EXPLORE WITH DYNAMIC MAP: GRAPH STRUCTURED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In reinforcement learning, a map with states and transitions built based on historical trajectories is often helpful in exploration and exploitation. Even so, learning and planning on such a map within a sparse environment remains a challenge. As a step towards this goal, we propose Graph Structured Reinforcement Learning (GSRL), which utilizes historical trajectories to slowly adjust exploration directions and learn related experiences while rapidly updating the value function estimation. GSRL constructs a dynamic graph on top of state transitions in the replay buffer based on historical trajectories, and develops an attention strategy on the map to select an appropriate goal direction, which decomposes the task of reaching a distant goal state into a sequence of easier tasks. We also leverage graph structure to sample related trajectories for efficient value learning. Results demonstrate that GSRL can outperform the state-of-the-art algorithms in terms of sample efficiency on benchmarks with sparse reward functions.

1 INTRODUCTION

How can humans learn to solve tasks with complex structure and delayed and sparse feedback? Take a complicated navigation task as an example in which the goal is to go from a start state to an end state. A straightforward approach that one often takes is by decomposing this complicated task into a sequence of easier ones, identifying/forming some intermediate goals that help to get to the final goal, and finally choosing a route with the highest return among a poll of candidate routes that lead to the final goal. Despite recent success and progress in reinforcement learning (RL) approaches (Mnih et al., 2013; Fakoor et al., 2020; Hansen et al., 2018; Huang et al., 2019; Silver et al., 2016), there is still a huge gap between how humans and RL agents learn.

In many real-world applications, an RL agent only has access to sparse and delayed rewards, which by itself leads to two major challenges. **(C1)** The first one is how can an agent effectively learn from sparse and delayed rewards? One possible solution is to build a new reward function, known as an intrinsic reward, that helps expedite the learning process and ultimately solve a given task. Although this may seem like an appealing solution, it is often not obvious how to formulate an effective intrinsic reward. Recent works have introduced goal-oriented RL (Schaul et al., 2015; Chiang et al., 2019) as a way of constructing intrinsic rewards. However, most goal-oriented RL algorithms require large amounts of reward shaping (Chiang et al., 2019) or human demonstrations (Nair et al., 2018). **(C2)** The second issue is how to retrieve relevant past experiences that help to learn faster and improve sample efficiency. Recent attention has focused much on episodic RL (Blundell et al., 2016), which builds a non-parametric episodic memory to store past experiences, and thus can rapidly latch onto related ones through search with similarity. However, most episodic RL algorithms require additional memory space (Pritzel et al., 2017).

In this work, we propose Graph Structured Reinforcement Learning (GSRL), which constructs a state-transition graph, leverages structural graph information and presents solutions for the problems raised above. As shown in Figure 1, when we encounter a complex task, we draw a map for planning with a long horizon (Eysenbach et al., 2019), and scan through the graph to seek for related experiences within a short horizon (Lee et al., 2019).

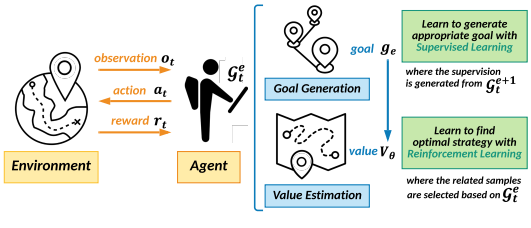


Figure 1: An illustration for motivation of GSRL.

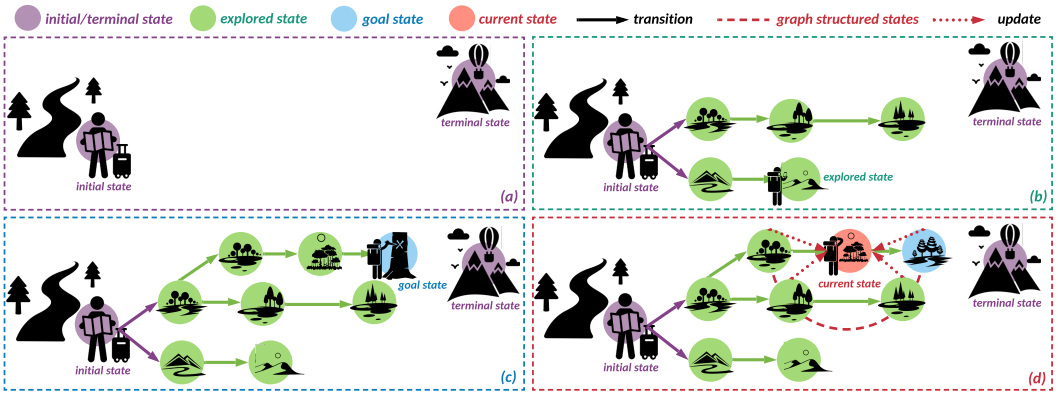


Figure 2: GSRL allows joint optimization of map construction and agent exploration from scratch. (a) an agent is located at initial state and aims at terminal state with empty map at the beginning. (b) an agent explores the environment and records the explored states on the map. (c) an agent generates a goal from previous states to decompose the whole task into a sequence of easier tasks. (d) an agent updates current policy with related trajectories selected by structured map information.

