RewardFlow: Can Action-Level Reward Be *Estimated* from State-Graph Topology in Agentic RL?

Xiao Feng[1] **Bo Han**[1†] **Zhanke Zhou**[1] **Jiaqi Fan**[2]
Jiangchao Yao[3] **Ka Ho Li**[2] **Dahai Yu**[2] **Michael K. Ng**[1]
[1]Hong Kong Baptist University [2]TCL Corporate Research (HK) Co., Ltd
[3]CMIC, Shanghai Jiao Tong University

Abstract

In agentic tasks such as household assistance and web shopping, large language model (LLM) agents take multiple actions before receiving a final outcome indicating success or failure. For reinforcement learning (RL), this outcome obscures individual actions' contributions: a shopping trajectory may find the right product but fail to complete the purchase. To address this credit-assignment problem, we introduce RewardFlow, which connects multiple trajectories for the same task through *shared states* (states grouped by task-relevant equivalence or semantic similarity). The resulting directed *state graph* links observed trajectory segments. Therefore, an action in a failed trajectory may reach a state from which another trajectory succeeds. It propagates reward signals backward from states where the task succeeds, assigning *state potentials* that decrease with shortest-path distance to success in the graph. *Potential differences* then reward actions that reduce this distance, bringing the agent closer to success in the graph. These estimated rewards yield action-level advantages that complement trajectory-level advantages from outcome rewards during policy optimization, without a learned process reward model or manual action-level annotations. Experiments on ALFWorld, WebShop, Sokoban, and search tasks show gains over RL baselines across model scales. With 3B policies, graph construction and reward estimation take 1.47-2.39 seconds per training step on ALFWorld, WebShop, and Sokoban (Fig. 1). Code is available at https://github.com/tmlr-group/RewardFlow.

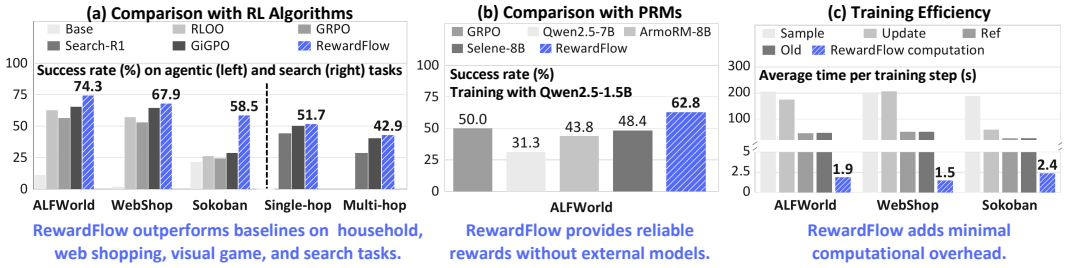


Figure 1: **RewardFlow improves agent learning. (a)** Averaged across evaluated model scales, RewardFlow achieves higher success rates on ALFWorld, WebShop, and Sokoban and higher search accuracy than the RL baselines. **(b)** On ALFWorld, Qwen2.5-1.5B trained with RewardFlow achieves a 62.8% success rate, compared with 50.0% for GRPO and 48.4% for GRPO with Selene-8B, the strongest evaluated PRM baseline. **(c)** Graph construction and reward estimation take 1.47-2.39 s per training step, using Qwen2.5-3B on ALFWorld and WebShop and Qwen2.5-VL-3B on Sokoban.

†Correspondence to Bo Han (bhanml@comp.hkbu.edu.hk).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

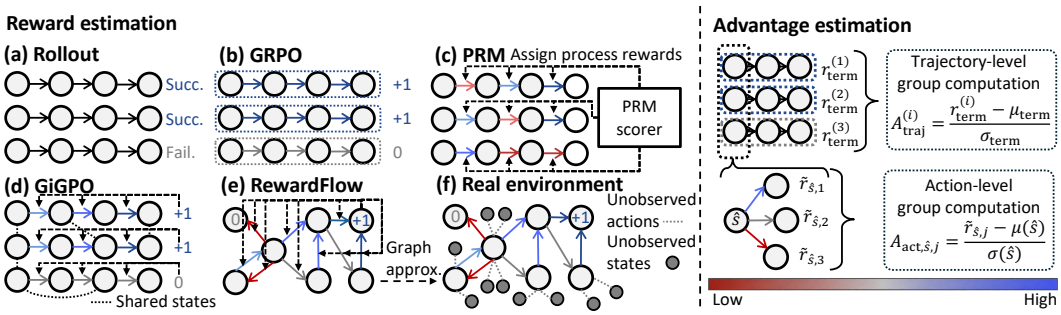


Figure 2: **From trajectory outcomes to graph-derived action rewards.** Given sampled trajectories (a), GRPO (b) uses terminal rewards, PRMs (c) score actions, and GiGPO (d) compares trajectory-local returns at shared states. RewardFlow (e) connects observed transitions into a graph that partially captures the environment's transition structure (f). Shared states link actions to successful continuations from other trajectories. Graph distances to success define potentials, whose differences yield action rewards. Right: Action and terminal rewards yield action- and trajectory-level advantages. The index j identifies action occurrences at graph node $\hat{s}$.

1 Introduction

In agentic tasks such as household assistance [43], web shopping [56], and tool-assisted reasoning [17, 60], large language model (LLM) agents interact with environments through sequences of actions. Reinforcement learning (RL) often trains these agents using a reward available only when a trajectory ends, indicating success or failure. Yet an action's contribution to task progress can differ from the trajectory's final outcome: a shopping agent may find the right product but later select the wrong option and fail to complete the requested purchase. Conversely, a successful trajectory may include avoidable detours. This mismatch creates an action-level credit-assignment problem: outcome rewards alone provide limited guidance on which actions to reinforce or discourage.

One approach to this credit-assignment problem is to use process reward models (PRMs) to score individual actions [26, 48]. However, a fixed scorer can become unreliable as the policy changes [10] and be exploited through reward hacking [3]. An alternative is to derive feedback directly from sampled interactions and terminal rewards. Group-in-Group Policy Optimization (GiGPO) [9] compares actions sampled at matching states across trajectories. However, each action is scored using discounted rewards from that action onward in the same trajectory. Later mistakes may therefore lower the score of an earlier action that made progress. This raises a question: *Can action-level rewards be estimated from the state-transition structure revealed collectively by sampled trajectories?*

To answer this question, we connect observed transitions across trajectories through *shared states*. These states represent task-relevant situations identified as equivalent or semantically similar using observations and relevant interaction history. Such states recur in trajectories sampled for the same task (Fig. 3, left), allowing their transitions to form a directed *state graph*. In the shopping example, two trajectories may reach the same product page with the same selected options. From that state, one later fails while the other completes the purchase. The graph connects the first trajectory's route to this shared state with the second's successful continuation, using only observed transitions. The graph's *topology* therefore encodes reachability to observed successful states and the minimum number of transitions required to reach them.

Building on this insight, we introduce RewardFlow, which estimates *action-level rewards from state-graph topology* for agentic RL (Fig. 2). For each task, RewardFlow constructs the graph from the current group of sampled trajectories and marks successful terminal nodes using outcome rewards. Reward signals propagate backward through the graph to assign *state potentials* that decrease with shortest-path distance to observed success. *Potential differences* then reward actions that bring the agent closer to success in the graph and penalize moves farther away. Comparing these rewards at shared states gives action-level advantages that complement outcome-based trajectory-level advantages during policy optimization, without a learned PRM or manual action-level annotations.

Empirically, we evaluate RewardFlow across four task families: household tasks (ALFWorld), web shopping (WebShop), visual puzzles (Sokoban), and search tasks. Averaged across evaluated model scales, it achieves absolute success-rate improvements over GiGPO of 6.2% for ALFWorld and

WebShop combined and 29.7% on Sokoban. It also achieves higher average search accuracy than GiGPO, with both methods using the same encoder for state grouping. A separate ALFWorld-3B control supports the contribution of the reward estimator: RewardFlow achieves an absolute success-rate improvement of 4.2% over GiGPO after state preprocessing and invalid-action filtering (Sec. 5.3). It also outperforms the evaluated PRM-based baselines on ALFWorld. Graph processing takes 1.47-2.39 seconds per training step with 3B policies on ALFWorld, WebShop, and Sokoban (Fig. 1).

Our contributions are:

- **State graphs for cross-trajectory analysis.** We connect observed transitions through shared states to construct directed graphs from sampled trajectories. These graphs reveal paths to success across trajectories and characterize each state's reachability and distance to observed success (Sec. 3).
- **Topology-aware action-level reward estimation.** We derive action rewards from potential differences for policy optimization. We analyze approximation error under incomplete graph coverage and conditions for cumulative raw rewards to favor successful trajectories (Sec. 4).
- **Controlled empirical evaluation.** Experiments show gains over RL baselines across four task families and multiple model scales. A matched preprocessing control and advantage ablations support the contribution of graph-derived rewards to policy learning (Sec. 5).

2 Preliminaries

Problem formulation. Consider an agentic setting where an LLM agent interacts with an environment to solve a task $x \sim P(X)$. We model these environments as partially observable Markov decision processes (POMDPs) [20]. Since the current observation may not fully reveal the environment state, we use a state representation $s_t \in \mathcal{S}$ that combines the task description, current observation, and relevant interaction history. At turn t, the policy π_θ generates a token sequence o_t conditioned on s_t, with $o_{t,k}$ denoting its k-th token. An action $a_t \in \mathcal{A}$ is parsed from o_t and executed in the environment. The environment returns feedback used to update the representation to s_{t+1}. Repeating this interaction yields a trajectory $\tau = (s_0, a_0, \ldots, a_{T-1}, s_T)$, where T is the number of action turns. In many agentic tasks, rewards are available only when the outcome is known: terminal rewards indicate success or failure, while intermediate actions receive zero task reward.

Group-based RL. For each task, the policy samples a group of G trajectories $\{\tau^{(i)}\}_{i=1}^{G}$, where trajectory i has T_i action turns and a binary terminal outcome $r_{\text{term}}^{(i)} \in \{0, 1\}$. Group-based methods such as GRPO [41] and RLOO [1, 21] use these outcomes to compute trajectory-level advantages for policy optimization. An outcome-based advantage is shared by all generated tokens in a trajectory, without distinguishing useful, unnecessary, and counterproductive actions within it. GiGPO [9] further groups actions at shared states and computes action-level advantages from their trajectory-local returns. However, trajectory-local returns can undervalue useful actions in failed trajectories and overvalue unnecessary or counterproductive actions in successful ones. Therefore, our goal is to estimate action-level rewards $\tilde{r}_t^{(i)}$ from the state-transition structure revealed across sampled trajectories. Together with trajectory-level terminal rewards, they yield complementary advantages to improve agentic RL without action-level annotations or a learned reward model.

3 State Graphs for Cross-Trajectory Analysis

To estimate action-level rewards from paths observed across trajectories, RewardFlow connects sampled trajectories through shared states into a *state graph* (Fig. 5, Steps 1-2). First, we describe how to identify shared states. Then, we construct the graph from observed state transitions. Finally, we analyze reachability and shortest-path distance to observed success and explain how shared nodes connect these paths.

3.1 State Identification across Trajectories

Across environments, the gap between total and unique counts of states and actions shows repeated visits in sampled trajectories (Fig. 3, left). Trajectories sampled for the same task can reach shared intermediate states, even when their subsequent actions and final outcomes differ (Fig. 3, right).

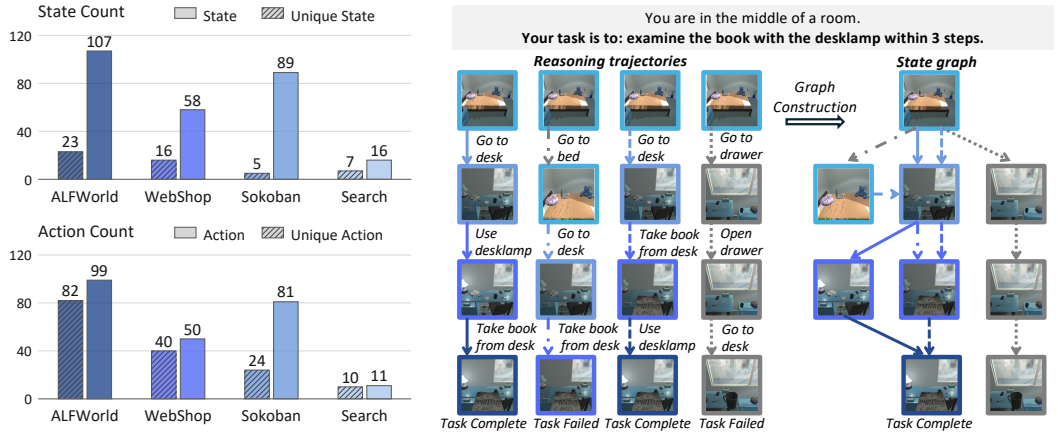


Figure 3: **Shared states connect sampled trajectories across environments. (left)** Per-task average counts of total and unique states and actions, computed from 128 trajectories sampled across 8 tasks using Qwen2.5-(VL)-3B-Instruct on ALFWorld, WebShop, Sokoban, and Search, reveal repeated visits to states and actions. **(right)** Successful and failed trajectories can reach the same states; merging these shared states connects their observed paths into a single state graph.

Identifying these meeting points lets one trajectory inform another about actions and paths observed from a shared state, even when their final outcomes differ.

We group task-relevant situations identified as equivalent or semantically similar through a map $f : \mathcal{S} \to \mathcal{V}$: states assigned to the same graph node $\hat{s}_v$ are treated as shared states. Under partial observability, state identification uses task-relevant observations and interaction history. Task-specific state preprocessing is detailed in Appendix B.1. For example, $f(s_1^{(1)}) = f(s_2^{(2)}) = f(s_t^{(i)}) = \hat{s}_v$ joins visits from three trajectories. If a failed trajectory reaches this node and another trajectory reaches success from it, the graph connects the first trajectory's path to the node with the second trajectory's path to success. This cross-trajectory path provides evidence for evaluating actions in the failed trajectory beyond its own terminal outcome.

3.2 Constructing the State Graph

Shared states form graph nodes, and observed state transitions form directed edges labeled by their actions. We include environment-valid state transitions and exclude invalid actions before constructing paths (Fig. 4, left). Appendix B.2 gives the validity rules.

Let $\mathcal{I}_{\text{valid}} \subseteq \{(i,t) : 1 \le i \le G,\ 0 \le t < T_i\}$ contain the indices of environment-valid state transitions. The directed state graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}_\mathcal{G}, \mathcal{E})$ is defined by

$$\mathcal{A}_\mathcal{G} = \{a_t^{(i)} : (i,t) \in \mathcal{I}_{\text{valid}}\} \subseteq \mathcal{A}, \qquad \mathcal{E} = \left\{\left(f(s_t^{(i)}), a_t^{(i)}, f(s_{t+1}^{(i)})\right) : (i,t) \in \mathcal{I}_{\text{valid}}\right\}. \quad (1)$$

Thus $\mathcal{V} = \bigcup_{i=1}^{G} \bigcup_{t=0}^{T_i} \{f(s_t^{(i)})\}$ includes all sampled states after applying f, including terminal states, while $\mathcal{A}_\mathcal{G}$ contains only labels of retained state transitions. We mark successful terminal nodes using the trajectories' terminal outcomes; these nodes anchor reward propagation in Sec. 4.1.

Each edge $(\hat{s}_v, a, \hat{s}_w) \in \mathcal{E}$ records an observed state transition, allowing paths to continue across trajectories at shared states. Appendix B.1 provides further examples and statistics.

3.3 Analyzing State-Graph Structure

The state graph connects intermediate states to task completion through environment-valid state transitions. Two complementary properties reveal this structure (Fig. 4, right):

- **Reachability.** A state is success-reachable when it has a path to an observed successful terminal. The shortest path measures its distance to success and supplies the basis for action-level reward estimation. Bright blue branches lack an observed path to success.

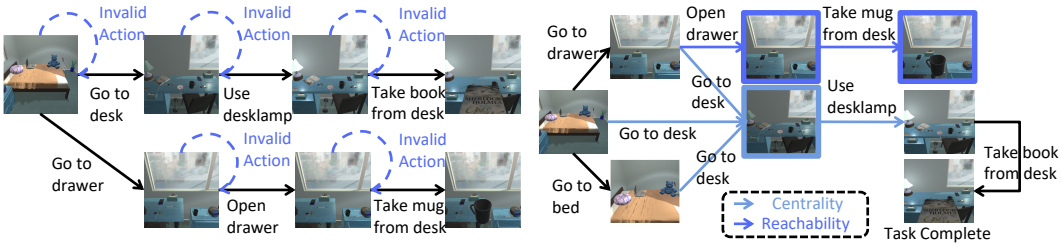


Figure 4: **State graph construction and structure. (left)** Environment-valid state transitions (black arrows) define the graph; blue dashed loops indicate invalid actions excluded from path construction. **(right)** The graph reveals reachability and centrality: bright blue branches lack an observed path to success, while the light blue desk state links incoming and outgoing state transitions.

- **Centrality.** Incoming and outgoing state transitions show how a state connects different routes. The desk state links paths involving the book and desklamp, allowing a path entering this shared state to follow actions observed in another trajectory.

By connecting sampled states to observed success, these analyses help assess each state's contribution to task completion. Therefore, the resulting state graph enables action-level reward estimation across trajectories, providing more reliable feedback than trajectory-local estimation.

4 Reward Estimation from State Graphs

Building on the state graph, RewardFlow estimates action-level rewards to complement trajectory-level supervision in agentic RL (Fig. 5, Steps 3-4). We first present action-level reward estimation, then describe how action- and trajectory-level advantages are combined for policy optimization. Finally, we analyze approximation error and trajectory-level reward ordering.

