PLUG-AND-PLAY GRAPH-BASED INTRINSIC REWARD FOR EXPLORATION OF COST-AWARE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Introducing action costs increases the difficulty of exploration in complex, sparse-reward state spaces for reinforcement learning (RL). To address this challenge, we propose a novel graph-based reward shaping method, termed Graph Intrinsic Reward (GIR), which uses graph to model the structure of explored state space to enable efficient exploration-exploitation in cost-aware RL. Specifically, GIR incrementally constructs a graph in which each node maintains both visit counts and value estimates, from which a count-based exploration reward and a potential-based exploitation reward are derived to guide the agent in escaping initial suboptimal states under costly actions. Through reward-gated modulation, GIR reduces ineffective exploration, while potential-based value propagation over the graph guides the agent toward high-return regions. We further analyze the impact of GIR on policy optimality and show that, from a theoretical perspective, GIR asymptotically preserves the optimal policy. Extensive experiments demonstrate that the proposed reward shaping strategy effectively guides exploration towards goal states and outperforms state-of-the-art exploration methods in cost-aware environments. Our implementation is available at `https://anonymous.4open.science/r/GIR-48BD/`.

1 INTRODUCTION

Reinforcement learning (RL) has achieved remarkable success in solving sequential decision-making problems Lei et al. (2024); Lu et al. (2025). However, exploration remains a fundamental challenge in environments characterized by *sparse rewards* and *costly actions*, which is a common feature of practical applications Suttle et al. (2021); Zhang et al. (2023); Devidze et al. (2022). In such settings, each action incurs a penalty, which discourages the agent from visiting unknown regions of the state space Cao et al. (2024). As a result, the agent often converges prematurely to suboptimal policies that avoid cost but fail to discover trajectories leading to high long-term returns Ladosz et al. (2022). This challenge is further aggravated when reward signals are sparse, providing limited guidance during early training.

To address exploration inefficiency, a wide range of intrinsic motivation methods have been proposed. Count-based approaches encourage visitation of novel states by assigning higher rewards to infrequently visited regions Ostrovski et al. (2017). Curiosity-driven methods leverage prediction errors of learned dynamics models as intrinsic rewards, encouraging exploration of uncertain transitions Pathak et al. (2017); Yang et al. (2025). Additionally, methods like *Go-Explore* Adrien Ecoffet et al. (2021) explicitly revisit promising state to improve exploration efficiency, while hindsight experience replay Andrychowicz et al. (2017); McCarthy et al. (2021) enhances sample efficiency in sparse-reward settings.

Despite these advances, existing exploration strategies exhibit notable limitations in *cost-aware* RL. Most methods treat exploration independently of action cost, leading to behaviors that either over-explore without regard to incurred penalties or under-explore due to immediate cost sensitivity. Furthermore, many approaches rely on local signals, including state novelty and prediction error, without capturing the global structure of the explored state space Ostrovski et al. (2017); Mahankali et al. (2024); Chen et al. (2021). This limits their ability to guide exploration toward regions that are

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

not only novel but also *potentially valuable* in terms of long-term returns Nair et al. (2018); Forbes et al. (2024). Consequently, the agent may fail to discover high-reward trajectories that require multi-step exploration.

In this work, we propose **Graph Intrinsic Reward** (GIR), a novel training-free reward shaping framework that explicitly models the structure of the explored state space using a dynamically constructed graph. In GIR, each node corresponds to a state representation augmented with visitation statistics and value estimates, while edges encode observed transitions. Based on this graph, we derive two complementary intrinsic reward signals: (i) a count-based exploration reward that encourages visiting under-explored regions, and (ii) a potential-based exploitation reward that propagates value over the graph to guide the agent toward high-return areas.

From a theoretical perspective, we analyze the effect of GIR within the framework of reward shaping and show that the proposed method preserves the optimal policy. Empirically, GIR can be seamlessly integrated into a wide range of RL algorithms and improves performance in challenging cost-aware environments with sparse rewards. Our experiments on *MountainCar* Moore (1990), *PointMaze* Fu et al. (2020), *MiniGrid* Chevalier-Boisvert et al. (2023) environments demonstrate that leveraging the structural information of the state space enables more efficient and stable exploration compared to existing intrinsic motivation methods.

In summary, this paper makes the following contributions:

- We propose a graph-based reward shaping framework that captures the structural and value information of the explored state space for cost-aware RL.
- We introduce a unified intrinsic reward formulation named GIR that combines count-based exploration with potential-based value propagation.
- We provide theoretical analysis and empirical experiments to validate the effectiveness and scalability of GIR.

2 PRELIMINARIES

2.1 NOTATIONS

We consider a discounted Markov decision process (MDP) that formalizes the cost-aware industrial decision-making problem. An MDP is defined by the tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, r_{env}, γ). Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a directed graph with a set of vertices $\mathcal{V}$ and a set of directed edges $\mathcal{E}$. The size of a node, denoted n_v, counts how many state samples have been assigned to that node. The adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ stores the smoothed expected immediate rewards for transitions from node i to node j. From the graph we also compute a node potential Φ_v, which estimates the maximum expected returns achievable from node v via multi-step transitions in graph $\mathcal{G}$.

2.2 DEEP REINFORCEMENT LEARNING

RL provides a mathematical framework for sequential decision-making under uncertainty, where an agent learns an optimal policy through interactions with the environment. In recent years, the integration of deep neural networks has given rise to deep RL, which has achieved remarkable success.

Deep RL algorithms can be broadly categorized into three families, including value-based methods, policy-based methods, and actor-critic methods. Value-based methods Mnih et al. (2015); Van Hasselt et al. (2016); Wang et al. (2016) learn an optimal action-value function $Q^*(s, a)$ and derive the policy implicitly by selecting actions that maximize the estimated Q-value. Policy-based methods Schulman et al. (2017); Shao et al. (2024) directly parameterize and optimize the policy $\pi_\theta(a|s)$ using gradient ascent on the expected return. Actor-critic (AC) methods Mnih et al. (2016); Haarnoja et al. (2018); Lillicrap (2015); Fujimoto et al. (2018) combine the strengths of both approaches, maintaining an actor that represents the policy and a critic that estimates the value function to reduce gradient variance; these methods achieve state-of-the-art performance on a variety of tasks.

2.3 INTRINSIC MOTIVATION

Intrinsic motivation augments the environment reward r_{env} with an auxiliary reward signal to encourage efficient exploration, especially in environments with sparse or delayed feedback. Formally, the agent is trained using a shaped reward:

$$r(s, a, s') = r_{env}(s, a, s') + \beta r_{int}(s, a, s'), \tag{1}$$

where r_{int} denotes the intrinsic reward and β controls the trade-off between exploration and exploitation.

A large body of work has explored different forms of intrinsic rewards. Count-based methods assign higher rewards to infrequently visited states, typically using state visitation counts or their approximations in high-dimensional spaces Ostrovski et al. (2017); Burda et al. (2019). These methods encourage exploration by prompting state novelty, but they do not account for the long-term utility of visited states. Curiosity-driven approaches define intrinsic rewards based on prediction errors of learned forward or inverse dynamics models, motivating the agent to explore transitions that are difficult to predict Pathak et al. (2017); Durugkar et al. (2021). Although effective in some domains, these methods rely on local signals and still lack an explicit mechanism to capture the global structure of the explored state space. This limitation is particularly critical in cost-aware settings, where exploration incurs penalties.

3 DYNAMIC STATE TRANSITION GRAPH

The core of our method is a dynamically evolving directed graph $\mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t\}$ that stores online interaction history, where each node $v \in \mathcal{V}_t$ represents a region of the state space, and each directed edge $e_{uv} \in \mathcal{E}_t$ denotes the expected immediate reward of the transition from region u to region v. The graph is constructed and updated incrementally as the agent collects experience from environment or samples interactions from the replay buffer Eysenbach et al. (2019).

3.1 NODE CREATION AND UPDATE

We construct the node set $\mathcal{V}$ via an online clustering process over observed states. Each node $v \in \mathcal{V}$ represents a cluster that summarizes a subset of the state space, which can be maintained depending on the choice of clustering method, for example, incremental k-means, Welford's update algorithm Efanov et al. (2021), and so on.

Given a new state s_t, it is either assigned to an existing node or used to initialize a new node according to a similarity criterion. Let $D_v(s_t)$ denote a distance (or divergence) between s_t and node v, which can be defined by the underlying clustering algorithm (e.g. Euclidean distance, Mahalanobis distance, or learned metrics). If $\min_{v \in \mathcal{V}} D_v(s_t) \leq \tau$, where τ is a predefined threshold, the state is assigned to the closest node $v^* = \arg\min_v D_v(s_t)$. Otherwise, a new node is created.

The update of node is algorithm-agnostic and depends on the chosen incremental clustering scheme. In general, the update can be written as Equation 2.

$$v_{t+1} \leftarrow \mathcal{U}(v_t, s_t) \tag{2}$$

where $\mathcal{U}(\cdot)$ denotes an arbitrary online update operator. This formulation allows the use of a wide range of incremental clustering methods, enabling flexibility in how the state space is represented.

3.2 EDGE WEIGHT UPDATE

For each transition (s_t, a_t, s_{t+1}) with associated environment reward r_{env}, we first identify the corresponding nodes $i = node(s_t)$ and $j = node(s_{t+1})$. The edge weight A_{ij} in the adjacency matrix is then updated via an exponential moving average as Equation 3 shows.

$$A_{ij} \leftarrow \alpha \cdot r_{env}(s, a, s') + (1 - \alpha) A_{ij}, \tag{3}$$

where $\alpha \in (0, 1)$ is a smoothing factor, and r_{env} is the environment reward. If the edge does not exist, it is initialized as $A_{ij} = r_{env}(s, a, s')$. Moreover, the adjacency matrix is dynamically resized as new nodes are added.