Inspired by this, we propose a method to address the first problem **(C1)** by learning to generate an appropriate goal g_e based on map $\mathcal{G}_t^e$ at episode e and form the intrinsic rewards through goal generation. This goal generation can be optimized in hindsight. That is, we can update the goal generation model under the supervision generated by planning algorithms based on map $\mathcal{G}_t^{e+1}$ at episode $e+1$. We can solve the second problem **(C2)** by updating the value estimation of the current state with the trajectories that include neighborhood states. Note that GSRL draws a map to help agents plan an appropriate direction in the long horizon and learn an optimal strategy in the short horizon, which can be widely incorporated with various learning strategies such as DQN (Mnih et al., 2013), DDPG (Lillicrap et al., 2015), *etc*.

However, arriving at a well-explored map in terms of state-transitions in the first place from scratch remains a challenging problem. In this regard, GSRL should be able to optimize map construction, goal selection and value estimation jointly. As illustrated in Figure 2, we design GSRL mainly following four steps: (a) at the beginning, an agent is located at the initial state and aims at the terminal state with an empty map; (b) the agent then explores the environment with current policy and records those explored states on the map; (c) considering the far distance to the terminal state and sparse reward during the exploration, we decompose the whole task into a sequence of easier tasks guided by a goal; (d) we leverage structured map information to select related trajectories to update the current policy; the agent can repeat procedures (b)-(d) and finally reach the terminal state with a well-explored map.

Our primary contribution is a novel algorithm that constructs a dynamic state-transition graph from scratch and leverages structured information to find an appropriate goal in goal selection with long horizon and sample related trajectories in value estimation with short horizon. This framework provides several benefits: (1) How to balance exploration and exploitation is a fundamental problem in any reinforcement learning (Andrychowicz et al., 2017; Nachum et al., 2018). GSRL learns an attention mechanism on the graph to decide whether to explore the states with high uncertainty or exploit the states with high value. (2) Almost all the existing reinforcement learning approaches (Paul et al., 2019; Eysenbach et al., 2019) utilize randomly sampled trajectories to update the current policy leading to inefficiency. GSRL selects highly related trajectories for updates through the local structure in the graph. (3) Unlike previous approaches (Kipf et al., 2020; Shang et al., 2019), GSRL constructs a dynamic graph through agent exploration without any additional object representation techniques and thus is not limited to several specific environments. Empirically, we find that our method constructs a useful map and learns an efficient policy for highly structured tasks. Comparisons with state-of-the-art RL methods show that GSRL is substantially more successful for boosting the performance of a graph-structured RL algorithm.

2 Related Work

Structured Models of Environments. Recent work has investigated leveraging structured environments to make great strides in improving predictive accuracy (Kipf et al., 2018; Xu et al., 2019; Chang et al., 2017; Battaglia et al., 2016; Watters et al., 2017; Sanchez-Gonzalez et al., 2018) and accelerating reinforcement learning procedures (Shang et al., 2019; Kipf et al., 2020; Eysenbach

et al., 2019; Wang et al., 2018). These methods mainly first construct some form of graph neural network where node update functions model the dynamics of individual objects, parts or agents and edge update functions model their interactions and relations through unsupervised object discovery (Xu et al., 2019), contrastive learning (Kipf et al., 2020) or agent exploration (Eysenbach et al., 2019). Unlike these previous works based on well-organized graphs, our model GSRL constructs and makes use of the graph from scratch. Specially, GSRL constructs a dynamic map via exploration, and builds efficient exploration with structured information from this map, which allows our approach to be widely deployed into more complex environments besides object interaction simulations.

Goal-oriented Reinforcement Learning. Goal-oriented RL allows an agent to generate intrinsic rewards, which is defined with respect to target subsets of the state space called goals (Florensa et al., 2018; Paul et al., 2019). Recently, goal-oriented RL has been investigated widely in various deep RL scenarios such as imitation learning (Pathak et al., 2018; Srinivas et al., 2018), disentangling task knowledge from the environment (Mao et al., 2018; Ghosh et al., 2019), constituting lower-level controller in hierarchical RL (Shang et al., 2019; Nachum et al., 2018) and organizing cooperation in multi-agent RL (Jin et al., 2019; Yang et al., 2020). Hence, how to generate appropriate goals is the essential technique in any goal-oriented RL (Andrychowicz et al., 2017; Ren et al., 2019). Eysenbach et al. (2019) proposed to utilize the shortest-path search on replay buffer to generate a sequence of goals. Instead of planning goals from a well-explored state-transition graph, in this paper we investigate guiding exploration with structured information from a dynamic graph, which can largely improve the performance in terms of sample efficiency in tasks with sparse reward signals.