4.1 Action-Level Reward Estimation

State potentials. Successful terminals anchor reward estimation in observed task completion. Let $\mathcal{V}_{\text{succ}} \subseteq \mathcal{V}$ be these states in the graph. Starting from these anchors, we propagate a reward signal backward along incoming state transitions, so that states leading to observed success receive a potential. Each additional step away from success attenuates the signal by a factor of γ; when several paths are available, the shortest path determines the potential. We compute these distances using reverse breadth-first search (BFS) [33] and define the potential as

$$d_{\mathcal{G}}(\hat{s}_v) := \min_{\hat{s}_w \in \mathcal{V}_{\text{succ}}} \text{dist}_{\text{hop}}(\hat{s}_v \rightsquigarrow \hat{s}_w), \quad \Phi_{\mathcal{G}}(\hat{s}_v) := \begin{cases} \gamma^{d_{\mathcal{G}}(\hat{s}_v)}, & d_{\mathcal{G}}(\hat{s}_v) < \infty, \\ 0, & d_{\mathcal{G}}(\hat{s}_v) = \infty, \end{cases} \quad \gamma \in (0,1). \tag{2}$$

We set $d_{\mathcal{G}} = \infty$ when no path reaches observed success, including when $\mathcal{V}_{\text{succ}}$ is empty. Successful terminals have potential 1; states without such a path have potential 0. Shorter paths receive higher potentials, including paths assembled from different trajectories. Each state's potential thus measures its proximity to observed success using experience pooled across trajectories. We further compare alternative graph propagation rules for potential estimation in Appendices E.7 and E.8.

Action-level rewards. Let $\phi_t^{(i)}$ denote the graph potential at state $s_t^{(i)}$ in trajectory i. An environment-valid action receives the potential change along its observed state transition:

$$\phi_t^{(i)} := \Phi_{\mathcal{G}}\big(f(s_t^{(i)})\big), \qquad \tilde{r}_t^{(i)} := \phi_{t+1}^{(i)} - \phi_t^{(i)}. \tag{3}$$

Positive rewards recognize progress toward success in the state graph, while negative rewards penalize detours that increase the distance to success. An action in a failed trajectory can therefore receive a positive reward through a route discovered by another trajectory. Appendix D.2 illustrates this effect with a worked example comparing RewardFlow's cross-trajectory reward estimation with GiGPO's trajectory-local estimates.

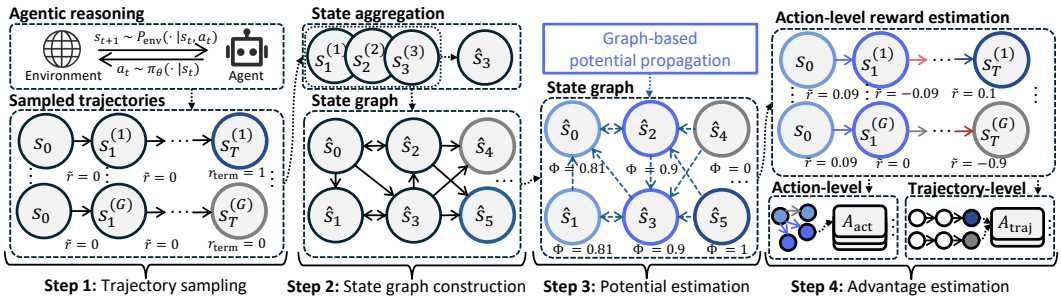


Figure 5: **Action-level reward estimation with RewardFlow.** For each task, policy π_θ samples G trajectories (Step 1). Shared states become nodes $\hat{s}_v$ connecting observed state transitions across trajectories (Step 2). Potentials Φ decay with shortest-path distance to successful terminals ($\gamma = 0.9$ here; Step 3). Their differences define action rewards $\tilde{r}$, which complement terminal outcomes r_{term} through action- and trajectory-level advantages A_{act} and A_{traj} in policy optimization (Step 4).

4.2 Complementary Advantage Estimation

Action-level advantage. Comparing rewards at the same graph node favors actions making greater graph-estimated progress relative to alternatives from that state. Let $\mathcal{I}_v = \{(i,t) \in \mathcal{I}_{\text{valid}} : f(s_t^{(i)}) = \hat{s}_v\}$ index these occurrences, counting repeated visits separately. Let $\mu(\hat{s}_v)$ be their mean reward and $\sigma(\hat{s}_v) > 0$ be a normalization scale obtained by applying F_{norm} to the reward multiset, such as the standard deviation plus $\varepsilon_{\text{norm}}$. The action-level advantage is

$$A^{(i)}_{\text{act},t} := (\tilde{r}^{(i)}_t - \mu(f(s^{(i)}_t)))/\sigma(f(s^{(i)}_t)). \tag{4}$$

For a singleton group, we set μ to its action reward and $\sigma = 1$, giving zero action advantage.

Trajectory-level advantage. Terminal outcomes provide complementary supervision for task completion. Let μ_{term} be a task-group baseline computed from the sampled terminal outcomes and $\sigma_{\text{term}} > 0$ a normalization scale. Following GRPO [41], we define

$$A^{(i)}_{\text{traj}} := (r^{(i)}_{\text{term}} - \mu_{\text{term}})/\sigma_{\text{term}}. \tag{5}$$

Thus, action-level advantages compare progress from a shared state, whereas trajectory-level advantages compare task-completion outcomes across attempts at the same task. Training configurations are provided in Appendix E.1.

Combining action- and trajectory-level advantages. Combining the two signals gives the policy guidance on individual actions while retaining supervision from overall task completion. Each generated token k in action turn t receives the same complementary advantage, with weights $\alpha_{\text{act}}, \alpha_{\text{traj}} \geq 0$:

$$A^{(i)}_{t,k} := \alpha_{\text{act}} A^{(i)}_{\text{act},t} + \alpha_{\text{traj}} A^{(i)}_{\text{traj}}. \tag{6}$$

With binary success as the only reward, both advantages vanish in an all-failure group; action advantages can still distinguish progress in an all-success group (Appendix D.4).

4.3 Policy Optimization

Using the complementary advantage, we follow GRPO [41] to optimize the policy with the clipped PPO objective [40]. For each task $x \sim P(X)$, we sample G trajectories with policy $\pi_{\theta_{\text{old}}}$, retaining their generated token sequences. With $o^{(i)}_{t,<k}$ denoting preceding tokens, the importance ratio is $\rho^{(i)}_{t,k} := \frac{\pi_\theta(o^{(i)}_{t,k}|s^{(i)}_t, o^{(i)}_{t,<k})}{\pi_{\theta_{\text{old}}}(o^{(i)}_{t,k}|s^{(i)}_t, o^{(i)}_{t,<k})}$. The objective combines clipped policy improvement with a KL penalty:

$$J_{\text{RewardFlow}}(\theta) = \mathbb{E}_{\substack{x\sim P(X),\\ \{\tau^{(i)}\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=0}^{T_i-1}\frac{1}{|o^{(i)}_t|}\sum_{k=1}^{|o^{(i)}_t|} \left(\min\left[\rho^{(i)}_{t,k}A^{(i)}_{t,k}, \text{clip}(\rho^{(i)}_{t,k}, 1-\varepsilon_{\text{clip}}, 1+\varepsilon_{\text{clip}})A^{(i)}_{t,k}\right] - \beta\mathbb{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\right)\right]. \tag{7}$$

Here $\pi_{\theta_{\text{old}}}(\cdot \mid x)$ denotes the trajectory distribution induced by the behavior policy and environment. The objective averages over trajectories, action turns, and tokens. The parameter $\varepsilon_{\text{clip}} > 0$ controls clipping, while $\beta \geq 0$ weights the token-context KL penalty relative to the fixed policy π_{ref}.

4.4 Theoretical Analysis

We first analyze graph approximation, then its effect on state potentials and action-level rewards, and finally trajectory-level reward ordering. For graph approximation, assume sufficient POMDP information states and a fixed map f into a countable shared-state space that preserves admissible actions, terminal status and outcomes, and possible successor classes. Let $\mathcal{G}^f_{\text{env}}$ contain reachable shared states and their admissible state transitions, and let $d^*_{\text{env}}(s)$ denote its shortest hop distance from $f(s)$ to success, with distance ∞ when no such path exists. Full proofs are in Appendix C.

Proposition 4.1 (POMDP State-Transition Graph Approximation)**.** *Under these conditions, trajectory sampling yields $\mathcal{G} \subseteq \mathcal{G}^f_{\text{env}}$ and $d_{\mathcal{G}}(f(s)) \geq d^*_{\text{env}}(s)$ for every represented s, with probability one. Distances agree once a shortest path and its successful endpoint are sampled.*

This relation provides the reference for analyzing distance-based state potentials.

Remark 4.2 (POMDP-Induced State Potential). Under Prop. 4.1, let $\Phi_{\text{env}}(s) := \gamma^{d^*_{\text{env}}(s)}$, with $\gamma \in (0, 1)$ and $\gamma^\infty = 0$. This potential measures shortest-path proximity to success in $\mathcal{G}^f_{\text{env}}$. For stochastic state transitions with successful-terminal value 1 and zero non-terminal rewards, the optimal expected discounted terminal value over admissible actions satisfies $V^*(s) \leq \Phi_{\text{env}}(s)$.

Missing paths can make the sampled distance to success longer than the environment's shortest distance. Since the potential decays with distance, the sampled graph then assigns a lower potential. The following result expresses this difference and identifies when it vanishes.

Proposition 4.3 (State-Potential Approximation)**.** *Under Rem. 4.2, the potential error at any represented information state s is*

$$\delta(s) := \Phi_{\text{env}}(s) - \Phi_{\mathcal{G}}(f(s)) = \gamma^{d^*_{\text{env}}(s)} - \gamma^{d_{\mathcal{G}}(f(s))} \geq 0.$$

The error vanishes once a shortest environment path and its successful endpoint are sampled, and converges almost surely to zero under the cumulative-sampling conditions in Appendix C.2.

Action rewards are differences between the potentials at the two endpoints of a state transition. Their approximation error therefore depends on the difference between the endpoint potential errors. For a retained state transition, write $\tilde{r}(s, a, s') := \Phi_{\mathcal{G}}(f(s')) - \Phi_{\mathcal{G}}(f(s))$.

Corollary 4.4 (Action-Level Reward Approximation)**.** *For retained (s, a, s') and δ from Prop. 4.3,*

$$\tilde{r}(s, a, s') = \big(\Phi_{\text{env}}(s') - \Phi_{\text{env}}(s)\big) - \big(\delta(s') - \delta(s)\big).$$

The error relative to the reference potential change is at most $\max\{\delta(s), \delta(s')\}$. As both coverage errors vanish, the graph-derived action reward recovers this POMDP-induced progress measure.

Finally, summing raw action rewards along a trajectory whose state transitions are all retained cancels intermediate potentials, leaving the final potential minus the initial potential. For such trajectories starting at the same graph node, their total raw action rewards are ordered by their endpoint potentials even when graph coverage is incomplete.

Proposition 4.5 (Trajectory-Level Reward Ordering)**.** *For a successful trajectory $\tau^{(+)}$ and a failed trajectory $\tau^{(-)}$ whose state transitions are all retained in $\mathcal{G}$, let T_+ and T_- be their action counts. If both start at the same canonical node $f(s_0^{(+)}) = f(s_0^{(-)}) = \hat{s}_{\text{init}}$, their raw action rewards satisfy $\sum_{t=0}^{T_+-1} \tilde{r}_t^{(+)} - \sum_{t=0}^{T_--1} \tilde{r}_t^{(-)} = \phi_{T_+}^{(+)} - \phi_{T_-}^{(-)}$, regardless of graph coverage. Since $\phi_{T_+}^{(+)} = 1$, the successful trajectory receives a strictly larger total reward whenever the failed endpoint is not merged with a success node, so $\phi_{T_-}^{(-)} < 1$.*

5 Experiments

We evaluate RewardFlow both in environments with well-defined state structures and in search tasks, where state identification requires an additional semantic definition. We first compare task performance, then examine graph-derived reward estimation and the complementarity of action- and trajectory-level advantages, and finally evaluate robustness and training efficiency.

Table 1: Main results (mean ± std over three seeds). We report success rates (%) for ALFWorld (per-subtask and overall average), and both task score and success rate for WebShop and Sokoban. **Bold** and underline denote the best and second-best results per column.

Type	Method	ALFWorld							WebShop		Sokoban	
		Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.	Score	Succ.
Qwen2.5-1.5B-Instruct												
Prompting	Base	$9.3_{\pm 0.2}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$10.5_{\pm 0.0}$	$3.9_{\pm 0.0}$	$20.8_{\pm 1.5}$	$4.9_{\pm 2.0}$	–	–
RL Training	RLOO	**$73.3_{\pm 5.9}$**	$16.7_{\pm 0.0}$	$46.8_{\pm 4.0}$	$74.6_{\pm 6.9}$	$54.2_{\pm 4.2}$	$52.6_{\pm 9.1}$	$56.0_{\pm 2.5}$	$80.8_{\pm 1.2}$	$56.5_{\pm 7.4}$	–	–
RL Training	GRPO	$72.2_{\pm 3.2}$	$13.9_{\pm 4.8}$	$34.2_{\pm 10.4}$	$42.1_{\pm 8.4}$	**$58.3_{\pm 7.2}$**	$49.1_{\pm 12.2}$	$49.5_{\pm 3.0}$	$64.8_{\pm 2.0}$	$45.1_{\pm 0.5}$	–	–
RL Training	GiGPO	$65.9_{\pm 5.7}$	**$26.7_{\pm 17.3}$**	$54.7_{\pm 8.2}$	$33.8_{\pm 9.1}$	$50.6_{\pm 6.2}$	$48.6_{\pm 10.3}$	$50.8_{\pm 2.8}$	$81.7_{\pm 5.5}$	$60.7_{\pm 4.8}$	–	–
RL Training	RewardFlow	$72.1_{\pm 3.5}$	$16.7_{\pm 0.0}$	**$76.0_{\pm 7.6}$**	**$95.2_{\pm 8.2}$**	$27.8_{\pm 6.4}$	**$77.2_{\pm 6.1}$**	**$62.8_{\pm 1.8}$**	**$83.8_{\pm 4.8}$**	**$66.4_{\pm 5.9}$**	–	–
Qwen2.5-(VL)-3B-Instruct												
Prompting	Base	$36.2_{\pm 15.1}$	$13.9_{\pm 4.8}$	$0.0_{\pm 0.0}$	$2.4_{\pm 4.1}$	$5.6_{\pm 2.4}$	$12.3_{\pm 3.0}$	$13.5_{\pm 3.3}$	$6.1_{\pm 1.7}$	$0.5_{\pm 0.9}$	$0.8_{\pm 0.3}$	$18.0_{\pm 3.4}$
RL Training	RLOO	**$88.7_{\pm 1.7}$**	$16.7_{\pm 0.0}$	$55.7_{\pm 5.8}$	$53.7_{\pm 6.0}$	$41.7_{\pm 4.2}$	$42.1_{\pm 9.1}$	$55.5_{\pm 0.0}$	$70.3_{\pm 6.0}$	$49.2_{\pm 5.9}$	$0.7_{\pm 0.1}$	$16.7_{\pm 0.5}$
RL Training	GRPO	$83.5_{\pm 2.0}$	$16.7_{\pm 0.0}$	$45.6_{\pm 1.0}$	$34.9_{\pm 1.4}$	$41.7_{\pm 4.2}$	$36.8_{\pm 15.8}$	$49.2_{\pm 3.6}$	$63.1_{\pm 1.9}$	$49.0_{\pm 4.0}$	$1.0_{\pm 0.2}$	$22.9_{\pm 3.7}$
RL Training	GiGPO	$84.8_{\pm 2.9}$	$41.7_{\pm 8.3}$	$65.9_{\pm 9.6}$	**$55.9_{\pm 2.2}$**	$53.6_{\pm 14.0}$	$65.8_{\pm 14.7}$	$64.6_{\pm 0.4}$	$77.9_{\pm 3.4}$	$62.0_{\pm 5.9}$	$0.9_{\pm 0.3}$	$19.8_{\pm 3.6}$
RL Training	RewardFlow	$87.0_{\pm 7.9}$	**$56.7_{\pm 11.5}$**	**$85.3_{\pm 4.6}$**	$43.1_{\pm 6.8}$	**$73.3_{\pm 5.1}$**	**$76.0_{\pm 7.4}$**	**$75.5_{\pm 3.2}$**	**$78.8_{\pm 4.6}$**	**$62.7_{\pm 3.2}$**	**$2.1_{\pm 0.2}$**	**$49.5_{\pm 1.2}$**
Qwen2.5-(VL)-7B-Instruct												
Prompting	Base	$45.3_{\pm 6.8}$	$16.7_{\pm 0.0}$	$8.8_{\pm 2.0}$	$2.4_{\pm 4.1}$	$6.9_{\pm 2.4}$	$0.0_{\pm 0.0}$	$16.4_{\pm 2.1}$	$3.3_{\pm 0.4}$	$0.3_{\pm 0.5}$	$1.3_{\pm 0.1}$	$25.0_{\pm 2.1}$
RL Training	RLOO	$94.8_{\pm 1.9}$	$36.1_{\pm 4.8}$	$78.4_{\pm 11.8}$	$79.0_{\pm 0.8}$	$68.1_{\pm 2.4}$	$75.4_{\pm 6.1}$	$76.3_{\pm 2.7}$	$76.6_{\pm 2.8}$	$65.6_{\pm 0.8}$	$1.4_{\pm 0.2}$	$35.7_{\pm 3.3}$
RL Training	GRPO	$90.7_{\pm 0.2}$	$25.0_{\pm 0.0}$	$63.4_{\pm 10.7}$	$72.4_{\pm 10.7}$	$70.8_{\pm 0.0}$	$73.7_{\pm 9.1}$	$70.6_{\pm 2.0}$	$78.5_{\pm 3.2}$	$65.1_{\pm 3.5}$	$1.3_{\pm 0.3}$	$26.0_{\pm 5.1}$
RL Training	GiGPO	$90.7_{\pm 6.2}$	$43.3_{\pm 31.8}$	$87.3_{\pm 9.6}$	$83.3_{\pm 10.9}$	**$73.2_{\pm 2.2}$**	$85.9_{\pm 3.1}$	$80.7_{\pm 2.0}$	$83.8_{\pm 1.5}$	$70.6_{\pm 0.9}$	$1.4_{\pm 0.2}$	$37.8_{\pm 3.5}$
RL Training	RewardFlow	**$95.9_{\pm 3.6}$**	**$62.3_{\pm 14.7}$**	**$92.4_{\pm 7.7}$**	**$84.2_{\pm 2.5}$**	$67.5_{\pm 12.5}$	**$91.5_{\pm 5.6}$**	**$84.6_{\pm 4.5}$**	**$85.3_{\pm 0.9}$**	**$74.7_{\pm 3.0}$**	**$3.5_{\pm 0.4}$**	**$67.4_{\pm 4.5}$**

Table 2: Performance comparison on search tasks (trained on NQ and HotpotQA following GiGPO). Reported as average accuracy (%). †in-distribution, * out-of-distribution.