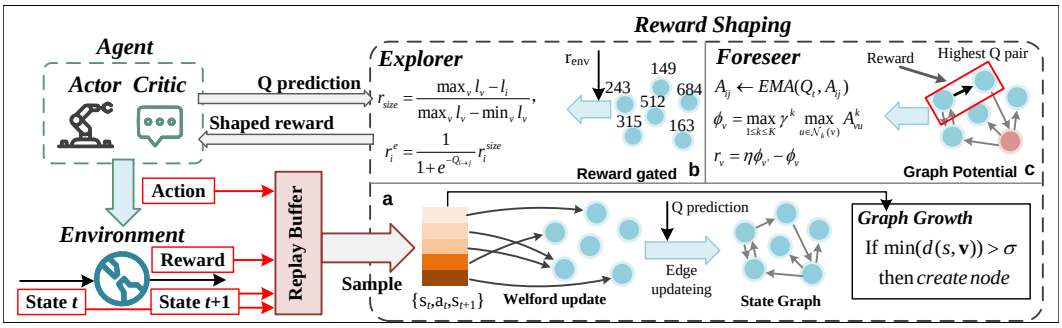


Figure 1: Overall framework of the proposed reward shaping method. (a) Dynamic state transition graph construction; (b) Reward-gated exploration reward; (c) Graph potential reward.

The dynamic graph thus serves as an adaptive memory that continuously refines its representation of the state space and transition rewards, providing the foundation for the intrinsic rewards. The overall algorithm is shown in Algorithm 1.

4 Graph Intrinsic Reward

The overall framework of GIR, as shown in Figure 1, includes the aforementioned three components. At each training step, the agent interacts with the environment and stores the transition with a graph $\mathcal{G}$. Then, it computes the two intrinsic rewards, including the reward-gated exploration reward and the graph potential reward.

Algorithm 1 Dynamic State Transition Graph Update

Require: $\mathcal{G} = (\mathcal{V}, A)$ with node statistics $\{\theta_v\}$; transition (s, s', r_{env}); threshold τ; smoothing α; clustering operator $\mathcal{U}$
Ensure: Updated $\mathcal{G}$

Node assignment and update:
for $x \in \{s, s'\}$ **do**
  if $\mathcal{V} = \emptyset$ **or** $\min_{v \in \mathcal{V}} D_v(x) > \tau$ **then**
    Create new node v with $\theta_v \leftarrow \textsc{Init}(x)$
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{v\}$
  else
    $v \leftarrow \arg\min_{u \in \mathcal{V}} D_u(x)$
    $\theta_v \leftarrow \mathcal{U}(\theta_v, x)$
Edge update:
$(i, j) \leftarrow$ node indices of (s, s')
$w \leftarrow r_{env}$
if $(i, j) \in A$ **then**
  $A_{ij} \leftarrow \alpha w + (1 - \alpha) A_{ij}$
else
  $A_{ij} \leftarrow w$

4.1 Reward-Gated Exploration Reward

The node-size exploration reward is designed to address the exploration trap that commonly arises in cost-sensitive industrial tasks, where agents tend to converge to regions with low immediate costs, even when those regions lead to suboptimal returns. By rewarding visitation of under-explored areas, the node-size exploration reward encourages the agent to visit under-explored regions of the state space, preventing early convergence to low-cost local optima. For a transition from state s to next state s', let $i = node(s')$ denote the node of the next state. The exploration reward is defined as follows:

$$r_{expl}(s, a, s') = \lambda_t \cdot \sigma(r_{env}(s, a)) \cdot r_{size}(s'), \tag{4}$$

where λ_t is a non-increasing exploration schedule with $\lambda_t \to 0$ as $t \to \infty$ (we use $\lambda_t = 1/\sqrt{1+t}$, with t the number of environment steps), $\sigma(\cdot)$ is the sigmoid function $\sigma(r_{env}) = \frac{1}{1+e^{-r_{env}(s,a)}}$, and $r_{size}(s')$ is the node-size reward computed from visit counts as Equation 5 shows.

$$r_{size}(s') = \beta_{size} \cdot \frac{\max_v \log(1 + n_v) - \log(1 + n_i)}{\max_v \log(1 + n_v) - \min_v \log(1 + n_v) + \epsilon} \tag{5}$$

where $\beta_{size} > 0$ is a scaling factor, n_v denotes the visit count of node v, and ϵ is a small constant to prevent division by zero. The logarithmic transformation compresses the count range, making the

bonus sensitive to relative differences in low-count regions; the schedule λ_t then anneals the bonus as training progresses. The normalization ensures the reward lies in [0, β_{size}].

The reward gate $\sigma(r_{env}(s, a))$ acts as a cost-aware filter that dampens the exploration bonus on transitions with low immediate reward (i.e., high action cost), preventing novelty-seeking from incurring excessive penalties. Moreover, the sigmoid function provides a smooth, bounded weighting that remains stable throughout learning, avoiding the instability that might arise from using raw rewards directly.

4.2 Graph Potential Reward

A graph potential reward is introduced, which leverages the multi-step transition structure encoded in the dynamic graph. This reward encourages the agent to move toward regions of the state space that have high potential for future returns, effectively serving as a form of value propagation across the graph to avoid unwanted exploration.

From the current adjacency matrix, we precompute a node potential Φ_v for each node, which estimates the maximum discounted return achievable from node v within a finite horizon. The potential is obtained by running value iteration over the graph for K steps, as Equation 6 shows.

$$\Phi_v = V_v^{(K)}, \qquad V_v^{(0)} = 0, \qquad V_v^{(k)} = \max_{u \in \mathcal{N}(v)} \big(A_{vu} + \gamma\, V_u^{(k-1)}\big), \tag{6}$$

where $\mathcal{N}(v)$ denotes the set of nodes reachable from v in one step, K is the number of propagation steps, and γ is the discount factor. The potential Φ_v thus estimates the maximum discounted return achievable from region v by looking ahead up to K transitions.

The graph potential reward for a state transition (s, a, s') with corresponding nodes $i = node(s)$ and $j = node(s')$ is defined as Equation 7 shows.

$$r_{pot}(s, a, s') = \beta_{pot} \cdot \big(\gamma \cdot \Phi_j - \Phi_i\big), \tag{7}$$

where $\beta_{pot} > 0$ is a scaling factor. The graph potential reward serves as a coarse-grained guidance for long-term return expectations, which mitigates the instability of the exploration reward and prevents performance degradation in the later stage of training.

4.3 Asymptotic Optimality

We now show that GIR asymptotically preserves the optimal policy of the original MDP, with the complete derivation deferred to Appendix A. We assume that the visitation distribution over transitions converges to a stationary limit (a standard ergodicity condition), so that the edge-weight moving averages have well-defined limits; we then establish that the *limiting* MDP induced by GIR shares the optimal policy of the original MDP, while convergence of the online learner to that optimum is inherited from the base RL algorithm.

The argument proceeds in three steps. First, since $r_{size}(s')$ in Eq. 5 is bounded in $[0, \beta_{size}]$ and the exploration schedule satisfies $\lambda_t \to 0$, the exploration reward decays to zero:

$$0 \;\le\; r_{expl}^{(t)}(s, a, s') \;=\; \lambda_t\, \sigma(r_{env})\, r_{size}^{(t)}(s') \;\le\; \lambda_t\, \beta_{size} \;\to\; 0. \tag{8}$$

Second, each edge weight A_{ij} follows the exponential moving average in Eq. 3, so $A_{ij}^{(t)} \xrightarrow{a.s.} A_{ij}^* = \mathbb{E}[r_{env}]$ and the adjacency matrix converges to a fixed limit A^*. Because the value-iteration mapping $A \mapsto \Phi$ in Eq. 6 is continuous, the node potentials converge as well: $\Phi_v^{(t)} \to \Phi_v^* = g(A^*)_v$.

Third, in the limit the total intrinsic reward reduces to a potential-based shaping term:

$$r_{expl}^{(t)}(s, a, s') + r_{pot}^{(t)}(s, a, s') \;\xrightarrow{a.s.}\; \beta_{pot}\big(\gamma\, \Phi_j^* - \Phi_i^*\big). \tag{9}$$

By the classic result on potential-based reward shaping Ng et al. (1999), a shaped reward of the form $r_{env} + \gamma\Psi(s') - \Psi(s)$ preserves the optimal policy for any potential Ψ. Taking $\Psi = \beta_{pot}\Phi^*$ (the constant factor β_{pot} does not affect policy invariance) and defining $Q^\dagger(s, a) = Q_\infty^*(s, a) + \beta_{pot}\Phi_s^*$, substituting into the Bellman optimality equation of the shaped MDP recovers the original Bellman equation; hence $\arg\max_a Q_\infty^*(s, a) = \arg\max_a Q_{env}^*(s, a)$. Therefore GIR asymptotically preserves the optimal policy of the cost-aware MDP.