Episodic Reinforcement Learning. Episodic RL is inspired by psycho-biological and cognitive studies of human memory (Sutherland & Rudy, 1989; Marr et al., 1991; Lengyel & Dayan, 2008; Botvinick et al., 2019), and follows the idea of instance-based decision theory (Gilboa & Schmeidler, 1995). For example, model-free episodic control (Blundell et al., 2016) utilizes a completely non-parametric model that keeps the best Q values in tabular-based memory and replays the sequence of actions with the highest return. Recent works (Pritzel et al., 2017; Hansen et al., 2018; Lin et al., 2018) have been proposed to integrate episodic control with parametric DQN. Zhu et al. (2019) proposed to associate memory by a graph, which enables leveraging not only intra-episode but also inter-episode information. Different from this prior work, which adopts either tabular, semi-tabular or graph-based memory, our work builds a state-transition graph on the replay buffer, where we mine global structure for goal generation and local structure for value estimation.

3 PRELIMINARIES

Let $M = (S, A, E, P, R)$ be an Markov Decision Process (MDP) where S is the state space, A is the action space whose size is bounded by a constant, $E \in \mathbb{Z}_+$ 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 the environment, called extrinsic, are usually sparse and therefore difficult to learn. To address this, goal-oriented RL introduces goal $g \in G$ to build dense intrinsic rewards $R_g : S \times A \times G \to \Delta(\mathbb{R})$. In order to make the learning procedure stable, the goal is limited to be updated episodically. A policy $\pi : S \times G \to \Delta(A)$ prescribes a distribution over actions for each state and goal. 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 episode terminates only after T timesteps even when the agent reaches 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 value function:

$$Q^\pi(s, a, g) = \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t \cdot r_g(a_t, s_t) \mid s_t = s, a_t = a, \pi\right], \; V^*(s, g) = \max_a Q^*(s, a, g). \tag{1}$$

We use off-policy algorithms named DQN (Mnih et al., 2013) for discrete action space and DDPG (Lillicrap et al., 2015) for continuous action space to learn Q-functions and policies by utilizing off-policy data (*i.e.*, data stored in the replay buffer).

4 GRAPH STRUCTURED REINFORCEMENT LEARNING

As illustrated in Figure 1, GSRL constructs a dynamic graph on top of state transitions. However, learning and planning algorithms on the state transition graph are non-trivial. Firstly, we provide a theoretical analysis to show that exploration without any constraint would lead to the explosion of the graph. We then propose to divide states into several groups and incorporate them with an attention mechanism to select an appropriate state as a goal to encourage further exploration. In order to help RL agents learn efficiently, we update goal generation hindsight and value estimation with related trajectories.

4.1 EXPLORE WITH DYNAMIC GRAPH

We directly build a dynamic and directed graph using sample trajectories from the replay buffer. We denote the state-transition graph at timestep t in episode e by $\mathcal{G}_t^e =< \mathcal{V}_t^e, \mathcal{E}_t^e >$ with relations $\mathcal{R}_t^e$, the node space $\mathcal{V}_t^e$ represents state space $S_{\mathcal{G}_t^e}$, edge space $\mathcal{E}_t^e$ represents action space $A_{\mathcal{G}_t^e}$, and relations $\mathcal{R}_t^e$ represents transition $P_{\mathcal{G}_t^e}$. We denote the graph's boundary by $\partial\mathcal{G}_t^e$, which consists of observed but not fully explored states. An illustrated example of these notations above is available in Appendix A.

Let an agent starts from an initial state s_0, $\mathcal{G}_t^e$ grows from a single node, *i.e.*, $\mathcal{G}_0^0 = \{s_0\}$ and expands itself each timestep, leading to a sequence of $\{\{\mathcal{G}_i^0\}_{i=0}^{T-1}, \{\mathcal{G}_i^1\}_{i=0}^{T-1}, \dots, \{\mathcal{G}_i^{E-1}\}_{i=0}^{T-1}\}$ where T denotes trajectory length and E is the number of episodes. Given that state-transition graph $\mathcal{G}_t^e$ is always a connected directed graph, we describe the expansion behavior as consecutive expansion, which means no jumping across neighborhood allowed. We denote graph increment at timestep t in episode e as $\Delta\mathcal{G}_t^e$, and then we can ensure that $\mathcal{G}_t^e \cup \Delta\mathcal{G}_t^e = \mathcal{G}_{t+1}^e \subseteq \mathcal{G}_t^e \cup \partial\mathcal{G}_t^e$.

Proposition 1. *Given the full state-transition graph $\mathcal{G}_{full}$, we assume that the probability of degree of an arbitrary state being less than or equal to d is larger than p (*i.e.*, $P(deg(s) \leq d) > p, \forall s \in S_{\mathcal{G}}$). Considering a sequence of consecutively expanding sub-graphs $\{\{\mathcal{G}_i^0\}_{i=0}^{T-1}, \{\mathcal{G}_i^1\}_{i=0}^{T-1}, \dots, \{\mathcal{G}_i^{E-1}\}_{i=0}^{T-1}\}$, starting with $\mathcal{G}_0^0 = \{s_0\}$, for all $t \geq 1, e \geq 0$, we can ensure that $P\big(|S_{\mathcal{G}_t^e}| \leq \epsilon\big) > p^\epsilon$, where $\epsilon = \frac{d\cdot(d-1)^{T\cdot e+t}-2}{d-2}$ when $d > 2$ and $\epsilon = 1, 3$ when $d = 1, 2$ respectively.*