Type	Method	Single-hop QA			Multi-hop QA				Avg.
		NQ†	TriviaQA*	PopQA*	HotpotQA†	2Wiki*	Musique*	Bamboogle*	
Qwen2.5-3B-Instruct									
RL Training	R1-Instruct	27.0	53.7	19.9	23.7	29.2	7.2	29.3	27.1
RL Training	Search-R1	34.1	54.5	37.8	32.4	31.9	10.3	26.4	32.5
RL Training	ZeroSearch	41.4	57.4	**44.8**	27.4	30.0	9.8	11.1	31.7
RL Training	StepSearch	–	–	–	34.5	32.0	**17.4**	34.4	–
RL Training	GiGPO	42.0	59.5	42.4	36.9	37.0	12.6	**64.1**	42.1
RL Training	RewardFlow	**44.4**	**60.9**	44.1	**40.3**	**41.2**	15.2	63.7	**44.3**
Qwen2.5-7B-Instruct									
RL Training	R1-Instruct	21.0	44.9	17.1	20.8	27.5	6.0	19.2	22.4
RL Training	Search-R1	39.3	61.0	39.7	37.0	40.1	14.6	36.8	38.5
RL Training	ZeroSearch	43.6	61.8	**51.5**	34.6	35.2	18.4	27.8	39.1
RL Training	StepSearch	–	–	–	38.6	36.6	**22.6**	40.0	–
RL Training	GiGPO	46.4	64.7	46.1	41.6	43.6	18.9	68.9	47.2
RL Training	RewardFlow	**47.4**	**65.2**	48.1	**44.7**	**47.1**	19.9	**71.4**	**49.1**

5.1 Experimental Setup

Experiments cover text-based household tasks in **ALFWorld** [43], web shopping in **WebShop** [56], visual puzzles in **Sokoban** [39], and retrieval-based question answering in **search tasks** [18]. In ALFWorld, WebShop, and Sokoban, environment-defined object states, page configurations, and board layouts provide a concrete basis for state identification, with interaction history retained where needed. Search tasks instead require an additional definition of shared states: we group observations by semantic similarity to connect related stages of information gathering. We train Qwen2.5-1.5B/3B/7B-Instruct on ALFWorld and WebShop, Qwen2.5-3B/7B-Instruct on search tasks, and Qwen2.5-VL-3B/7B-Instruct on Sokoban.

RLOO, GRPO, and GiGPO are the RL baselines for ALFWorld, WebShop, and Sokoban. For search tasks, we follow GiGPO's setup and also compare R1-Instruct, Search-R1 [18], ZeroSearch [45], and StepSearch [51]. GiGPO and RewardFlow use the same text encoder for semantic grouping (Appendix B.1). Appendix E.1 details the benchmarks, metrics, and training configurations; supplementary controls specify their protocols separately.

5.2 Task Performance

Agentic tasks. As illustrated in Tab. 1, RewardFlow achieves the highest overall ALFWorld success rate and the highest WebShop and Sokoban success rates at every evaluated model scale. On ALFWorld, it improves success over GiGPO from 64.6% to 75.5% at 3B and from 80.7% to 84.6% at 7B. It also leads in both task-match score and success rate on WebShop, showing improvements in both product matching and task completion. The improvements extend to visual tasks: on Sokoban, RewardFlow exceeds the strongest RL baseline by 26.6% at 3B and 29.6% at 7B. Together, these results demonstrate the value of graph-derived action rewards across tasks with different observation modalities and action spaces.

Search tasks. Search tasks pose a further challenge for graph construction: retrieved passages vary with the query, and related stages of information gathering need not produce identical observations. We therefore, following GiGPO's setting, identify shared states through approximate semantic matching rather than exact observation equality. Despite this challenge, RewardFlow achieves average accuracy of 44.3% at 3B and 49.1% at 7B, outperforming GiGPO by 2.2% and 1.9%, respectively (Tab. 2). With both methods using the same state-preprocessing method, these additional improvements suggest that RewardFlow's graph-based reward estimation better captures progress through observed environment interactions than GiGPO's trajectory-local reward estimation.

Comparison with process reward models. Baselines using PRM-estimated rewards underperform GRPO alone, whereas RewardFlow's action-level rewards improve RL success from 50.0% to 62.8% without a learned process scorer (Fig. 1, middle). Full details are provided in Appendix E.2; we discuss fixed-scorer transfer failures in Appendix E.3.

5.3 Ablation Studies

We examine how reward estimation and the two advantage levels contribute to learning:

- **Action-level reward estimation improves performance beyond preprocessing.** GiGPO's success increases from 75.2% to 78.6% using RewardFlow's state preprocessing method and invalid-action filtering. RewardFlow reaches 82.8%, a further 4.2% absolute improvement (Fig. 8). Full details see Appendix E.5.

Table 4: Propagation strategy ablation. We report success rates (%) on ALFWorld with Qwen2.5-1.5B and Sokoban with Qwen2.5-VL-3B.

Propagation strategy	ALFWorld	Sokoban
Mean aggregation	62.5	33.6
Personalized PageRank (PPR)	52.3	35.2
Shortest-path (BFS / Max)	**62.8**	**49.5**

- **Shortest-path propagation achieves the highest success rates.** It outperforms PPR in both environments. Compared with mean aggregation, success is similar on ALFWorld (62.8% vs. 62.5%) but increases from 33.6% to 49.5% on Sokoban (Tab. 4). Full details are provided in Appendix E.7 and Appendix E.8.
- **Action- and trajectory-level advantages are complementary.** Combining both at unit weight yields 75.5% success, versus 59.4% with trajectory advantages alone and 7.0% with action advantages alone. Halving the action or trajectory weight lowers success to 67.2% or 68.8%, respectively (Fig. 7). Full details are provided in Appendix E.9.

5.4 Robustness and Training Efficiency

We examine robustness to budget, distribution, and state-grouping changes, and measure runtime:

- **RewardFlow outperforms GiGPO across training and rollout budgets.** RewardFlow stays ahead in all four larger-training-budget comparisons by 2.6-7.6% in success rate and across rollout budgets of 4, 6, and 8 per training step, including 42.2% versus 28.9% with 4 rollouts (Figs. 6 and 9). Full details are provided in Appendix E.11 and E.12.
- **Performance improvements extend to unseen environments and datasets.** RewardFlow ranks first in overall success in unseen ALFWorld rooms and layouts at all three scales, including 66.1% versus GiGPO's 46.9% at 1.5B. On OOD search datasets, it exceeds GiGPO on four of five at 3B and all five at 7B (Tabs. 2-3). Full details are provided in Appendix E.1.

Table 3: Out-of-distribution evaluation on ALFWorld. The agent solves household tasks with familiar objects in entirely novel environments (different rooms, layouts, furniture). We report the average success (%); higher is better.

Method	ALFWorld						
	Pick	Look	Clean	Heat	Cool	Pick2	All
Qwen2.5-1.5B-Instruct							
Base	$2.4_{\pm 4.1}$	$0.0_{\pm 0.0}$	$5.7_{\pm 2.8}$	$3.4_{\pm 3.4}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$2.6_{\pm 1.2}$
RLOO	$66.7_{\pm 10.9}$	$35.1_{\pm 3.0}$	$37.5_{\pm 3.5}$	$41.4_{\pm 6.9}$	$26.6_{\pm 4.1}$	$33.3_{\pm 13.3}$	$39.3_{\pm 2.3}$
GRPO	$69.0_{\pm 4.1}$	$26.3_{\pm 13.9}$	$33.7_{\pm 4.9}$	$34.5_{\pm 6.9}$	**$32.5_{\pm 17.3}$**	$17.8_{\pm 3.8}$	$34.6_{\pm 2.0}$
GiGPO	$69.0_{\pm 4.1}$	$57.9_{\pm 5.3}$	$53.8_{\pm 3.0}$	$48.3_{\pm 6.0}$	$28.7_{\pm 4.5}$	$13.3_{\pm 0.0}$	$46.9_{\pm 2.8}$
RewardFlow	**$83.3_{\pm 8.2}$**	**$84.2_{\pm 5.3}$**	**$69.2_{\pm 1.2}$**	**$74.7_{\pm 10.0}$**	$28.3_{\pm 11.6}$	**$44.4_{\pm 3.8}$**	**$66.1_{\pm 1.8}$**
Qwen2.5-3B-Instruct							
Base	$47.6_{\pm 14.9}$	$3.5_{\pm 3.0}$	$3.9_{\pm 1.7}$	$5.7_{\pm 2.0}$	$4.2_{\pm 7.2}$	$2.2_{\pm 3.8}$	$8.9_{\pm 2.4}$
RLOO	$90.5_{\pm 4.1}$	$10.5_{\pm 5.3}$	$53.9_{\pm 4.6}$	**$44.8_{\pm 6.0}$**	$53.2_{\pm 8.2}$	$20.0_{\pm 0.0}$	$45.3_{\pm 2.1}$
GRPO	**$95.2_{\pm 4.1}$**	$28.1_{\pm 8.0}$	$46.1_{\pm 5.1}$	$37.9_{\pm 3.4}$	$38.7_{\pm 6.6}$	$44.4_{\pm 13.9}$	$45.8_{\pm 2.5}$
GiGPO	$92.9_{\pm 0.0}$	$47.4_{\pm 9.1}$	$54.8_{\pm 5.5}$	$39.1_{\pm 5.3}$	$40.9_{\pm 7.9}$	**$57.8_{\pm 3.8}$**	$52.9_{\pm 0.9}$
RewardFlow	$92.9_{\pm 0.0}$	**$52.6_{\pm 10.5}$**	**$61.6_{\pm 9.0}$**	$33.3_{\pm 5.3}$	**$55.1_{\pm 6.5}$**	$53.3_{\pm 6.7}$	**$55.5_{\pm 4.1}$**
Qwen2.5-7B-Instruct							
Base	$28.6_{\pm 12.4}$	$12.3_{\pm 3.0}$	$16.4_{\pm 7.5}$	$8.0_{\pm 4.0}$	$12.1_{\pm 10.5}$	$0.0_{\pm 0.0}$	$12.8_{\pm 4.3}$
RLOO	**$92.9_{\pm 0.0}$**	$64.9_{\pm 3.0}$	**$86.5_{\pm 3.6}$**	$62.1_{\pm 3.4}$	$81.6_{\pm 0.6}$	**$84.4_{\pm 10.2}$**	$77.6_{\pm 3.0}$
GRPO	$83.3_{\pm 4.1}$	$47.4_{\pm 0.0}$	$73.0_{\pm 6.3}$	$57.5_{\pm 5.3}$	$81.6_{\pm 0.6}$	$60.0_{\pm 17.6}$	$66.4_{\pm 4.9}$
GiGPO	**$92.9_{\pm 0.0}$**	$49.1_{\pm 8.0}$	$80.8_{\pm 7.2}$	$65.5_{\pm 6.0}$	$79.5_{\pm 4.0}$	$68.9_{\pm 15.4}$	$72.4_{\pm 4.4}$
RewardFlow	$90.5_{\pm 4.1}$	**$87.7_{\pm 16.9}$**	$77.9_{\pm 1.9}$	**$69.0_{\pm 0.0}$**	**$89.8_{\pm 3.4}$**	$80.0_{\pm 6.7}$	**$80.5_{\pm 4.1}$**

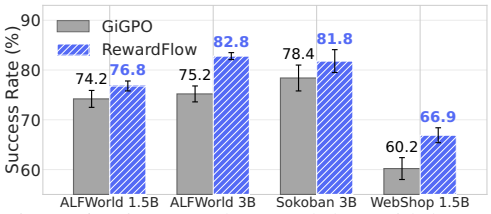


Figure 6: GiGPO and RewardFlow with larger training budgets.

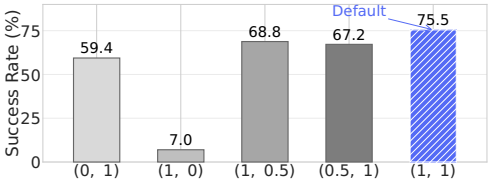


Figure 7: Advantage-weight ablation ($\alpha_{\text{act}}, \alpha_{\text{traj}}$) on ALFWorld (Qwen2.5-3B).

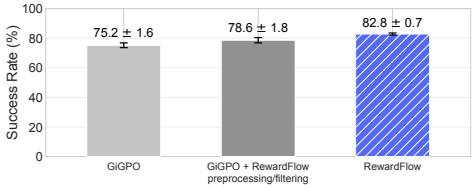


Figure 8: Preprocessing control on ALFWorld with Qwen2.5-3B.

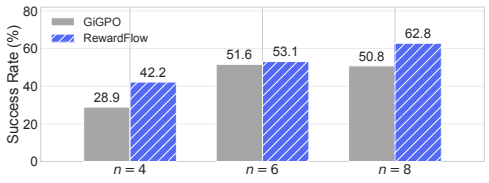


Figure 9: GiGPO and RewardFlow with 4-8 rollouts budget on ALFWorld (Qwen2.5-1.5B).

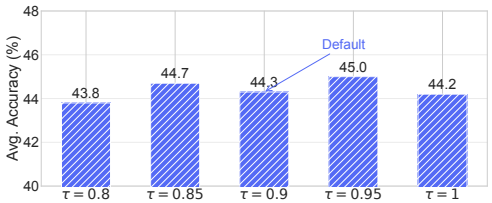


Figure 10: Sensitivity to cosine-similarity threshold τ on search tasks (Qwen2.5-3B).

- **Search accuracy remains stable across cosine-similarity thresholds.** Across semantic similarity thresholds 0.80, 0.85, 0.90, 0.95, and 1.00 for encoded observations using the embedding model, RewardFlow's accuracy stays within 43.8-45.0%, with 44.3% at the default 0.90. Higher thresholds produce finer state groupings (Fig. 10). Full details are provided in Appendix E.6.
- **RewardFlow's computation adds little training overhead.** The introduced components of RewardFlow, state graph construction and action-level reward estimation, take overall 1.87, 1.47, and 2.39 seconds per training step on ALFWorld, WebShop, and Sokoban, respectively, without additional model calls (Fig. 1, right). Full details are provided in Appendix E.13.

6 Conclusion

We present RewardFlow, a method for estimating action-level rewards from state-graph topology. By connecting sampled trajectories through shared states, RewardFlow defines state potentials from shortest-path distances to observed success and uses their differences as action-level rewards. These graph-derived rewards complement trajectory-level supervision to improve performance across household, shopping, visual puzzle, and search tasks without a learned process scorer. Consistent improvements and robustness across evaluation conditions support RewardFlow as a reliable source of action-level supervision. Tool-integrated math reasoning experiments provide preliminary evidence of applicability to open-ended tasks, as detailed in Appendix E.15. Future work will develop more reliable state identification across reasoning paths and evaluate this extension more broadly.

Acknowledgements

XF, ZKZ, and BH were supported by Guangdong Basic and Applied Basic Research Foundation No. 2024A1515012399, RGC Research Fellow Scheme No. RFS2627-2S02, RGC Young Collaborative Research Grant No. C2005-24Y, RGC General Research Fund No. 12202026, RGC General Research Fund No. 12200725, NSFC Major Research Plan No. 92570109, NSFC General Program No. 62376235, TCL Collaborative Research Fund, Guangdong and Hong Kong Universities "1+1+1" Joint Research Collaboration Scheme, HKBU Faculty Niche Research Areas No. RC-FNRA-IG/25-26/SCI/05, and HKBU CSD Departmental Incentive Scheme. JCY is supported by National Natural Science Foundation of China (No. 62676245), the Lingang Laboratory (Grant No. LGL-8889-01), STCSM (No. 22DZ2229005) and 111 plan (No. BP0719010). MKN is supported in part by the Department of Science and Technology of Guangdong Province (GDSTC): Guangdong and Hong Kong Universities "1+1+1" Joint Research Collaboration Scheme, the National Key Research and Development Program of China (Grant No. 2024YFE0202900), and Research Grants Council (RGC) of Hong Kong under the General Research Fund (GRF) (Grant No. 12300125).

References

[1] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

[2] Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, et al. Web-shepherd: Advancing prms for reinforcing web agents. *arXiv preprint arXiv:2505.15277*, 2025.

[3] Jie Cheng, Gang Xiong, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Yisheng Lv, and Fei-Yue Wang. Stop summation: Min-form credit assignment is all process reward model needs for reasoning. In *NeurIPS*, 2025. URL https://arxiv.org/abs/2504.15275.

[4] Xin Cheng, Shuo He, Lang Feng, HaiYang Xu, Ming Yan, Lei Feng, and Bo An. Beyond trajectory-level attribution: Graph-based credit assignment for agentic reinforcement learning. *arXiv preprint arXiv:2605.26684*, 2026. URL https://arxiv.org/abs/2605.26684v2.

[5] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

[6] Aladin Djuhera, Swanand Ravindra Kadhe, Farhan Ahmed, Heiko Ludwig, and Holger Boche. Tsr: Trajectory-search rollouts for multi-turn rl of llm agents. *arXiv preprint arXiv:2602.11767*, 2026.

[7] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.

[8] Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.

[9] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *NeurIPS*, 2025.

[10] Lukas Fluri, Leon Lang, Alessandro Abate, Patrick Forré, David Krueger, and Joar Max Viktor Skalse. The perils of optimizing learned reward functions: Low training error does not guarantee low regret. In *ICML*, 2025. URL https://proceedings.mlr.press/v267/fluri25a.html.

[11] Daniel Furelos-Blanco, Mark Law, Anders Jonsson, Krysia Broda, and Alessandra Russo. Induction and exploitation of subgoal automata for reinforcement learning. *Journal of Artificial Intelligence Research*, 2021.

[12] Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *ICLR*, 2026.

[13] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.

[14] Zhiyuan Hu, Shiyun Xiong, Yifan Zhang, See-Kiong Ng, Anh Tuan Luu, Bo An, Shuicheng Yan, and Bryan Hooi. Guiding vlm agents with process rewards at inference time for gui navigation. *arXiv preprint arXiv:2504.16073*, 2025.

[15] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *ICML*, 2018.

[16] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 2022.

[17] Dongfu Jiang, Yi Lu, Zhuofeng Li, Zhiheng Lyu, Ping Nie, Haozhe Wang, Alex Su, Hui Chen, Kai Zou, Chao Du, et al. Verltool: Towards holistic agentic reinforcement learning with tool use. *arXiv preprint arXiv:2509.01055*, 2025.

[18] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[19] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[20] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

[21] Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 REINFORCE samples, get a baseline for free!, 2019.

[22] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 2019.

