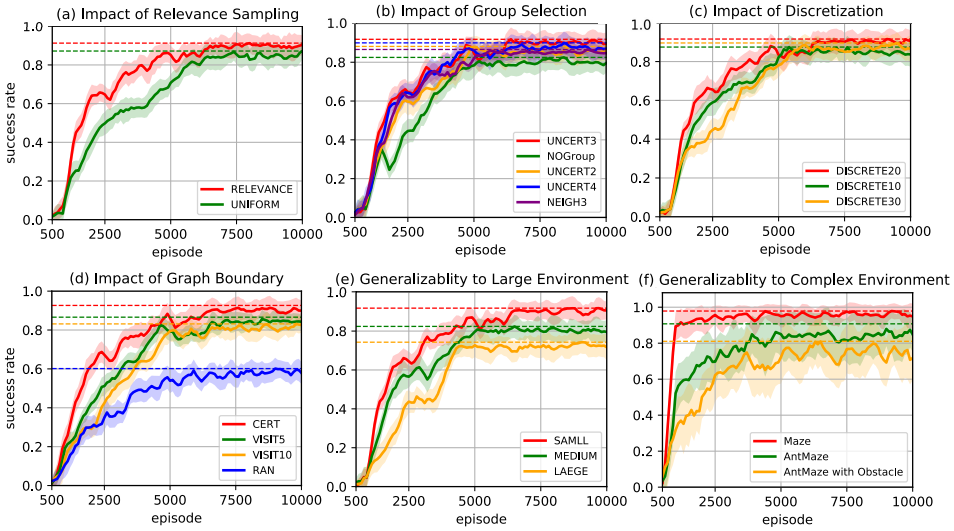


Figure 4: Comparison of learning curves of G2RL in different settings mainly on Ant Maze average across 10 different random seeds. These ablation studies verify our architecture design and evaluate the generalizability.

Generalizablity to Large Environment. To investigate whether G2RL can be well adapted in the environments with different sizes, we extend the AntMaze with a larger size, which usually means more sparse rewards. We report the performance comparisons on the three sizes namely *SMALL*, *MEDIUM*, *LARGE*, where *SMALL* is the original size. More details of these environment configuration are available in Appendix A5.1. Results shown in Figure 4(f) indicate that G2RL can well address the sparse reward issues in various environments.

Generalizability to Complex Environment. For further investigation on whether G2RL can be well adapted in the environments with different complexity levels, we create *AntMaze with Obstacle* environment, where we extend the AntMaze with an obstacle that can not be removed. For ease of the comparison, we also depict the performance of G2RL in the Maze, AntMaze environment. Results are illustrated in Figure 4(g), showing that G2RL performs well in these environments.

We test G2RL with different levels (i.e., $\sigma = 0.5, 0.75, 1$) of Gaussian noises $\widetilde{\mathcal{N}}(0, \sigma^2)$ on state-transition graph, provide detailed settings in Appendix A5.1 and report results in Appendix A6.1.

From above ablation studies, we can conclude the performance gain of G2RL mainly from our differentiable goal generation algorithm and slightly from our relevance sampling algorithm, which relies on a deterministic (or low-stochastic) MDPs and an inaccurate state-transition graphs, consistent with the assumptions introduced in Theorem 1.

5 Related Work

GoRL (Florensa et al., 2018; Paul et al., 2019) allows the agent to generate intrinsic rewards, which is defined with respect to target subsets of the state space called goals. Then, how to generate appropriate goals is the essential technique in any GoRL (Andrychowicz et al., 2017; Ren et al., 2019). There are previous literature (Levy et al., 2017; Kulkarni et al., 2016b; Şimşek et al., 2005; Noelle, 2019; Nachum et al., 2018b;a; Schaul et al., 2015) focusing on the goal generation based on potentially pivotal states; however, are not adaptively aligned with the exploration status and thus often sub-optimal. Several prior works have investigate to build an environmental graph for high-level planning and used searching algorithm to find nearby graph nodes as reachable goals for the low-level (Zhang et al., 2018; Eysenbach et al., 2019; Huang et al., 2019; Savinov et al., 2018; Laskin et al., 2020; Klissarov & Precup, 2020; Jiang et al., 2021). However, these approaches use hard-coded the high-level planning, instead of a learning fashion and thus are limited in scalability. G2RL leverages the state-transition graph to build a differentiable goal generator for the exploration.

6 Conclusion

In this paper, we propose a novel GoRL framework called G2RL, which leverages structure information of the abstraction graph to develop a differentiable goal generator for explorative goal generation and a novel relevance sampling techniques for efficient policy learning. For future work, it is interesting to further investigate theoretical supports for incorporating graph structure with GoRL in context of stochastic MDPs.

REFERENCES

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 25

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NeurIPS*, 2017. 2, 7, 9, 20, 23, 24

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006. 4, 20

Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, Joseph Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, 2018. 3, 18

Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *NeurIPS*, 2016. 18

Richard Bellman. Dynamic programming. *Science*, 1966. 17

Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995. 17

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 7

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 7, 22

Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *In Neural Information Processing Systems (NeurIPS)*, 2020a. 7, 18, 20, 22, 23, 25

Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155*, 2020b. 25

Michael Colombo, Tom Fernandez, Katsuki Nakamura, and Charles G Gross. Functional differentiation along the anterior-posterior axis of the hippocampus in monkeys. *Journal of Neurophysiology*, 80(2):1002–1005, 1998. 3, 18

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *NeurIPS*, 2019. 1, 2, 3, 4, 7, 9, 14, 18

Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. In *NeurIPS*, 2019. 1, 7, 23

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *ICML*, 2017. 7, 20

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *ICML*, 2018. 2, 7, 9, 23

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 7, 18, 22, 23

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017. 7, 18, 22, 23

Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. *NeurIPS*, 2019. 1, 3, 4, 7, 9, 14, 18, 20, 23

Zhengyao Jiang, Pasquale Minervin, Minqi Jiang, and Tim Rocktäschel. Grid-to-graph: Flexible spatial relational inductive biases for reinforcement learning. In *AAMAS 2021*, 2021. 7, 9, 23, 25

Matan Keidar and Gal A Kaminka. Robot exploration with fast frontier detection: theory and experiments. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 113–120, 2012. 4, 19

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 2016. 7, 22

Martin Klissarov and Doina Precup. Reward propagation using graph convolutional networks. *arXiv preprint arXiv:2010.02474*, 2020. 9, 18

Sotiris Kotsiantis and Dimitris Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006. 2, 3, 23

Tejas D Kulkarni, Karthik R Narasimhan, Ardavan Saeedi, and Joshua B Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *NeurIPS*, 2016a. 1

Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016b. 9

Michael Laskin, Scott Emmons, Ajay Jain, Thanard Kurutach, Pieter Abbeel, and Deepak Pathak. Sparse graphical memory for robust planning. *NeurIPS*, 2020. 1, 4, 9, 14

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *ICLR*, 2017. 1, 9

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv*, 2015. 7, 23

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014. 22

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017. 22

Ashique Rupam Mahmood, Hado Van Hasselt, and Richard S Sutton. Weighted importance sampling for off-policy learning with linear function approximation. In *NIPS*, 2014. 7

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. On the effectiveness of least squares generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(12):2947–2960, 2018. 23

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 7

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018a. 2, 7, 9

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *NeurIPS*, 2018b. 2, 7, 9, 23

Soroush Nasiriany, Vitchyr H Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. *NeurIPS*, 2019. 1

David C Noelle. Unsupervised methods for subgoal discovery during intrinsic motivation in model-free hierarchical reinforcement learning. In *KEG@AAAI*, 2019. 9

John O'Keefe and Jonathan Dostrovsky. The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 1971. 3, 18

Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *ICML*, 2017. 18

Sujoy Paul, Jeroen van Baar, and Amit K Roy-Chowdhury. Learning from trajectories via subgoal discovery. *NeurIPS*, 2019. 1, 2, 9

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 18

Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. In *NeurIPS*, 2019. 7, 9, 20

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *ICLR*, 2018. 2, 3, 7, 9, 18, 20, 21, 22, 23, 25

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9339–9347, 2019. 22

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *ICML*, 2015. 9

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607. Springer, 2018. 23

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015. 7

Wenling Shang, Alex Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Learning world graphs to accelerate hierarchical reinforcement learning. In *ICML Workshop*, 2019. 1

Özgür Şimşek, Alicia P Wolfe, and Andrew G Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *ICML*, 2005. 9

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 17

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 6, 14

Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pp. 146–151. IEEE, 1997. 4, 19

Amy Zhang, Sainbayar Sukhbaatar, Adam Lerer, Arthur Szlam, and Rob Fergus. Composable planning with attributes. In *ICML*, 2018. 9

Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *NeurIPS*, 2020. 1, 2, 7, 14, 16, 23

Guangxiang Zhu, Zichuan Lin, Guangwen Yang, and Chongjie Zhang. Episodic reinforcement learning with associative memory. In *ICLR*, 2019. 1, 3, 14, 18

A1 ILLUSTRATIONS OF NOTATIONS

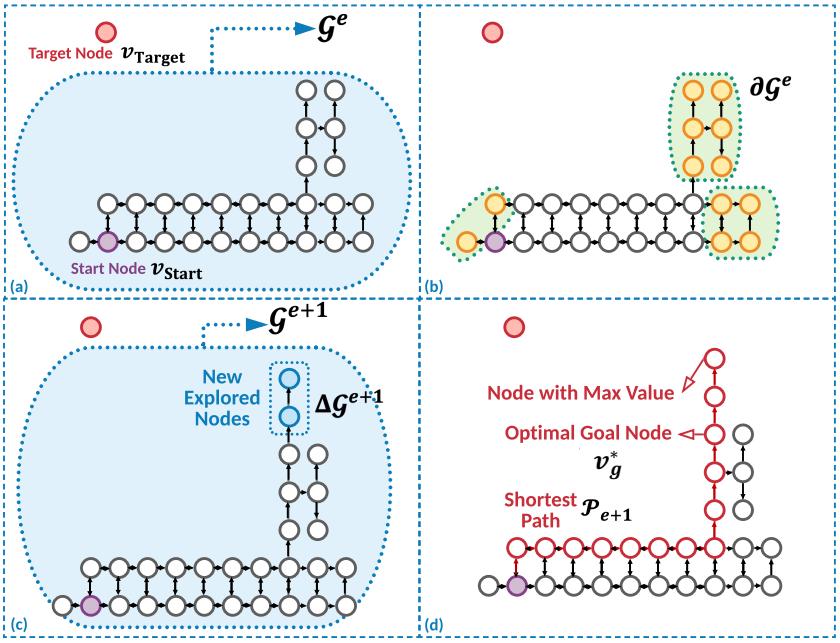


Figure A1: An illustrated example of notations in G2RL. We use purple circle to denote the start node $v_{\texttt{start}}$

In this paper, we construct a state-transition graph $\mathcal{G}$ on top of the replay buffer $\mathcal{D}$. Specifically, as Figure A1(a) shows, we use $\mathcal{G}^e$ to denote the graph built based on historical explored trajectories at the beginning in episode e. We use $v_{\texttt{start}}$ and $v_{\texttt{target}}$ to denote the start and target nodes. Except the whole state-transition graph containing all possible transitions, there exist many poorly-explored states. We measure the exploration (defined as certainty in Section 2.3) of these states according to the number of their untaken candidate actions. As an untaken action may lead to certain unvisited state, we further introduce the definition of the boundary of graph as the set of all the states with at least one untaken action. We use $\partial\mathcal{G}^e$ as the boundary of $\mathcal{G}^e$, as Figure A1(b) shows. As shown in Figure A1(c), after the explorations of one episode, the agent encounters several new states, the set of which is defined as the graph increment. Formally, we use $\Delta\mathcal{G}^{e+1}$ to denote the graph increment from $\mathcal{G}^e$ to $\mathcal{G}^{e+1}$, namely $\mathcal{G}^{e+1} = \mathcal{G}^e \cup \Delta\mathcal{G}^{e+1}$.

We provide the illustration of finding the optimal goal node v_g for episode e in Figure A1(d). Beginning with $\mathcal{G}^{e+1}$, we first draw the optimal path by planning the shortest path connecting the start node to the node with the highest value. The value of each node is defined by its highest value of states in it. We use $\mathcal{P}_{e+1}$ to denote the optimal path. We then calculate the set of nodes visited both in $\mathcal{P}_{e+1}$ and $\mathcal{G}^e$, where we select one node with the highest value as the goal node. Besides these notations on the state-transition graph, we also introduce $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_N$ to denote N group divisions of candidate nodes.