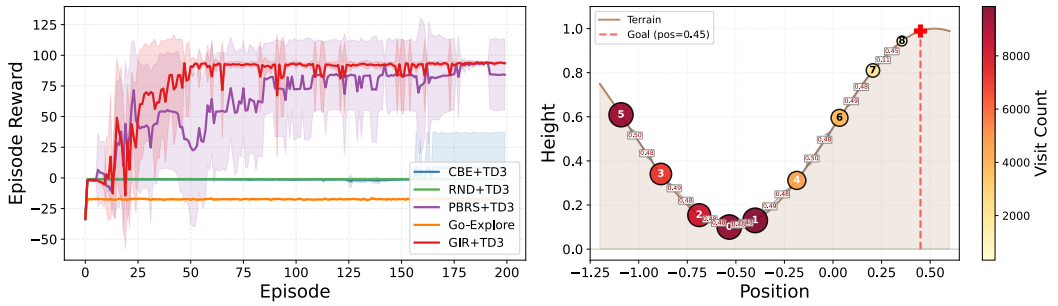


Figure 2: (**Left**) **Learning progress.** Lines show the cumulative reward averaged over 10 experiments (shaded area shows mean±SD). Only PBRS and our method (GIR+TD3) allow the agent to discover the goal in the experiments. However, PBRS+TD3 is obviously unstable compared to our method. (**Right**) **Graph structure.** Colored points and directed edges illustrate the final graph structure. The color represents the visit count of nodes and the value of an edge denotes the value of the state transition. Since the agent needs to accumulate velocity in the valley, the graph edges in the valley significantly exceed those near the goal.

5 Experiments

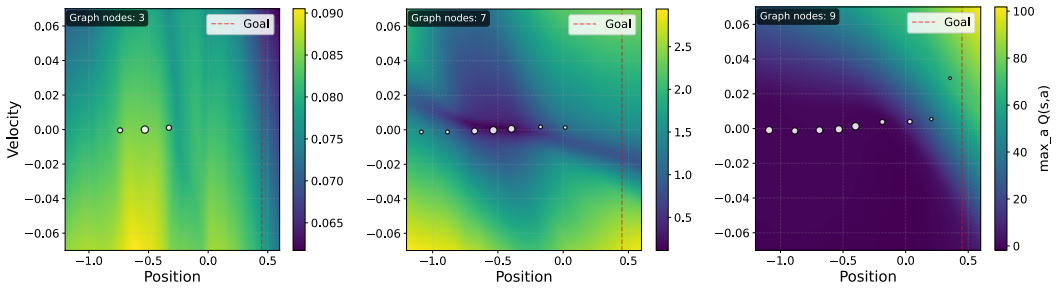
We empirically evaluate GIR on a suite of cost-aware reinforcement learning benchmarks to assess its effectiveness as a structured reward shaping mechanism. Our experiments are designed to validate both the efficiency of exploration and the quality of the learned policies, as well as the generality and modularity of GIR as a plug-and-play component. In particular, we mainly answer the following questions:

- Does GIR speed up the exploration process in a cost-aware environment?
- Does GIR converge to a high-return policy in cost-aware environments?
- Do the individual modules of GIR contribute to learning?

To answer these questions, we integrate GIR with multiple representative RL algorithms and evaluate performance across diverse environments with varying degrees of sparsity and cost sensitivity. We report learning curves, final performance, and ablation studies to isolate the contributions of each component in the proposed framework. The additional experiments and their configurations can be found in the Appendix B.

To benchmark GIR against established exploration strategies, we conduct comparative experiments on three representative cost-aware environments: *MountainCar* Moore (1990), *PointMaze* Fu et al. (2020), and *MiniGrid* Chevalier-Boisvert et al. (2023). We report *MountainCar* and *PointMaze* in this section, and defer *MiniGrid*, which tests GIR on discrete pixel-based observations, to Appendix C.5. We compare against four widely used baselines, including:

- **Random Network Distillation (RND Burda et al. (2019)):** RND encourages exploration by rewarding states with high prediction error from a fixed random target network, serving as a scalable novelty signal.
- **Count-based Exploration (CBE) Ostrovski et al. (2017):** Assigns intrinsic rewards based on state visitation frequency, promoting exploration of rarely visited regions via explicit or approximated counts.
- **Potential-based Reward Shaping (PBRS) Forbes et al. (2024):** Adds a shaping term defined by a potential function over states, providing dense guidance while theoretically preserving the optimal policy.
- **Go-Explore Adrien Ecoffet et al. (2021):** Improves exploration by explicitly storing and revisiting promising states before systematically exploring from them, enabling efficient discovery of sparse rewards.



(a) (b) (c)

Figure 3: **Q-value heatmaps** for agents trained for 0(a), 20(b) and 100(c) episodes in the MountainCar environment along with the corresponding graph node locations. In the early stage of training, smaller graph nodes correspond to higher Q-values. As training progresses, the visitation counts increase and the exploration schedule λ_t decays, leading to a decrease in their Q-values, and the agent becomes inclined to visit smaller and more promising nodes. In the later stage of training, the influence of graph node Q-values diminishes, and the agent converges to the optimal policy.

MountainCar As a toy benchmark and illustrative example, the environment is a classical control task that naturally exhibits sparse rewards and action costs. The agent must repeatedly accelerate back and forth to accumulate sufficient momentum to reach the goal state. However, due to action costs, the agent is prone to a degenerate local optimum in which it remains stationary to avoid penalties, making efficient exploration essential. We evaluate GIR on the continuous control variant using the representative actor-critic algorithm TD3 Fujimoto et al. (2018). Action costs are incorporated as a penalty proportional to the magnitude of the applied force. All methods share identical network architectures and training configurations where applicable.

The results are shown in Figure 2, and the change of area of interests guided by the GIR is shown in Figure 3 along with the corresponding graph node locations. In the early stage, the graph is sparse and low-visit nodes provide strong intrinsic rewards, encouraging broad exploration and helping the agent escape the stationary local optimum. As training progresses, visitation statistics become more reliable: the count-based reward suppresses frequently visited regions, the reward gate dampens high-cost transitions, and the potential term guides the agent toward high-value areas. In the later stage, the influence of intrinsic rewards diminishes and the policy converges to the optimal solution.

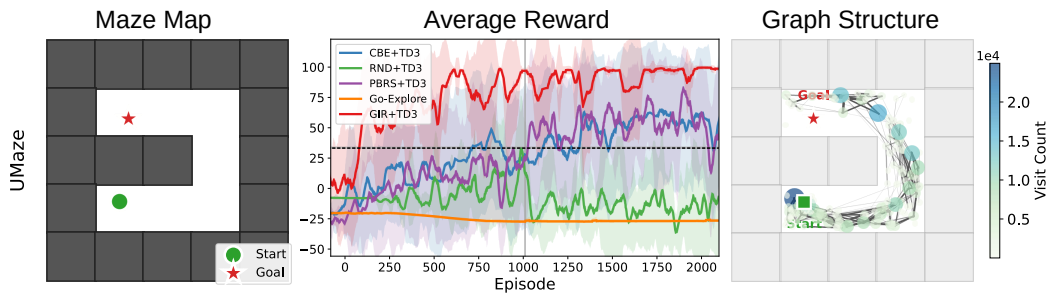
In the appendix, we present a more detailed training process and the mechanism of the GIR guidance, additional visualization results illustrating the behavior of GIR at different stages of training.

PointMaze The *PointMaze* environment is a navigation task with continuous control, where the agent must reach a goal position through a maze with obstacles under sparse rewards. We manually add action costs, which further increases the difficulty, as inefficient exploration leads to accumulated penalties without meaningful progress. In such settings, the agent must discover long-horizon trajectories that require coordinated movements while avoiding excessive exploration cost.

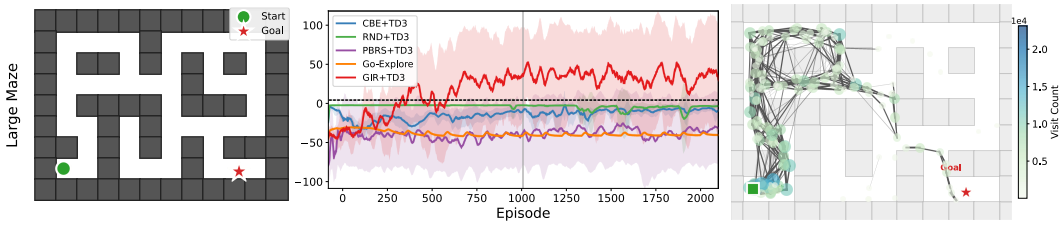
We evaluate GIR using TD3 under the same training configurations as in MountainCar. The agent receives a sparse reward only upon reaching the goal, and action costs are defined as penalties on control magnitude. Baseline methods include RND, CBE, PBRS, and Go-Explore. The learning processes on UMaze and large-scale maze environments are presented, while additional maze environments are provided in the appendix.

In Figure 4, experimental results show that GIR significantly improves exploration efficiency compared to baseline methods. Methods based on local novelty such as RND, CBE and PBRS tend to explore inefficiently in the maze, often revisiting irrelevant regions and incurring high costs and instability. Go-Explore fails to reach the goal and gets trapped in a local optimum with high cost.

In contrast, GIR leverages the graph structure to capture transition relationships between regions in the maze, enabling value information to propagate across distant states. This allows the agent to progressively focus on promising paths toward the goal. As a result, GIR achieves faster convergence



(a)



(b)

Figure 4: (**Left**) **Maze maps.** The top view of maze environments. Start and goal locations are determined by a fixed seed (provided in the appendix) to generate the hardest exploration task in both the UMaze and large maze. (**Middle**) **Learning progress.** Learning curves of averaged reward over 10 experiments (shaded area shows mean±SD). The exploration rate and stability in the cost-aware maze exploration task are clearly better for our method (GIR+TD3). (**Right**) **Graph structure.** The trajectory guided by the graph is illustrated. In the absence of the reward-gated mechanism and potential-based value, less-visited nodes with low potential remain unvisited in the final graph structure. This phenomenon highlights the effectiveness of these two mechanisms.

and higher success rates, demonstrating its advantage in structured, long-horizon exploration tasks under action costs.