[23] Xuan Li, Zhanke Zhou, Zongze Li, Jiangchao Yao, Tongliang Liu, and Bo Han. Learning to evolve: Open-ended molecular optimization with progress-shaped RL. In *3rd AI for Math Workshop: Toward Self-Evolving Scientific Agents*, 2026. URL https://openreview.net/forum?id=qEe4ypDaQp.

[24] Xuan Li, Zhanke Zhou, Zongze Li, Jiangchao Yao, Yu Rong, Lu Zhang, and Bo Han. RePO: Reference-guided policy optimization for molecular optimization via LLM reasoning. In *International Conference on Learning Representations*, 2026.

[25] Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025.

[26] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

[27] Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Junge Zhang, and Jianbin Jiao. Agentic reinforcement learning with implicit step rewards. *arXiv preprint arXiv:2509.19199*, 2025.

[28] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

[29] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

[30] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *ACL*, 2023.

[31] Maja J. Matarić. Reward functions for accelerated learning. In *ICML*, 1994.

[32] Yu Meng, Mengzhou Xia, and Danqi Chen. SimPO: Simple preference optimization with a reference-free reward. In *NeurIPS*, 2024.

[33] E. F. Moore. The shortest path through a maze. In *Proc. International Symposium on the Theory of Switching, Part II*. harvard, 1959.

[34] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, 1999.

[35] Evan L. Porteus. Bounds and transformations for discounted finite markov decision chains. *Oper. Res.*, 23:761–784, 1975.

[36] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

[37] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

[38] Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, 1998.

[39] Max-Philipp B. Schrader. Gym-Sokoban. https://github.com/mpSchrader/gym-sokoban, 2018.

[40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[41] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[42] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *CVPR*, 2020.

[43] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *ICLR*, 2021.

[44] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

[45] Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*, 2025.

[46] Hongduan Tian, Xiao Feng, Ziyuan Zhao, Xiangyu Zhu, Rolan Yan, and Bo Han. Multi-agent debate with memory masking. In *International Conference on Learning Representations (ICLR)*, 2026.

[47] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 2022.

[48] Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. Spa-rl: Reinforcing llm agents via stepwise progress attribution. *arXiv preprint arXiv:2505.20732*, 2025.

[49] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[50] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *ACL*, 2024.

[51] Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. *arXiv preprint arXiv:2505.15107*, 2025.

[52] Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, et al. Agentprm: Process reward models for llm agents via step-wise promise and progress. In *WWW*, 2026.

[53] Yi Xie, Zhanke Zhou, Chentao Cao, Qiyu Niu, Tongliang Liu, and Bo Han. From debate to equilibrium: Belief-driven multi-agent LLM reasoning via Bayesian Nash Equilibrium. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=RQwexjUCxm.

[54] Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025.

[55] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.

[56] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*, 2022.

[57] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *NeurIPS*, 2025.

[58] Zizhuo Zhang, Jianing Zhu, Xinmu Ge, Zihua Zhao, Zhanke Zhou, Xuan Li, Xiao Feng, Jiangchao Yao, and Bo Han. Co-rewarding: Stable self-supervised RL for eliciting reasoning in large language models. In *International Conference on Learning Representations*, volume 2026, pages 82520–82558, 2026.

[59] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*, 2025.

[60] Zhanke Zhou, Chentao Cao, Xiao Feng, Xuan Li, Zongze Li, Xiangyu Lu, Jiangchao Yao, Weikai Huang, Tian Cheng, Jianghangfan Zhang, Tangyu Jiang, Linrui Xu, Yiming Zheng, Brando Miranda, Tongliang Liu, Sanmi Koyejo, Masashi Sugiyama, and Bo Han. AlphaApollo: A system for deep agentic reasoning. *arXiv preprint arXiv:2510.06261*, 2025. URL https://arxiv.org/abs/2510.06261.

[61] Zhanke Zhou, Xiao Feng, Zhaocheng Zhu, Jiangchao Yao, Sanmi Koyejo, and Bo Han. From passive to active reasoning: Can large language models ask the right questions under incomplete information? In *ICML*, 2025. URL https://proceedings.mlr.press/v267/zhou25e.html.

[62] Zhanke Zhou, Zongze Li, Weikai Huang, Xuan Li, Chentao Cao, Xiao Feng, Xiangyu Lu, Jinbo Hu, Yi Xie, Nico Pelleriti, Shiyang Liu, Menghan Lu, Max Zimmer, Brando Miranda, Jiangchao Yao, Bo Liu, Sanmi Koyejo, Sebastian Pokutta, and Bo Han. AlphaDiana: A measurement system for agents in verifiable reasoning tasks. *arXiv preprint arXiv:2610.20675*, 2026. URL https://arxiv.org/abs/2610.20675.

[63] Zhanke Zhou, Xiangyu Lu, Chentao Cao, Brando Miranda, Tongliang Liu, Bo Han, and Sanmi Koyejo. The easy, the hard, and the learnable: Confidence and difficulty-adaptive policy optimization for LLM reasoning. In *ICML*, 2026. URL https://proceedings.mlr.press/v306/zhou26ah.html.

[64] Zhanke Zhou, Zhaocheng Zhu, Xuan Li, Mikhail Galkin, Xiao Feng, Sanmi Koyejo, Jian Tang, and Bo Han. Landscape of thoughts: Visualizing the reasoning process of large language models. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/79201c5a56d6f39520fb8a06b1ec415c-Abstract-Conference.html.

Appendix

A Related Work **18**

B Further Method Details **18**

B.1 State Preprocessing 18

B.2 Filtering Invalid Actions 20

B.3 From Trajectory States to Token-Level Advantages 20

B.4 Prompt Templates 22

C Proofs **24**

C.1 Information States and State-Transition-Preserving Grouping 24

C.2 Proof of Proposition 4.1 and Cumulative-Sampling Convergence 25

C.3 State-Transition-Support Potential in a POMDP 25

C.4 Proof of Proposition 4.3 (State-Potential Approximation) 26

C.5 Proof of Corollary 4.4 (Action-Level Reward Approximation) 26

C.6 Proof of Proposition 4.5 (Trajectory-Level Reward Ordering) 26

D Further Discussion **27**

D.1 Relation to Potential-Based Reward Shaping and Reward Machines 27

D.2 Shared-State Grouping versus Graph-Based Reward Estimation 27

D.3 State Potentials and Directed State Transitions 28

D.4 Exploration and Graph Coverage 28

D.5 Graph-Based Rewards versus Learned Process Scorers 29

D.6 Learned Graph Models for Reward Estimation 29

E Further Experiments **29**

E.1 Detailed Experimental Setting 29

E.2 Comparison with Process Reward Models 31

E.3 Fixed-Scorer Transfer Diagnostics 31

E.4 State Preprocessing and Invalid-Action Filtering 31

E.5 Separating Reward Estimation from Preprocessing and Filtering 32

E.6 Semantic-Aggregation Threshold Sensitivity 32

E.7 Shortest-Path versus Mean Propagation 32

E.8 BFS vs. Personalized PageRank (PPR) 33

E.9 Sensitivity to α_{act} and α_{traj} 33

E.10 Longer Training 33

E.11 Comparison under GiGPO's Original Training Settings 33

E.12 Rollout-Budget Sensitivity 34

E.13 Graph Size and Runtime 34

E.14 Offline Synthesis of a State Normalizer 35

E.15 Supplementary Math-TIR Evaluation 35
E.16 State Graph Cases 37

A Related Work

RL for LLM Reasoning. Policy optimization for LLMs depends both on the feedback available and on how that feedback is converted into an update. PPO [40] supplies a clipped policy objective, while GRPO [41] and RLOO [1, 21] estimate baselines from groups of sampled outputs. Refinements such as DAPO [57] and Dr. GRPO [28] address aspects of the training objective and sampling procedure. CoDaPO [63] adapts policy-update weights and question sampling to rollout confidence and empirical difficulty. Preference-based approaches, including DPO [37] and SimPO [32], provide a different route for learning from comparisons. Self-supervised RL can also derive reward signals from agreement among model-generated answers without annotated solutions [58]. Verifiable feedback also supports policy optimization in scientific domains such as molecular optimization [23, 24]. RewardFlow estimates action-level rewards from the state-transition structure revealed across sampled trajectories; its graph-derived advantages are used in a clipped policy update.

RL for Agentic Scenarios. Agentic reasoning spans retrieval-based tasks [18, 44, 59], reasoning with code-execution tools [8, 17, 25, 60], reasoning under incomplete information [61], and multi-agent debate [46, 53]. Such tasks require the policy to respond to feedback over several turns. Evaluation and analysis tools provide information on task outcomes, reasoning processes, tool usage, and resource consumption [62, 64], which can guide RL optimization to improve policy performance. Rollout construction is one way to improve learning: ARPO [7] guides exploration with entropy, SimpleTIR [54] filters void turns, and TSR [6] uses training-time tree search. These choices shape the collected experience, while reward estimation guides learning from it. GiGPO [9] and HGPO [12] use repeated states to organize process-level comparisons. Concurrent work GraphGPO [4] also studies cross-trajectory graphs for reward estimation. RewardFlow estimates action rewards from graph-derived potential differences and analyzes their approximation error under incomplete trajectory coverage.

Process Reward Estimation for LLM Reasoning. Process reward models provide supervision below the level of a complete response. Work on mathematical reasoning uses human step labels [26], rollout-derived labels [29, 50], or implicit process rewards [5]. For agents, judging an intermediate action also requires its environment context and its relation to a future goal. AgentPRM [52], OPRL [27], and Web-Shepherd [2] investigate process feedback in interactive settings; process rewards can also guide actions at inference time [14]. RewardFlow recomputes action rewards from current observed state transitions and terminal outcomes without training a separate process scorer.

B Further Method Details

B.1 State Preprocessing

State preprocessing f assigns graph-node identities to the agent-side representations defined in Sec. 2. Advantage normalization refers to reward centering and scaling in Sec. 4.2. Retained history distinguishes observations that look alike but require different actions; embedding-based matching joins related text into shared nodes.

ALFWorld: retaining task-relevant history. After an agent cleans an apple, the next observation may still say that it is carrying an apple without recording that change. We retain inventory, held-object status, transformative actions such as cleaning or heating, and placement progress in pick-two tasks. These features distinguish interaction histories that share the same current observation.

Search tasks: semantic state grouping. Different search results can describe related information without matching as strings. We use E5 text embeddings [49] to aggregate observations that meet a cosine-similarity threshold (default 0.9). Fig. 10 and Appendix E.6 evaluate sensitivity to this threshold.

Within each task's trajectory group, retrieved observations are processed in input order. Each state group has a fixed representative observation. An observation joins the first existing group whose representative has embedding cosine similarity at least τ to it. If no representative meets the threshold, the observation starts a new group and becomes its representative. Members of a group share the same graph-node identity.

The following schematic summarizes these task-specific grouping rules:

$$f(s) = \begin{cases} \text{cluster representative}(s) & \text{if } s \text{ meets the semantic-similarity criterion,} \\ s \oplus \text{critical features} & \text{if } s \text{ is ambiguous,} \\ s & \text{otherwise.} \end{cases} \tag{8}$$

The right-hand sides specify canonical node keys; critical features are extracted from the observation and history already retained in s. Occurrences with the same key map to one node $\hat{s}_v$. For the identity interface $f(s) = s$, the representation itself is the key; node identifiers remain independent of trajectory and time indices.

Task-specific interfaces, shared graph operations. The map $f(\cdot)$ reflects the information exposed by each benchmark. Sokoban uses exact board configurations, $f(s) = s$, while text environments can require retained history or approximate matching. Graph construction, distance computation, and advantage formulas are shared; validity handling and training configurations are specified for each environment.

Invalid-action handling is a parallel interface choice. ALFWorld's "Nothing happens" reports an ineffective command. Filtering prevents this generic feedback from becoming a shared node across unrelated situations. Invalid actions in WebShop and Sokoban can induce state-transition self-loops, which preserve BFS distance and have zero potential difference.

Table 5: State-construction interfaces for the four main benchmarks and the supplementary Math-TIR evaluation.

Environment	Node identity / retained information	Role in graph construction
ALFWorld	Observation plus inventory, object status, and prior placement progress	Retains task-relevant changes across turns.
WebShop	Page observation with persistent selected attributes	Distinguishes product configurations.
Sokoban	Exact board configuration, $f(s) = s$	Joins identical board layouts.
Search tasks	E5 similarity aggregation of observations	Connects semantically related retrieved information.
Math-TIR	Exact interaction histories at reasoning/tool-action boundaries	Shares nodes along common interaction prefixes.

Here s in $f(s)$ is the state representation defined in Sec. 2, including the observation and retained interaction history. Within each task's trajectory group, the pipeline maps these representations to nodes, adds valid directed state transitions, marks observed successful endpoints, runs reverse BFS, and looks up node potentials for each retained trajectory occurrence.

Node construction for tool-integrated reasoning. In mathematical tool-integrated reasoning (Math-TIR), a model alternates written reasoning with Python execution. We call the supplementary node-construction procedure *Structured-Decoding*: generation branches at reasoning and tool-action boundaries, and the exact interaction history up to each boundary identifies a node. Shared prefixes connect the sampled branches. Executed Python actions and the resulting observed states supply directed edges; execution requires tool state consistent with the interaction history. RewardFlow and Graph-GRPO use the same branching rules, and RewardFlow applies its graph-potential formula to the resulting graph. A frozen LLM provides a separate state-construction option by mapping histories to shared node identities.

Recurrence and graph diagnostics. The method assigns rewards from paths to observed success. These diagnostics describe node recurrence, connectivity, and retained state transitions.

Repeated visits create opportunities for sharing. Fig. 3 shows repeated states and actions across sampled trajectories, providing opportunities to connect their paths.

Connectivity and reachability. Fig. 4 (left) illustrates state-transition filtering. Fig. 4 (right) shows success reachability and the incoming and outgoing state transitions at graph nodes. Reachability supplies the distances used to define state potentials; degree describes how trajectories connect.

Coverage conditions. Prop. 4.1 in Sec. 4.4 relates sampled graph distances to environment distances. Appendix C.2 gives the cumulative-sampling conditions used by the potential analysis.

B.2 Filtering Invalid Actions

A generated response can fail at two different levels: it may not parse as an action, or its parsed action may be inadmissible in the current environment. The retention predicate $\textsc{valid}(s_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)})$ checks the state transition using environment-specific validity and state-change feedback. The set $\mathcal{I}_{\text{valid}}$ in Sec. 3.2 contains exactly the indices for which this predicate equals 1.

In ALFWorld, an ineffective command can return generic feedback without creating a new environment state. Treating that string as node identity would merge unrelated situations and permit spurious paths. In the example below, "invalid state" labels generic feedback excluded from node construction.

An example of invalid actions in ALFWorld

STEP: 1
STATE: -= Welcome to TextWorld, ALFRED! =-

You are in the middle of a room. Looking quickly around you, you see a cabinet 10, a cabinet 9, a cabinet 8, a cabinet 7, a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable 1, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Your task is to: put a cool apple in microwave.
ACTION: go to cabinet 1
—
STEP: 2
STATE: You arrive at cabinet 1. The cabinet 1 is closed.
ACTION: go to cabinet 1 [INVALID ACTION]
—
STEP: 3
STATE: Nothing happens. [INVALID STATE]
ACTION: open cabinet 1
—
STEP: 4
STATE: You open the cabinet 1. The cabinet 1 is open. In it, you see nothing.

We exclude this generic feedback from effective propagation. The filtering ablation in Tab. 10 reports its contribution to task performance alongside the other interface components.

B.3 From Trajectory States to Token-Level Advantages

This section uses the same notation as the main text throughout the state-to-token computation. Trajectory indices are superscripts (i); action turns and generated tokens are subscripts t and k. A graph node has an independent identifier v. Terminal outcomes, node potentials, occurrence-level lookups, and action rewards are distinct quantities. The formulas use binary terminal outcomes.

In Fig. 2, j indexes action occurrences grouped at the same graph node $\hat{s}$. Their rewards are written as $\tilde{r}_{\hat{s},j}$, with group mean $\mu(\hat{s})$ and normalization scale $\sigma(\hat{s})$; the resulting action advantage is $A_{\text{act},\hat{s},j}$.

Table 6: Unified notation for states, graph nodes, and the state-to-token computation.

Symbol	Domain and meaning
$s_t^{(i)} \in \mathcal{S}$; $a_t^{(i)} \in \mathcal{A}$	State-representation and action occurrences at turn t of trajectory i.
$\hat{s}_v \in \mathcal{V}$; $f(s_t^{(i)})$	Canonical graph node and the node assigned to occurrence (i,t). The identifier v is not a time index.

Symbol	Domain and meaning (continued)
$\mathcal{G} = (\mathcal{V}, \mathcal{A}_{\mathcal{G}}, \mathcal{E})$	Sampled graph: nodes, observed action labels, and retained directed state transitions.
$\mathcal{V}_{\text{succ}}$	Observed successful terminal nodes in $\mathcal{G}$.
$d_{\mathcal{G}}(\hat{s}_v)$	Shortest directed hop distance to $\mathcal{V}_{\text{succ}}$, or ∞ if unreachable.
$d^*_{\text{env}}(s)$	Shortest hop distance from $f(s)$ to a successful class in the reference graph $\mathcal{G}^f_{\text{env}}$.
$r_{\text{env}}(s, a, s')$	Environment reward for an observed state transition.
$r^{(i)}_{\text{term}} \in \{0, 1\}$	Observed binary terminal success of trajectory i.
$\Phi_{\mathcal{G}}(\hat{s}_v)$	Graph-node potential, computed from sampled paths.
$\phi^{(i)}_t$	Node potential looked up at occurrence (i, t).
$\tilde{r}^{(i)}_t$	Raw graph-derived reward of the retained action at (i, t).
$\mathcal{I}_v$	Indices of retained action occurrences whose source maps to $\hat{s}_v$.
$\mu(\hat{s}_v)$; $\sigma(\hat{s}_v)$	State-group mean reward and positive normalization scale; singleton groups use their sole reward as the mean and scale 1.
μ_{term}; σ_{term}	Task-group baseline computed from sampled terminal outcomes and positive normalization scale.
$A^{(i)}_{\text{act},t}$; $A^{(i)}_{\text{traj}}$	State-conditioned action advantage and trajectory advantage. Neither is separately estimated per token.
$o^{(i)}_t$; $o^{(i)}_{t,k}$	Generated token sequence at turn t and its k-th token.
$A^{(i)}_{t,k}$; $\rho^{(i)}_{t,k}$	Complementary advantage broadcast to token k and its policy importance ratio.
$\varepsilon_{\text{norm}}$; $\varepsilon_{\text{clip}}$	Normalization stabilizer and PPO clipping parameter, respectively.