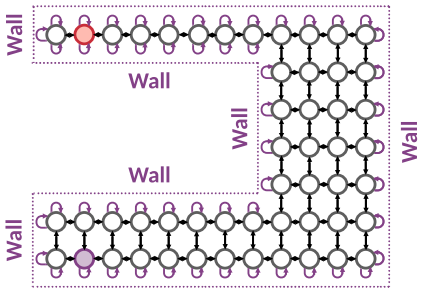


Figure A2: An illustrated example for the state-transition graph.

Remark. Note that the sizes of action space (i.e., the maximum out-degree) of all the states are the same. When applying the proposed algorithm to the environments with obstacles, the state-transitions would draw self-loops on the states near the obstacles (e.g., walls). As shown in Figure A2, for those states neighboring the wall, their actions toward the wall would result in the self-loop edges (highlighted in purple colors).

Notably, for convenience and clarity, all the illustrations in this paper do not include the self-loop connections (i.e., purple edges in Figure A2), and assume that all the actions resulting in the self-loop connections have been taken.

A2 ALGORITHM

A2.1 OVERALL ALGORITHM ANALYSIS

Novelties. From Algorithm 1, there are mainly three differences respect to previous work: a way to generate the graph, a way to generate the goals, and a way to sample transitions from replay buffer, where the first one is regarding the graph construction and the second two are about use of the graph to enhance GoRL. As stated in Section 1, the main insight of G2RL is how to leverage the hidden structure in state-transition graphs to enhance GoRL, which is introduced in Section 3 including a novel effective goal generation strategy (i.e., *differentiable goal generation with hindsight graph supervisions*) and a new efficient policy learning method (i.e., *relevance trajectory sampling technique*), which is highlighted in purple color in Algorithm 1. And, the K-bins discretization technique, used to build a discretized state-transition graph from replay buffer, would be regarded as a data pre-processing procedure of G2RL (See Appendix A4.1 for further discussion).

Limitations. The main limitation of our theoretical analysis is that all our results are derived in the context of deterministic MDPs, following the assumption in (Zhang et al., 2020). However, we argue that these findings are still instructive for designing practical GoRL algorithms, as we note that many real-world applications can be approximated as deterministic environments where the stochasticity is mainly induced by noise.

Empirically, we demonstrate that G2RL is robust to low stochasticity level (see Figure 3(c)) but fails at high stochasticity (see Figure A8(b)), which we argue is the general bottleneck of any GoRL (Huang et al., 2019; Eysenbach et al., 2019; Laskin et al., 2020; Zhu et al., 2019) using the state-transition graph, as the performances of these methods (e.g., SoRB, GTG) significantly drop when comparing Figure A8(b) to Figure 3(b). We leave rigorous theoretical analysis in future work.

A2.2 IMPLEMENTATION DETAILS OF ATTENTION

In Section 3.1, we introduce a self-attention mechanism (Vaswani et al., 2017) parameterized by ϕ denoted as $\texttt{ATT}_\phi$ to select an appropriate group over all the candidate groups. Concretely, we first structure the embedding vectors of groups, denoted as $F = [f_1, \dots, f_N]^\top \in \mathbb{R}^{N \times F_d}$ where F_d is the embedding dimension of each group $\mathcal{C}_n$, $n = 1, 2, \dots, N$. Each feature f_n is the one-hot representation vector of n, whose meaning is specific regarding the approach to building these groups. If *Neighbor States* is applied, then f_n is the one-hot representation vector of the layer number of neighbors (i.e., n); while if *Uncertain States* is applied, then f_n is the one-hot representation vector of the certainty of states or nodes (i.e., $|A| - n$ or $|\widehat{A}| - n$).

We then build $F_Q = F_K = F_V = F$, and adopt the attention network considering the features of all groups as

$$\texttt{ATT}_\phi(\mathcal{C}_1, \dots, \mathcal{C}_N) = \texttt{softmax}\Big(\frac{F_Q F_K^T}{\sqrt{N}}\Big) F_V. \tag{7}$$

A3 PROOFS

A3.1 PROOF OF PROPOSITION 1

Proposition 1. (Exploration Bound on Graph) *In the context of deterministic MDPs, we assume that the probability of out-degree of an arbitrary node in $\mathcal{G}_{\texttt{whole}}$ being less than or equal to d is larger than p (i.e., $P(\texttt{deg}(s) \le d) \ge p, \forall s \in S_{\mathcal{G}_{\texttt{whole}}}$). We can ensure that for any $t \ge 0, e \ge 1$, $P\big(|S_{\mathcal{G}_t^e}| \le \epsilon\big) \ge p^\delta$ holds, where $\epsilon = \sum_{t'=0}^{t} \min(e, d^{t'}) \cdot d^{t'} + \sum_{t'=t+1}^{T-1} \min(e-1, d^{t'}) \cdot d^{t'}$ and $\delta = \sum_{t'=0}^{t-1} \texttt{ind}(e > d^{t'}) + \sum_{t'=t}^{T-2} \texttt{ind}(e-1 > d^{t'})$, $\texttt{ind}(\cdot)$ is an indicator function.*

Proof. Considering that the graph of G2RL is the state-transition graph, we only can obtain new states by the new explored trajectories instead of consecutively adding nodes which naturally encode the dependency of states. We illustrate an example to expand from $\mathcal{G}^e$ to $\mathcal{G}^{e+1}$ in Figure A3(a)-(c). Hence, we can see that the original expansion order of the sub-graphs at the timestep level is $\{\{\mathcal{G}_t^1\}_{t=0}^{T-1}, \{\mathcal{G}_t^2\}_{t=0}^{T-1}, \dots, \{\mathcal{G}_t^E\}_{t=0}^{T-1}\}$, starting with $\mathcal{G}_0^0 = \{s_0\}$. We then can re-order the expansion order of these sub-graphs as $\{\{\mathcal{G}_0^e\}_{e=1}^{E}, \{\mathcal{G}_0^e\}_{e=1}^{E}, \dots, \{\mathcal{G}_0^e\}_{e=1}^{E}\}$, starting with $\mathcal{G}_0^0 = \{s_{\texttt{start}}\}$.

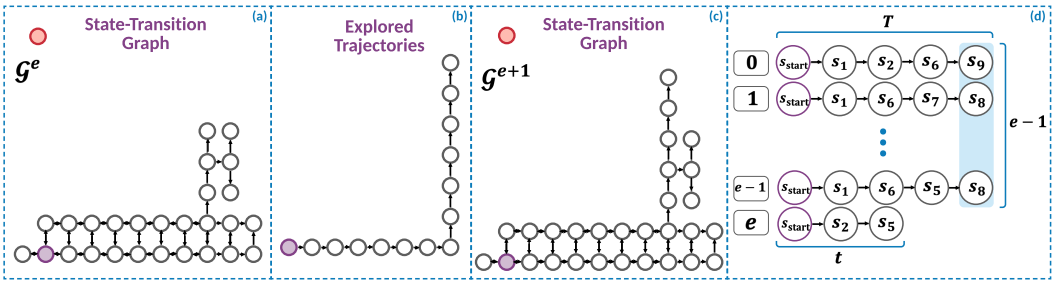


Figure A3: An illustrated example of the expansion of the state-transition graph and intuition of the proof. The natural expansion of the graph is by adding new trajectories, as shown in (a)-(c). In the proof of Proposition 1, we consider the expansion by involving the nodes within the neighborhood, namely adding nodes column-wisely instead of row-wisely in (d).

Let $\Delta\mathcal{G}_t^e$ denote the graph increment at timestep t in episode e, and we also define $\Delta\mathcal{G}_0^1 = 0$. We can ensure that $\mathcal{G}_{t+1}^e = \mathcal{G}_t^e \cup \Delta\mathcal{G}_t^e$ holds for any e and t. Then, the re-order operation can be formulated as

$$\begin{aligned}\mathcal{G}_t^e &= \mathcal{G}_{T-1}^{e-1} \cup (\cup_{t'=0}^{t} \Delta\mathcal{G}_{t'}^e) = \mathcal{G}_0^0 \cup (\cup_{e'=1}^{e-1}\{\cup_{t'=0}^{T-1}\Delta\mathcal{G}_{t'}^{e'}\}) \cup (\cup_{t'=0}^{t}\Delta\mathcal{G}_{t'}^e) \\ &= \mathcal{G}_0^0 \cup (\cup_{t'=0}^{T-1}\{\cup_{e'=1}^{e-1}\Delta\mathcal{G}_{t'}^{e'}\}) \cup (\cup_{t'=0}^{t}\Delta\mathcal{G}_{t'}^e)\end{aligned} \tag{8}$$

holds at any timestep t in episode e. Eq. (8) allows us to adding the node according to their timesteps. One can easily see that at timestep t in any episode, since the start point is always the start state, the explored state is at most t-hop neighbor of the initial state. Notably, all the subgraphs $\mathcal{G}_t^e$ are a connected graph, which is preserved after the re-order operation.

Here, we here first consider $\mathcal{G}_{T-1}^{e-1}$. For simplicity, we omit the mark of episode here. We define $\Delta\mathcal{G}_t = \cup_{e'=1}^{e-1}\Delta\mathcal{G}_{t'}^{e'}$, and $\mathcal{G}_{t'} = \mathcal{G}_0 \cup \Delta\mathcal{G}_1 \cup \cdots \cup \Delta\mathcal{G}_{t'}$ where $\mathcal{G}_0 = \mathcal{G}_0^0$. Let ΔS_t denote $S_{\Delta\mathcal{G}_t}$ for short. Based on this, we study the sequence $\{\Delta\mathcal{G}_1, \ldots, \Delta\mathcal{G}_{T-1}\}$, where $\mathcal{G}_{T-1}^{e-1} = \mathcal{G}_{T-1} = \mathcal{G}_0 \cup \Delta\mathcal{G}_1 \cup \cdots \cup \Delta\mathcal{G}_{T-1}$. Since all the graphs in the sequence $\{\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{T-1}\}$ are the connected graph. Although each node in ΔS_t already has at least one edge within $\mathcal{G}_{t-1}$ due to the definition of connected graphs, we here only measure the out-degree, and thus have

$$P\big(|\Delta S_t| \le |\Delta S_{t-1}| \cdot d\big) \ge p^{|\Delta S_{t-1}|}. \tag{9}$$

For $e = 0$ and $t = 0$, there is the only the start state in the graph. Therefore, we have $P(|S_{\mathcal{G}_0^0}| \le 1) = 1$ and thus

$$P(|S_{\mathcal{G}_0^0}| \le 1) \ge p^0. \tag{10}$$

For $e \ge 1$ and $t \ge 0$, we first analyze the consecutive expansion of the state-transition graph $\mathcal{G}_{T-1}^{e-1}$ as

$$\begin{aligned}&\mathcal{G}^1 \to \mathcal{G}^2 \to \cdots \to \mathcal{G}^{e-1} \\ \Rightarrow &\underbrace{\mathcal{G}_0^1 \to \mathcal{G}_1^1 \to \cdots \to \mathcal{G}_{T-1}^1}_{\mathcal{G}^1} \to \underbrace{\mathcal{G}_0^2 \to \mathcal{G}_1^2 \to \cdots \to \mathcal{G}_{T-1}^2}_{\mathcal{G}^2} \to \cdots \to \underbrace{\mathcal{G}_0^{e-1} \to \mathcal{G}_1^{e-1} \to \cdots \to \mathcal{G}_{T-1}^{e-1}}_{\mathcal{G}^{e-1}} \\ \Rightarrow &\underbrace{\mathcal{G}_0^1 \to \mathcal{G}_0^2 \to \cdots \to \mathcal{G}_0^{e-1}}_{\mathcal{G}_1} \to \underbrace{\mathcal{G}_1^1 \to \mathcal{G}_1^2 \to \cdots \to \mathcal{G}_1^{e-1}}_{\mathcal{G}_2} \to \cdots \to \underbrace{\mathcal{G}_{T-1}^1 \to \mathcal{G}_{T-1}^2 \to \cdots \to \mathcal{G}_{T-1}^{e-1}}_{\mathcal{G}_{T-1}}.\end{aligned} \tag{11}$$

Based on $|S_{\mathcal{G}_{T-1}^{e-1}}| = |S_{\mathcal{G}_0^0}| + |\Delta S_{\mathcal{G}_1}| + |\Delta S_{\mathcal{G}_2}| + \cdots + |\Delta S_{\mathcal{G}_{T-1}}|$, we have

$$P\big(|S_{\mathcal{G}_{T-1}^{e-1}}| \le 1 + d + d \cdot d + \cdots + d^{T-1}\big) \ge p^{0+1+d+\cdots+d^{T-2}}. \tag{12}$$

However, for $\mathcal{G}_{T-1}^{e-1}$ we notice that for each timestep t, the graph increment is always bound with $e-1$, namely

$$P(|\Delta S_t| \le e - 1) = 1 \tag{13}$$

holds for any $t \le T-1$. Hence, we can re-formulate Eq. (12) as

$$P\big(|S_{\mathcal{G}_{T-1}^{e-1}}| \le \sum_{t'=0}^{T-1} \min(e-1, d^{t'})\big) \ge p^{\sum_{t'=0}^{T-2} \mathtt{ind}(e-1 > d^{t'})}, \tag{14}$$

where $\texttt{ind}(\cdot)$ denotes an indicator function that $\texttt{ind}(\texttt{true}) = 1$ and $\texttt{ind}(\texttt{false}) = 0$ hold. The intuition behind Eq. (15) is straightforward that for any timestep t when the expansion $|\Delta S_t|$ is larger than $e-1$ then this expansion is constricted by episode number (see Eq. (13)), and otherwise is bound by the out-degree of $\mathcal{G}_{T-1}^{e-1}$ (see Eq. (9)).