The proof of Proposition 1 can be found in Appendix C.1. The proposition implies the guarantee of upper-bounding $|S_{\mathcal{G}_t^e}|$ becomes looser and weaker as t, e get larger even if the given assumption has a small d and a large p (close to 1). In other words, randomly appending $\mathcal{G}_t^e$ through exploration would enhance computation complexity for learning and plan on the dynamic graph. In order to prevent $\mathcal{G}_t^e$ from explosion, we need to constrain $\Delta\mathcal{G}_t^e$ by guiding exploration with goals. In this paper, we propose to leverage graph structure to assist agent exploration. Before detailed introduction of exploration strategy, we firstly introduce the definition of certainty as

Definition 1. *Given a state s, we define its certainty as to the number of candidate actions that have been already taken. In this paper, we build state-transition graph $\mathcal{G}$ on the replay buffer and thus can adopt the out degree[1] of each state to approximately measure the certainty. In other words, we can approximate the certainty of state $cert(s) \approx deg(s)$, where $deg(s)$ denotes the degree of node s.*

Note that certainty of state in Definition 1 can be served as a local measurement to show the extent of exploration on some states, which is actually proportioned to the global measurement (*i.e.*, the number of visited states) in a deterministic environment (See Appendix D.1 for details). In addition, we also can define uncertainty of state as the number of untaken candidate actions.

Exploration Strategy. A simple but effective way is to do exploration based on goal-oriented RL, which decomposes the whole task into a sequential of goal-oriented tasks. In this case, newly explored states in each episode are guided by a specific goal. Therefore, we constrain the state-transition graph from the explosion and balance the exploration and exploitation through assigning an appropriate goal to the agent. In order to further discuss the definition of an appropriate goal, we here provide the definition of the optimal goal. As shown in Figure 4, once we are accessible to the whole state-transition graph $\mathcal{G}_{\text{full}}$, we can obtain the optimal solution by short path planning algorithm such as Dijkstra's Algorithm. We define the optimal goal as the terminal state on the path.

[1] In this paper, we use degree to represent out-degree, unless otherwise stated.

Definition 2. *For fully explored graph $\mathcal{G}_{full}$, we define the optimal goal g^*_{full} as the terminal state (*e.g.*, s_{10} in Figure 4(a)). For any not fully explored state-transition graph $\mathcal{G}^e_0$, we define the optimal goal g^*_e hindsightly, where we generate the optimal solution path $\mathcal{P}_{e+1}$ based on the shortest path planning on the next episode graph $\mathcal{G}^{e+1}_0$ and regard the reachable terminal state included both in $\mathcal{G}^e_0$ and $\mathcal{P}_{e+1}$ as g^*_e (*e.g.*, s_6 in Figure 4(c)).*

One should be noted that the optimal goal is hindsight generated and keep approaching the terminal state during exploration. We further analyze and discuss the relations between goals in this paper and previous goal-oriented RL literatures in Appendix D.2.

Proposition 2. *Assume that Q-value of each state is well estimated (*i.e.*, $Q = Q^*$), then optimal goal g^*_e at the beginning timestep of any episode e is always included in the boundary of the state-transition graph $\mathcal{G}^e_0$ (*i.e.*, $\partial\mathcal{G}^e_0$).*

The proof of Proposition 2 can be found in Appendix C.2. The proposition implies that the candidate set for each goal generation can be limited to the boundary. However, learning to generate an appropriate goal is non-trivial respect to a dynamic and large-scale state-transition graph $\mathcal{G}^e_t$. One intuitive solution is to divide the whole candidate goal space (*i.e.*, $\partial\mathcal{G}^e_t$) into several candidate groups $\mathcal{C}_1, \dots, \mathcal{C}_N$, where N is the number of groups. In order to both cover all the states and control exploration size in these groups, our group segment should follow the principle that $\cup_{n=1}^{N}\mathcal{C}_n = \partial\mathcal{G}^e_t$ and $\mathcal{C}_m \cap \mathcal{C}_n = \emptyset$ for any $m \neq n; m, n = 1, 2, \dots, N$. Here, we set the last visited state s_{last} as $\mathcal{C}_0$, since s_{last} often is both close to the terminal state and has high uncertainty. As shown in Figure 3, based on the last explored state, we build group $\mathcal{C}_n, n = 1, \dots, N$ by extending scales in two different perspectives. We list two perspectives, as follows:

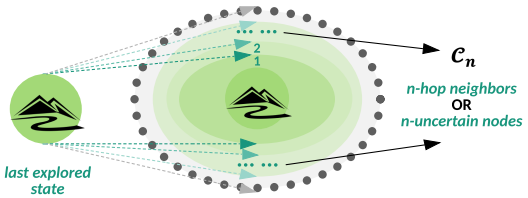


Figure 3: An illustration of exploration strategy.