Graph nodes and occurrence lookup. Several occurrences can map to one node: $f(s^{(1)}_1) = f(s^{(2)}_2) = f(s^{(i)}_t) = \hat{s}_v$. Each occurrence retains its original (i, t) identity for reward assignment. For a fixed state graph $\mathcal{G}$, define

$$\Phi_{\mathcal{G}}(\hat{s}_v) = \begin{cases} \gamma^{d_{\mathcal{G}}(\hat{s}_v)}, & d_{\mathcal{G}}(\hat{s}_v) < \infty, \\ 0, & d_{\mathcal{G}}(\hat{s}_v) = \infty. \end{cases} \tag{9}$$

Then $\phi^{(i)}_t = \Phi_{\mathcal{G}}(f(s^{(i)}_t))$ and $\tilde{r}^{(i)}_t = \phi^{(i)}_{t+1} - \phi^{(i)}_t$. This is a lookup through f, not a single-valued inverse. The preimage of a node is a set of representations; $\mathcal{I}_v$ additionally distinguishes repeated occurrences of those representations. Only sampled-graph distances determine these rewards. For a generic observed state transition, $\tilde{r}(s, a, s') = \Phi_{\mathcal{G}}(f(s')) - \Phi_{\mathcal{G}}(f(s))$ is the same action-reward definition with the successor explicit.

Occurrence groups and normalization. Let

$$\mathcal{I}_v = \{(i, t) : f(s^{(i)}_t) = \hat{s}_v,\ (i, t) \text{ is retained}\}.$$

Repeated visits are counted separately, rather than deduplicating action strings. For non-singleton groups, $\mu(\hat{s}_v)$ is the mean action reward and $\sigma(\hat{s}_v) > 0$ is the positive factor obtained by applying F_{norm} to their reward multiset, as defined in Sec. 4.2. For singleton groups, μ equals the sole action reward and $\sigma = 1$, so the action advantage is zero.

With the same nonzero-denominator convention for terminal-outcome normalization,

$$A^{(i)}_{\text{act},t} = \frac{\tilde{r}^{(i)}_t - \mu(f(s^{(i)}_t))}{\sigma(f(s^{(i)}_t))}, \qquad A^{(i)}_{\text{traj}} = \frac{r^{(i)}_{\text{term}} - \mu_{\text{term}}}{\sigma_{\text{term}}}, \tag{10}$$

$$A^{(i)}_{t,k} = \alpha_{\text{act}} A^{(i)}_{\text{act},t} + \alpha_{\text{traj}} A^{(i)}_{\text{traj}}. \tag{11}$$

A constant-outcome group contributes zero trajectory advantage. The complementary advantage is broadcast to the valid generated tokens of the action, whereas $\rho^{(i)}_{t,k}$ depends on the conditional

probability of each token under the current and behavior policies. Thus only the final broadcast and probability ratio carry a token index; graph construction and action-level reward estimation operate at action-turn boundaries.

B.4 Prompt Templates

These are the prompts used to sample trajectories during RL training. The no-history variant starts a trajectory; the history variant supplies context for subsequent turns. Prompt contents are reproduced without modification.

Prompt of ALFWorld without history

You are an expert agent operating in the ALFRED Embodied Environment.
Your current observation is: {current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].

Now it's your turn to take an action.
You should first reason step-by-step about the current situation. This reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Prompt of ALFWorld with history

You are an expert agent operating in the ALFRED Embodied Environment. Your task is to: {task_description}
Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].

Now it's your turn to take an action.
You should first reason step-by-step about the current situation. This reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Prompt of WebShop without history

You are an expert autonomous agent operating in the WebShop e-commerce environment.
Your task is to: {task_description}.
Your current observation is: {current_observation}.
Your admissible actions of the current situation are:
[
{available_actions}
].

Now it's your turn to take one action for the current step.
You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Prompt of WebShop with history

You are an expert autonomous agent operating in the WebShop e-commerce environment.
Your task is to: {task_description}.
Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}.
Your admissible actions of the current situation are: [{available_actions}].

Now it's your turn to take one action for the current step.
You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Prompt of Sokoban

You are an expert agent operating in the Sokoban environment. Your goal is to push all the boxes onto the target spots. Once all boxes are on the targets, you win!

Rules
You can only push boxes. You can't pull them, so plan ahead to avoid getting stuck.
You can't walk through or push boxes into walls.
To avoid traps, do not push boxes into corners or against walls where they can't be moved again.

Visual Elements in the Image:
Character: A small, green alien-like figure with two antennae and black eyes. It represents you.
Box: A yellow crate marked with an orange "X" across its front. It is the box you need to push.
Target: A black tile outlined in red, with a small red diamond shape in the center. It marks the destination where a box should be pushed.

Current Step
Your current observation is shown in the image: <image>
Your admissible actions are ["up", "down", "left", "right"].

Now it's your turn to make a move (choose ONE action only for the current step).
You should first reason step-by-step about the current situation — observe the positions of boxes and targets, plan a path to push a box toward a target, and avoid traps like corners or walls. This reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Prompt for search tasks without history

You are an expert agent tasked with answering the given question step-by-step.
Your question: {task_description}

Now it's your turn to respond for the current step.
You should first conduct reasoning process. This process MUST be enclosed within <think> </think> tags.
After completing your reasoning, choose only one of the following actions (do not perform both):
(1) If you find you lack some knowledge, you can call a search engine to get more external information using format: <search> your query </search>.
(2) If you have enough knowledge to answer the question confidently, provide your final answer within <answer> </answer> tags, without detailed illustrations. For example, <answer>Beijing</answer>.

Prompt for search tasks with history

You are an expert agent tasked with answering the given question step-by-step.
Your question: {task_description}

Prior to this step, you have already taken {step_count} step(s). Below is the interaction history where <search> </search> wrapped your past search queries and <information> </information> wrapped the corresponding search results returned by the external search engine. History:
{memory_context}

Now it's your turn to respond for the current step.
You should first conduct reasoning process. This process MUST be enclosed within <think> </think> tags.
After completing your reasoning, choose only one of the following actions (do not perform both):
(1) If you find you lack some knowledge, you can call a search engine to get more external information using format: <search> your query </search>.
(2) If you have enough knowledge to answer the question confidently, provide your final answer within <answer> </answer> tags, without detailed illustrations. For example, <answer>Beijing</answer>.

C Proofs

We first establish how sampled state transitions approximate the POMDP's state-transition structure, then derive support-potential and action-reward guarantees that permit stochastic state transitions. We also distinguish this potential from expected discounted value. Throughout, s is an agent-side information state, $\hat{s}_v = f(s)$ is its shared graph state, and $\gamma^\infty = 0$.

C.1 Information States and State-Transition-Preserving Grouping

Fix a task x. Let y_t denote the observation at turn t and $h_t = (x, y_0, a_0, \ldots, y_t)$ its observation-action history. Write the agent-side representation from Sec. 2 as $s_t = \psi(h_t)$, where ψ retains the current observation and task-relevant history. The policy-environment interaction induces a distribution $P_{\text{env}}(\cdot \mid h_t, a_t)$ over the next representation $s_{t+1} = \psi(h_{t+1})$. A representation s_t is a *sufficient information state* if this conditional distribution depends on the history only through s_t and the chosen action:

$$\Pr(s_{t+1} \in B \mid h_t, a_t) = K(B \mid s_t, a_t)$$

for every measurable set B. Admissible actions and terminal status, including success or failure, must also be determined by s_t. Relevant task memory and any horizon affecting future outcomes are part of this information. A belief over hidden states, augmented with such task information, is a standard POMDP information state [20].

Let f be a fixed measurable map into a finite or countable shared-state set. The information-state space itself may be continuous. For a shared state $\hat{s}_w$, define its successor-class probability by

$$K_f(\hat{s}_w \mid s, a) := \int_{f^{-1}(\hat{s}_w)} K(\mathrm{d}u \mid s, a).$$

We call f *state-transition-preserving* when states with the same image have the same admissible actions and terminal status/outcome, and, for every common admissible action a,

$$f(s) = f(u) \quad \Longrightarrow \quad \{\hat{s}_w : K_f(\hat{s}_w \mid s, a) > 0\} = \{\hat{s}_w : K_f(\hat{s}_w \mid u, a) > 0\}.$$

This condition equates successor supports while allowing different state-transition probabilities. The vertices of $\mathcal{G}^f_{\text{env}}$ are the shared states reachable from the task's initial distribution under the full interaction process, including feedback after invalid actions. Its action-labelled edges $(\hat{s}_v, a, \hat{s}_w)$ include only admissible actions with $K_f(\hat{s}_w \mid s, a) > 0$ for $f(s) = \hat{s}_v$. Thus every sampled state belongs to the reference vertex set, while invalid state transitions are excluded from both graphs. Successful terminal classes form its success set. Each edge counts as one hop. We write $d^*_{\text{env}}(s)$ for the shortest distance from $f(s)$ to this set, with infinite distance when no such path exists. The reference graph represents the POMDP's information-state transition support.

C.2 Proof of Proposition 4.1 and Cumulative-Sampling Convergence

Proof. Every sampled vertex is reachable under the full interaction process, and every retained edge comes from an observed, environment-admissible state transition. Because the shared-state partition is countable, its observed successor class has positive conditional probability almost surely. Hence every vertex and edge lies in $\mathcal{G}^f_{\text{env}}$; every observed successful endpoint has the same terminal label in that graph. Consequently $\mathcal{G} \subseteq \mathcal{G}^f_{\text{env}}$ almost surely, with a subset of the reference success states.

The state-transition-preserving condition allows combining paths at shared states. Given a finite path $(\hat{s}_{v_0}, a_0, \hat{s}_{v_1}, \dots, a_{\ell-1}, \hat{s}_{v_\ell})$, its first state transition has positive probability from every information state in $f^{-1}(\hat{s}_{v_0})$. The same property holds at each subsequent class. Inductively, following the listed actions has positive probability of realizing the entire class sequence. Terminal-label preservation ensures that a path ending at a successful class ends in success. Thus each abstract path represents a possible environment path, even when its edges were sampled in different trajectories.

Since all paths in $\mathcal{G}$ are also in the reference graph and its observed success set is a subset of the full success set,

$$d_{\mathcal{G}}(f(s)) \geq d^*_{\text{env}}(s).$$

If a shortest reference path and its successful endpoint are represented, the reverse inequality holds as well, giving equality.

For convergence, suppose $\mathcal{G}^f_{\text{env}}$ is finite and trajectories are independent draws from a fixed sampler. Let $\mathcal{C}$ contain its vertices, action-labelled edges, and successful-terminal labels. For every component $c \in \mathcal{C}$, assume the probability p_c that one sampled trajectory contributes c is positive. This requires sufficient exploration and rollout support for each component. Let $\mathcal{G}_n$ be the graph formed from the first n sampled trajectories, retaining their observed terminal labels. Independence gives a missing-component probability $(1 - p_c)^n$. Therefore

$$\Pr(\mathcal{G}_n \neq \mathcal{G}^f_{\text{env}}) \leq \sum_{c \in \mathcal{C}} (1 - p_c)^n \leq |\mathcal{C}| e^{-n p_{\min}}, \qquad p_{\min} := \min_{c \in \mathcal{C}} p_c > 0.$$

Each component has a finite first-appearance time almost surely; there are finitely many components. Their maximum is therefore finite almost surely, after which $\mathcal{G}_n = \mathcal{G}^f_{\text{env}}$. All represented shortest distances then agree with their reference distances. This proves almost-sure convergence. □

The convergence statement concerns cumulative samples with a fixed state map and sampling distribution. During training, RewardFlow constructs a separate graph from each current batch. Environment-invalid state transitions are excluded before paths are formed; success reachability is then determined within the resulting graph. Stochastic decoding randomizes the policy's action choices, whereas $K(\cdot \mid s, a)$ describes the next information state conditional on an action; both sources of randomness are allowed.

C.3 State-Transition-Support Potential in a POMDP

For Rem. 4.2, define the reference potential $\Phi_{\text{env}}(s) := \gamma^{d^*_{\text{env}}(s)}$ from the POMDP's state-transition-support graph. It is 1 at successful terminals and 0 when success is unreachable. Under state-transition-preserving grouping, every finite reference path has positive probability from each information state in its starting class, as shown above. Thus Φ_{env} measures the discounted length of a shortest possible route to success, whether the information-state transitions are deterministic or stochastic.

To relate this potential to expected value, let τ_{succ} be the number of state transitions until successful termination, taking value ∞ when success is never reached. For policies choosing admissible actions, use the terminal-value convention

$$V^\pi(s) := \mathbb{E}^\pi_s[\gamma^{\tau_{\text{succ}}} \mathbf{1}\{\tau_{\text{succ}} < \infty\}], \qquad V^*(s) := \sup_\pi V^\pi(s).$$

A successful terminal has value 1. Almost surely, every finite successful realization follows edges of $\mathcal{G}^f_{\text{env}}$, so $\tau_{\text{succ}} \geq d^*_{\text{env}}(s)$. Its discounted return is therefore at most $\Phi_{\text{env}}(s)$, giving $V^*(s) \leq \Phi_{\text{env}}(s)$ after taking expectations and the supremum over policies. If no success path exists, both quantities are zero.

Equality holds when a policy reaches success in exactly $d^*_{\text{env}}(s)$ state transitions almost surely. Deterministic information-state transitions are one sufficient special case: a shortest supported path

can then be followed with certainty. In general, the only action may reach success with probability $p \in (0,1)$ and terminal failure otherwise, giving $\Phi_{\text{env}}(s) = \gamma$ but $V^*(s) = p\gamma$. The approximation results below compare $\Phi_{\mathcal{G}}$ with the state-transition-support potential Φ_{env}.

C.4 Proof of Proposition 4.3 (State-Potential Approximation)

Proof. Under the POMDP information-state conditions of Rem. 4.2, we compare $\Phi_{\mathcal{G}}(f(s)) = \gamma^{d_{\mathcal{G}}(f(s))}$ with $\Phi_{\text{env}}(s) = \gamma^{d^*_{\text{env}}(s)}$.

Prop. 4.1 gives $\mathcal{G} \subseteq \mathcal{G}^f_{\text{env}}$ and hence

$$d_{\mathcal{G}}(f(s)) \geq d^*_{\text{env}}(s).$$

Since $\gamma \in (0,1)$, the function γ^x is strictly decreasing, so

$$\Phi_{\mathcal{G}}(f(s)) = \gamma^{d_{\mathcal{G}}(f(s))} \leq \gamma^{d^*_{\text{env}}(s)} = \Phi_{\text{env}}(s).$$

The coverage error is

$$\begin{aligned}\delta(s) := \left|\Phi_{\mathcal{G}}(f(s)) - \Phi_{\text{env}}(s)\right| &= \Phi_{\text{env}}(s) - \Phi_{\mathcal{G}}(f(s)) \\ &= \gamma^{d^*_{\text{env}}(s)} - \gamma^{d_{\mathcal{G}}(f(s))} \geq 0.\end{aligned}$$

For a success-reachable state, the error vanishes once the graph contains a shortest reference path from $f(s)$ to success and its successful endpoint; for an unreachable state, both potentials are already zero. Under the cumulative-sampling conditions in Appendix C.2, the distance and potential eventually equal $d^*_{\text{env}}(s)$ and $\Phi_{\text{env}}(s)$ almost surely. These conclusions use only state-transition support and thus hold with stochastic state transitions. □

C.5 Proof of Corollary 4.4 (Action-Level Reward Approximation)

Proof. Since $\Phi_{\mathcal{G}}(f(s)) = \Phi_{\text{env}}(s) - \delta(s)$ by definition of δ, we expand directly:

$$\begin{aligned}\tilde{r}(s,a,s') &= \Phi_{\mathcal{G}}(f(s')) - \Phi_{\mathcal{G}}(f(s)) \\ &= \big(\Phi_{\text{env}}(s') - \delta(s')\big) - \big(\Phi_{\text{env}}(s) - \delta(s)\big) \\ &= \big(\Phi_{\text{env}}(s') - \Phi_{\text{env}}(s)\big) - \big(\delta(s') - \delta(s)\big).\end{aligned}$$

Nonnegativity of the two coverage errors gives

$$\left|\tilde{r}(s,a,s') - \big(\Phi_{\text{env}}(s') - \Phi_{\text{env}}(s)\big)\right| = |\delta(s') - \delta(s)| \leq \max\{\delta(s), \delta(s')\}.$$

By Prop. 4.3, both errors vanish when the relevant shortest paths and their successful endpoints are covered, so the action reward recovers $\Phi_{\text{env}}(s') - \Phi_{\text{env}}(s)$. The same conclusion holds almost surely under cumulative sampling.

Since $\tilde{r}(s,a,s') = \gamma^{d_{\mathcal{G}}(f(s'))} - \gamma^{d_{\mathcal{G}}(f(s))}$ and γ^x is strictly decreasing, the sign of the raw action reward identifies progress through the two sampled-graph distances. □

C.6 Proof of Proposition 4.5 (Trajectory-Level Reward Ordering)

Proof. For a trajectory whose state transitions are all retained, $\tilde{r}^{(i)}_t = \phi^{(i)}_{t+1} - \phi^{(i)}_t$ at every turn, so the sum of raw action rewards telescopes:

$$\sum_{t=0}^{T_i-1} \tilde{r}^{(i)}_t = \phi^{(i)}_{T_i} - \phi^{(i)}_0.$$

The successful and failed trajectories share the same canonical initial node $\hat{s}_{\text{init}}$, so $\phi^{(+)}_0 = \phi^{(-)}_0 = \Phi_{\mathcal{G}}(\hat{s}_{\text{init}})$ cancels:

$$\begin{aligned}\sum_{t=0}^{T_+-1} \tilde{r}^{(+)}_t - \sum_{t=0}^{T_--1} \tilde{r}^{(-)}_t &= \big(\phi^{(+)}_{T_+} - \Phi_{\mathcal{G}}(\hat{s}_{\text{init}})\big) - \big(\phi^{(-)}_{T_-} - \Phi_{\mathcal{G}}(\hat{s}_{\text{init}})\big) \\ &= \phi^{(+)}_{T_+} - \phi^{(-)}_{T_-}.\end{aligned}$$

This holds for any state graph, regardless of coverage or approximation quality. With $\phi^{(+)}_{T_+} = 1$, strict ordering holds if the failed endpoint is not mapped to a successful node, so $\phi^{(-)}_{T_-} < 1$. The gap equals 1 when the failed endpoint has no path to observed success and thus has potential zero. A timeout can end at a success-reachable nonterminal with positive potential. This identity applies to raw action rewards before state-group normalization. □

D Further Discussion

D.1 Relation to Potential-Based Reward Shaping and Reward Machines

RewardFlow uses $\tilde{r}(s, a, s') = \Phi_{\mathcal{G}}(f(s')) - \Phi_{\mathcal{G}}(f(s))$ to turn a node potential into an action reward. Scalar-potential differences are established in control and RL [31, 35, 38]; the question here is how current trajectories supply the potential. RewardFlow constructs $\Phi_{\mathcal{G}}$ from state transitions and successful endpoints observed across trajectories, without a predefined task graph or a learned process scorer.