We then investigate $\mathcal{G}_t^e$ with arbitrary e and t. As illustrated in Figure A3(d), in episode e, since the timestep $t \leq T$, we can need to divide the timesteps into two parts. One part is $t' \leq t$, where the expansion for each timestep is bound by e while in the other part, the expansion for each timestep is bound by $e-1$. Combining these two parts together, we have

$$P\big(|S_{\mathcal{G}_{T-1}^{e-1}}| \leq \sum_{t'=0}^{t} \min(e, d^{t'}) \cdot d^{t'} + \sum_{t'=t+1}^{T-1} \min(e-1, d^{t'}) \cdot d^{t'}\big) \geq p^{\sum_{t'=0}^{t-1} \texttt{ind}(e > d^{t'}) + \sum_{t'=t}^{T-2} \texttt{ind}(e-1 > d^{t'})}. \tag{15}$$

Notably, we here don't explicitly consider the effect of K-bins discretization. However, considering that K-bins discretization only can mapping multiple states into one, thus the inequality still holds. □

A3.2 Proof of Theorem 1

Theorem 1. (Goal Optimality Preserved in Graph Boundary) *In the context of deterministic MDPs, assume that if there exists a path connecting a state s and a goal g whose length is shorter than any other paths connecting another state s' and the goal g, then $V^\pi(s, g) > V^\pi(s', g)$ holds for $\forall s, s' \in S, g \in G$. Then, we have that g_e^* is always included in the boundary of the graph $\mathcal{G}^e$ (i.e., $g_e^* \in \partial\mathcal{G}^e$) holds for any episode e.*

Proof. We first introduce the following lemma to show that the K-bins discretization can preserve the optimal goals if our discretization is under adjacent region constraint.

Lemma 1. (Goal Optimality Preserved with Adjacent Region Constraint) *(Zhang et al., 2020) Let $s \in S$, $g \in G$, let π^* be an optimal goal-conditioned policy. Under the assumptions that the MDP is deterministic and the MDP states are strongly connected, for all $k \in \mathbb{N}_+$ satifying $k \leq \texttt{distance}(s, \phi^{-1}(g))$ where ϕ is a mapping function from S to G, then the optimal goal space are preserved by a k-step adjacent region.*

The key intuition is that distant goals can be substituted by closer goals, as long as they drive the agent to move towards the same "direction". Since our K-bins discretization also can be regarded as a space reduction technique constructing adjacent regions, thus we can conclude that if satisfying some distance constrictions, K-bins discretization also preserve the optimal goal space.

We then show that the optimal goals are always in the graph boundary. According to Definition 1, in the whole state-transition graph $\mathcal{G}_{\texttt{whole}}$, the optimal goal g_{whole}^* is the target state. The intuitive explanation behind this is very natural, where the environment in this case is fully explored, and thus the agent is ready to aim at the target state.

In the other cases, we generate the optimal goal g_e^* of episode e at the episode $e+1$. Specially, we find the shortest path to the highest value state in $\mathcal{G}^{e+1}$ as the optimal solution path $\mathcal{P}_{e+1}$. For instance, suppose that in the episode $e+1$, the state with the highest value in $\mathcal{G}^{e+1}$ is s_9 and the optimal solution path in this case is $\mathcal{P}_{e+1} = \langle s_{\texttt{start}}, s_1, s_3, s_6, s_9 \rangle$. We then compare the explored states in $\mathcal{G}^e$ with the states in $\mathcal{P}_{e+1}^{\texttt{inverse}}$, where $\mathcal{P}_{e+1}^{\texttt{inverse}} = \langle s_9, s_6, s_3, s_1, s_{\texttt{start}} \rangle$ is the inverse order of $\mathcal{P}_{e+1}$. Assume that $\mathcal{G}^e$ involving $s_{\texttt{start}}, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8$, we can find the first overlapping state is s_6 and we assign s_6 as the optimal goal g_e^*. As stated above, it's easy to find that there are two cases in the optimal goal generation. One is the last node of solution path $\mathcal{P}_{e+1}$. The other is one of the rest nodes in $\mathcal{P}_{e+1}$ except the last one. We then prove that in both of these cases, optimal goal g_e^* is always included in the boundary of the state-transition graph (i.e., $\partial\mathcal{G}^e$).

Case I: Node at Last. Assume that if there exists a path connecting a state s and a goal g whose length is shorter than any other paths connecting another state s' and the goal g, then $V^\pi(s, g) > V^\pi(s', g)$ holds for $\forall s, s' \in S, g \in G$. We assign g as g_{e+1} here. Then, if g_e^* is not in the boundary, there must be one neighbor node closer to the goal g_{e+1}. Otherwise, g_e^* is the dead end and thus should not be regarded as the optimal goal. And if there is one neighbor node closer to g_{e+1}, then this neighbor node should be regarded as the optimal goal. Therefore, we obtain a contradiction.

Case II: Node Not at Last. If the optimal goal is not the last state, then there must exist the state unexplored at episode e. In the previous example, if we take s_6 as the optimal goal g_e^*, state s_9 must be unexplored in $\mathcal{G}^e$ and explored in $\mathcal{G}^{e+1}$. If g_e^* is not included in $\partial\mathcal{G}^e$, then there should not exist any unexplored state that is included in its neighborhood. According to the definition of the boundary of the graph, we have proved the proposition by contradiction.

In summary, we have proved the proposition in both two cases by contradiction. □

A3.3 Proof of Proposition 2

Proposition 2. (Convergence of G2RL with Q-learning) *Denote the Bellman backup operator in* $Q^\pi(s,a,g) \leftarrow r_g(s,a)+\gamma\cdot\max_{a'} Q^\pi(s',a',g)$ *as* $\mathcal{B}:\mathbb{R}^{|S|\times|A|\times|G|}\rightarrow\mathbb{R}^{|S|\times|A|\times|G|}$ *and a mapping* $Q: S\times A\times G\rightarrow\mathbb{R}$ *with* $|S|<\infty$ *and* $|A|<\infty$. *Sufficient repeated applications of the operator* $\mathcal{B}$ *for our graph-based goal-conditioned state-action value estimate* $\widehat{Q}_\mathcal{G}$ *converges to a unique optimal value* $Q^*_{\mathcal{G}^*}$ *with the graph* $\mathcal{G}^*$ *including the optimal solution* $\mathcal{P}_{\texttt{whole}}$.

Proof. The proof is developed in two main steps. The first step is to show that our state-transition graph $\mathcal{G}$ can converge to the whole state-transition graph $\mathcal{G}_{\texttt{whole}}$. Here, we define $\mathcal{G}^*$ as the graph that includes the optimal solution $\mathcal{P}_{\texttt{whole}}$ defined based on the whole state-transition graph $\mathcal{G}_{\texttt{whole}}$. In the second step, we prove that given graph $\mathcal{G}$, our graph-based method can converge to unique optimal value $Q^*_\mathcal{G}$.

Step I. Since $|S|<\infty$ and $|A|<\infty$, we can obtain that $|\mathcal{V}_\mathcal{G}|<\infty$ and $|\mathcal{E}_\mathcal{G}|<\infty$. Note that the state-transition graph $\mathcal{G}$ is a dynamic graph, and goals g generated on $\mathcal{G}$ are updated at the beginning timestep of each episode. Hence, there is a sequence of goals denoted as $(g_1, g_2, \cdots, g_E)$ and corresponding sequence of graphs denoted as $(\mathcal{G}^1, \mathcal{G}^2, \cdots, \mathcal{G}^E)$, where E here is the number of episodes. Given that $|S|<\infty$ and $|A|<\infty$, the number of nodes and edges in the whole state-transition graph $\mathcal{G}_{\texttt{whole}}$ is also bounded. Based on the explore strategy introduced in Section 3, we know that GoRL will first search for a path leading to the target state. After that, the target state will be included in $\mathcal{G}$. Then the agent will seek the shortest path to the terminal state because the agent is given a negative reward at each timestep. Hence, the optimal solution path $\mathcal{P}_{\texttt{whole}}$ will be involved. Hence, we can obtain that

$$\mathcal{G}^1\subseteq\mathcal{G}^2\subseteq\cdots\subseteq\mathcal{G}^*\Rightarrow\mathcal{G}\rightarrow\mathcal{G}^*. \tag{16}$$

Assume that E is large enough, our state-transition graph $\mathcal{G}$ can finally converge to the graph $\mathcal{G}^*$.

Step II. Note that the proof of convergence for our graph-based GoRL is quite similar to Q-learning (Bellman, 1966; Bertsekas et al., 1995; Sutton & Barto, 2018). The differences between our approach and Q-learning are that Q value $Q(s,a,g)$ is also conditioned on goal g, and that the state-transition probability $P_\mathcal{G}(s'|s,a)$ can be reflected by graph $\mathcal{G}$. We provide detailed proof as follows:

For any state-transition graph $\mathcal{G}$, we can obtain goal $g\in G$ conditioned on $\mathcal{G}$ from Step I. Based on that, our estimated graph-based action-value function $\widehat{Q}_\mathcal{G}$ can be defined as

$$\mathcal{B}\widehat{Q}_\mathcal{G}(s,a,g)=r_g(s,a)+\gamma\cdot\max_{a'\in A}\sum_{s'\in S}P_\mathcal{G}(s'|s,a)\cdot\widehat{Q}_\mathcal{G}(s',a',g). \tag{17}$$

For any action-value function estimates $\widehat{Q}^1_\mathcal{G}, \widehat{Q}^2_\mathcal{G}$, we study that

$$\begin{aligned}
&|\mathcal{B}\widehat{Q}^1_\mathcal{G}(s,a,g)-\mathcal{B}\widehat{Q}^2_\mathcal{G}(s,a,g)|\\
&=\gamma\cdot|\max_{a'\in A}\sum_{s'\in S}P_\mathcal{G}(s'|s,a)\cdot\widehat{Q}^1_\mathcal{G}(s',a',g)-\max_{a'\in A}\sum_{s'\in S}P_\mathcal{G}(s'|s,a)\cdot\widehat{Q}^2_\mathcal{G}(s',a',g)|\\
&\leq\gamma\cdot\max_{a'\in A}|\sum_{s'\in S}P_\mathcal{G}(s'|s,a)\cdot\widehat{Q}^1_\mathcal{G}(s',a',g)-\sum_{s'\in S}P_\mathcal{G}(s'|s,a)\cdot\widehat{Q}^2_\mathcal{G}(s',a',g)|\\
&=\gamma\cdot\max_{a'\in A}\sum_{s'\in S}P_\mathcal{G}(s'|s,a)\cdot|\widehat{Q}^1_\mathcal{G}(s',a',g)-\widehat{Q}^2_\mathcal{G}(s',a',g)|\\
&\leq\gamma\cdot\max_{s\in S,a\in A}|\widehat{Q}^1_\mathcal{G}(s,a,g)-\widehat{Q}^2_\mathcal{G}(s,a,g)|
\end{aligned} \tag{18}$$

So the contraction property of Bellman operator holds that

$$\max_{s\in S,a\in A}|\mathcal{B}\widehat{Q}^1_\mathcal{G}(s,a,g)-\mathcal{B}\widehat{Q}^2_\mathcal{G}(s,a,g)|\leq\gamma\cdot\max_{s\in S,a\in A}|\widehat{Q}^1_\mathcal{G}(s,a,g)-\widehat{Q}^2_\mathcal{G}(s,a,g)| \tag{19}$$

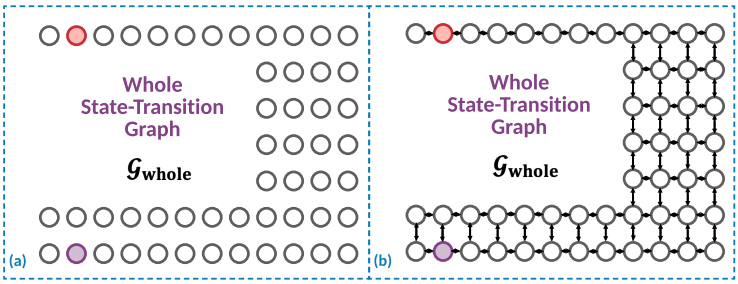


Figure A4: An illustrated example for connection between certainty and number of visited states.