The additional experimental results on all environments are provided in the Appendix C.

Ablation Study To better understand the contribution of each component in GIR, we conduct a series of ablation experiments by systematically removing or modifying key design elements.

(1) Effect of intrinsic reward components. We decompose GIR into two parts: the count-based exploration reward and the potential-based shaping reward. We evaluate three variants: (i) exploration-only (w/o potential), (ii) potential-only (w/o exploration), and (iii) full GIR. Results in Figure 5 show that removing either component leads to degraded performance. The exploration-only variant improves state coverage but suffers from unstable learning due to lack of long-term guidance, while the potential-only variant converges prematurely to suboptimal policies due to insufficient exploration. The full model achieves the best performance by balancing exploration and exploitation.

(2) Effect of reward-gated modulation. We remove the reward gate $\sigma(r_{env})$ and directly apply the intrinsic reward without cost-aware modulation. This results in Figure 5 in excessive exploration in low-value regions, significantly slowing down convergence. This demonstrates that the reward gate is critical for suppressing ineffective exploration under action costs.

(3) Effect of graph structure. To evaluate the importance of the graph abstraction, we replace the graph with a flat count-based representation that ignores transition structure. As shown in Figure 5, this variant fails to propagate long-range value information and performs significantly worse in sparse-reward environments. This confirms that the graph structure plays a key role in capturing transition dependencies.

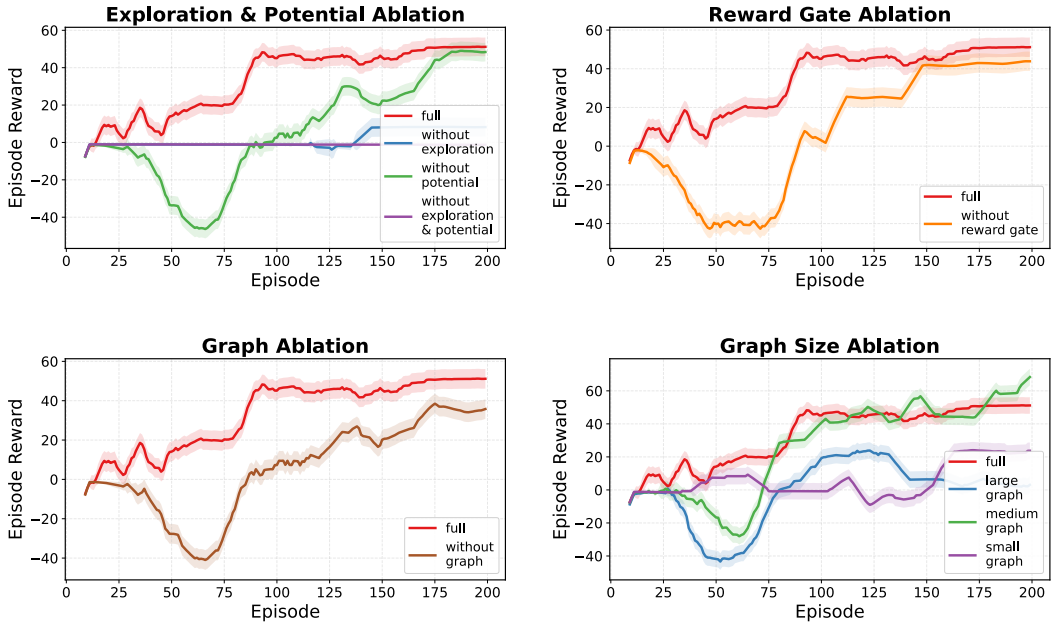


Figure 5: **(Ablation Study)** Learning curves of GIR and the reduced variants of it. The experimental results are obtained over 10 experiments. **(Upper Left)** Effect of the intrinsic rewards; **(Upper Right)** Effect of the reward gate; **(Lower Left)** Effect of graph structure; **Lower Right** Effect of graph size.

(4) Effect of node granularity. We vary the number of nodes by adjusting the clustering threshold. As shown in Figure 5, a very small number of nodes leads to insufficient exploration, while an excessively large graph increases noise and computational cost.

Overall, these results demonstrate that each component of GIR contributes to its effectiveness, and that the proposed design achieves a robust balance between exploration efficiency and learning stability.

Discussion and Future Work

In the work, we propose Graph Intrinsic Reward (GIR), a structured reward shaping framework for cost-aware reinforcement learning. By constructing a dynamic state transition graph, GIR unifies count-based exploration and potential-based value propagation within a single formulation. The proposed reward-gated mechanism further enables adaptive regulation of exploration under action costs.

Despite these advantages, several limitations remain. First, the scalability of the graph structure is a key concern. As the number of visited states grows, the number of nodes and edges may increase substantially, leading to higher computational and memory overhead.

Second, the current implementation relies on features extracted from a random network to measure state similarity of high-dimensional environments. While this approach is simple and empirically effective in several environments, it lacks interpretability and may not capture semantically meaningful structure in the state space.

Future work will focus on addressing these limitations by exploring scalable graph abstraction techniques, and integrating learned or structured feature embeddings for more interpretable state clustering. Another promising direction is to extend GIR to more complex settings, including offline RL, multi-agent systems, and real-world decision-making tasks with richer dynamics and constraints.

AI USE

In this work, we used generative AI tools for editing a research paper to improve readability, interpret results. We have not used generative AI tools to Generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ACKNOWLEDGEMENTS

REFERENCES

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021. ISSN 1476-4687. doi: 10.1038/s41586-020-03157-9.

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. *Advances in Neural Information Processing Systems*, 30, 2017.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458, 2017.

Yuri Burda, Harrison Edwards, A. Storkey, and Oleg Klimov. Exploration by random network distillation. *International Conference on Learning Representations*, 2019.

Buqing Cao, Hongfan Ye, Jianxun Liu, Bing Tang, Zhi Tao, and Shuiguang Deng. Smart: Cost-aware service migration path selection based on deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 25(9):12421–12436, 2024.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097, 2021.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA*, December 2023.

Rati Devidze, Parameswaran Kamalaruban, and Adish Singla. Exploration-guided reward shaping for reinforcement learning under sparse rewards. *Advances in Neural Information Processing Systems*, 35:5829–5842, 2022.

Ishan Durugkar, Mauricio Tec, Scott Niekum, and Peter Stone. Adversarial intrinsic motivation for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8622–8636, 2021.

Andrey A. Efanov, Sergey A. Ivliev, and Alexey G. Shagraev. Welford's algorithm for weighted statistics. In *2021 3rd International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, pp. 1–5, 2021. doi: 10.1109/REEPE51337.2021.9387973.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Grant C. Forbes, Nitish Gupta, Leonardo Villalobos-Arias, Colin M. Potts, Arnav Jhala, and David L. Roberts. Potential-based reward shaping for intrinsic motivation. In *International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '24, pp. 589–597, Richland, SC, 2024. ISBN 979-8-4007-0486-4.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870, 2018.

Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

Kun Lei, Peng Guo, Yi Wang, Jian Zhang, Xiangyin Meng, and Linmao Qian. Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning. *IEEE Transactions on Industrial Informatics*, 20(1):1007–1018, 2024. ISSN 1941-0050. doi: 10.1109/TII.2023.3272661.

Timothy P Lillicrap. Continuous control with deep reinforcement learning. *arXiv*, 2015.

Renzhi Lu, Zonghe Shao, Yuemin Ding, Ruijuan Chen, Dongrui Wu, Housheng Su, Tao Yang, Fumin Zhang, Jun Wang, Yang Shi, Zhong-Ping Jiang, Han Ding, and Hai-Tao Zhang. Discovery of the reward function for embodied reinforcement learning agents. *Nature Communications*, 16 (1):11064, 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-66009-y.

Srinath V. Mahankali, Zhang-Wei Hong, Ayush Sekhari, Alexander Rakhlin, and Pulkit Agrawal. Random latent exploration for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 34219–34252, 2024.

Robert McCarthy, Qiang Wang, and Stephen J. Redmond. Imaginary hindsight experience replay: Curious model-based learning for sparse reward tasks. https://arxiv.org/abs/2110.02414v2, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 1476-4687. doi: 10.1038/nature14236.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937. PMLR, 2016.

Andrew William Moore. Efficient memory-based learning for robot control. Technical report, University of Cambridge, 1990.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *International Conference on Robotics and Automation*, pp. 6292–6299, 2018. doi: 10.1109/ICRA.2018.8463162.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, pp. 278–287, 1999.

Georg Ostrovski, Marc G. Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International Conference on Machine Learning*, pp. 2721–2730, 2017.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pp. 2778–2787, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, abs/1707.06347, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

Wesley Suttle, Kaiqing Zhang, Zhuoran Yang, Ji Liu, and David Kraemer. Reinforcement learning for cost-aware markov decision processes. In *International Conference on Machine Learning*, pp. 9989–9999, 2021.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1995–2003. PMLR, 2016.

Shih-Min Yang, Martin Magnusson, Johannes A. Stork, and Todor Stoyanov. Kea: Keeping exploration alive by proactively coordinating exploration strategies in curiosity-driven exploration. *Machine Learning Research*, 267:70927–70942, 2025.

Jingwei Zhang, Long Cheng, Cong Liu, Zhiming Zhao, and Ying Mao. Cost-aware scheduling systems for real-time workflows in cloud: An approach based on genetic algorithm and deep reinforcement learning. *Expert Systems with Applications*, 234:120972, 2023.