- *Extending from Neighbor Nodes*. One perspective is to leverage local structure on graph, where $\mathcal{C}_n := \{s\}, s \in \mathcal{N}^n(s_{\text{last}}) \cap \partial\mathcal{G}^e_t$ for $n = 1, 2, \dots, N-1$, and $\mathcal{C}_N := \partial\mathcal{G}^e_t - \cup_{n=1}^{N-1}\mathcal{C}_n$. $\mathcal{N}^n(s_{\text{last}})$ means n-hop neighbors of s_{last}. The intuitive motivation behind this is to keep the learning procedure stable by slowly adjusting the goal to explore the surrounding environment.
- *Extending from Uncertain Nodes*. The other perspective is to utilize certainty information to guide goal generation, where $\mathcal{C}_n := \{s\}, s \in S_{d=|A|-n} \cap \partial\mathcal{G}^e_t$ for $n = 1, 2, \dots, N-1$, and $\mathcal{C}_N := \partial\mathcal{G}^e_t - \cup_{n=1}^{N-1}\mathcal{C}_n$. $S_{d=|A|-n}$ denotes set of states whose degree equals $|A| - n$, and $|A|$ is the size of action space. The intuitive motivation behind this is to eliminate uncertainty in the graph through exploration.

We provide the complexity analysis for two perspectives as follows. Let $d_{\partial\mathcal{G}^e_t}$ denote the maximum degree of states in $\partial\mathcal{G}^e_t$, and $|S_{\partial\mathcal{G}^e_t}|$ denote the number of states in $\partial\mathcal{G}^e_t$. The complexity to construct $\mathcal{C}_1, \dots, \mathcal{C}_N$ by extending from neighbor nodes is $\mathcal{O}(d^{N-1}_{\partial\mathcal{G}^e_t})$ and by extending from uncertain nodes is $\mathcal{O}(|S_{\partial\mathcal{G}^e_t}|)$. The detailed analysis and method are available in Appendix D.3.

Attention Strategy. One should be noted that extending from either neighbor or uncertain nodes can guarantee the number of groups $\mathcal{C}_1, \dots, \mathcal{C}_N$ on a dynamic graph $\mathcal{G}^e_t$ is fixed. In order to keep all the groups not empty, we append s_{last} to them at the beginning timestep of each episode. However, RL algorithms face an exploration-exploitation dilemma, which requires the agent to makes appropriate selection rather than random exploration over these groups. In order to the trade-off between exploration and exploitation, we apply an attention mechanism to select an appropriate group on the current situation. In this paper, the attention module runs over N groups at each episode, and select the appropriate one to balance exploration and exploitation, which can be formulated as

$$\mathcal{C}_{ATT} = \arg\max_{\mathcal{C}_n} ATT_\phi(\mathcal{C}_1, \dots, \mathcal{C}_N), \tag{2}$$

where ATT_ϕ denotes self-attention function parameterized by ϕ, and $\mathcal{C}_{ATT}$ means the group selected according to the attention score. At the beginning timestep of any episode e, we have access to state-transition graph $\mathcal{G}^e_0$ and its boundary $\partial\mathcal{G}^e_0$. We then divide $\partial\mathcal{G}^e_0$ into groups $\mathcal{C}_1, \dots, \mathcal{C}_N$, and obtain $\mathcal{C}_{ATT}$ through the attention mechanism. We select one state with the highest value in $\mathcal{C}_{ATT}$

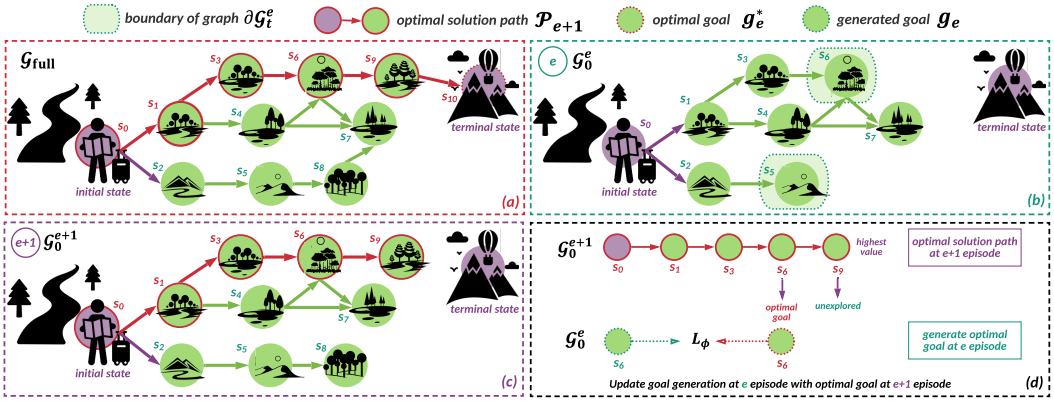


Figure 4: Illustrations of optimal goals. States surrounded by red circles and connected by red arrows are optimal solution path, which can be obtained by the shortest path planning algorithm on the fully explored graph $\mathcal{G}_{\text{full}}$ in (a). When the graph is not fully explored ($\mathcal{G}_0^e$ in (b)), we generate the optimal solution path hindsight, where we regard the state with the highest value in the next episode ($\mathcal{G}_0^{e+1}$ in (c)) as the terminal state, and shortest path to it as the optimal solution path ($< s_0, s_1, s_3, s_6, s_9 >$ in (d)). We define the optimal goal as the reachable terminal state in the optimal solution path (*e.g.*, s_{10} in $\mathcal{G}_{\text{full}}$ and s_6 in $\mathcal{G}_0^e$).

as the goal, which can be formulated as

$$g_e = \arg\max_s V(s),\ \forall s \in \mathcal{C}_{ATT}. \tag{3}$$

By this way, we can generate appropriate goal g_e to guide the agent exploration in the episode e.