For the fixed point $Q^*_{\mathcal{G}}$, we have that

$$\max_{s\in S,a\in A} |\mathcal{B}\widehat{Q}_{\mathcal{G}}(s,a,g) - \mathcal{B}\widehat{Q}^*_{\mathcal{G}}(s,a,g)| \leq \gamma \cdot \max_{s\in S,a\in A} |\widehat{Q}_{\mathcal{G}}(s,a,g) - \widehat{Q}^*_{\mathcal{G}}(s,a,g)| \Rightarrow \widehat{Q}_{\mathcal{G}} \to Q^*_{\mathcal{G}}. \tag{20}$$

Combining Step I and II, we can conclude that our graph-based estimated state-action value $\widehat{Q}_{\mathcal{G}}$ can converge to a unique optimal value $Q^*_{\mathcal{G}^*}$. □

A4 Discussions

A4.1 Discussion on Graph Construction

We do not directly adopt the previous graph construction methods due to following reasons. Firstly, as G2RL generates the goal supervisions (i.e., optimal goals in Definition 1) based on the shortest path searching algorithm on the (global) state-transition graph, existing approaches (Zhu et al., 2019; Huang et al., 2019; Klissarov & Precup, 2020) organizing the graph based on sampled trajectories are not suitable in our case. Secondly, when considering large-scale discrete environments and continuous environments without image inputs (i.e., Ant Maze, (Low and High) Ant Maze, FetchPush, FetchPush with Obstacle shown in Figure A6(b)(c)(d)), directly using the whole state-transition graphs is not a possible solution as performing training and inference would be too costly and impractical (Klissarov & Precup, 2020). Prior works (Savinov et al., 2018; Eysenbach et al., 2019) proposing a parameterized distance network are not suitable either, as it would introduce a batch of parameters to train and might severely affect the final performance. A feasible solution is to consider discretizing the state and action spaces, as the evidence of that animals explore their environments with the help of grid cells activated on particular intervals (O'Keefe & Dostrovsky, 1971; Banino et al., 2018; Colombo et al., 1998). We select the K-bins discretization technique here, as it can be conveniently applied by scikit-learn package (Pedregosa et al., 2011).

For the environments with image inputs (e.g., VizDoom, Matterport3D in Figure A6(e)(f)), unfortunately, we can not directly employ the K-bins discretization technique to build the graph. Hence, we follow previous literature (Chaplot et al., 2020a; Savinov et al., 2018) to first pre-train a ResNet-18 (He et al., 2016) and Mask-RCNN (He et al., 2017) to obtain representations from RGB images for VizDoom and Matterport3D respectively, which can further be used to generate a graph given a pre-defined similarity threshold. Note that the generated graph is normally large-scale for G2RL, as our proposed algorithm runs the shortest path searching to generate the goal supervisions (See Section 2.3) and the group division to produce candidate groups (See Section 2.4). Therefore, we further apply K-bins discretization technique to reduce the scale of the graph.

A4.2 Discussion on Certainty of State

In this section, we further discuss the relationship between the certainty of state and the number of states. In the previous exploration RL literature (Ostrovski et al., 2017; Bellemare et al., 2016), the performance of exploration often is measured by the number of the visited states. Namely, given a fixed number of episodes, more visited states, better performance. In this paper, we propose to utilize a new measurement, i.e., certainty of state in Section 2.3. We conclude the relations between certainty and the number of visited states as Proposition 3.

Proposition 3. (Connection between Certainty of Node ⟨Local Measurement⟩ and Number of Visited Node ⟨Global Measurement⟩) *Given the whole state-transition graph $\mathcal{G}_{\texttt{whole}}$, we can regard the certainty of states as the local measurement and the number of states as the global measurement for exploration, which share a similar trend during the exploration.*

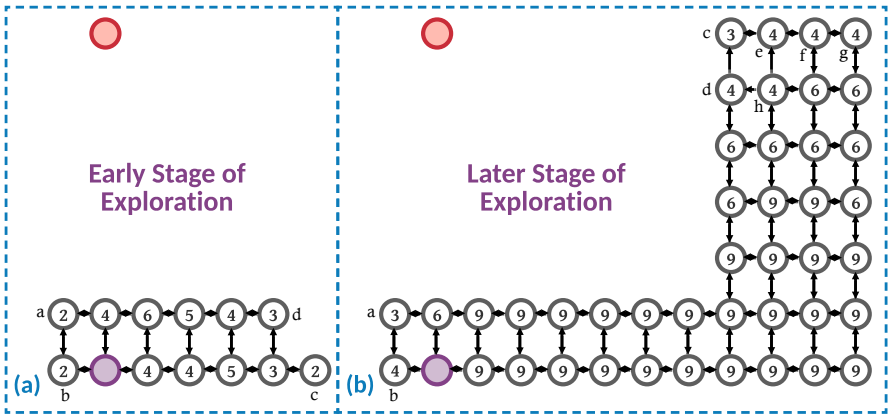


Figure A5: An illustrated example for comparison of using the certainty of states and using the visitation counts of states to define the graph boundary. The number at the center of each state denote its visitation number.

Proof. We illustrate and prove the proposition hindsightly. If we have the full observation for states as shown in Figure A4(a), we can model the agent finding new states as connecting new states with visited states. In other words, since the state-transition graph $\mathcal{G}_t^e$ must keep being a fully connected graph at any timestep t in any episode e. Hence, adding new states into the visited state set can always be regarded as finding new edge between new states and the visited state set. And each directed edge in the state-transition graph, as shown in Figure A4(b) is determined by action and state-transition function. If the environment is determined, we can roughly regard the number of edges as the approximate measurement for exploration. The certainty of states is the local perspective for this measurement. □

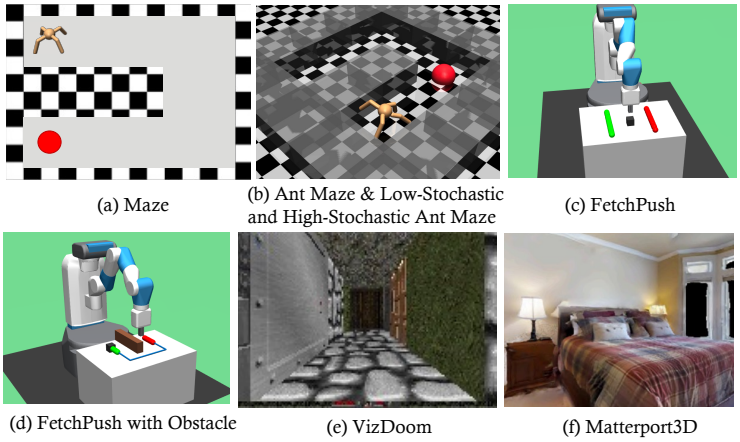
A4.3 Discussion on Graph Boundary

As described in Section 2.3, the boundary of $\mathcal{G}_t^e$ (i.e., $\partial\mathcal{G}_t^e$) is defined according to the certainty of states, namely the number of taken actions of states. Another intuitive way is to use the visitation number of states to define the graph boundary. Specifically, we can construct the graph boundary by including infrequently visited states. Take Figure A5 as an example, where (a) illustrates the case at early stage of exploration and (b) shows the case at later stage of exploration. The number at the center of each state denotes its visitation number. From (a), we can see that if the graph boundary is built based on the visitation numbers of states, then the states s_a, s_b, s_c will be treated equally. As the states s_a, s_b are located on the left side of the initial state, if these states are further assigned as the goals, the agent will be leaded to the wrong direction. Similar scenario occurs in (b), where the states s_a and s_c are treated equally, and the states s_b and s_d, s_e, s_f, s_g, s_h are treated equally, when building the graph boundary according to the states' visitation numbers. In contrast, if defining the graph boundary by the number of the taken actions of each state, then only the states s_c and s_d will construct the the graph boundary in the case (a) and only the state s_c will construct the graph boundary in the case (b).

We argue that these two cases in Figure A5, or similar cases, are common for GoRL: if the goals are generated towards the target state (namely on the right side of the initial state), then the visitation counts of the states on the left side of the initial state are likely to be small; however, these states are in the wrong direction for exploration.

We further provide the reasons for the differences of these two definitions of graph boundary as follows. The aim of the exploration of RL is to discover new states. Note that new states are always the results of taking new actions of visited states. Hence, the number of taken actions of each state directly shows the potential of the state to explore new states. We also provide the theoretical support that the goal optimality is preserved in the graph boundary defined based on the certainty of states in Theorem 1. In contrast, the idea of using the visitation count of states to define the graph boundary is intuitive but not as straightforward as using the certainty of states. We show the empirical comparison of these two definition of graph boundary in Figure 4(d).

There are existing approaches that introduce the similar concepts and use them for exploration. For example, classical method (Yamauchi, 1997) proposes to first detect the frontier by a frontier detection approach (Keidar & Kaminka, 2012). The frontier is defined as the boundary between unexplored and explored space; however, is measured by whether the states have been visited. In-



(a) Maze (b) Ant Maze & Low-Stochastic and High-Stochastic Ant Maze (c) FetchPush

(d) FetchPush with Obstacle (e) VizDoom (f) Matterport3D

Figure A6: Environments used in our experiments. (a) Maze, where the ant agent learns to reach the specified target position (ϵ-ball depicted in red) located at the other end of the U-turn. (b) Ant Maze constructed based on Maze but high-dimensional, where the state space is of 8 dimensions and the action space is of 30 dimensions, and Low-Stochastic and High-Stochastic Ant Maze built based on Ant Maze where the connectivity between each pair of neighbor cells will change at the probability of 5% and 20% in each episode respectively. (c) FetchPush, where the agent is trained to fetch an object from the initial position (rectangle in green) to a distant position (rectangle in red). (d) FetchPush with Obstacle constructed based on FetchPush but harder, where the brown block is a static wall that cannot be moved. (e) VizDoom built based on maze but image data as input, where the agent is required to navigate towards the goal in a maze environment with a image as the state. (f) Matterport3D built based on maze but scenes, which are 3D reconstructions of real-world environments, as input, where the agent is required to reach the target scenes. Some of these illustrations are adopted from the previous literature (Florensa et al., 2017; Savinov et al., 2018; Ren et al., 2019; Chaplot et al., 2020a).

stead, our graph boundary is measured by the exploration status of states (i.e., the number of untaken actions). Recent attempt (Huang et al., 2019) introduce landmark defined as farthest points, and can be sampled by a farthest point sampling algorithm (Arthur & Vassilvitskii, 2006), which is definitively regardless of the exploration status of states either. Detailed discussions about the difference between our certainty and the visitation of the states have been provided in the above parts in this subsection.

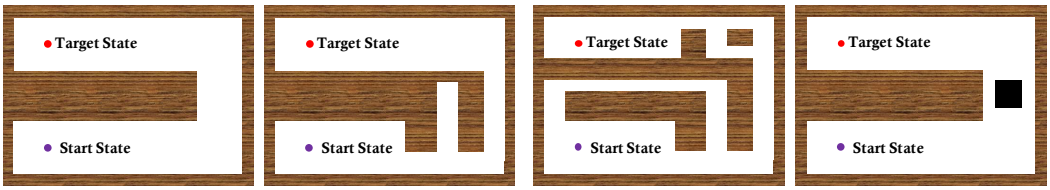
We further discuss whether there will be too many states/nodes in the graph boundary as follows. As the certainty of states are measured by its untaken actions, two different cases can happen: (i) The action space is small (e.g., forward, left, right in navigation task), and then it's unlikely to have too many states being uncertain. (ii) The action space is large (e.g., continuous action space), and then we can apply K-bins discretization on the action space to reduce it to a small action space. Hence, it's unlikely to have too many uncertain nodes in the graph boundary. We also argue that the impact the number of uncertain nodes is limited, and show the reasons as follows. (i) For the differentiable goal generator, our group division will divide these states into a fixed number of groups, and the supervision signals of the goal generator are on groups instead of states. Hence, the effect of too uncertain states is limited. (ii) For the relevance sampling technique, there is no explicit effect as the relevant states are determined by the neighborhood structure, regardless of whether the nodes are uncertain or not.