A Optimality Analysis

We analyze the asymptotic effect of GIR on policy optimality. The key insight is that the graph-based intrinsic reward converges to a potential-based shaping term, which is known to preserve the optimal policy.

Assumptions. Throughout, we assume that the visitation distribution over graph edges converges to a stationary limit, so that the moving averages in Eq. 3 admit well-defined almost-sure limits. The result characterizes the optimal policy of the limiting shaped MDP; the convergence of the underlying online learner to that optimum is inherited from the base RL algorithm and is outside the scope of this analysis.

A.1 Decay of Exploration Reward

The node-size reward $r_{size}(s')$ in Eq. 5 is normalized to lie in $[0, \beta_{size}]$, and the exploration reward is further modulated by the schedule λ_t with $\lambda_t \to 0$ (Eq. 4). Since $\sigma(r_{env}) \in (0, 1)$, we obtain:

$$0 \;\le\; r_{expl}(s, a, s') \;=\; \lambda_t \, \sigma(r_{env}) \, r_{size}(s') \;\le\; \lambda_t \, \beta_{size} \;\to\; 0, \tag{10}$$

so the exploration reward vanishes almost surely as $t \to \infty$.

A.2 Convergence of Edge Weights

The edge weight A_{ij} follows an exponential moving average of environment rewards (Eq. 3). Expanding the recurrence gives:

$$A_{ij}^{(t)} = (1-\alpha)^{N_{ij}(t)} A_{ij}^{(0)} + \alpha \sum_{k=1}^{N_{ij}(t)} (1-\alpha)^{N_{ij}(t)-k} \, r_{env}^{(k)}, \tag{11}$$

where $N_{ij}(t)$ is the number of transitions assigned to edge (i, j) up to time t. Since $(1-\alpha)^{N_{ij}(t)} \to 0$ and the weighted average converges to the expectation, the edge weight converges almost surely:

$$A_{ij}^{(t)} \xrightarrow{a.s.} A_{ij}^{*} = \mathbb{E}_{(s,a,s') \sim d_{ij}^{*}} \big[r_{env}(s, a, s') \big], \tag{12}$$

where d_{ij}^{*} is the limiting visitation distribution over transitions from node i to j. Therefore, the adjacency matrix converges to a fixed limit A^{*} almost surely.

A.3 Convergence of Node Potentials

The node potential $\Phi_v = V_v^{(K)}$ is defined by the value-iteration mapping in Eq. 6. The mapping $g : A \mapsto \Phi$ is a composition of additions, scalar multiplications by γ, and $\max$ operators, all of which are continuous. By the continuous mapping theorem, $A^{(t)} \xrightarrow{a.s.} A^*$ implies:

$$\Phi_v^{(t)} \xrightarrow{a.s.} \Phi_v^* = g(A^*)_v. \tag{13}$$

Thus, node potentials converge to stationary values determined by the limiting graph structure.

A.4 Limiting Intrinsic Reward

Combining the above results, as $t \to \infty$:

- The exploration reward vanishes: $r_{expl}^{(t)} \to 0$ (Eq. 10).
- The node potentials converge: $\Phi^{(t)} \to \Phi^*$ (Eq. 13).

Consequently, the total intrinsic reward converges almost surely to a potential-based shaping term:

$$\lim_{t\to\infty} \big(r_{expl}^{(t)} + r_{pot}^{(t)}\big) = \beta_{pot}\big(\gamma\Phi_{s'}^* - \Phi_s^*\big), \qquad \text{a.s.} \tag{14}$$

where s and s' denote the current and next states of a transition, and $\beta_{pot} > 0$ is the constant potential scaling factor in Eq. 7.

A.5 Asymptotic Bellman Optimality

Consider the limiting total reward:

$$r_\infty(s, a, s') = r_{env}(s, a, s') + \beta_{pot}\big(\gamma\Phi_{s'}^* - \Phi_s^*\big). \tag{15}$$

Let Q_∞^* be the optimal Q-function under r_∞:

$$Q_\infty^*(s, a) = \mathbb{E}_{s'}\big[r_\infty(s, a, s') + \gamma \max_{a'} Q_\infty^*(s', a')\big]. \tag{16}$$

Define the transformed Q-function $Q^\dagger(s, a) = Q_\infty^*(s, a) + \beta_{pot}\Phi_s^*$. Substituting into Eq. 16:

$$\begin{aligned} Q^\dagger(s, a) &= \mathbb{E}_{s'}\big[r_{env} + \beta_{pot}\gamma\Phi_{s'}^* - \beta_{pot}\Phi_s^* + \gamma \max_{a'} Q_\infty^*(s', a')\big] + \beta_{pot}\Phi_s^* \\ &= \mathbb{E}_{s'}\big[r_{env} + \gamma \max_{a'} \big(Q_\infty^*(s', a') + \beta_{pot}\Phi_{s'}^*\big)\big] \\ &= \mathbb{E}_{s'}\big[r_{env}(s, a, s') + \gamma \max_{a'} Q^\dagger(s', a')\big]. \end{aligned} \tag{17}$$

Equation 17 is precisely the standard Bellman optimality equation for the original MDP with reward r_{env}. Hence $Q^\dagger = Q_{env}^*$, the optimal Q-function of the original MDP. Since $\beta_{pot}\Phi_s^*$ is independent of the action, $\arg\max_a Q_\infty^*(s, a) = \arg\max_a Q^\dagger(s, a)$, and the greedy policy derived from Q_∞^* coincides with the optimal policy of the original MDP.

Therefore, GIR asymptotically preserves the optimal policy: the limiting policy under the shaped reward is optimal for the original cost-aware MDP.

B Experimental Implementation Details

In this section, we provide the hyperparameters and implementation details of our algorithm along with the baseline methods for *MountainCar*, *PointMaze*, and *MiniGrid* environments.

B.1 GIR Pseudo-code

Below, we provide the pseudocode for GIR in Algorithm 2. The full implementation in python can be found at `https://anonymous.4open.science/r/GIR-48BD/`.

Algorithm 2 Graph-based Intrinsic Reward (GIR)

Phase 1: Online Graph Construction

for each transition (s_t, r_t, s_{t+1}) **do**
  $v_t \leftarrow \text{AssignOrCreateNode}(s_t)$ ▷ *incremental clustering*
  $v_{t+1} \leftarrow \text{AssignNode}(s_{t+1})$
  $w \leftarrow r_t$ ▷ *edge weight = environment reward*
  $\mathbf{A}[v_t, v_{t+1}] \leftarrow \alpha \cdot w + (1-\alpha) \cdot \mathbf{A}[v_t, v_{t+1}]$ ▷ *EMA update*

Phase 2: Node Potential via Value Iteration

$\mathbf{V}^{(0)} \leftarrow \mathbf{0}$
for $k = 1$ **to** K **do**
  $V_v^{(k)} \leftarrow \max_{u \in \mathcal{N}(v)} \left(A_{vu} + \gamma \cdot V_u^{(k-1)}\right),\ \forall v$ ▷ *propagate along outgoing edges*
$\mathbf{\Phi} \leftarrow \mathbf{V}^{(K)}$ ▷ *node potentials*

Phase 3: Intrinsic Reward

for each transition (s_t, r_t, s_{t+1}) **do**
  $v_t \leftarrow \text{AssignNode}(s_t), \quad v_{t+1} \leftarrow \text{AssignNode}(s_{t+1})$
  $n_v \leftarrow \text{VisitCount}(v)$

  Exploration reward:
  $\tilde{r}_{\text{expl}} \leftarrow \lambda_t \cdot \sigma(r_t) \cdot \beta_{\text{size}} \cdot \dfrac{\max_v \log(1+n_v) - \log(1+n_{v_{t+1}})}{\max_v \log(1+n_v) - \min_v \log(1+n_v) + \epsilon}$

  Potential-based shaping:
  $\tilde{r}_{\text{pot}} \leftarrow \beta_{\text{pot}} \cdot \left(\gamma \cdot \Phi_{v_{t+1}} - \Phi_{v_t}\right)$

  Total intrinsic reward:
  $r_t^{\text{int}} \leftarrow \tilde{r}_{\text{expl}} + \tilde{r}_{\text{pot}}$

B.2 Continuous Mountain Car

The *MountainCarContinuous* environment Moore (1990) is a classic continuous control benchmark with sparse rewards. A car is situated in a one-dimensional valley and must build up sufficient momentum to reach the goal at the top of the right hill.

State space. The state is two-dimensional,

$$s = (x, \dot{x}) \in [-1.2, 0.6] \times [-0.07, 0.07],$$

where x is the horizontal position and $\dot{x}$ is the velocity.

Action space. The action is a continuous scalar force

$$a \in [-1.0, 1.0],$$

where negative values accelerate the car leftward and positive values accelerate it rightward.

Reward structure. The environment provides a reward of $-0.1\,a^2$ at every time step. An episode terminates when the car reaches the goal position $x \geq 0.5$, at which point we provide an extrinsic reward of $+100$. The maximum episode length is 999 steps.

Hyperparameters. Table 1 lists the key configuration parameters used in our experiments.

Role in evaluation. We use *MountainCarContinuous* to measure exploration efficiency in a sparse-reward continuous control task. Because the state space is only two-dimensional, the graph structure learned by GIR can be readily visualized and analyzed, providing insight into the exploration behavior of each algorithm.

B.3 PointMaze

The *PointMaze* environments Fu et al. (2020) are continuous-control navigation tasks in which a point mass must navigate through a maze to reach a goal location. We use three variants from

Table 1: Configuration for *MountainCarContinuous-v0*.