4.2 Learn with Graph Structured Reinforcement Learning

As illustrated in Figure 1, we update goal generation under supervised learning and value estimation under reinforcement learning. Specifically, we introduce learning strategies as follows:

Goal Learning Strategy. As mentioned above, we learn the goal generation hindsight. As illustrated in Figure 4, at the beginning timestep of each episode $e+1$, we first compute the shortest path distance between initial state (s_0 in (c)) and highest value state (s_9 in (c)) to obtain the solution path ($\mathcal{P}_{e+1} =< s_0, s_1, s_3, s_6, s_9 >$ in (d)). We then find the optimal goal g_e^* at episode e based on the solution path generated at episode $e+1$. Specifically, we search for the intersection state between $\mathcal{P}_{e+1}$ and $\partial\mathcal{G}_t^e$ in the inverse order of the solution path. Given that s_9 is unexplored and unreachable in the episode e, we select s_6 as the optimal goal g_e^*. We then find the optimal group $\mathcal{C}^*$ that contains goal g_e^*. With this hindsight supervision, we are able to update attention mechanism in Eq. (2) as

$$L_\phi = \mathbb{E}_{(s,a,g,s',r)\sim\mathcal{D}}\left[(\mathcal{C}^* - \mathcal{C}_{ATT})^2 + \alpha \cdot \|\phi\|_2\right], \tag{4}$$

where $\|\phi\|_2$ is regularizer and α is the corresponding hyper-parameter. This method updating goal generation under the supervision for the group instead of goal can eliminate instability brought from potentially inaccurate value estimation because our group division does not involve the result from value estimation.

Value Learning Strategy. With a generated goal at the beginning of each episode, we can build the critical block of our method, *i.e.*, a goal-conditioned policy and its associated value function. We consider a goal-reaching agent interacting with an environment. The agent observes its current state $s \in S$ and a goal state $g \in G$. The dynamics are governed by the distribution $P(s_{t+1}|s_t, a_t)$. At every timestep, the agent samples an action $a \sim \pi(a|s, g)$ and receives a corresponding reward $r_g(s, a)$ that indicates whether the agent has reached the goal. The episode terminates after T timesteps, even the agent reaches the goal. The agent's task is to maximize its cumulative and undiscounted reward. We use an off-policy algorithm to learn such a policy, as well as its associated goal-conditioned Q-function. For example, we obtain a policy by acting greedily, *w.r.t.*, the Q-function as

$$Q(s, a, g) \leftarrow r_g(s, a) + \gamma \cdot \max_{a'} Q(s, a', g) \tag{5}$$

In order to improve data efficiency and obtain good value estimation, we consider to choose off-policy RL algorithm with goal relabelling and update parameters with related trajectories. Specifically, we choose an off-policy RL algorithm with goal relabelling, such as Andrychowicz et al. (2017). We define related data as those trajectories that contain neighborhood nodes of the current

state. Formally, when we update Q-function of state s, we sample $\mathcal{D}_{\text{related}} := \{\tau\}$ from replay buffer $\mathcal{D}$, where τ denotes the trajectory that contains at least one state in s's neighborhood $\mathcal{N}^1(s)$. The Q-network is learned by minimizing the following objective function as

$$L_\theta = \mathbb{E}_{(s,a,g,s',r)\sim\mathcal{D}_{\text{related}}}\left[\left(r_g + \gamma \cdot \max_a Q_\theta(s',a,g) - Q_\theta(s,a,g)\right)^2 + \beta \cdot \|\theta\|_2\right], \tag{6}$$

where β is the weight of the regularization term. Besides, as we show in Proposition 3, our GSRL algorithm can converge to a unique optimal point, if Q-learning strategy is adopted. The proof is shown in Appendix C.3.

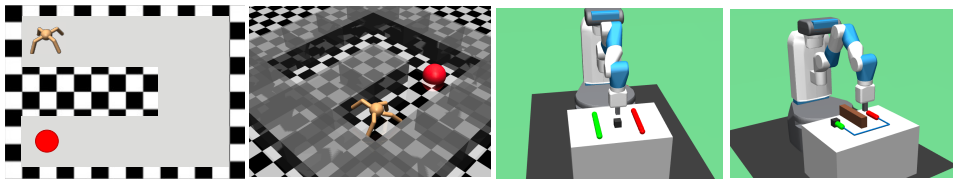
Proposition 3. *Denote the Bellman backup operator in Eq. (5) as $\mathcal{B} : \mathbb{R}^{|S|\times|A|\times|G|} \to \mathbb{R}^{|S|\times|A|\times|G|}$ and a mapping $Q : S \times A \times G \to \mathbb{R}^{|S|\times|A|\times|G|}$ with $|S| < \infty$ and $|A| < \infty$. Repeated application of the operator $\mathcal{B}$ for our graph-based state-action value estimate $\hat{Q}_{\mathcal{G}}$ converges to a unique optimal value $\hat{Q}^*_{\mathcal{G}^*}$ with well explored graph $\mathcal{G}^*$ including optimal solution path.*

Overall Algorithm. We provide the overall algorithm in Algorithm 1 in Appendix B. One can see that our state-transition graph can be applied to both the learning and planning phase. Previous graph-based RL algorithms either focus on learning (Zhu et al., 2019) or planning (Eysenbach et al., 2019) based on structured information of the state-transition graph.