A4.4 Discussion on Optimal Goal

In the previous GoRL literature (Andrychowicz et al., 2017; Ren et al., 2019), what kind of generated goals is helpful for the agent to efficiently learn a well-performed policy is one of the key questions to be answered. The basic idea of GoRL architecture is to generate goals to decompose the complex task into several goal-oriented tasks. In this paper, we analyze our generated goals from two perspectives, namely reachability and curriculum.

Reachability. The first property required in the optimal goal is that the generated goal is guaranteed to be reachable for the agent. To this end, in this paper, the candidate goal set is constrained into the visited states. In other words, the goal generated in the episode e must be visited before the episode e. Therefore, we can guarantee that the generated goal of any episode is reachable.

Curriculum. The second property is the curriculum, which means that our optimal goals are required to approach the target state during the exploration. Our goal generation under the supervision



(a) TRAIN Ant Maze (b) TEST1 Ant Maze (c) TEST2 Ant Maze (d) Ant Maze with Obstacle

Figure A7: Visualization of three different mazes designed for Ant Maze environment including *TRAIN* (a), *TEST1*(b), *TEST2*(c). We train G2RL in *TRAIN* and test it in *TEST1* and *TEST2* to evaluate the generalizability. Beside, we also design *Ant Maze with Obstacle* to evaluate the performance of G2RL when encountering more complex environment.

of forward-looking planning in the next episode will focus on the potential highest value states in the future, which is actually the target state when the agent has the full observation of states.

A4.5 Discussion on Group Division

Motivation. The intuitive motivation behind the group division is very natural. Proposition 1 implies that exploration on the state-transition graph $\mathcal{G}_t^e$ at timestep t in episode e without any constraint may lead to explosion of graph and inefficiency of exploration. Therefore, the agent is expected to do exploration within a limited domain. Considering that $\mathcal{G}^e$ is always changing and the number of nodes, i.e., $|S_{\mathcal{G}^e}|$) keeps increasing, it is non-trivial for the agent to learn to select state as the goal for further exploration. Hence, we first restrict the exploration within the boundary of state-transition graph $\partial\mathcal{G}^e$ according to Proposition 1. We then consider partitioning $\partial\mathcal{G}^e$ into several groups.

We set the last visited state $s_{\texttt{last}}$ as the original point because $s_{\texttt{last}}$ is likely to be close to the target state and reachable for current policy. As introduced in Section 3, we propose to construct groups from $s_{\texttt{last}}$ by adding two classes of nodes, namely neighbor nodes of $s_{\texttt{last}}$ and uncertain nodes in the graph.

Complexity. Let $d_{\partial\mathcal{G}^e}$ denote the maximum degree of states in $\partial\mathcal{G}^e$, and $|S_{\partial\mathcal{G}^e}|$ denote the number of states in $\partial\mathcal{G}^e$. Note that $\partial\mathcal{G}^e$ is always a directed fully connected graph. If we want to find the n-hop neighbors of $s_{\texttt{last}}$, we need to iteratively go through related nodes' neighborhood. In other words, the computation complexity should be $O(d^n_{\partial\mathcal{G}^e})$. Hence, the complexity to construct $\mathcal{C}_1, \ldots, \mathcal{C}_N$ by extending from neighbor nodes is $O(d^1_{\partial\mathcal{G}^e}) + O(d^2_{\partial\mathcal{G}^e}) + \cdots + O(d^{N-1}_{\partial\mathcal{G}^e}) = O(d^{N-1}_{\partial\mathcal{G}^e})$. If we want to find nodes whose uncertainty equals n, we need to go through the graph once. In this case, the computation complexity should be $O(|S_{\partial\mathcal{G}^e}|)$. Hence, the complexity to construct $\mathcal{C}_1, \ldots, \mathcal{C}_N$ extending from uncertain nodes is $O(|S_{\partial\mathcal{G}^e}|)$.

Example. Take Figure A5(b) as an example, where the graph boundary consists of v_c, v_e. Assume that the last visited state/node is v_c, then the first way is to add the neighbor nodes of v_c within the graph boundary (e.g., v_e is the first-hop neighbor node) into the group, while the second way is to add the uncertain node (e.g., v_c is uncertain node with two action untaken) into the group.

A5 Experiments

A5.1 Environment Configuration

Maze. As shown in Figure A6(a), in the maze environment, a point in a $2D$ U-maze aims to reach a goal represented by a red point. The size of maze is 15×15, the state space and is in this $2D$ U-maze, and the goal is uniformly generated on the segment from $(0, 0)$ to $(15.0, 15.0)$. The action space is from $(-1.0, -1.0)$ to $(1.0, 1.0)$, which represents the movement in x and y directions.

AntMaze. As shown in Figure A6(b), in the AntMaze environment, an ant is put in a U-maze, and the size of the maze is 12×12. The ant is put on a random location on the segment from $(-2.0, -2.0)$ to $(10.0, 10.0)$, and the goal is uniformly generated on the segment from $(-2.0, -2.0)$ to $(10.0, 10.0)$. The state of ant is 30-dimension, including its positions and velocities. For AntMaze environment, we follow (Savinov et al., 2018) to develop three different maze designs, namely *TRAIN*, *TEST1* and *TEST2*, which are illustrated in Figure A7(a)-(c). We use *TRAIN* to train G2RL and evaluate the performance in *TEST1* and *TEST2* to test the generalizability of the model. One can easily find that *TEST1* is much more similar to *TRAIN* than *TEST2*.

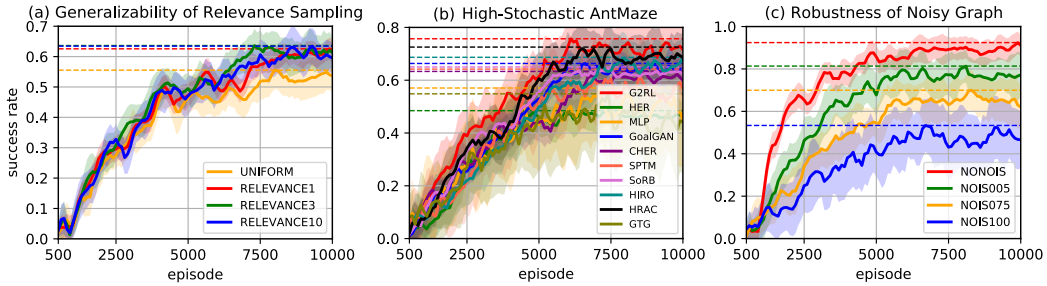


Figure A8: Comparison of learning curves of our models G2RL in different settings mainly on Ant Maze environment average across 10 different random seeds. The solid curves depict the mean, the shaded areas indicate the standard deviation, and dashed horizontal lines show the asymptotic performance.

Low-Stochastic and High-Stochastic Ant Mazes. These two environments is stochastic versions of Ant Maze, where the connectivity between each pair of neighbor cells will change at the probability of 5% for Low-Stochastic Ant Maze and 10% for High-Stochastic Ant Maze in each episode.

FetchPush. As shown in Figure A6(c), in the fetch environment, the agent is trained to fetch an object from the initial position (rectangle depicted in green) to a distant position (rectangle depicted in red). Let the origin $(0, 0, 0)$ denote the projection of the gripper's initial coordinate on the table. The object is uniformly generated on the segment from $(-0.0, -0.0, 0)$ to $(8, 8, 0)$, and the goal is uniformly generated on the segment from $(-0.0, -0.0, 0)$ to $(8, 8, 0)$.

FetchPush with Obstacle. As shown in Figure A6(d), in the fetch with obstacle environment, we create an environment based on FetchPush with a rigid obstacle, where the brown block is a static wall that can't be moved. The object is uniformly generated on the segment from $(-0.0, -0.0, 0)$ to $(8, 8, 0)$, and the goal is uniformly generated on the segment from $(-0.0, -0.0, 0)$ to $(8, 8, 0)$.

AntMaze with Obstacle. This environment is an extended version of AntMaze, where a 1×1 rigid obstacle is put in U-maze, which is shown in Figure A7(d).

VizDoom. As shown in Figure A6(e), this environment introduced in (Kempka et al., 2016) is based on the classical first-person shooter video game. It allows developing bots that play the game using the screen buffer. In contrast to the other popular visual learning environments such as Atari 2600, ViZDoom provides a 3D, semi-realistic, first-person perspective virtual world. In this paper, we follow (Savinov et al., 2018) and use this environment for navigation task, where we adopt ResNet-18 (He et al., 2016) to handle the image-based observations. Concretely, stacks of two consecutive RGB images obtained from the environment, at resolution 160×120 pixels are fed into ResNet-18 for state representation learning.

Matterport3D. As shown in Figure A6(f), this environment proposed in (Chang et al., 2017) is based on Habitat simulator (Savva et al., 2019). It consists of scenes which are 3D reconstructions of real-world environments. We use the standard train and test splits. The observation space consists of RGB images of size $4 \times 640 \times 480$, base sensor of size 3×1 denoting the change in agent's x-y coordinates and orientation, and goal object category represented as an integer. The action space consists of four actions: moving forward, turning left, turning right and stopping. We follow (Chaplot et al., 2020a) to pre-train a Mask-RCNN (He et al., 2017) using Feature Pyramid Networks (Lin et al., 2017) with ResNet-50 (He et al., 2016) backbone on MS-COCO dataset (Lin et al., 2014) for object detection and instance segmentation.

Note that all the baseline methods use the same (positional or image) inputs as G2RL.

Different from the stochastic versions of Ant Maze introduced above, a batch of more challenging environments are noisy environments. Note that these stochastic versions operating on the connectivity of neighbor nodes, might accelerate the exploration by creating a short cut from the start point to the target point. Instead, for the noisy version of Ant Maze (called as Noisy Ant Maze), we directly generate noises from Gaussian distributions $\mathcal{N}(\mu, \sigma^2)$ where we set $\mu = 0$, and $\sigma = 0.5, 0.75, 1$ (denoted as *NOIS050*, *NOIS075*, *NOIS100*). For each episode, we generate a Gaussian noise matrix (denoted as $\widetilde{G}$) with the same shape to the adjacency matrix of the current state-transition graph (denoted as $\widetilde{A}$), and update $\widetilde{A}$ by summing up these two matrix (i.e., $\widetilde{A} \leftarrow \widetilde{A} + \widetilde{G}$). Then, each entity of $\widetilde{A}$ will be set as 1 if its value is larger than 0.5, or 0, otherwise. We can see that this approach can both connect and disconnect each pair of nodes.

Table A1: Result comparisons with baselines on unseen environments.

Environments	TEST1 AntMaze		TEST2 AntMaze		(TEST) Matterport3D	
	Success	*SPL*	*Success*	*SPL*	*Success*	*SPL*
HER	0.482	0.296	0.365	0.182	0.262	0.078
MLP	0.715	0.505	0.473	0.236	0.311	0.124
GoalGAN	0.695	0.426	0.412	0.206	0.284	0.100
CHER	0.587	0.360	0.395	0.198	0.281	0.101
SPTM	0.773	0.610	0.496	0.262	0.306	0.122
SoRB	0.802	0.636	0.535	0.278	0.321	0.128
HIRO	0.700	0.432	0.511	0.255	0.287	0.097
HRAC	0.721	0.443	0.533	0.272	0.298	0.104
GTG	0.686	0.480	0.378	0.190	0.278	0.097
G2RL	**0.852**	**0.677**	**0.578**	**0.318**	**0.387**	**0.164**

A5.2 Evaluation Details

- All curves are plotted from 10 runs with random task initialization and seeds.
- The shaded region indicates 60% population around the median.
- All curves are plotted using the same hyper-parameters (except the ablation section).
- Following (Andrychowicz et al., 2017), an episode is considered successful if $\|g - s_{\texttt{object}}\|_2 \leq \delta_g$ is achieved, where $s_{\texttt{object}}$ is the object position at the end of the episode. δ_g is the threshold.
- The max timestep for each episode is set as 200 for training and 500 for tests.
- The average success rate using in the curve is estimated by 10^2 samples.