Potential-based reward shaping (PBRS) [34] uses $F_{\mathrm{PBRS}}(s, s') = \gamma\Psi(s') - \Psi(s)$ for a scalar potential Ψ in the discounted setting, with its applicable terminal conditions. RewardFlow uses the consecutive difference of its graph potential. Its finite-trajectory sum is $\phi^{(i)}_{T_i} - \phi^{(i)}_0$. This identity concerns raw action rewards before state-group normalization and their combination with trajectory-level advantages. Appendix C derives the corresponding reward-ordering result.

Reward Machines [15, 16] and the finite-state automaton (FSA) induction work of Furelos-Blanco et al. [11] likewise structure rewards using finite automata, but these methods rely on either hand-specified or learned symbolic abstractions. RewardFlow constructs nodes from preprocessed observations and rebuilds its state graph from each batch's trajectories. Tab. 7 compares how these methods obtain and use their reward structures.

Table 7: Conceptual roles of reward-shaping paradigms. PBRS specifies a reward transformation rather than a unique procedure for obtaining its potential.

	PBRS	**Reward Machines / learned FSAs**	**RewardFlow**
Required input	A scalar potential	Symbolic events and a specified or learned automaton	Canonical observations, observed state transitions, and outcomes
Main operation	Transform rewards given a potential	Track symbolic reward-relevant structure	Estimate potentials over the state graph
Structure source	Not prescribed by the shaping theorem	Specification or automaton induction	Current on-policy trajectory group

RewardFlow joins observed paths across trajectories before assigning action rewards. The following example shows why that changes the information available compared with computing returns along each trajectory separately.

D.2 Shared-State Grouping versus Graph-Based Reward Estimation

GiGPO [9] already supplies action-level rewards: it compares trajectory-local discounted returns among actions sampled at the same anchor state. RewardFlow uses shared states additionally as graph nodes, merges their observed state transitions, and propagates over the combined topology before computing state-conditioned action advantages. Plain trajectory propagation, state grouping, and graph propagation are therefore different operations.

A minimal deterministic example. Assume exact state identity, terminal rewards $0/1$, and $\gamma = 0.9$. Let $s_{\mathrm{A}}, s_{\mathrm{B}}, s_{\mathrm{C}}$ name three nonterminal states, and $s_{\mathrm{F}}, s_{\mathrm{S}}$ failed and successful terminals. Consider the two observed trajectories

$$\tau^{(1)} : s_{\mathrm{A}} \xrightarrow{a} s_{\mathrm{C}} \xrightarrow{b} s_{\mathrm{F}},$$

$$\tau^{(2)} : s_{\mathrm{A}} \xrightarrow{c} s_{\mathrm{B}} \xrightarrow{d} s_{\mathrm{C}} \xrightarrow{e} s_{\mathrm{S}}.$$

Each named state maps to its correspondingly named canonical node, for example $f(s_{\mathrm{C}}) = \hat{s}_{\mathrm{C}}$. For trajectory returns, the terminal reward is attached to the final state transition. The merged graph instead has terminal potential $\Phi_{\mathcal{G}}(\hat{s}_{\mathrm{S}}) = 1$ and node potentials $\Phi_{\mathcal{G}}(\hat{s}_{\mathrm{F}}) = 0$, $\Phi_{\mathcal{G}}(\hat{s}_{\mathrm{C}}) = 0.9$, $\Phi_{\mathcal{G}}(\hat{s}_{\mathrm{B}}) = 0.81$, and $\Phi_{\mathcal{G}}(\hat{s}_{\mathrm{A}}) = 0.81$. Tab. 8 illustrates the resulting advantages by subtracting each state group's mean reward for both methods.

Table 8: Illustrative raw signals and centered advantages at shared nodes $\hat{s}_{\mathrm{A}}$ (actions a, c) and $\hat{s}_{\mathrm{C}}$ (actions b, e). Entries are ordered pairs.

Method	Raw (a, c)	Centered (a, c)	Raw (b, e)	Centered (b, e)
GiGPO	$(0, 0.81)$	$(-0.405, 0.405)$	$(0, 1)$	$(-0.5, 0.5)$
RewardFlow	$(0.09, 0)$	$(0.045, -0.045)$	$(-0.9, 0.1)$	$(-0.5, 0.5)$

Both methods distinguish b from e at $\hat{s}_{\mathrm{C}}$. GiGPO assigns a the return recorded from that action onward in the failed trajectory. RewardFlow rewards a for reaching $\hat{s}_{\mathrm{C}}$, where another trajectory supplies the successful action e, while penalizing b. The graph path $\hat{s}_{\mathrm{A}} \xrightarrow{a} \hat{s}_{\mathrm{C}} \xrightarrow{e} \hat{s}_{\mathrm{S}}$ composes existing edges without adding a state transition.

D.3 State Potentials and Directed State Transitions

Potentials at shared nodes. Reverse BFS assigns one distance and hence one potential to each canonical node, including nodes with several observed successors. Under the state-transition-preserving assumptions in Sec. 4.4, shared information states admit the same actions and possible successor classes, while allowing stochastic state transitions. Each action occurrence then receives the potential difference along its observed state transition.

Directed state transitions and dead ends. Every retained edge records an observed directed state transition. Shortest-hop distance measures proximity to success through the support of these state transitions; Appendix C relates the resulting potential to expected value under stochastic state transitions. Reverse BFS follows incoming graph edges to compute distances, while environment state transitions retain their original direction. A state transition from a success-reachable node to an unreachable dead end has negative potential difference.

D.4 Exploration and Graph Coverage

Graph construction from sampled experience. RewardFlow constructs a graph from the current task's sampled trajectories. After node construction, reverse BFS costs $O(|\mathcal{V}| + |\mathcal{E}|)$ in the observed nodes and edges. Rollout generation and state preprocessing have their own computational costs. Successful endpoints seed propagation through the observed state transitions, and additional trajectories can expose further paths to success.

No successful trajectory. With no successful endpoint in a trajectory group, $\mathcal{V}_{\mathrm{succ}} = \varnothing$, every graph potential and consecutive difference is zero, giving zero graph-derived action advantages. In the binary-success-only case, the trajectory advantage is also zero under the nonzero-denominator convention in Appendix B.3, leaving only the KL term. In the experimental recipe, environment-return differences can also contribute to the advantages (Appendix E.1). Conversely, if every trajectory succeeds and binary success is the only reward, equal outcomes give zero trajectory advantage, while potential differences can supply nonzero action advantages.

Graph budget in search tasks. The four-step, five-trajectory search budget supplies at most twenty state transitions per group before merging, fewer than the structured-task budgets. Cross-trajectory paths are formed within this sampled graph. Tab. 2 reports improvements over GiGPO at both model scales, and Appendix B.1 describes the state interfaces used in search tasks and supplementary mathematical reasoning.

Rollout-budget sensitivity. Tab. 16 reports higher success than GiGPO at rollout budgets of 4, 6, and 8 on ALFWorld.

Appendix E.13 reports how entropy, graph size, and success vary during training. The state-graph cases in Appendix E.16 illustrate connectivity across one or more sampled trajectories.

D.5 Graph-Based Rewards versus Learned Process Scorers

Process supervision has been studied in mathematical reasoning [26] and agentic interaction. For long-horizon tasks, a process scorer must account for actions, observations, and their changing relationship to task completion. Comparisons should distinguish learned scorers from methods that derive action rewards from outcomes: GiGPO [9], for example, already combines action and trajectory advantages without a separate learned PRM. RewardFlow additionally uses the topology of observed state transitions across trajectories, as illustrated in Appendix D.2. It does not require human step labels or an auxiliary learned process-reward scorer.

Fixed per-step scorers can become misaligned with a changing policy. RewardFlow instead reconstructs its graph from current trajectories and observed outcomes to update action rewards. Appendix E.2 provides the PRM comparison results and training protocol; Appendix E.3 examines frozen-scorer transfer to new policies.

D.6 Learned Graph Models for Reward Estimation

A graph neural network (GNN) could use state and action features as well as connectivity. Such an extension would require a training objective, supervision source, and update schedule, along with additional optimization and inference.

RewardFlow uses BFS to compute the distance to observed success directly. Its cost is linear in the constructed graph after node processing, and the resulting distances determine potentials without fitting another model.

E Further Experiments

E.1 Detailed Experimental Setting

Evaluation benchmarks. The four main benchmarks expose different observations and actions:

- **ALFWorld** [43] is a synthetic text-based simulator aligned with the embodied benchmark ALFRED [42], comprising 3,827 household tasks spanning six types: *Pick & Place* (Pick), *Examine in Light* (Look), *Clean & Place* (Clean), *Heat & Place* (Heat), *Cool & Place* (Cool), and *Pick Two & Place* (Pick2).
- **WebShop** [56] is a large-scale benchmark where the agent must fulfill natural-language shopping instructions by issuing text commands such as "search" and "click" to web pages with over 1.18 million real Amazon products.
- **Sokoban** [39] is a classic puzzle game in which agents must observe and analyze the grid layout and push boxes onto target locations, taxing spatial understanding and long-horizon planning. In this work, we randomly generate 6×6 Sokoban boards for training and evaluation.
- For **search tasks** [18], agents answer knowledge-intensive questions using a search engine to retrieve documents. The environment state consists of retrieved documents, and the agent's actions issue search queries or submit a final answer. Retrieved passages can vary between queries, making exact observation matching insufficient for sharing research progress. We evaluate on single-hop tasks: Natural Questions (NQ) [22], TriviaQA [19], PopQA [30], and multi-hop tasks: HotpotQA [55], 2WikiMultiHopQA (2Wiki) [13], Musique [47], and Bamboogle [36]. Training uses NQ and HotpotQA; the other five datasets evaluate out-of-distribution transfer.

ALFWorld out-of-distribution evaluation. The evaluation uses familiar objects in unseen rooms and layouts (Tab. 3).

Metrics and evaluation repeats. Success rate is the fraction of evaluated episodes that satisfy the environment's task-completion check. For ALFWorld, All pools task-completion outcomes across evaluated episodes, while the six subtask columns report rates within each task type. WebShop and Sokoban additionally report their task scores. WebShop's task score measures how well the purchased product matches the requested attributes and is displayed on a 0-100 scale; Sokoban's task

score measures progress in solving the puzzle. For search tasks, we report answer accuracy, with an unweighted mean across the listed datasets. Rate improvements are reported as absolute differences in %.

Tabs. 1 and 3 report variation across three inference seeds at fixed checkpoints. Appendix E.5 specifies the separate 128-task in-distribution evaluation used for the ALFWorld-3B larger-budget and preprocessing comparisons.

Training configuration. Tab. 9 records the training and algorithm settings for the main experiments, implemented with Verl-Agent [9]. Supplementary controls state their differing configurations in their respective subsections.

Table 9: Details of training configuration of experiments.

Type	Hyperparameter	ALFWorld	WebShop	Sokoban	Search tasks
		Qwen2.5-(VL)-1.5B/3B-Instruct			
Learning	Training batch size	16	16	16	256
Learning	Minibatch size for policy update	256	64	256	512
Learning	Max action turns per trajectory	25	15	15	4
Learning	Max prompt length	2048	4096	1024	4096
Learning	Max response length	512	512	512	512
Learning	Training steps	100	100	100	200
Learning	Temperature	0.4	0.4	0.4	0.4
Learning	Learning rate	1e-6	1e-6	1e-6	1e-6
Learning	Invalid action penalty	0.1	0.1	0.1	0.01
Learning	Computational cost	2x(A100 & h20)	2x(A100 & h20)	2x(A100 & h20)	2x(A100 & h20)
		Qwen2.5-(VL)-7B-Instruct			
Learning	Training batch size	16	16	16	256
Learning	Minibatch size for policy update	256	64	256	512
Learning	Max action turns per trajectory	25	15	15	4
Learning	Max prompt length	2048	4096	1024	4096
Learning	Max response length	512	512	512	512
Learning	Training steps	100	100	100	200
Learning	Temperature	0.4	0.4	0.4	0.4
Learning	Learning rate	1e-6	1e-6	1e-6	1e-6
Learning	Invalid action penalty	0.1	0.1	0.1	0.01
Learning	Computational cost	4x(A100 & h20)	4x(A100 & h20)	4x(A100 & h20)	4x(A100 & h20)
		GRPO			
Algorithm	Group size	8	8	8	5
Algorithm	KL-divergence loss coefficient	0.01	0.01	0.01	0.001
		RLOO			
Algorithm	Group size	8	8	8	5
Algorithm	KL-divergence loss coefficient	0.01	0.01	0.01	0.001
		GiGPO			
Algorithm	Discount factor γ	0.95	0.95	0.95	0.95
Algorithm	weighting coefficient w	1	1	1	1
Algorithm	Group size	8	8	8	5
Algorithm	KL-divergence loss coefficient	0.01	0.01	0.01	0.001
		RewardFlow			
Algorithm	Discount factor γ	0.90	0.90	0.90	0.90
Algorithm	Max graph propagation iteration	1000	1000	1000	1000
Algorithm	Action-level advantage weight α_{act}	1	1	1	1
Algorithm	Trajectory-level advantage weight α_{traj}	1	1	1	1
Algorithm	Group size	8	8	8	5
Algorithm	KL-divergence loss coefficient	0.01	0.01	0.01	0.001

Reward assignment. The training recipe uses environment episode returns for outcome-based supervision. Graph cleanup selects state transitions for propagation, while all active generated responses remain in the policy-update batch. Graph-derived rewards are mapped back to their response positions; missing node potentials are zero, and response rows without a mapped graph reward retain their environment reward for the corresponding state transition as the action-reward entry. Each response receives its trajectory's episode return as its outcome score.

Advantage computation and token masks. Action-level rewards are grouped by task and source state, including active responses excluded from graph propagation. Singleton groups receive zero action-level advantage. Task-group baselines are computed by averaging outcome scores over active responses. The action- and trajectory-level advantages are combined and broadcast to generated response tokens; padding is masked.

E.2 Comparison with Process Reward Models

Fig. 1 reports comparisons against three baselines using PRM-estimated rewards (LLM-as-PRM with Qwen2.5-7B-Instruct, ArmoRM-8B, and Selene-8B) and plain GRPO on ALFWorld with Qwen2.5-1.5B-Instruct.

PRM comparison protocol. All three comparisons use the ALFWorld Qwen2.5-1.5B-Instruct setting in Appendix E.1: 100 training steps, task batch size 16, eight rollouts per task, a maximum of 25 action turns, sampling temperature 0.4, and learning rate 10^{-6}. Each PRM supplies the action-level reward for the generated response in its interaction context. Actions are grouped using GiGPO's state-grouping procedure; their PRM rewards provide the process-level advantage, which is combined with the outcome-based trajectory advantage. The policy, environment, training budget, and evaluation procedure follow the ALFWorld-1.5B setup, while the action-reward source changes to the corresponding PRM. Plain GRPO supplies the outcome-only reference at 50.0% success.

Comparison results. RewardFlow achieves 62.8% success, with absolute improvements over the strongest PRM baseline (GRPO + Selene-8B at 48.4%) of 14.4% and over the no-PRM GRPO baseline (50.0%) of 12.8%.

E.3 Fixed-Scorer Transfer Diagnostics

Complementing the PRM training comparison in Appendix E.2, the following diagnostics examine transfer with two additional agent-oriented scorers.

We examine released AgentPRM[2] and Web-Shepherd [2] scorers as frozen process-reward sources for Qwen2.5-1.5B policies in ALFWorld and WebShop, respectively. The AgentPRM integration scores responses from the Qwen policy with a fixed reward model. The Web-Shepherd integration uses a single-judgment probability adapter.

AgentPRM: action admissibility versus a high score. The run was stopped near step 57, with checkpoint 50 retained for diagnosis. Four sampled trajectories contain 100 action turns. All 100 parsed action strings are outside the pre-action admissible-action lists, yet their mean raw scorer logit is 97.785; none of the four trajectories succeeds. On these same 100 states, responses from the released AgentPRM policy are all admissible and have mean logit 0.191. The transferred Qwen policy frequently combines several commands into one action. The integration checked output syntax but did not enforce candidate membership before assigning the process reward. The reported scorer outputs are unbounded raw logits.

Web-Shepherd: proxy/outcome divergence. The one-judgment run completed 100 training steps with batch size 16, eight trajectories per task, horizon 15, and training seed 0. Comparing training-window means over steps 1-10 and 91-100, the raw process score rises from 0.1931 to 0.9111, while environment success rate decreases from 3.89% to 0.56%. The checkpoint-100 evaluation on 128 tasks per inference seed $(0, 1, 2)$ yields success rates 1.5625%, 1.5625%, and 0.7813%: mean 1.3021% with population standard deviation 0.3683%. The adapter's historical trajectory input contains past thoughts/actions but not past page observations; inspected traces narrate unsupported progress. The increasing process score and declining task success show proxy/outcome divergence in this adapter-policy pairing.

Implication for graph-based reward estimation. These diagnostics show divergence between transferred process scores and environment outcomes. RewardFlow instead computes action rewards from observed state transitions and terminal outcomes.

E.4 State Preprocessing and Invalid-Action Filtering

Tab. 10 shows the contribution of state preprocessing and invalid-action filtering in RewardFlow on ALFWorld (Qwen2.5-3B-Instruct). Removing either state preprocessing or invalid-action filtering

[2]Here AgentPRM refers to Choudhury, *Process Reward Models for LLM Agents: Practical Framework and Directions* (2025), https://arxiv.org/abs/2502.10325; released implementation: https://github.com/sanjibanc/agent_prm.

lowers success in this configuration. Methodological details of state preprocessing and invalid-action filtering are provided in Appendices B.1 and B.2 respectively.