5 Experiments

In this section, we perform an experimental evaluation of our algorithm and compare it with other state-of-the-art methods. Our experiment environments are based on the standard robotic manipulation environments in the OpenAI Gym (Brockman et al., 2016). We provide the experimental results to answer the following questions:

1. Can GSRL obtain better convergence in various environments?
2. Can GSRL tackle a high-dimensional and continuous environment with obstacle?
3. Can GSRL perform higher sample efficiency?



(a) Maze (b) Ant Maze (c) FetchPush (d) FetchPush with Obstacle

Figure 5: Visualization of robotic manipulation environments.

In order to answer the first question, we demonstrate our method in various robotic manipulation tasks including reach task (see Figure 5(a)) and fetch task (see Figure 5(c)). To answer the second one, we investigate how our method performs on environment with high-dimensional state and action space (see Figure 5(b)) and obstacle (see Figure 5(d)). We answer the third one by comparison on the convergence of learning curves in Figure 6 and further analysis on sample efficiency in Appendix E.4. Specifically, we conduct experiments with previous approaches:

- **HER**: Andrychowicz et al. (2017) generated imaginary goals in a simple heuristic way to tackle sparse reward issue.
- **MAP**: Huang et al. (2019) explicitly modeled 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.

One should be noted that graph structure in GSRL is constructed on top of replay buffer for goal generation and value estimation, which can be closely incorporated with policy networks such as DQN (Mnih et al., 2013), DDPG (Lillicrap et al., 2015), *etc*. To demonstrate the real performance gain from our GSRL, we set the policy network with DDPG for GSRL and all the baselines. The detailed description of environments, experiment settings, and implementation details can be found in Appendix E.1.

Maze. We first test our method and other strong baselines in the Maze environment, 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 as shown in Figure 5(a). This environment is quite simple, where most RL approaches can converge with a success rate 1. As Figure 6(a) illustrates, GSRL performs with high sample efficiency.

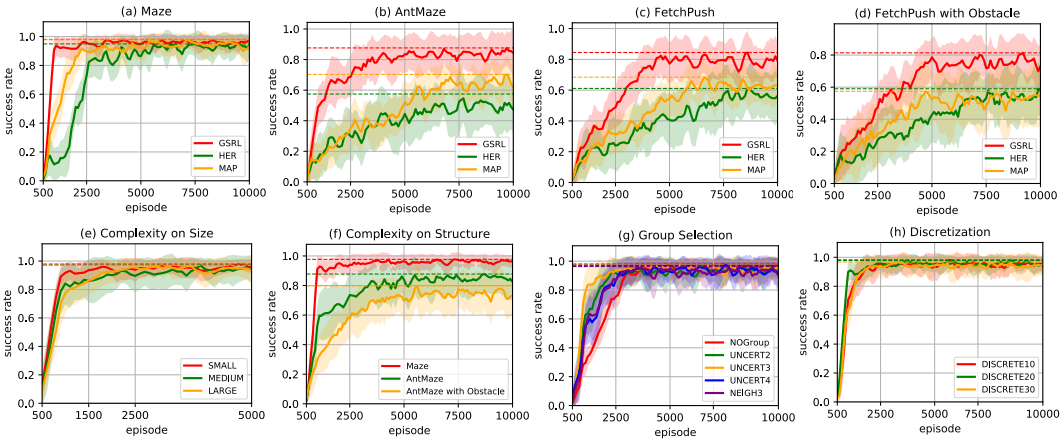


Figure 6: Learning curves of GSRL, HER and MAP on various environments with 10 random seeds, where the solid curves depict the mean, the shaded areas indicate the standard deviation, and dashed horizontal lines show the asymptotic performance.

AntMaze. We show that our GSRL is efficient at a complex environment of robotic agent navigation task, as illustrated in Figure 5(b), where the state space is of 8 dimensions and the action space is of 30 dimensions. Duan et al. (2016) showed that standard RL methods are unable to solve it. As shown in Figure 6(b), GSRL achieves the best asymptotic performance and sample efficiency.

FetchPush. As shown in Figure 5(c), in the fetch environment, the agent is trained to fetch an object from the initial position (rectangle in green) to a distant position (rectangle in red). Although the fetch tasks are more complicated than they reach ones in the maze, GSRL also yields large performance gain, as shown in Figure 6(c).

FetchPush with Obstacle. As Figure 5(d) illustrates, we create an environment based on FetchPush with a rigid obstacle, where the brown block is a static wall that cannot be moved. Experimental results in Figure 6(d) suggests that the graph structure of state-transitions can provide additional useful information, which leads to better guidance for goal generation and value estimation.