Parameter	Value
General	
`Total Episodes`	2000
`Max Timesteps`	999
`Optimizer`	Adam
`Learning Rate`	0.0005
`Batch Size`	256
`Discount Rate` γ	0.99
`Policy Noise Coefficient`	0.2
`Exploration Noise Coefficient`	0.1
`Clipping Coefficient`	0.5
`Replay Buffer Size`	100,000
`Target Update Frequency`	2
`Policy Network Architecture`	MLP(256, 256, 1)
`Value Network Architecture`	MLP(256, 256, 1)
`Network Activation Function`	Tanh
CBE	
`Clipping Count`	100
`Bin Size`	0.2
RND	
`Intrinsic Reward Coefficient`	0.6
`Predictor Network`	MLP(512, 512, 512)
`Target Network`	MLP(512, 512, 512)
`Random Batch Size`	256
PBRS	
`Potential Reward Weight`	0.5
`Potential Network`	MLP(256, 256, 1)
`Bin Size`	0.2
Go-Explore	
`Robustification Epochs`	100
`Bin Size`	0.2
`Exploration Steps`	0.2
`Stochasticity Coefficient`	0.5
GIR	
`Cluster Method`	Welford
`Cluster Threshold`	0.2
`Value Propagation Step`	4
`Explore Reward Factor`	0.2
`Exploration Decay Schedule`	$1/\sqrt{1+t}$
`Potential Reward Factor`	0.2

the `gymnasium-robotics` library: *PointMaze_UMaze-v3*, *PointMaze_Medium-v3*, and *PointMaze_Large-v3*.

State space. The observation is a 4-dimensional vector

$$s = (x, y, \dot{x}, \dot{y}) \in \mathbb{R}^4,$$

where (x, y) is the position and $(\dot{x}, \dot{y})$ is the velocity of the point mass in world coordinates.

For the large maze, the coordinate bounds are approximately

$$x \in [-6.5, 6.5], \quad y \in [-5.0, 5.0],$$

and for the U-shaped maze,

$$x, y \in [-2.7, 2.7].$$

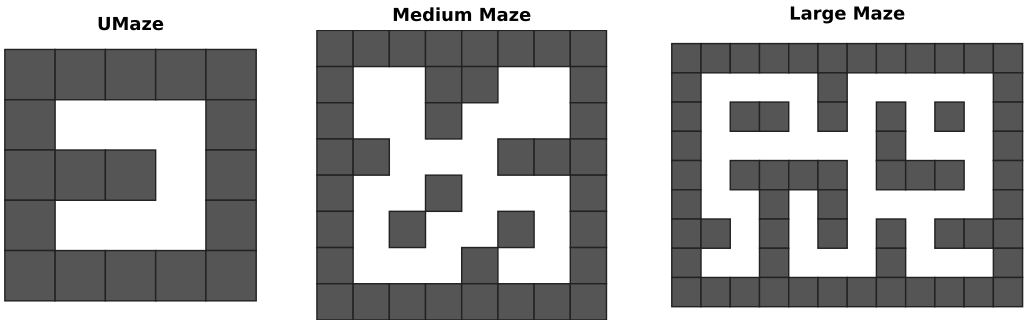


Figure 6: The layout of the mazes, start points and goal points are generated randomly.

Action space. The action is a 2-dimensional continuous force

$$a \in [-1, 1]^2,$$

applied to the point mass in the x and y directions.

Reward structure. The environment uses a sparse reward:

$$R(s, a, s') = \begin{cases} -0.1 \, \|a\|_2, & \text{if the goal is not reached,} \\ +100, & \text{if the goal is reached.} \end{cases}$$

An episode terminates when the agent reaches the goal or after 300 (UMaze), 600 (Medium) or 800 (Large) timesteps.

Maze layouts. Figure 6 illustrates the three maze configurations. The U-shaped maze is a simple 5×5 grid with a single wall forming a U-shaped corridor. The Medium maze is a 8×8 grid and the Large maze is a 12×9 grid with more complex wall geometry, requiring hierarchical exploration to reach the goal.

Hyperparameters. Table 2 lists the shared and maze-specific configuration parameters.

Role in evaluation. PointMaze provides a challenging sparse-reward continuous-control benchmark where exploration is critical. The UMaze variant tests whether an agent can learn to navigate around a simple obstacle, while the Large variant requires sustained exploration over a much larger state space. These environments are particularly suitable for evaluating graph-based methods because the discrete structure of maze corridors aligns naturally with graph representations.

B.4 MiniGrid

The *MiniGrid* environments Chevalier-Boisvert et al. (2023) are a suite of discrete partially-observable grid-world tasks. We evaluate on four variants: *SimpleCrossingS11N5*, *DistShift2*, *LavaGapS7*, and *FourRooms*.

Observation space. All environments use pixel-based observations. After preprocessing (resizing to 84×84 and stacking 3 frames), the observation tensor has shape

$$o \in \mathbb{N}^{9 \times 84 \times 84},$$

where the channel dimension is $9 = 3$ (RGB) $\times$ 3 (frames).

Action space. The action space is discrete with 7 possible actions: turn left, turn right, move forward, pick up an object, drop an object, toggle, and done.

Table 2: Configuration for *PointMaze*. Multiple parameters stands for *UMaze*, *MediumMaze*, and *LargeMaze*, respectively.

Parameter	**Value**
General	
`Total Episodes`	2000
`Max Timesteps`	300/600/800
`Optimizer`	Adam
`Learning Rate`	0.0003
`Batch Size`	256
`Discount Rate` γ	0.99
`Policy Noise Coefficient`	0.2
`Exploration Noise Coefficient`	0.1
`Clipping Coefficient`	0.5
`Replay Buffer Size`	100,000
`Target Update Frequency`	2
`Policy Network Architecture`	MLP(256, 256, 1)
`Value Network Architecture`	MLP(256, 256, 1)
`Network Activation Function`	Tanh
CBE	
`Clipping Count`	100
`Bin Size`	0.2/0.2/0.3
RND	
`Intrinsic Reward Coefficient`	0.6/0.6/1.0
`Predictor Network`	MLP(512, 512, 512)
`Target Network`	MLP(512, 512, 512)
`Random Batch Size`	256
PBRS	
`Potential Reward Weight`	0.85
`Potential Network`	MLP(256, 256, 1)
`Bin Size`	0.1/0.1/0.3
Go-Explore	
`Robustification Epochs`	100/100/150
`Bin Size`	0.1/0.2/0.2
`Exploration Steps`	0.2/0.2/0.2
`Stochasticity Coefficient`	0.4/0.4/0.8
GIR	
`Cluster Method`	Welford
`Cluster Threshold`	1.5/1.5/1.5
`Value Propagation Step`	3/5/3
`Explore Reward Factor`	0.2/0.2/0.4
`Exploration Decay Schedule`	$1/\sqrt{1+t}$
`Potential Reward Factor`	0.1/0.1/0.2

Table 3: Configuration for *MiniGrid*. Multiple parameters stands for *Crossing*, *DistShift*, *FourRooms*, *Lava*, respectively.

Parameter	Value
General	
Total Episodes	2000
Max Timesteps	100/200/300/200
Optimizer	Adam
Learning Rate	0.0003
Batch Size	256
Discount Rate γ	0.99
Policy Noise Coefficient	0.2
Exploration Noise Coefficient	0.1
Clipping Coefficient	0.5
Replay Buffer Size	100,000
Target Update Frequency	2
Policy Network Architecture	MLP(256, 256, 1)
Value Network Architecture	MLP(256, 256, 1)
Network Activation Function	Tanh
CBE	
Clipping Count	500
Bin Size	0.2/0.2/0.3/0.3
RND	
Intrinsic Reward Coefficient	0.6/0.6/1.0/1.0
Predictor Network	MLP(512, 512, 512)
Target Network	MLP(512, 512, 512)
Random Batch Size	256
PBRS	
Potential Reward Weight	0.85
Potential Network	MLP(256, 256, 1)
Bin Size	0.1/0.2/0.2/0.3
Go-Explore	
Robustification Epochs	100/100/150/150
Bin Size	0.1/0.2/0.2/0.3
Exploration Steps	0.2/0.2/0.2/0.4
Stochasticity Coefficient	0.4/0.4/0.6/0.8
GIR	
Cluster Method	KMeans
Cluster Threshold	0.06
Value Propagation Step	3/4/4/5
Explore Reward Factor	0.2/0.2/0.3/0.4
Exploration Decay Schedule	$1/\sqrt{1+t}$
Potential Reward Factor	0.1/0.1/0.1/0.1

Reward structure. A sparse reward is used across all tasks:

$$R(s,a,s') = \begin{cases} 1 - 0.9 \cdot (t/t_{\max}), & \text{if the goal is reached,} \\ 0, & \text{if a reserved action is used,} \\ -0.1, & \text{otherwise,} \end{cases}$$

where t is the current timestep and $t_{\max}$ is the episode horizon. The reward decreases linearly with the episode length to encourage faster solutions.

Task descriptions.

- **SimpleCrossingS11N5** (11×11 grid): The agent must navigate around randomly placed walls to reach the goal on the opposite side of the map. This tests basic obstacle avoidance and path planning.
- **DistShift-2** (8×8 grid): The agent must reach the goal in a maze composed of L-shaped obstacles. The layout is shifted by a random offset at each episode, testing the agent's ability to generalize across different spatial layouts.
- **LavaGapS7** (7×7 grid): The agent must cross a gap of lava tiles by walking along a narrow safe path. Falling into lava terminates the episode with zero reward. This tests risk-aware exploration.
- **FourRooms** (19×19 grid): The agent starts in one room and must reach the goal in another room, passing through narrow doorways. This tests hierarchical exploration and long-horizon credit assignment.