Table 10: Performance of RewardFlow with or without each state/action interface component on ALFWorld using Qwen2.5-3B-Instruct. Success rate (%).

Method	**Success Rate (%)**
Base	13.5
RewardFlow (w/o state preprocessing)	53.9
RewardFlow (w/o invalid-action filtering)	60.2
RewardFlow	**75.5**

E.5 Separating Reward Estimation from Preprocessing and Filtering

To distinguish the state preprocessing and invalid-action filtering from graph-based action-level reward estimation, we additionally evaluate GiGPO with RewardFlow's state preprocessing and invalid-action filtering enabled, but with RewardFlow graph construction and propagation disabled. Tab. 11 presents this control alongside the ALFWorld-3B rows from Tab. 15. Each method uses one training run (seed 0), evaluated at step 150 on the same 128-task in-distribution split, with one sampled trajectory per task, horizon 50, and temperature 0.4. The three repeats vary inference seeds $0, 1, 2$; error bars are population standard deviations of these three success rates. The additional control has success counts $104/128$, $99/128$, and $99/128$.

Table 11: Preprocessing control on ALFWorld-3B. Success rate (%), mean $\pm$ population std across three inference seeds at fixed checkpoints after 150 training steps. The outer rows also appear in Table 15.

Method	Success rate (%)
GiGPO	75.2 ± 1.6
GiGPO + RewardFlow preprocessing/filtering	78.6 ± 1.8
RewardFlow	82.8 ± 0.7

Adding RewardFlow's state preprocessing and invalid-action filtering improves GiGPO by 3.4% in absolute success rate. The complete RewardFlow estimator improves success by a further 4.2%. Tab. 10 separately reports single-component removals.

E.6 Semantic-Aggregation Threshold Sensitivity

Tab. 12 gives the values underlying the Qwen2.5-3B threshold analysis on search tasks in Fig. 10. Lower cosine thresholds permit broader merging; higher thresholds retain finer distinctions.

Table 12: Average accuracy (%) on search tasks across semantic-aggregation thresholds.

Threshold τ	0.80	0.85	0.90	0.95	1.00
Average accuracy	43.8	44.7	44.3	45.0	44.2

Accuracy ranges from 43.8% to 45.0% across cosine-similarity thresholds 0.80-1.00, an absolute spread of 1.2%.

E.7 Shortest-Path versus Mean Propagation

We compare maximum and mean aggregation in decay-based graph propagation. Using the unit-success potential convention of Sec. 4.1, potentials start at 1 for successful terminals and 0 elsewhere. At each iteration, maximum aggregation takes the largest value among a node's current potential and its successors' discounted potentials. This recovers the shortest-hop potential in Eq. (2), since shorter distances give larger discounted values. The mean variant averages these same candidates. Writing

$\mathcal{N}^+(\hat{s}_v)$ for the distinct successor nodes, its update is

$$\Phi_{\ell+1}(\hat{s}_v) = \frac{\Phi_\ell(\hat{s}_v) + \gamma \sum_{\hat{s}_w \in \mathcal{N}^+(\hat{s}_v)} \Phi_\ell(\hat{s}_w)}{1 + |\mathcal{N}^+(\hat{s}_v)|}.$$

Both variants update all nodes synchronously for the configured 1,000 iterations. Terminal sinks retain their initial potentials. Nodes with no path to success remain at zero; with no successful terminal, all potentials remain zero. Mean aggregation combines successor values even when the graph has only one successful terminal, whereas maximum aggregation selects the strongest propagated value. Maximum aggregation performs better in both comparisons, including a 15.9% absolute success-rate improvement on Sokoban (Tab. 4).

Appendix E.8 additionally compares BFS with personalized PageRank (PPR).

E.8 BFS vs. Personalized PageRank (PPR)

BFS propagation yields absolute success-rate improvements over personalized PageRank of 10.5% on ALFWorld and 14.3% on Sokoban (Tab. 4). BFS uses hop distance to define $\Phi_{\mathcal{G}}(\hat{s}_v) = \gamma^{d_{\mathcal{G}}(\hat{s}_v)}$ for success-reachable nodes, whereas PPR assigns mass through a random-walk computation. BFS achieves higher success in both comparisons.

E.9 Sensitivity to α_{act} and α_{traj}

We vary the action- and trajectory-level advantage weights to examine their complementary roles.

Table 13: Ablation on the action-level and trajectory-level advantage weights $(\alpha_{\text{act}}, \alpha_{\text{traj}})$ on ALFWorld using Qwen2.5-3B-Instruct. Success rate (%); higher is better. Default $(1, 1)$ performs best among the tested weight pairs.

α_{act}	α_{traj}	**ALFWorld 3B (%)**
0	1	59.4
1	0	7.0
1	0.5	68.8
0.5	1	67.2
1	1	**75.5**

Using only the action advantage obtains a 7.0% success rate. Combining action and trajectory advantages at $(\alpha_{\text{act}}, \alpha_{\text{traj}}) = (1, 1)$ yields a 16.1% absolute success-rate improvement over using only the trajectory advantage (75.5% vs. 59.4%), supporting the complementary roles of both signals.

E.10 Longer Training

We continue the main Sokoban checkpoints for an additional 100 steps (total 200 steps) with the same hyperparameters.

Table 14: Extended training on Sokoban (Qwen2.5-VL-3B-Instruct), comparing success after 100 and 200 steps.

Method	**Success @100 steps (%)**	**Success @200 steps (%)**	**Absolute Δ (%)**
RLOO	16.7	41.4	**+24.7**
GRPO	22.9	39.1	+16.2
GiGPO	19.8	33.6	+13.8
RewardFlow (ours)	**49.5**	**70.3**	+20.8

RewardFlow retains its lead at the later checkpoint: its absolute success-rate gap over the strongest baseline is 26.6% at 100 steps and 28.9% at 200 steps (Tab. 14). The benefit therefore persists under this extension.

E.11 Comparison under GiGPO's Original Training Settings

We compare GiGPO and RewardFlow using the larger training settings reported by GiGPO [9]: 150 steps for ALFWorld and WebShop, 200 steps for Sokoban, an ALFWorld interaction horizon of 50,

and a Sokoban training batch size of 32. Tab. 15 reports mean ± standard deviation over 3 inference seeds under this protocol. For ALFWorld-3B, both methods use training seed 0 and checkpoint 150; the same two estimates are reused in the preprocessing control, whose full evaluation protocol is in Appendix E.5.

Our GiGPO reproduction is below the originally reported performance. We compare both methods using the same specified training settings on NVIDIA H20 and A100 GPUs.

Table 15: Comparison of GiGPO and RewardFlow under GiGPO's original training settings (150 steps for ALFWorld/WebShop, 200 steps for Sokoban, batch size 32 for Sokoban). Success rate (%, mean ± std over 3 inference seeds at fixed checkpoints); **bold** indicates the better method.

Method	ALFWorld 1.5B	ALFWorld 3B	Sokoban 3B	WebShop 1.5B
GiGPO	74.2 ± 1.7	75.2 ± 1.6	78.4 ± 2.6	60.2 ± 2.2
RewardFlow	$\mathbf{76.8 \pm 1.0}$	$\mathbf{82.8 \pm 0.7}$	$\mathbf{81.8 \pm 2.3}$	$\mathbf{66.9 \pm 1.5}$

RewardFlow exceeds GiGPO in all four larger-training-budget comparisons.

E.12 Rollout-Budget Sensitivity

We vary the rollout budget while comparing RewardFlow with GiGPO. Tab. 16 reports the resulting success rates and graph sizes. RewardFlow exceeds GiGPO at budgets of 4, 6, and 8, including an absolute success-rate improvement of 13.3% with 4 trajectories.

Table 16: Ablation on number of trajectories per training task using Qwen2.5-1.5B-Instruct in ALFWorld. Success rate (%) reported; RewardFlow results include average graph statistics.

Method	Trajectories	Avg. Nodes	Avg. Edges	Success Rate (%)
GiGPO	4	-	-	28.9
	6	-	-	51.6
	8	-	-	50.8
RewardFlow	4	17.1	28.8	**42.2**
	6	22.9	41.3	**53.1**
	8	28.8	55.9	**62.8**

E.13 Graph Size and Runtime

Tab. 17 reports graph sizes for 8 sampled trajectories per task. ALFWorld and WebShop use Qwen2.5-1.5B-Instruct with horizons of 25 and 15 interactive steps, respectively; Sokoban uses Qwen2.5-VL-3B-Instruct with horizon 15. The separate runtime profile in Fig. 1 (right) reports graph construction and reward propagation times for Qwen2.5-3B-Instruct on ALFWorld and WebShop and Qwen2.5-VL-3B-Instruct on Sokoban. These graph operations use no additional model calls. LLM-based state construction has a separate inference cost.

Table 17: Average and maximum size of induced state graphs from 8 on-policy trajectories (Qwen2.5-1.5B/VL-3B-Instruct). Maxima refer to the sampled graphs in these configurations.

Environment	Avg. Nodes	Avg. Edges	Max Nodes	Max Edges
ALFWorld	28.8	55.9	42	77
WebShop	36.8	48.2	54	65
Sokoban	14.0	21.0	27	39

Entropy and graph size during training. We examine how policy entropy, retained graph size, and success vary across checkpoints. We measure policy entropy as $H(p) = -\sum_{z \in \mathcal{W}} p(z) \log p(z)$, where $\mathcal{W}$ is the token vocabulary and z a vocabulary token; $\mathcal{W}$ is distinct from the graph-node set $\mathcal{V}$. To study the interplay between entropy and graph structure, we evaluate three checkpoints of the Qwen2.5-1.5B-Instruct model during RewardFlow training on ALFWorld: step 0 (initial supervised model), step 50, and step 100. For each checkpoint, we collect $G = 8$ trajectories per task using group sampling, compute the entropy, and record graph statistics along with validation performance.

Table 18: Training progress of Qwen2.5-1.5B-Instruct on the target task. Entropy, invalid rates, graph size, and success rate across training steps.

Training Step	Entropy	Invalid State Rate (%)	Invalid Action Rate (%)	Avg. Node	Avg. Edge	Success Rate (%)
0	1.099	33.1	41.0	24.1	46.9	4.1
50	0.555	0.3	0	33.1	55.9	44.5
100	0.295	0.1	0	19.9	30.6	62.8

At step 0, the policy has the highest entropy among these checkpoints (1.099) and generates many invalid actions, which are removed during graph construction. At step 50, entropy is lower (0.555), invalid actions are rare, and the observed graphs are largest among these checkpoints (33.1 nodes and 55.9 edges on average). At step 100, entropy is lowest (0.295), the graphs are smaller (19.9 nodes and 30.6 edges), and validation success is highest (62.8%). Across these checkpoints, success increases as invalid actions decrease, while graph size first increases and then decreases.

E.14 Offline Synthesis of a State Normalizer

We test whether an executable ALFWorld normalizer can be synthesized without exposing the handwritten normalizer or its reward outputs to the proposer and validation judge. The GPT-5.6 proposer/judge procedure uses reward-blind interaction traces and feedback on state-transition consistency and merge consistency. Its collection contains 1,024 trajectories over 256 task identities, split by task into 161 development, 49 validation, and 46 final-test identities. The reported R04 candidate uses merge supervision.

After freezing the candidate, we compare its induced action rewards with the handwritten interface on a separate fixed comparison dump of 128 trajectories (16 groups of eight, 2,407 state transitions), using potential discount factor 0.95. This post-hoc agreement assessment applies the same trajectory cleaner to both normalizers.

Table 19: Post-hoc action-reward agreement with the handwritten normalizer on the same fixed trajectories. Exact agreement compares reward values; direction agreement compares positive, zero, or negative signs.

Normalizer	Exact (%)	Direction (%)	Nonzero-support exact (%)
Identity	84.67	90.78	34.11
Frozen merge-supervised R04	95.26	96.88	77.69

The absolute increases in all-action exact and direction agreement are 10.59% and 6.10%. Because both interfaces assign zero reward to many actions, we also report exact agreement over the union of their nonzero-reward supports; the R04 figure is $397/511 = 77.69\%$, and its full-dump reward correlation is 0.601. The synthesized normalizer more closely reproduces the reference action rewards on this comparison dump.

E.15 Supplementary Math-TIR Evaluation

Mathematical tool-integrated reasoning (Math-TIR) is a setting in which a model solves math problems by alternating written reasoning with Python execution. The Structured-Decoding variant identifies nodes by exact interaction histories and branches generation at reasoning/tool-action boundaries (Appendix B.1). Its baseline, Graph-GRPO, runs GRPO with the same branching rules. Both use Qwen2.5-3B-Instruct, training seed 0, and checkpoints after 200 updates. We also report a RewardFlow variant that uses a frozen Qwen2.5-14B state normalizer as its state-construction interface.

Here mean@8 is average sample accuracy, pass@8 is the fraction of problems with at least one correct sample, and maj@8 is majority-vote accuracy. Structured-Decoding improves mean@8 on MATH-500 and AIME26 and pass@8 on AIME26; its AIME24 and AIME25 results are lower or tied across metrics. These experiments extend graph-based action-level reward estimation to mathematical reasoning through two state-construction interfaces. The frozen-LLM entries retain the source report's two-decimal percentage precision.

Table 20: Supplementary Math-TIR results (%). Offline audited accuracy on MATH-500 (500 problems) and AIME24/25/26 (30 each), with eight samples per problem and inference seed 0.

Benchmark	Method / state interface	mean@8	pass@8	maj@8
MATH-500	Graph-GRPO / exact prefix	62.250	82.600	71.400
	RewardFlow / Structured-Decoding	62.775	82.000	70.600
	RewardFlow / frozen LLM normalization	59.80	81.60	70.20
AIME24	Graph-GRPO / exact prefix	7.500	20.000	13.333
	RewardFlow / Structured-Decoding	6.667	20.000	10.000
	RewardFlow / frozen LLM normalization	7.08	16.67	10.00
AIME25	Graph-GRPO / exact prefix	5.000	16.667	10.000
	RewardFlow / Structured-Decoding	2.500	13.333	3.333
	RewardFlow / frozen LLM normalization	3.33	10.00	6.67
AIME26	Graph-GRPO / exact prefix	4.583	13.333	10.000
	RewardFlow / Structured-Decoding	6.667	26.667	10.000
	RewardFlow / frozen LLM normalization	6.67	16.67	13.33

Scoring audit. The reported metrics use offline `qwen_math` scoring of the completed evaluations. Audits found 19 online false negatives for Structured-Decoding and 16 for Graph-GRPO among 4,720 samples, all on MATH-500, due to a symbolic-equivalence fallback that fails in worker threads. One additional Structured-Decoding sample was initially over-counted by offline reconstruction from a stale earlier boxed answer; the audited MATH-500 mean is corrected to $2511/4000 = 62.775\%$. The LLM-normalized evaluation records 15 analogous online/offline discrepancies among 4,000 MATH-500 samples; independent offline rescoring matches its reported metrics.

E.16 State Graph Cases

We show the original state-graph visualizations constructed from 1, 2, and 3 sampled rollouts in ALFWorld, WebShop, and Sokoban. The WebShop and Sokoban rollouts were sampled from checkpoints after 100 training steps. The figures retain their original node and edge color scales and illustrate the state transitions shared across rollouts. Node 0 is the initial state, and ★ marks task completion. The accompanying lists retain the observations and action strings, including different states reached at the same location.

ALFWorld. These cases connect alternative routes through shared household states. The accompanying observations track the kettle's status during cleaning and placement.

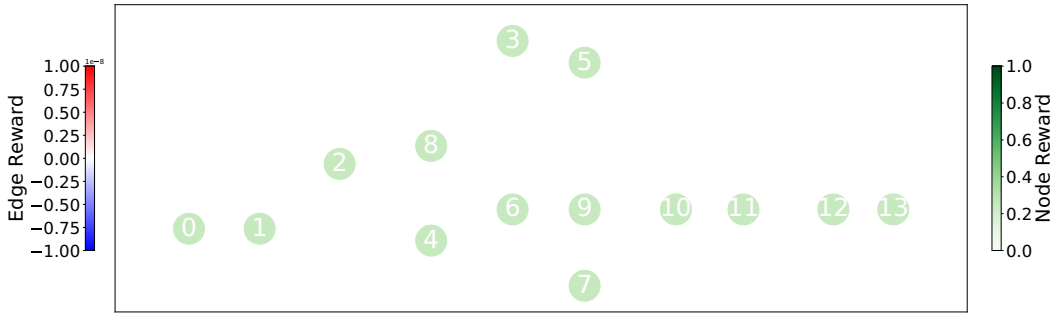


Figure 11: The original state graph constructed from 1 sampled rollout in ALFWorld. Darker nodes indicate higher values on the source plot's color scale. Node 0 is the initial state, and ★ is the successful terminal state. Node and action descriptions appear in Figures 14 and 15.

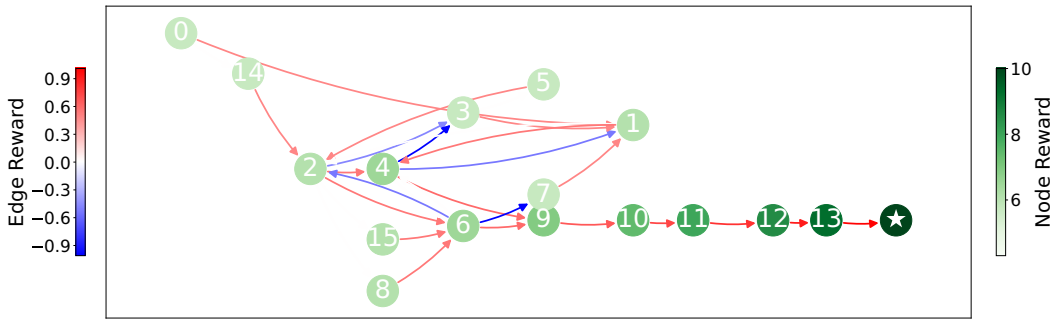


Figure 12: The original state graph constructed from 2 sampled rollouts in ALFWorld. Darker nodes indicate higher values on the source plot's color scale. Node 0 is the initial state, and ★ is the successful terminal state. Node and action descriptions appear in Figures 14 and 15.