Impact of Environment Complexity. In order to investigate whether GSRL can be well adapted in the environments with different complexity levels, we extend the Maze environment, as shown in Figure 5(a) in two aspects: one is to extend the maze's size, resulting in more sparse rewards, the other is to extend the environment with a more complex or high-dimensional structure. More details of these environments are available in Appendix E.1. We report the performance comparisons on the three environments with small, medium, and large sizes in Figure 6(e). We create AntMaze with Obstacle environment, where we extend AntMaze environment with an obstacle. The performance of GSRL in this environment is illustrated in Figure 6(f). One can see that GSRL can be well adapted in various environments with different complexity levels.

Impact of Group Selection. We provide two strategies to divide the boundary of the graph into several groups, namely *extending from neighbor nodes* and *extending from uncertain nodes* in Section 4. We demonstrate the performance of these two strategies in Figure 6(g). Results show that GSRL without any group selection would lead to inefficiency, since the goal generation strategy at the beginning is almost random. Both of these two strategies perform well as illustrated. We also provide complexity analysis in Appendix D.3. In the main experiment, we adopt *extending from uncertain nodes* with 3 groups.

Hyper-parameter Study. We apply K-bins discretization technique (Kotsiantis & Kanellopoulos, 2006) to discretize the continuous state and action spaces, where there is a wrapper that converts a Box observation into a single integer. We find that for the simple tasks, the choice of K is not critical, and show the performance in Figure 6(h). K is set at 20 in the main experiment.

6 CONCLUSION

In this paper, we propose a novel framework called GSRL, which leverages structure information of the state-transition graph for efficient goal generation and value estimation. We provide theoretical analysis to show the efficiency and converge property of our method. Results on various challenging robotic manipulation environments demonstrate that GSRL can outperform the state-of-the-art RL algorithms. In the future, this work would shed light on graph-structured RL for efficient learning and planning, where various graph-based algorithms can help RL agents learn and infer in highly structured environments.

REFERENCES

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

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *NeurIPS*, 2016.

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

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

Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv*, 2016.

Matthew Botvinick, Sam Ritter, Jane X Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 2019.

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

Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. In *ICLR*, 2017.

Hao-Tien Lewis Chiang, Aleksandra Faust, Marek Fiser, and Anthony Francis. Learning navigation behaviors end-to-end with autorl. *RA-L*, 2019.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *NeurIPS*, 2019.

Rasool Fakoor, Pratik Chaudhari, and Alexander J Smola. P3o: Policy-on policy-off policy optimization. In *UAI*, 2020.

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

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal-conditioned policies. In *ICLR*, 2019.

Itzhak Gilboa and David Schmeidler. Case-based decision theory. *The quarterly Journal of economics*, 1995.

Steven Hansen, Alexander Pritzel, Pablo Sprechmann, André Barreto, and Charles Blundell. Fast deep reinforcement learning using online adjustments from the past. In *NeurIPS*, 2018.

Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. In *NeurIPS*, 2019.

Jiarui Jin, Ming Zhou, Weinan Zhang, Minne Li, Zilong Guo, Zhiwei Qin, Yan Jiao, Xiaocheng Tang, Chenxi Wang, Jun Wang, et al. Coride: Joint order dispatching and fleet management for multi-scale ride-hailing platforms. In *CIKM*, 2019.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *ICML*, 2018.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *ICLR*, 2020.

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

Su Young Lee, Choi Sungik, and Sae-Young Chung. Sample-efficient deep reinforcement learning via episodic backward update. In *NeurIPS*, 2019.

Máté Lengyel and Peter Dayan. Hippocampal contributions to control: the third way. In *NeurIPS*, 2008.

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.

Zichuan Lin, Tianqi Zhao, Guangwen Yang, and Lintao Zhang. Episodic memory deep q-networks. *IJCAI*, 2018.

Jiayuan Mao, Honghua Dong, and Joseph J Lim. Universal agent for disentangling environments and tasks. In *ICLR*, 2018.

David Marr, David Willshaw, and Bruce McNaughton. Simple memory: a theory for archicortex. In *From the Retina to the Neocortex*. 1991.

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

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

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *ICRA*, 2018.

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

Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *CVPR Workshops*, 2018.

Sujoy Paul, Jeroen Vanbaar, and Amit Roy-Chowdhury. Learning from trajectories via subgoal discovery. In *NeurIPS*, 2019.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *ICML*, 2017.

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

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *ICML*, 2018.

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

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

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 2016.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks. In *ICML*, 2018.

Robert J Sutherland and Jerry W Rudy. Configural association theory: The role of the hippocampal formation in learning, memory, and amnesia. *Psychobiology*, 1989.

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

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *ICLR*, 2018.

Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *NeurIPS*, 2017.

Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. In *International Conference on Learning Representations*, 2020.

Zhenjia Xu, Zhijian Liu, Chen Sun, Kevin Murphy, William T Freeman, Joshua B Tenenbaum, and Jiajun Wu. Unsupervised discovery of parts, structure, and dynamics. In *ICLR*, 2019.

Jiachen Yang, Alireza Nakhaei, David Isele, Kikuo Fujimura, and Hongyuan Zha. Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning. In *ICLR*, 2020.

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