Hyperparameters. Table 3 lists the shared configuration parameters for all MiniGrid experiments.

Role in evaluation. MiniGrid provides a challenging testbed for exploration under pixel observations. The pixel-based observations require the graph-based methods to learn meaningful latent representations before constructing the exploration graph.

C Additional Experimental Results

C.1 Insights into Graph Progression in Environment MountainCar

We visualize the evolution of the agent's area of interest during training with GIR-guided TD3, as shown in Figure 7. In the early stage, the agent explores the environment as the guidance of GIR works. As training progresses, the exploration gradually becomes more focused, shifting toward regions that contribute to higher long-term returns. This transition illustrates how GIR reshapes the reward landscape to avoid trapping in local optima. Overall, the visualization demonstrates that GIR effectively guides the agent to escape local optima and discover meaningful trajectories toward the goal.

C.2 Visualization of Graph in Exploration of Environment PointMaze

We visualize the progression of the graph during training in the PointMaze environment, as shown in Figure 8. Initially, the graph is sparse and expands rapidly as the agent explores different regions of the maze. Over time, the graph structure becomes more organized, with nodes concentrating along feasible paths toward the goal. The edge weights gradually reflect the value of transitions, enabling the propagation of long-term return information across the graph. These results show that GIR captures both the structural layout of the environment and the value of trajectories, facilitating more efficient exploration in long-horizon navigation tasks.

C.3 Time Consumption Analysis

We evaluate the computational overhead introduced by GIR by measuring both inference time and training time across different methods. The results are summarized in Table 4.

All experiments are performed using Pytorch 2.3.1 and PyG 2.5.3 on Ubuntu 22.04. The hardware platform on which the experiments are performed is composed of a 13th Gen Intel(R) Core i7-13700KF CPU, a NVIDIA GeForce RTX 4090 GPU with 24 GB memory.

Compared to standard RL algorithms such as TD3 and SAC, GIR introduces only a modest computational overhead. Specifically, the additional cost mainly comes from graph construction and reward computation, which slightly increases the training time. However, the inference time remains comparable, indicating that the proposed method does not significantly affect action selection efficiency.

We further analyze the impact of graph size on computational cost by varying the number of graph nodes. The results show that both inference and training time increase mildly as the graph grows,

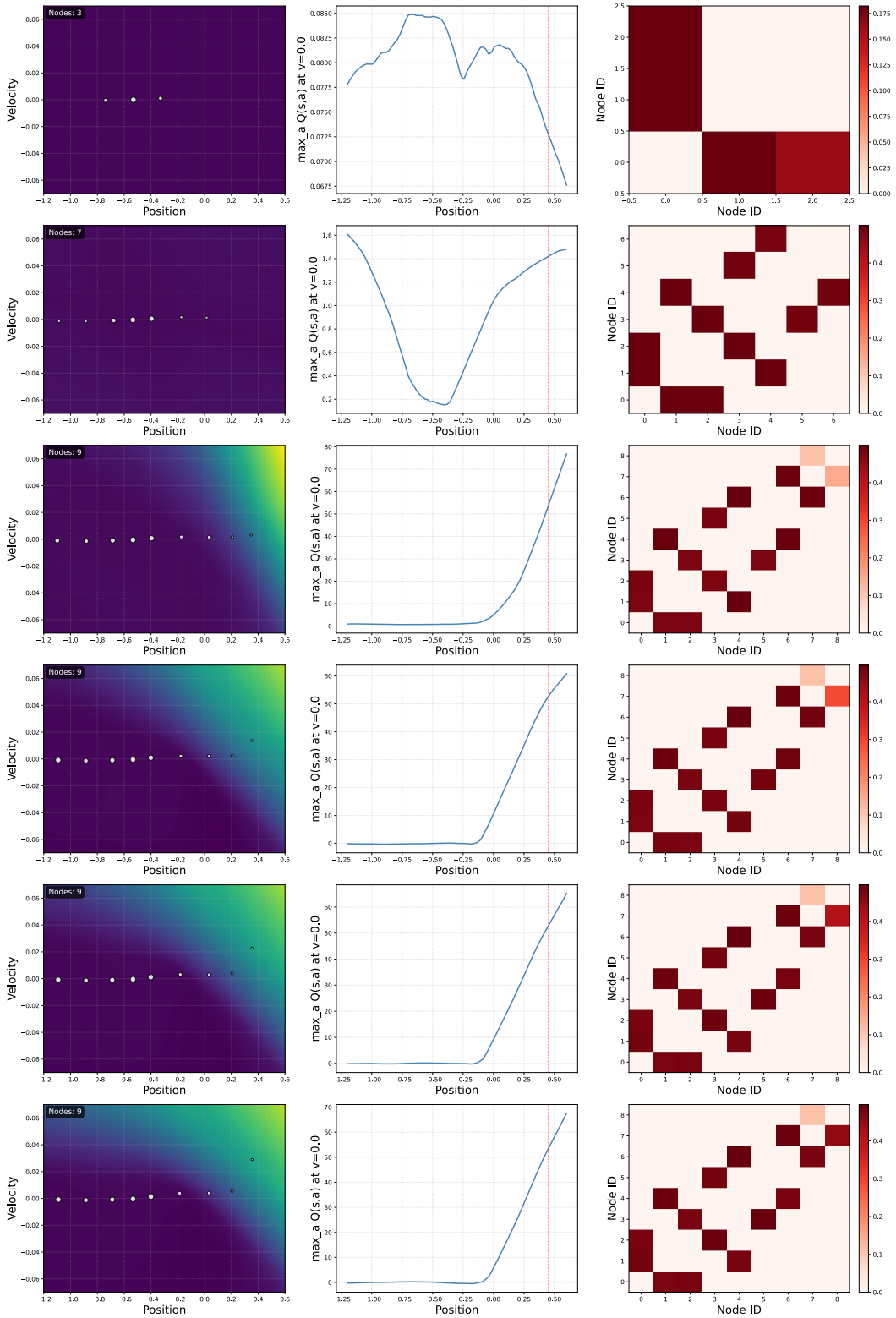
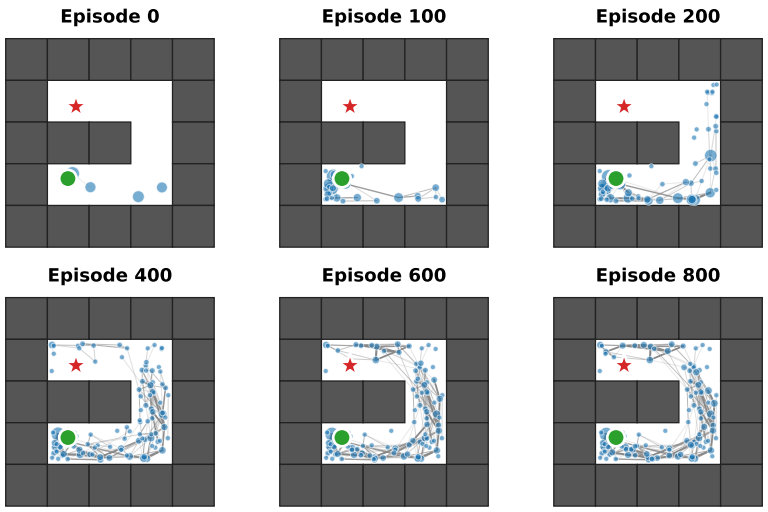
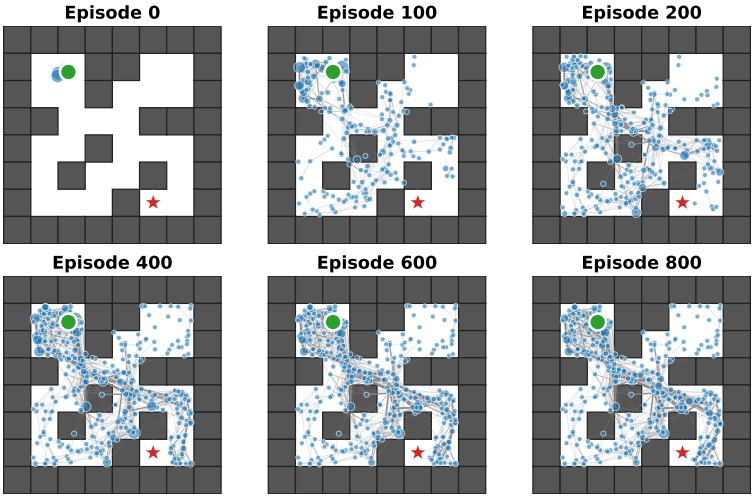


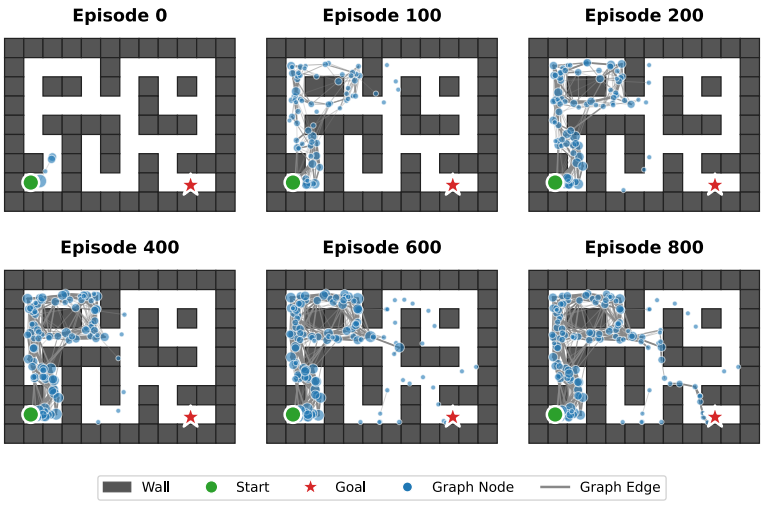
Figure 7: Episode Progression in Environment MountainCar



(a)



(b)



(c)

Figure 8: The progression of the graph constructed over time in the maze exploration.