A5.3 Baseline Methods

- *HER* (Andrychowicz et al., 2017) generates imaginary goals in a simple heuristic way to tackle sparse reward issue.
- *MLP* (Huang et al., 2019) explicitly models the environment in a hierarchical manner, with a high-level map abstracting the state space and a low-level value network to derive local decisions.
- *GoalGAN* (Florensa et al., 2018) leverages Least-Squares GAN (Mao et al., 2018) to mimic the set of GOID as an automatic goal generator.
- *CHER* (Fang et al., 2019) proposes to enforce more curiosity in earlier stages and changes to larger goal-proximity later.
- *SPTM* (Savinov et al., 2018) consists of a graph corresponding to environment and deep network capable of retrieving nodes from the graph.
- *HIRO* (Nachum et al., 2018b) introduces an off-policy correction to learn higher- and lower-level policies of Hierarchical RL.
- *HRAC* (Zhang et al., 2020) follows the hierarchical RL framework but generates the goals with an adjacent region constraint.
- *GTG* (Jiang et al., 2021) constructs the grid graph based on entities of x-y coordinates and use R-GCN (Schlichtkrull et al., 2018) to encode the graph structure. As the original work is proposed for the grid environments, we follow the same graph construction procedure of G2RL, namely using the K-bins discretization (Kotsiantis & Kanellopoulos, 2006) to extend GTG to continuous environments, and following (Savinov et al., 2018; Chaplot et al., 2020a) and using ResNet-18 (He et al., 2016) and Mask-RCNN (He et al., 2017) to handle RGB images obtained from the environments.

A5.4 Implementation Details

Almost all hyper-parameters using DDPG (Lillicrap et al., 2015) and HER (Andrychowicz et al., 2017) are kept the same as benchmark results, except these:

- Number of MPI workers: 1;
- Actor and critic networks: 3 layers with 256 units and ReLU activation;
- Adam optimizer with 5×10^{-4} learning rate;
- Polyak-averaging coefficient: 0.98;
- Action l_2-norm penalty coefficient: 0.5;
- Batch size: 256;
- Probability of random actions: 0.2;
- Scale of additive Gaussian noise: 0.2;
- Probability of HER experience replay: 0.8;

- Number of batches to replay after collecting one trajectory: 50.

Hyper-parameters in goal generation:

- Adam optimizer with 1×10^{-3} learning rate;
- K of K-bins discretization: 20;
- Number of groups to depart the graph: 3.

A5.5 Hardware Settings

The models are trained under the same hardware settings with an Amazon EC2 p3.8×large instance[‡], where the GPU processor is NVIDIA Tesla V100 processor and the CPU processor is Intel Xeon E5-2686 v4 (Broadwell) processor.

A6 Additional Results

A6.1 Comprehensive Transductive Experiments

From Figure 4(a), we know that our relevance sampling technique is beneficial to G2RL. We further investigate whether this technique can also improve the performance of other G2RL. Without loss of generality, we incorporate our relevance sampling technique with HER (Andrychowicz et al., 2017), a widely adopted GoRL method, and evaluate the performance on Ant Maze. We report the results in Figure A8(a), which indicates that the relevance sampling technique can seamlessly incorporate with other GoRL. Note that Figure 3 depicts the results of G2RL with strong baselines on the environments introduced in Appendix A5.1 except High-Stochastic Ant Maze and Matterport3D environments. Figure A8(b) reports the results on High-Stochastic Ant Maze. Comparing to Figure 3(b)(c) to Figure A8(b), we can see that GoRL methods leveraging the state-transition graph (e.g, G2RL, SoRB, MLP) are only robust to a certain level of stochasticity, which is consistent to our analysis of the limitation of G2RL in Appendix A2.1.

However, as discussed in Appendix A5.1, the above stochastic versions of Ant Maze might connect the pairs of nodes that are originally cut off by obstacles (e.g., wall), and thus acceletate the exploration. Therefore, we introduce Noisy Ant Mazes described in Appendix A5.1 to further investigate the robustness of G2RL. Results depicted in Figure A8(c) shows that G2RL based on the state-transition graph, which requires an accurate graph to achieve a good performance.

Notably, from the empirical results in Section 4 and this subsection, we can conclude the performance gain of G2RL mainly from our differentiable goal generation algorithm and slightly from our relevance sampling algorithm.

A6.2 Comprehensive Inductive Experiments

In Figure 3, we show the results of performance of using the same environments for both training and testing. One of the most challenging tasks of RL is to train and evaluate in two different environments. To this end, we investigate the generaliability to unseen environments of G2RL against strong baseline methods. Concretely, we draw two new Ant Mazes to test the performance of G2RL trained in the original Ant Maze (i.e., *TRAIN*). We introduce details of these environments in Appendix A5.1. Moreover, as we introduce in Appendix A5.1, Matterport3D dataset provides the standard train and test splits based on the Habitat simulator.

Recall that the main components of G2RL are (i) differentiable goal generator for effective goal generation and (ii) relevance sampling technique for policy learning. As no learning is allowed when generalizing to unseen environments, the generalizability of G2RL mainly depends on the goal generator, which is discussed as follows: (i) Our definition of the graph boundary providing the candidate states for the goal generation can seamlessly generalize to unseen environment, as it can re-compute the certainty of each state and construct the graph boundary by uncertain states. (ii) The group division based on the graph boundary can also seamlessly generalize to unseen environment, since the number of groups (i.e., N) is a hyper-parameter. (iii) The group selection over N groups can directly generalize to unseen environments by freezing the parameters (i.e., ϕ) in the attention

[‡]Detailed setting of AWS E2 instance can be found at https://aws.amazon.com/ec2/instance-types/?nc1=h_ls.

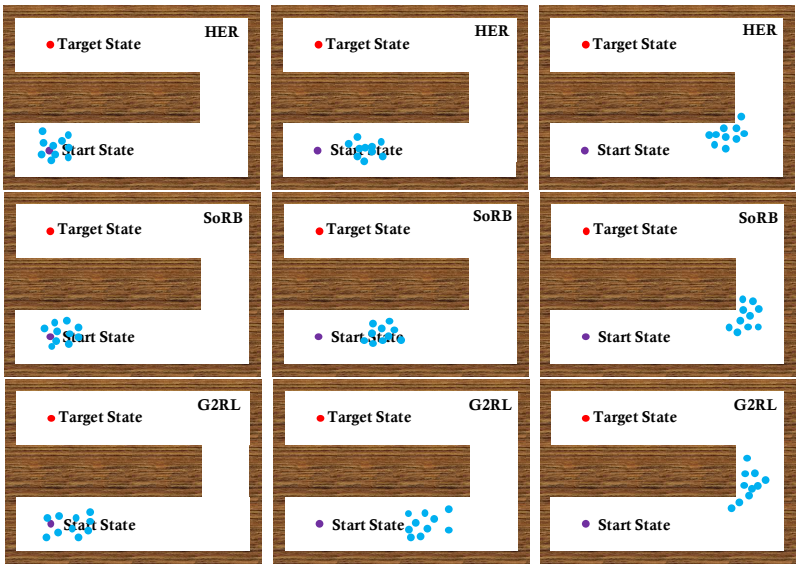


Figure A9: Visualization of generated goals by G2RL, SoRB and HER in AntMaze environment. The start and the target states are illustrated as purple and red circles respectively. The blue circles indicate ten recently generated goals. The subfigures in the top show the results of HER and the subfigures in the middle show the results of SoRB, while the subfigures in the bottom show the results of G2RL.

model (i.e., $\texttt{ATT}_\phi$). In our implementation, like related literature (Chaplot et al., 2020b;a; Savinov et al., 2018), we freeze the parameters of the goal generator (i.e., ϕ) and the policy (i.e., θ), and apply the trained goal generator and the trained policy $\pi : S \times G \to \Delta(A)$ (See Section 2.1 for background of GoRL) in the new environments.

We use two metrics for comparing all the methods, namely *Success*: success rate, the ratio of episodes where the method is successful; *SPL*: successes weighted by the path length proposed by (Anderson et al., 2018), which not only considers whether the method is successful but only consider the length of trajectories. Results reported in Table A1 indicate that G2RL can better generalize to the environments with similar structures than other strong methods. Also, we can notice that GTG (Jiang et al., 2021) using graph neural networks can not well generalize to unseen environments. Table A1 also reports the results of Matterport3D environment (on the test datasets), which shows the consistent superiority of G2RL in the complex navigation environment.

A6.3 Visualization of Goal Distributions

To investigate whether the generated goals truly guide the agent to the target state, we demonstrate the distribution of generated goals of G2RL, SoRB, and HER. As shown in Figure A9, G2RL generates goals that gradually move towards the target state. Since these goals are generated within the visited states, and thus they are considered to be achieved during training. In comparison, the goal distribution of HER has been stuck around the start state for many iterations, indicating the less effective exploration. And, the goal distribution of SoRB seems to locate at the middle between the distributions of HER and G2RL. One possible explanation for this observation is that G2RL and SoRB leverage the state-transition graph to guide the agent to explore, and G2RL is further benefited from the differentiable goal generation based on the graph boundary.

A6.4 Comparison on Sample Efficiency

By sample efficiency, we show the comparisons according to the number of states visited and actions taken. In other words, given the fix number of episodes, more unique states visited and actions taken usually denote the efficiency of exploration. We report the log files of G2RL and HER in Maze at 10, 50, 100 episodes, which contain the number of visited nodes and actions taken.

```
==================== Graph−enhanced Goal−oriented Reinforcement Learning (G2RL) ====================
episode is: 10
nodes: [22, 21, 11, 31, 42, 32, 41, 23, 43, 33, 44, 34, 13, 14, 15, 26, 25, 36, 35, 45, 16, 24, 37, 47, 38, 48, 46, 12, 49,
        59, 69, 79, 80, 78, 90, 89, 99, 109, 110, 100, 27, 28, 39, 50, 40]
number of nodes: 45
```