Table 4: Time consumption comparison (milliseconds per step).

Method	Inference (ms)	Training (ms)	Total (ms)
TD3	0.0377	4.1278	4.1655
SAC	0.0556	7.2524	7.3079
RND	0.1126	8.8403	8.9529
CBE	0.0737	5.0137	5.0874
PBRS	0.0897	5.7345	5.8242
Go-Explore	0.1178	0.1120	0.2298
GIR (0 nodes)	0.0582	4.8904	4.9486
GIR (10 nodes)	0.0618	5.0763	5.1381
GIR (50 nodes)	0.0511	5.3528	5.4038
GIR (100 nodes)	0.0523	5.4544	5.5067
GIR (500 nodes)	0.0581	5.4151	5.4732
GIR (1000 nodes)	0.0856	5.9367	6.0223

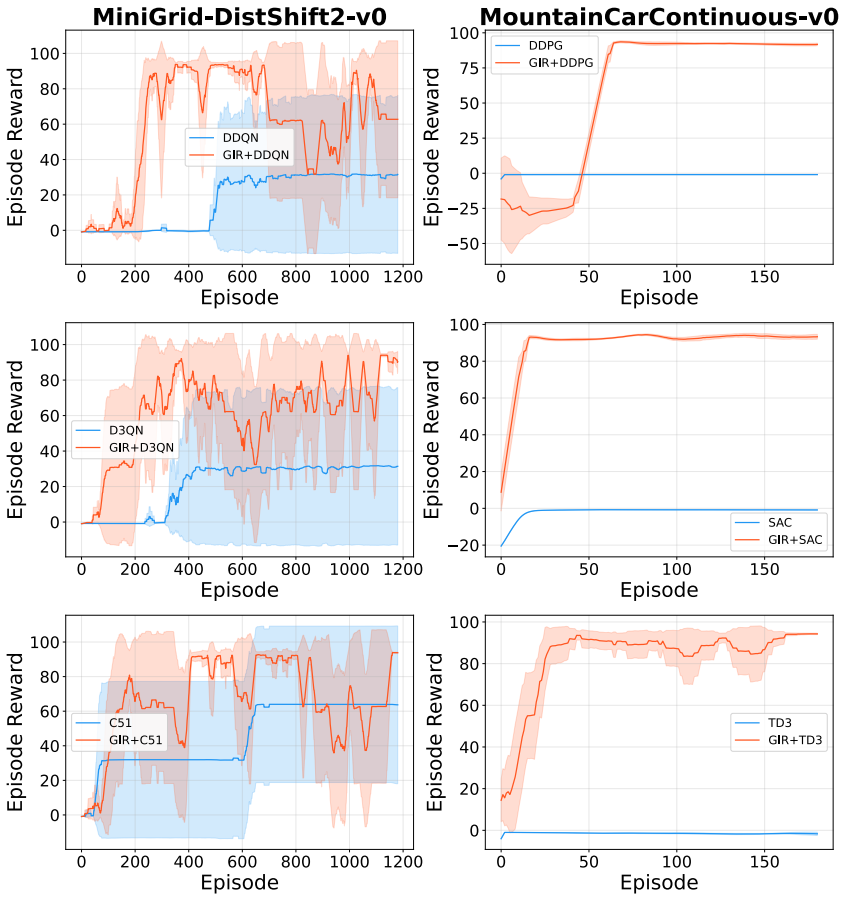


Figure 9: Comparison of GIR with other methods on the Gym environments.

but remain stable overall. Even with 1000 nodes, the total computational cost remains close to that of baseline methods, demonstrating good scalability in practice.

Overall, these results indicate that GIR achieves a favorable trade-off between performance improvement and computational efficiency, making it suitable as a plug-and-play module for existing RL algorithms.

C.4 Compatibility with Different RL Algorithms

To evaluate the generality of GIR as a plug-and-play module, we conduct compatibility experiments across a diverse set of reinforcement learning algorithms, covering value-based and actor-critic paradigms in both discrete and continuous action spaces.

For discrete control, we integrate GIR with value-based methods including Double DQN (DDQN), Dueling DQN (D3QN), and distributional RL methods such as C51 and QR-DQN. For continuous control, we evaluate GIR with actor-critic algorithms including DDPG, TD3, and SAC.

In all cases, GIR is incorporated as an auxiliary reward shaping module without modifying the original network architectures, loss functions, or optimization procedures.

The results in Figure 9 show that GIR consistently improves performance across all evaluated algorithms. For value-based methods, GIR enhances exploration by providing structured intrinsic rewards that complement value estimation. For actor-critic methods, it improves stability and sample efficiency by guiding both the actor and critic toward high-value regions. GIR mitigates sparse reward issues by reshaping the reward landscape, enabling more informative gradient updates.

Notably, the performance gains are consistent across both discrete and continuous domains, indicating that the effectiveness of GIR is largely independent of the underlying RL paradigm. Furthermore, we observe that GIR requires no algorithm-specific tuning, and the same hyperparameter configuration generalizes well across different methods.

These results demonstrate that GIR is a general and modular framework that can be seamlessly integrated with a wide range of reinforcement learning algorithms.

C.5 Compatibility with Different Input Type

We conduct experiments on a suite of MiniGrid tasks to evaluate the effectiveness of GIR in discrete, pixel-based cost-aware environments. Compared with continuous control benchmarks, MiniGrid introduces additional challenges due to high-dimensional visual observations, sparse rewards, and discrete action spaces, making efficient exploration substantially more difficult under action costs.

To construct the dynamic graph, state similarity is measured using cosine similarity between features extracted by a fixed random network, rather than conventional distance metrics defined directly in the observation space. This demonstrates that GIR is not restricted to low-dimensional continuous states, and can be naturally extended to image-based environments through feature-space abstraction.

We compare GIR against RND under identical training settings, using C51 Bellemare et al. (2017) as the base RL algorithm. The experimental results in Figure 10 show that baseline methods fail to discover effective goal-reaching policies under the imposed action costs, yielding near-random performance throughout training. In contrast, GIR consistently achieves substantial performance improvements across all evaluated tasks and is the only method that reliably learns successful policies.

These results indicate that the graph-based intrinsic reward remains effective in discrete visual domains, where the structured abstraction induced by the graph enables efficient propagation of long-term value information and guides exploration toward promising regions despite sparse external feedback and costly actions.

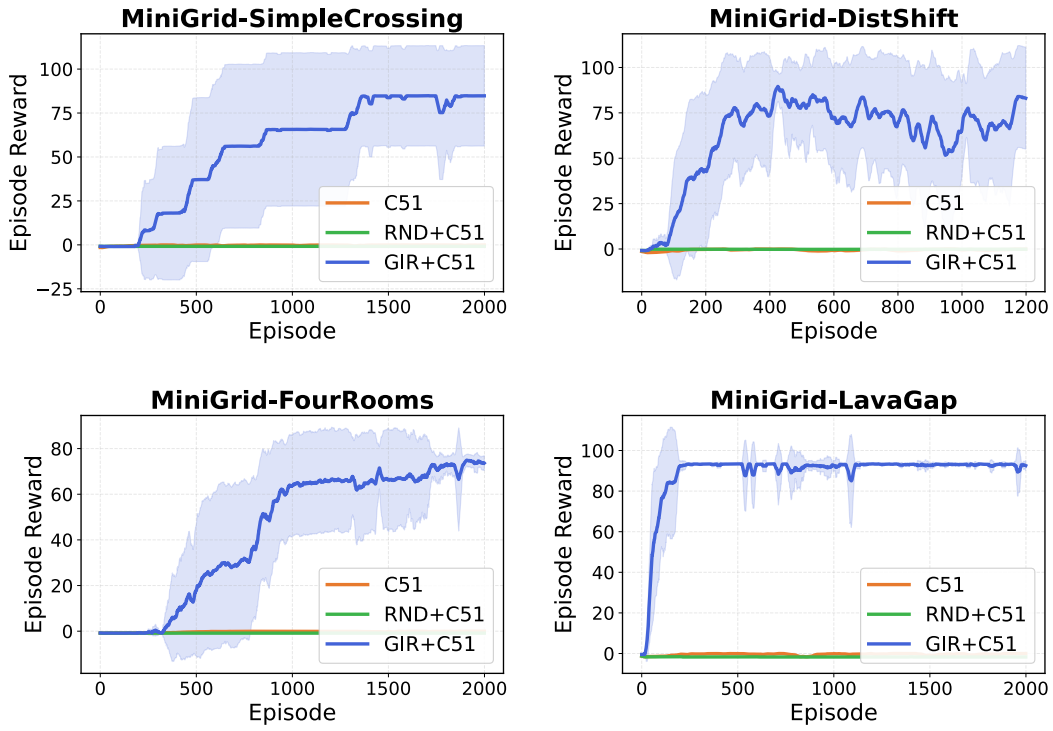


Figure 10: Learning curves of GIR on MiniGrid tasks over 10 experiments (Shaded area means standard deviation ±SD). GIR is able to achieve the goal of all four tasks in the MiniGrid environments, which significantly outperforms the other methods. The max timesteps for each task is 100(Simple Crossing), 200(DistShift), 300(FourRooms), 200(LavaGap).