edges: [(22, 22), (22, 21), (22, 23), (22, 32), (22, 33), (22, 13), (22, 31), (22, 12), (21, 21), (21, 11), (21, 31), (21, 32), (21, 22), (11, 11), (11, 21), (11, 12), (31, 31), (31, 42), (31, 41), (31, 22), (31, 32), (31, 21), (42, 42), (42, 32), (42, 43), (42, 41), (42, 31), (32, 31), (32, 32), (32, 42), (32, 43), (32, 33), (32, 23), (32, 22), (41, 41), (41, 31), (23, 22), (23, 23), (23, 32), (23, 34), (23, 33), (23, 24), (43, 32), (43, 43), (43, 42), (43, 33), (43, 44), (43, 34), (33, 33), (33, 44), (33, 32), (33, 43), (33, 23), (33, 34), (44, 44), (44, 33), (44, 43), (44, 35), (44, 34), (44, 45), (34, 43), (34, 33), (34, 34), (34, 24), (34, 35), (34, 44), (34, 45), (13, 13), (13, 14), (13, 23), (14, 15), (14, 25), (14, 14), (15, 15), (15, 26), (15, 25), (15, 14), (15, 16), (26, 26), (26, 25), (26, 36), (26, 16), (26, 37), (26, 27), (25, 26), (25, 25), (25, 15), (25, 36), (36, 35), (36, 26), (36, 36), (36, 37), (36, 27), (35, 35), (35, 45), (35, 26), (35, 34), (35, 36), (35, 44), (35, 25), (45, 35), (45, 44), (45, 45), (16, 26), (24, 35), (24, 34), (24, 25), (37, 47), (37, 37), (37, 38), (37, 48), (47, 37), (47, 47), (47, 48), (47, 46), (38, 47), (38, 48), (38, 37), (38, 49), (38, 28), (38, 39), (48, 38), (48, 48), (48, 49), (46, 47), (46, 46), (12, 11), (12, 21), (12, 23), (12, 22), (12, 13), (49, 48), (49, 59), (49, 50), (59, 69), (69, 69), (69, 79), (79, 80), (79, 78), (79, 79), (79, 90), (80, 79), (78, 79), (90, 89), (89, 99), (99, 99), (99, 109), (109, 110), (109, 109), (109, 100), (110, 100), (100, 109), (27, 36), (27, 27), (27, 38), (28, 28), (28, 38), (39, 50), (39, 40), (39, 39), (50, 39), (50, 40), (50, 50), (40, 49), (40, 39), (40, 50)]
number of edges: 166
episode is: 50
nodes: [22, 21, 11, 31, 42, 32, 41, 23, 43, 33, 44, 34, 13, 14, 15, 26, 25, 36, 35, 45, 16, 24, 37, 47, 38, 48, 46, 12, 49, 59, 69, 79, 80, 78, 90, 89, 99, 109, 110, 100, 27, 28, 39, 50, 40, 29, 30, 51, 57, 18, 19, 58, 68, 17, 60, 20, 67, 70, 71, 61]
number of nodes: 60
edges: [(22, 22), (22, 21), (22, 23), (22, 32), (22, 33), (22, 13), (22, 31), (22, 12), (22, 11), (21, 21), (21, 11), (21, 31), (21, 32), (21, 22), (21, 12), (11, 11), (11, 21), (11, 12), (11, 22), (31, 31), (31, 42), (31, 41), (31, 22), (31, 32), (31, 21), (31, 40), (31, 30), (42, 42), (42, 32), (42, 43), (42, 41), (42, 31), (32, 31), (32, 32), (32, 42), (32, 43), (32, 33), (32, 23), (32, 22), (32, 41), (32, 21), (41, 41), (41, 31), (41, 40), (41, 32), (23, 22), (23, 23), (23, 32), (23, 34), (23, 33), (23, 24), (23, 13), (23, 14), (23, 12), (43, 32), (43, 43), (43, 42), (43, 33), (43, 44), (43, 34), (33, 33), (33, 44), (33, 32), (33, 43), (33, 23), (33, 34), (33, 22), (33, 42), (33, 24), (44, 44), (44, 33), (44, 43), (44, 35), (44, 34), (44, 45), (34, 43), (34, 33), (34, 34), (34, 24), (34, 35), (34, 44), (34, 45), (34, 25), (13, 13), (13, 14), (13, 23), (13, 12), (13, 22), (13, 24), (14, 15), (14, 25), (14, 14), (14, 24), (14, 13), (14, 23), (15, 15), (15, 26), (15, 25), (15, 14), (15, 16), (26, 26), (26, 25), (26, 36), (26, 16), (26, 37), (26, 27), (26, 35), (26, 17), (25, 26), (25, 25), (25, 15), (25, 36), (25, 24), (25, 35), (25, 16), (25, 34), (36, 35), (36, 26), (36, 36), (36, 37), (36, 27), (36, 46), (36, 47), (36, 45), (35, 35), (35, 45), (35, 26), (35, 34), (35, 36), (35, 44), (35, 25), (35, 46), (35, 24), (45, 35), (45, 44), (45, 45), (45, 46), (45, 36), (45, 34), (16, 26), (16, 16), (16, 27), (16, 17), (16, 15), (16, 25), (24, 35), (24, 34), (24, 25), (24, 24), (24, 14), (24, 23), (24, 15), (24, 33), (37, 47), (37, 37), (37, 38), (37, 48), (37, 28), (37, 36), (37, 46), (37, 27), (47, 37), (47, 47), (47, 48), (47, 46), (47, 38), (47, 58), (47, 57), (47, 36), (38, 47), (38, 48), (38, 37), (38, 49), (38, 28), (38, 39), (38, 38), (38, 27), (48, 38), (48, 48), (48, 49), (48, 57), (48, 47), (48, 58), (48, 59), (46, 47), (46, 46), (46, 37), (46, 45), (46, 36), (46, 35), (12, 11), (12, 21), (12, 23), (12, 22), (12, 13), (12, 12), (49, 48), (49, 59), (49, 50), (49, 39), (49, 49), (49, 60), (59, 69), (59, 59), (59, 48), (59, 50), (59, 60), (59, 49), (59, 58), (69, 69), (69, 79), (69, 78), (69, 80), (79, 80), (79, 78), (79, 79), (79, 90), (79, 68), (80, 79), (80, 69), (80, 80), (80, 90), (78, 79), (78, 78), (78, 68), (78, 69), (78, 89), (90, 89), (90, 79), (90, 90), (89, 99), (89, 79), (99, 99), (99, 109), (109, 110), (109, 109), (109, 100), (110, 100), (100, 109), (27, 36), (27, 27), (27, 38), (27, 28), (27, 26), (27, 37), (27, 18), (27, 16), (27, 17), (28, 28), (28, 38), (28, 27), (28, 18), (28, 19), (28, 37), (28, 39), (28, 29), (39, 50), (39, 40), (39, 39), (39, 29), (39, 38), (50, 39), (50, 40), (50, 50), (50, 49), (50, 59), (50, 60), (50, 51), (40, 49), (40, 39), (40, 50), (40, 40), (40, 51), (40, 41), (40, 29), (40, 31), (40, 30), (29, 30), (29, 39), (29, 29), (29, 19), (29, 28), (30, 29), (30, 40), (30, 31), (30, 30), (30, 20), (30, 39), (51, 40), (57, 57), (57, 68), (57, 47), (57, 58), (57, 48), (18, 19), (18, 27), (18, 28), (18, 18), (18, 17), (18, 29), (19, 28), (19, 19), (19, 18), (19, 30), (19, 29), (58, 48), (58, 58), (58, 57), (58, 59), (58, 49), (58, 47), (58, 67), (68, 69), (68, 78), (68, 79), (17, 17), (17, 18), (17, 28), (17, 16), (17, 27), (60, 50), (60, 60), (60, 49), (60, 70), (60, 61), (60, 59), (20, 19), (67, 67), (67, 58), (70, 71), (70, 70), (70, 60), (70, 69), (71, 71), (71, 70), (61, 70)]
number of edges: 336
episode: 100
nodes: [22, 21, 11, 31, 42, 32, 41, 23, 43, 33, 44, 34, 13, 14, 15, 26, 25, 36, 35, 45, 16, 24, 37, 47, 38, 48, 46, 12, 49, 59, 69, 79, 80, 78, 90, 89, 99, 109, 110, 100, 27, 28, 39, 50, 40, 29, 30, 51, 57, 18, 19, 58, 68, 17, 60, 20, 67, 70, 71, 61, 88, 87, 96, 106, 105, 104, 114, 115, 81, 77, 97, 107, 86, 98, 108, 95, 85, 94, 103]
number of nodes: 79
edges: [(22, 22), (22, 21), (22, 23), (22, 32), (22, 33), (22, 13), (22, 31), (22, 12), (22, 11), (21, 21), (21, 11), (21, 31), (21, 32), (21, 22), (21, 12), (21, 20), (21, 30), (11, 11), (11, 21), (11, 12), (11, 22), (31, 31), (31, 42), (31, 41), (31, 22), (31, 32), (31, 21), (31, 40), (31, 30), (42, 42), (42, 32), (42, 43), (42, 41), (42, 31), (42, 33), (32, 31), (32, 32), (32, 42), (32, 43), (32, 33), (32, 23), (32, 22), (32, 41), (32, 21), (41, 41), (41, 31), (41, 40), (41, 32), (41, 42), (41, 50), (41, 30), (23, 22), (23, 23), (23, 32), (23, 34), (23, 33), (23, 24), (23, 13), (23, 14), (23, 12), (43, 32), (43, 43), (43, 42), (43, 33), (43, 44), (43, 34), (33, 33), (33, 44), (33, 32), (33, 43), (33, 23), (33, 34), (33, 22), (33, 42), (33, 24), (44, 44), (44, 33), (44, 43), (44, 35), (44, 34), (44, 45), (34, 43), (34, 33), (34, 34), (34, 24), (34, 35), (34, 44), (34, 45), (34, 25), (34, 23), (13, 13), (13, 14), (13, 23), (13, 12), (13, 22), (13, 24), (14, 15), (14, 25), (14, 14), (14, 24), (14, 13), (14, 23), (15, 15), (15, 26), (15, 25), (15, 14), (15, 16), (15, 24), (26, 26), (26, 25), (26, 36), (26, 16), (26, 37), (26, 27), (26, 35), (26, 17), (26, 15), (25, 26), (25, 25), (25, 15), (25, 36), (25, 24), (25, 35), (25, 16), (25, 34), (25, 14), (36, 35), (36, 26), (36, 36), (36, 37), (36, 27), (36, 46), (36, 47), (36, 45), (36, 25), (35, 35), (35, 45), (35, 26), (35, 34), (35, 36), (35, 44), (35, 25), (35, 46), (35, 24), (45, 35), (45, 44), (45, 45), (45, 46), (45, 36), (45, 34), (16, 26), (16, 16), (16, 27), (16, 17), (16, 15), (16, 25), (24, 35), (24, 34), (24, 25), (24, 24), (24, 14), (24, 23), (24, 15), (24, 33), (24, 13), (37, 47), (37, 37), (37, 38), (37, 48), (37, 28), (37, 36), (37, 46), (37, 27), (37, 26), (47, 37), (47, 47), (47, 48), (47, 46), (47, 38), (47, 58), (47, 57), (47, 36), (38, 47), (38, 48), (38, 37), (38, 49), (38, 28), (38, 39), (38, 38), (38, 27), (38, 29), (48, 38), (48, 48), (48, 49), (48, 57), (48, 47), (48, 58), (48, 59), (48, 37), (48, 39), (46, 47), (46, 46), (46, 37), (46, 45), (46, 36), (46, 35), (12, 11), (12, 21), (12, 23), (12, 22), (12, 13), (12, 12), (49, 48), (49, 59), (49, 50), (49, 39), (49, 49), (49, 60), (49, 58), (49, 40), (49, 38), (59, 69), (59, 59), (59, 48), (59, 50), (59, 60), (59, 49), (59, 58), (59, 68), (59, 70), (69, 69), (69, 79), (69, 78), (69, 80), (69, 70), (69, 68), (69, 59), (79, 80), (79, 78), (79, 79), (79, 90), (79, 68), (79, 88), (79, 89), (79, 69), (80, 79), (80, 69), (80, 80), (80, 90), (80, 89), (80, 81), (80, 70), (80, 71), (78, 79), (78, 78), (78, 68), (78, 69), (78, 89), (78, 87), (78, 67), (78, 77), (78, 88), (90, 89), (90, 79), (90, 90), (90, 80), (89, 99), (89, 79), (89, 80), (89, 89), (89, 88), (89, 90), (89, 78), (89, 98), (99, 99), (99, 109), (99, 88), (99, 89), (99, 98), (99, 100), (109, 110), (109, 109), (109, 100), (110, 100), (100, 109), (100, 99), (27, 36), (27, 27), (27, 38), (27, 28), (27, 26), (27, 37), (27, 18),

(27, 16), (27, 17), (28, 28), (28, 38), (28, 27), (28, 18), (28, 19), (28, 37), (28, 39), (28, 29), (28, 17), (39, 50), (39, 40), (39, 39), (39, 29), (39, 38), (39, 49), (39, 48), (39, 30), (50, 39), (50, 40), (50, 50), (50, 49), (50, 59), (50, 60), (50, 51), (50, 61), (50, 41), (40, 49), (40, 39), (40, 50), (40, 40), (40, 51), (40, 41), (40, 29), (40, 31), (40, 30), (29, 30), (29, 39), (29, 29), (29, 19), (29, 28), (29, 18), (29, 40), (29, 38), (29, 20), (30, 29), (30, 40), (30, 31), (30, 30), (30, 20), (30, 39), (30, 19), (30, 21), (51, 40), (51, 51), (51, 60), (51, 50), (57, 57), (57, 68), (57, 47), (57, 58), (57, 48), (57, 67), (18, 19), (18, 27), (18, 28), (18, 18), (18, 17), (18, 29), (19, 28), (19, 19), (19, 18), (19, 30), (19, 29), (19, 20), (58, 48), (58, 58), (58, 57), (58, 59), (58, 49), (58, 47), (58, 67), (58, 69), (58, 68), (68, 69), (68, 78), (68, 79), (68, 68), (68, 57), (68, 67), (68, 77), (68, 58), (17, 17), (17, 18), (17, 28), (17, 16), (17, 27), (60, 50), (60, 60), (60, 49), (60, 70), (60, 61), (60, 59), (60, 51), (60, 69), (60, 71), (20, 19), (20, 30), (20, 20), (20, 21), (20, 29), (67, 67), (67, 58), (67, 68), (67, 78), (67, 77), (67, 57), (70, 71), (70, 70), (70, 60), (70, 69), (70, 79), (70, 80), (70, 59), (71, 71), (71, 70), (71, 80), (61, 70), (61, 61), (61, 50), (61, 60), (88, 87), (88, 79), (88, 89), (88, 88), (88, 99), (88, 98), (88, 78), (88, 97), (87, 78), (87, 88), (87, 96), (87, 87), (87, 97), (87, 77), (87, 86), (96, 106), (96, 97), (96, 87), (96, 86), (96, 96), (96, 95), (106, 105), (106, 107), (106, 96), (105, 105), (105, 104), (105, 114), (105, 115), (104, 114), (104, 104), (104, 105), (114, 114), (114, 104), (114, 105), (115, 105), (81, 80), (77, 77), (77, 67), (77, 68), (77, 88), (77, 78), (97, 96), (97, 106), (97, 107), (97, 87), (107, 96), (107, 106), (107, 107), (107, 108), (86, 96), (86, 86), (98, 99), (98, 89), (98, 98), (95, 95), (95, 85), (95, 94), (85, 85), (85, 95), (94, 103), (103, 103)]
number of edges: 486
========== Hindsight Experience Replay (HER) ==========
episode is: 10
nodes: [22, 21, 31, 41, 42, 32, 23, 43, 33, 44, 34, 35, 45, 46, 36, 37, 47, 13, 12, 14, 15, 16, 17, 26, 25, 24, 48, 11, 27]
number of nodes: 29
edges: [(22, 21), (22, 23), (22, 22), (22, 32), (22, 33), (22, 12), (21, 21), (21, 31), (21, 22), (31, 31), (31, 41), (31, 21), (31, 42), (41, 42), (41, 41), (41, 32), (41, 31), (42, 41), (42, 42), (42, 32), (42, 43), (42, 31), (32, 31), (32, 32), (32, 42), (32, 33), (32, 43), (32, 23), (32, 41), (23, 22), (23, 13), (23, 14), (23, 33), (43, 32), (43, 43), (43, 42), (43, 33), (43, 44), (43, 34), (33, 33), (33, 44), (33, 32), (33, 43), (33, 34), (44, 44), (44, 33), (44, 43), (44, 34), (44, 35), (44, 45), (34, 43), (34, 33), (34, 35), (34, 34), (34, 45), (35, 45), (35, 46), (35, 35), (35, 34), (35, 25), (45, 46), (45, 45), (45, 35), (45, 44), (45, 34), (46, 46), (46, 45), (46, 35), (46, 36), (46, 37), (36, 36), (36, 37), (36, 47), (36, 26), (36, 46), (37, 47), (37, 37), (37, 36), (37, 48), (47, 47), (47, 37), (47, 36), (47, 46), (13, 13), (13, 12), (13, 14), (12, 13), (12, 11), (12, 12), (12, 23), (14, 14), (14, 13), (14, 15), (15, 15), (15, 16), (15, 26), (15, 25), (16, 16), (16, 17), (26, 25), (26, 26), (26, 37), (26, 36), (26, 27), (25, 24), (25, 36), (25, 26), (24, 15), (48, 37), (11, 12), (11, 11), (27, 27)]
number of edges: 112
episode is: 50
nodes: [22, 21, 31, 41, 42, 32, 23, 43, 33, 44, 34, 35, 45, 46, 36, 37, 47, 13, 12, 14, 15, 16, 17, 26, 25, 24, 48, 11, 27, 18, 19, 20, 58, 57, 49, 39, 60, 50, 59, 40, 38, 28, 29, 30, 69, 70, 80, 90, 101, 100, 67, 68, 77, 78, 61]
number of nodes: 55
edges: [(22, 21), (22, 23), (22, 22), (22, 32), (22, 33), (22, 12), (22, 13), (22, 11), (21, 21), (21, 31), (21, 22), (21, 11), (21, 12), (21, 32), (31, 31), (31, 41), (31, 21), (31, 42), (31, 32), (41, 42), (41, 41), (41, 32), (41, 31), (42, 41), (42, 42), (42, 32), (42, 43), (42, 31), (32, 31), (32, 32), (32, 42), (32, 33), (32, 43), (32, 23), (32, 41), (23, 22), (23, 13), (23, 14), (23, 33), (23, 34), (23, 24), (23, 23), (23, 12), (43, 32), (43, 43), (43, 42), (43, 33), (43, 44), (43, 34), (33, 33), (33, 44), (33, 32), (33, 43), (33, 34), (33, 23), (33, 22), (33, 42), (33, 24), (44, 44), (44, 33), (44, 43), (44, 34), (44, 35), (44, 45), (34, 43), (34, 33), (34, 25), (34, 24), (35, 45), (35, 46), (35, 35), (35, 34), (35, 25), (35, 44), (35, 36), (45, 46), (45, 45), (45, 35), (45, 44), (45, 34), (45, 36), (46, 46), (46, 45), (46, 35), (46, 36), (46, 37), (46, 47), (36, 36), (36, 37), (36, 47), (36, 26), (36, 46), (36, 45), (36, 35), (36, 27), (37, 47), (37, 37), (37, 36), (37, 48), (37, 46), (37, 27), (47, 47), (47, 37), (47, 36), (47, 46), (47, 48), (47, 38), (47, 57), (47, 58), (13, 13), (13, 12), (13, 14), (13, 22), (13, 23), (12, 13), (12, 11), (12, 12), (12, 23), (12, 22), (12, 21), (14, 14), (14, 13), (14, 15), (14, 23), (14, 25), (14, 24), (15, 15), (15, 16), (15, 26), (15, 25), (15, 14), (16, 16), (16, 17), (16, 15), (16, 26), (17, 17), (17, 18), (26, 25), (26, 26), (26, 37), (26, 36), (26, 27), (26, 15), (26, 16), (25, 24), (25, 36), (25, 26), (25, 16), (25, 15), (25, 14), (25, 25), (25, 35), (24, 15), (24, 35), (24, 24), (24, 25), (24, 33), (24, 23), (24, 34), (48, 37), (48, 47), (48, 48), (48, 58), (48, 49), (48, 59), (48, 38), (11, 12), (11, 11), (11, 21), (27, 27), (27, 28), (27, 38), (27, 37), (18, 18), (18, 19), (19, 19), (19, 20), (20, 20), (20, 19), (20, 30), (58, 57), (58, 69), (58, 58), (58, 67), (58, 59), (58, 48), (57, 48), (57, 57), (57, 58), (57, 67), (49, 39), (49, 60), (49, 50), (49, 59), (49, 49), (39, 49), (39, 29), (39, 39), (60, 49), (60, 50), (60, 70), (50, 60), (50, 49), (50, 40), (59, 49), (59, 59), (59, 69), (59, 58), (59, 60), (40, 49), (38, 47), (38, 28), (38, 38), (38, 39), (38, 48), (28, 29), (28, 38), (29, 29), (29, 30), (29, 39), (30, 30), (30, 20), (30, 29), (69, 69), (69, 70), (69, 58), (69, 68), (69, 59), (70, 70), (70, 80), (70, 61), (80, 80), (80, 90), (90, 90), (90, 101), (101, 101), (101, 100), (100, 101), (100, 100), (100, 90), (67, 68), (67, 67), (67, 77), (68, 58), (68, 69), (77, 77), (77, 78)]
number of edges: 255
episode: 100
nodes: [22, 21, 31, 41, 42, 32, 23, 43, 33, 44, 34, 35, 45, 46, 36, 37, 47, 13, 12, 14, 15, 16, 17, 26, 25, 24, 48, 11, 27, 18, 19, 20, 58, 57, 49, 39, 60, 50, 59, 40, 38, 28, 29, 30, 69, 70, 80, 90, 101, 100, 67, 68, 77, 78, 61, 88, 87, 97, 96, 106, 117, 107, 71, 79, 89, 86, 85, 95, 51, 99, 110]
number of nodes: 71
edges: [(22, 21), (22, 23), (22, 22), (22, 32), (22, 33), (22, 12), (22, 13), (22, 11), (22, 31), (21, 21), (21, 31), (21, 22), (21, 11), (21, 12), (21, 32), (21, 20), (31, 31), (31, 41), (31, 21), (31, 42), (31, 32), (31, 30), (31, 40), (31, 22), (41, 42), (41, 41), (41, 32), (41, 31), (42, 41), (42, 42), (42, 32), (42, 43), (42, 31), (42, 33), (32, 31), (32, 32), (32, 42), (32, 33), (32, 43), (32, 23), (32, 41), (32, 22), (32, 21), (23, 22), (23, 13), (23, 14), (23, 33), (23, 34), (23, 24), (23, 23), (23, 12), (43, 32), (43, 43), (43, 42), (43, 33), (43, 44), (43, 34), (33, 33), (33, 44), (33, 32), (33, 43), (33, 34), (33, 23), (33, 22), (33, 42), (33, 24), (44, 44), (44, 33), (44, 43), (44, 34), (44, 35), (44, 45), (34, 43), (34, 33), (34, 25), (34, 24), (35, 45), (35, 46), (35, 35), (35, 34), (35, 25), (35, 44), (35, 36), (45, 46), (45, 45), (45, 35), (45, 44), (45, 34), (45, 36), (46, 46), (46, 45), (46, 35), (46, 36), (46, 37), (46, 47), (36, 36), (36, 37), (36, 47), (36, 26), (36, 46), (36, 45), (36, 35), (36, 27), (37, 47), (37, 37), (37, 36), (37, 48), (37, 46), (37, 27), (47, 47), (47, 37), (47, 36), (47, 46), (47, 48), (47, 38), (47, 57), (47, 58), (13, 13), (13, 12), (13, 14), (13, 22), (13, 23), (12, 13), (12, 11), (12, 12), (12, 23), (12, 22), (12, 21), (14, 14), (14, 13), (14, 15), (14, 23), (14, 25), (14, 24), (15, 15), (15, 16), (15, 26), (15, 25), (15, 14), (16, 16), (16, 17), (16, 15), (16, 26), (16, 27), (17, 17), (17, 18), (17, 16), (17, 28), (17, 27), (17, 26), (26, 25), (26, 26), (26, 37), (26, 36), (26, 27), (26, 15), (26, 16), (26, 35), (26, 17), (25, 24), (25, 36), (25, 26), (25, 16), (25, 15), (25, 14), (25, 25), (25, 35), (24, 15), (24, 35), (24, 24), (24, 25), (24, 33), (24, 23), (24, 34), (24, 14), (48, 37), (48,

47), (48, 48), (48, 58), (48, 49), (48, 59), (48, 38), (48, 39), (48, 57), (11, 12), (11, 11), (11, 21), (27, 27), (27, 28), (27, 38), (27, 37), (27, 18), (27, 26), (27, 17), (18, 18), (18, 19), (18, 17), (19, 19), (19, 20), (19, 29), (19, 18), (19, 28), (20, 20), (20, 19), (20, 30), (20, 29), (20, 21), (58, 57), (58, 69), (58, 58), (58, 67), (58, 59), (58, 48), (58, 68), (58, 49), (57, 48), (57, 57), (57, 58), (57, 67), (57, 68), (49, 39), (49, 60), (49, 50), (49, 59), (49, 49), (49, 58), (49, 48), (39, 49), (39, 29), (39, 39), (39, 38), (39, 30), (39, 40), (60, 49), (60, 50), (60, 70), (60, 60), (60, 69), (60, 59), (60, 61), (50, 60), (50, 49), (50, 40), (50, 61), (50, 50), (50, 51), (59, 49), (59, 59), (59, 69), (59, 58), (59, 60), (59, 68), (40, 49), (40, 40), (40, 39), (40, 29), (40, 30), (40, 31), (40, 50), (38, 47), (38, 28), (38, 38), (38, 39), (38, 48), (38, 37), (38, 29), (38, 49), (28, 29), (28, 38), (28, 28), (28, 39), (28, 18), (28, 19), (29, 29), (29, 30), (29, 39), (29, 19), (29, 28), (29, 38), (29, 40), (30, 30), (30, 20), (30, 29), (30, 31), (30, 40), (69, 69), (69, 70), (69, 58), (69, 68), (69, 59), (69, 78), (69, 80), (69, 79), (70, 70), (70, 80), (70, 61), (70, 71), (70, 60), (80, 80), (80, 90), (80, 79), (80, 70), (80, 69), (80, 89), (90, 90), (90, 101), (90, 79), (90, 89), (101, 101), (101, 100), (100, 101), (100, 100), (100, 90), (100, 110), (67, 68), (67, 67), (67, 77), (67, 57), (68, 58), (68, 69), (68, 68), (68, 67), (68, 78), (77, 77), (77, 78), (77, 67), (77, 87), (78, 88), (78, 77), (78, 68), (78, 69), (61, 61), (61, 60), (61, 50), (61, 70), (88, 87), (87, 97), (87, 86), (97, 96), (96, 106), (106, 117), (106, 107), (117, 106), (107, 107), (71, 71), (71, 70), (79, 79), (79, 68), (79, 80), (79, 69), (79, 78), (89, 90), (89, 99), (86, 85), (85, 95), (51, 60), (99, 99), (99, 100)]
number of edges: 370