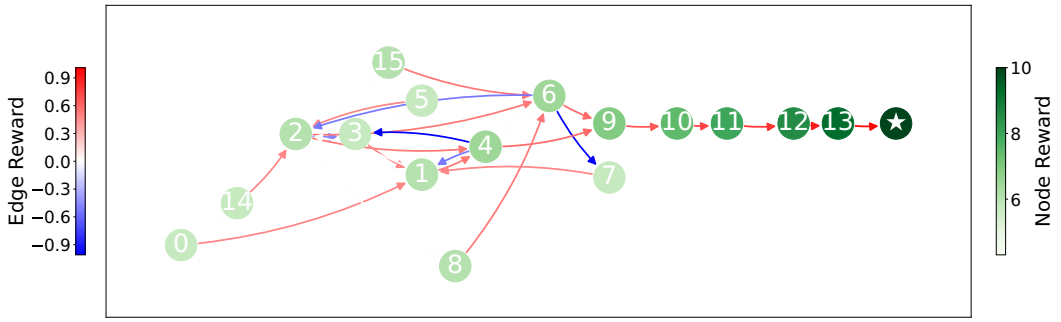


Figure 13: The original state graph constructed from 3 sampled rollouts in ALFWorld. Darker nodes indicate higher values on the source plot's color scale. Node 0 is the initial state, and ★ is the successful terminal state. Node and action descriptions appear in Figures 14 and 15.

Node list of cases in ALFWorld

Node 0: -= Welcome to TextWorld, ALFRED! =-
You are in the middle of a room. Looking quickly around you, you see a cabinet 13, a cabinet 12, a cabinet 11, a cabinet 10, a cabinet 9, a cabinet 8, a cabinet 7, a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 2, a countertop 1, a diningtable 1, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: clean some kettle and put it in shelf. You are not carrying anything.
Node 1: You arrive at coffeemachine 1. On the coffeemachine 1, you see nothing. You are not carrying anything.
Node 2: You arrive at countertop 1. On the countertop 1, you see a fork 1, and a lettuce 1. You are not carrying anything.
Node 3: You arrive at stoveburner 1. On the stoveburner 1, you see nothing. You are not carrying anything.
Node 4: You arrive at countertop 2. On the countertop 2, you see a book 2, a book 1, a butterknife 1, a dishsponge 2, a fork 2, a kettle 1, a lettuce 2, a plate 2, a sink 1, a statue 2, and a tomato 1. You are not carrying anything.
Node 5: You pick up the kettle 1 from the countertop 2. You are carrying: a kettle 1.
Node 6: You arrive at sinkbasin 1. On the sinkbasin 1, you see nothing. You are carrying: a kettle 1.
Node 7: You clean the kettle 1 using the sinkbasin 1. You are carrying: kettle 1 (cleaned).
Node 8: You arrive at shelf 1. On the shelf 1, you see nothing. You are carrying: kettle 1 (cleaned).
Node ★: You Won!
Node 10: You arrive at fridge 1. The fridge 1 is closed. You are not carrying anything.
Node 11: You open the fridge 1. The fridge 1 is open. In it, you see a apple 2, a bowl 1, a cup 1, a egg 1, a plate 1, and a potato 1. You are not carrying anything.
Node 12: You arrive at shelf 1. On the shelf 1, you see nothing. You are not carrying anything.
Node 13: You arrive at fridge 1. The fridge 1 is open. In it, you see a apple 2, a bowl 1, a cup 1, a egg 1, a plate 1, and a potato 1. You are not carrying anything.
Node 14: You arrive at diningtable 1. On the diningtable 1, you see a apple 1, a bread 2, a bread 1, a houseplant 1, a knife 1, a saltshaker 2, a soapbottle 1, a spatula 2, a spatula 1, a statue 1, a tomato 3, and a tomato 2. You are not carrying anything.
Node 15: You arrive at stoveburner 2. On the stoveburner 2, you see a pan 1. You are not carrying anything.
Node 16: You arrive at sinkbasin 1. On the sinkbasin 1, you see nothing. You are not carrying anything.

Figure 14: The node list of cases in ALFWorld in Figures 11, 12, and 13.

Edge list of cases in ALFWorld

Node 0 $->$ **Node 1:** go to coffeemachine 1
Node 0 $->$ **Node 14:** go to diningtable 1
Node 1 $->$ **Node 2:** go to countertop 1
Node 2 $->$ **Node 13:** go to fridge 1
Node 2 $->$ **Node 3:** go to stoveburner 1
Node 2 $->$ **Node 16:** go to sinkbasin 1
Node 2 $->$ **Node 12:** go to shelf 1
Node 2 $->$ **Node 10:** go to fridge 1
Node 3 $->$ **Node 13:** go to fridge 1
Node 3 $->$ **Node 14:** go to diningtable 1
Node 3 $->$ **Node 10:** go to fridge 1
Node 3 $->$ **Node 4:** go to countertop 2
Node 4 $->$ **Node 5:** take kettle 1 from countertop 2
Node 5 $->$ **Node 6:** go to sinkbasin 1
Node 6 $->$ **Node 7:** clean kettle 1 with sinkbasin 1
Node 7 $->$ **Node 8:** go to shelf 1
Node 8 $->$ **Node ★:** move kettle 1 to shelf 1
Node 10 $->$ **Node 11:** open fridge 1
Node 10 $->$ **Node 14:** go to diningtable 1
Node 11 $->$ **Node 2:** go to countertop 1
Node 12 $->$ **Node 13:** go to fridge 1
Node 13 $->$ **Node 2:** go to countertop 1
Node 13 $->$ **Node 4:** go to countertop 2
Node 13 $->$ **Node 15:** go to stoveburner 2
Node 13 $->$ **Node 3:** go to stoveburner 1
Node 14 $->$ **Node 3:** go to stoveburner 1
Node 14 $->$ **Node 2:** go to countertop 1
Node 15 $->$ **Node 14:** go to diningtable 1
Node 16 $->$ **Node 13:** go to fridge 1

Figure 15: The edge list of cases in ALFWorld in Figures 11, 12, and 13.

WebShop. Different search queries can reach the same product page, joining purchase paths that share the subsequent attribute-selection actions.

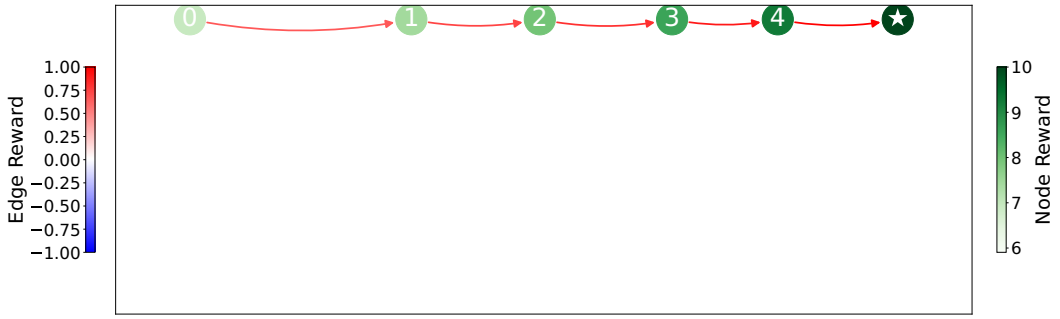


Figure 16: The original state graph constructed from 1 sampled rollout in WebShop. Darker nodes indicate higher values on the source plot's color scale. Node 0 is the initial state, and ★ is the successful terminal state. Node and action descriptions appear in Figures 19 and 20.

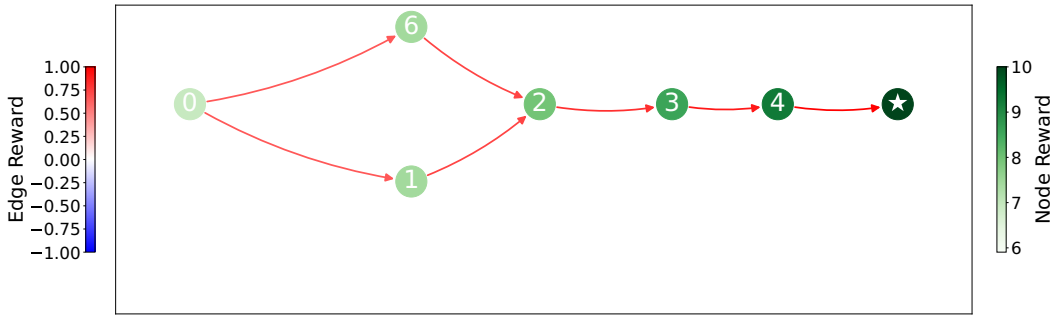


Figure 17: The original state graph constructed from 2 sampled rollouts in WebShop. Darker nodes indicate higher values on the source plot's color scale. Node 0 is the initial state, and ★ is the successful terminal state. Node and action descriptions appear in Figures 19 and 20.

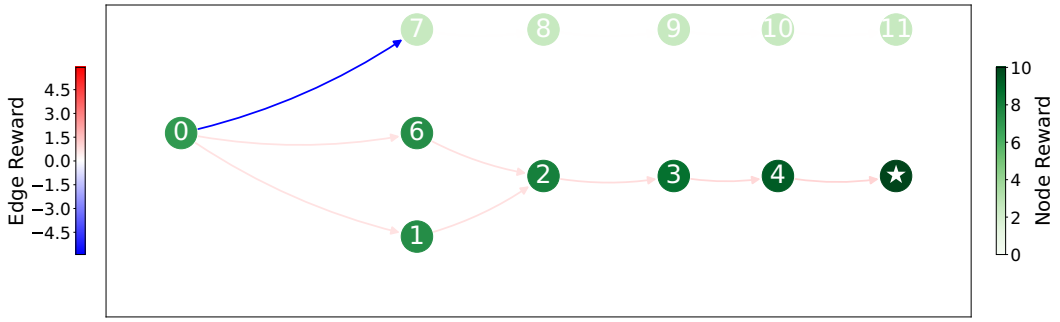


Figure 18: The original state graph constructed from 3 sampled rollouts in WebShop. Darker nodes indicate higher values on the source plot's color scale. Node 0 is the initial state, and ★ is the successful terminal state. Node and action descriptions appear in Figures 19 and 20.

Node list of examples in WebShop

Node 0: 'Search'
Node 1: 'Back to Search' [SEP] 'Page 1 (Total results: 50)' [SEP] 'Next >' [SEP] 'B07S7HDC88' [SEP] 'VanciLin Mens Casual Leather Fashion Slip-on Loafers' [SEP] '$33.99 to $35.99' [SEP] 'B00lTCBD3Y' [SEP] 'Laredo Womens Spellbound Studded Square Toe Dress Boots Mid Calf Low Heel 1-2" - Black' [SEP] '$132.0 to $150.26' [SEP] 'B09N9T673Q' [SEP] 'Women's Heeled Sandals Sexy Minimalistic Cut out Rear Zipper Wedge Roman Sandals Casual Solid Stiletto Pump for Night Party Comfy Lightweight Summer Shoes' [SEP] '$100.0' [SEP] 'B07KK7G5YJ' [SEP] 'Laeacco Steampunk Clock Backdrops 6.5x6.5ft Vinyl Photography Background Shabby Clock Photo Backdrop Old Clock Punk Grunge Culture Adults Brave Boy Portraits Photo Studio Props' [SEP] '$24.59' [SEP] 'B09P1HJ32G' [SEP] 'VDSOIUTYHFV Infrared Digital Night Vision Device, High-Definition Day and Night Dual-use, Outdoor Photos and Videos are All Black, for Adults Bird Watching, Traveling, Hiking' [SEP] '$840.99' [SEP] 'B089KGBJGW' [SEP] 'ZINUS 10 Inch Cooling Copper ADAPTIVE Pocket Spring Hybrid Mattress / Moisture Wicking Cover / Cooling Foam / Pocket Innersprings for Motion Isolation / Mattress-in-a-Box, Full' [SEP] '$289.0' [SEP] 'B09P57ZKGP' [SEP] '10 Pack 512MB USB Flash Drives Gift Stick Business Bidding Thumb Drive USB 2.0 high Speed Pen Drives (512MB, Black)' [SEP] '$100.0' [SEP] 'B001C6JXJU' [SEP] 'Premier HiDef Grey Electric Projection Screen with Low Voltage Motor Viewing Area: 50" H x 80" W' [SEP] '$2721.34' [SEP] 'B085HGD5Z3' [SEP] 'haoricu Womens Platform Sandals Summer Fashion Ladies Bow Tie Flats Rome Casual Basic Slipper' [SEP] '$14.98 to $25.98' [SEP] 'B09QKJHF4Q' [SEP] 'LIUEONG Lady Metal Buckle Platform Shoes Booties Comfortable High Top Boots Casual Shoes Comfort Women's Closed Toe Ankle Strap' [SEP] '$37.99'
Node 2: 'Back to Search' [SEP] '< Prev' [SEP] 'size' [SEP] '6.5' [SEP] '7' [SEP] '8' [SEP] '8.5' [SEP] '9' [SEP] '9.5' [SEP] '10' [SEP] '10.5' [SEP] '11' [SEP] '11.5' [SEP] '12' [SEP] '13' [SEP] '14' [SEP] 'color' [SEP] '1877black' [SEP] '1877blue' [SEP] '228black' [SEP] 'black137' [SEP] 'blue137' [SEP] 'l.brown137' [SEP] 'r.brown137' [SEP] '1877r.brown' [SEP] '228blue' [SEP] 'r.brown-hole228' [SEP] '1877brown' [SEP] '228r.brown-a' [SEP] 'y.brown-hole228' [SEP] 'black-hole228' [SEP] '228l.brown' [SEP] 'r.brown2070' [SEP] '228r.brown' [SEP] 'black2070' [SEP] 'r.brown2626' [SEP] 'r.brown-hole1887' [SEP] 'black2626' [SEP] 'black1901' [SEP] 'r.brown2060' [SEP] 'l.brown2060' [SEP] 'VanciLin Mens Casual Leather Fashion Slip-on Loafers' [SEP] 'Price: $33.99 to $35.99' [SEP] 'Rating: N.A.' [SEP] 'Description' [SEP] 'Features' [SEP] 'Reviews' [SEP] 'Buy Now'
Node 3: 'Back to Search' [SEP] '< Prev' [SEP] 'size' [SEP] '6.5' [SEP] '7' [SEP] '8' [SEP] '8.5' [SEP] '9' [SEP] '9.5' [SEP] '10' [SEP] '10.5' [SEP] '11' [SEP] '11.5' [SEP] '12' [SEP] '13' [SEP] '14' [SEP] 'color' [SEP] '1877black' [SEP] '1877blue' [SEP] '228black' [SEP] 'black137' [SEP] 'blue137' [SEP] 'l.brown137' [SEP] 'r.brown137' [SEP] '1877r.brown' [SEP] '228blue' [SEP] 'r.brown-hole228' [SEP] '1877brown' [SEP] '228r.brown-a' [SEP] 'y.brown-hole228' [SEP] 'black-hole228' [SEP] '228l.brown' [SEP] 'r.brown2070' [SEP] '228r.brown' [SEP] 'black2070' [SEP] 'r.brown2626' [SEP] 'r.brown-hole1887' [SEP] 'black2626' [SEP] 'black1901' [SEP] 'r.brown2060' [SEP] 'l.brown2060' [SEP] 'VanciLin Mens Casual Leather Fashion Slip-on Loafers' [SEP] 'Price: $33.99 to $35.99' [SEP] 'Rating: N.A.' [SEP] 'Description' [SEP] 'Features' [SEP] 'Reviews' [SEP] 'Buy Now' clicked attributes: [228r.brown-a]
Node 4: 'Back to Search' [SEP] '< Prev' [SEP] 'size' [SEP] '6.5' [SEP] '7' [SEP] '8' [SEP] '8.5' [SEP] '9' [SEP] '9.5' [SEP] '10' [SEP] '10.5' [SEP] '11' [SEP] '11.5' [SEP] '12' [SEP] '13' [SEP] '14' [SEP] 'color' [SEP] '1877black' [SEP] '1877blue' [SEP] '228black' [SEP] 'black137' [SEP] 'blue137' [SEP] 'l.brown137' [SEP] 'r.brown137' [SEP] '1877r.brown' [SEP] '228blue' [SEP] 'r.brown-hole228' [SEP] '1877brown' [SEP] '228r.brown-a' [SEP] 'y.brown-hole228' [SEP] 'black-hole228' [SEP] '228l.brown' [SEP] 'r.brown2070' [SEP] '228r.brown' [SEP] 'black2070' [SEP] 'r.brown2626' [SEP] 'r.brown-hole1887' [SEP] 'black2626' [SEP] 'black1901' [SEP] 'r.brown2060' [SEP] 'l.brown2060' [SEP] 'VanciLin Mens Casual Leather Fashion Slip-on Loafers' [SEP] 'Price: $33.99 to $35.99' [SEP] 'Rating: N.A.' [SEP] 'Description' [SEP] 'Features' [SEP] 'Reviews' [SEP] 'Buy Now' clicked attributes: [228r.brown-a, 6.5]
Node ★: Thank you for shopping with us! [SEP] Your code: [SEP] None [SEP] (Paste it in your MTurk interface.) [SEP] Purchased [SEP] asin [SEP] B07S7HDC88 [SEP] options [SEP] "color": "228r.brown-a", "size": "6.5" [SEP] attrs [SEP] None [SEP] category [SEP] None [SEP] query [SEP] None [SEP] product category [SEP] None [SEP] Target [SEP] asin [SEP] options [SEP] attrs [SEP] price upper [SEP] instuction text [SEP] category [SEP] product category [SEP] query [SEP] Goal [SEP] None [SEP] Reward [SEP] Your score (min 0.0, max 1.0) [SEP] 1.0 [SEP] Reward Details [SEP] None
Node 6: 'Back to Search' [SEP] 'Page 1 (Total results: 23)' [SEP] 'Next >' [SEP] 'B07S7HDC88' [SEP] 'VanciLin Mens Casual Leather Fashion Slip-on Loafers' [SEP] '$33.99 to $35.99' [SEP] 'B07YCGBPRD' [SEP] 'OTAO Privacy Screen Protector for iPhone 11 Pro Max/iPhone Xs Max 6.5 Inch True 28°Anti Spy Tempered Glass Full-Coverage (2-pack)' [SEP] '$9.98' [SEP] 'B085HGD5Z3' [SEP] 'haoricu Womens Platform Sandals Summer Fashion Ladies Bow Tie Flats Rome Casual Basic Slipper' [SEP] '$14.98 to $25.98' [SEP] 'B07KK7G5YJ' [SEP] 'Laeacco Steampunk Clock Backdrops 6.5x6.5ft Vinyl Photography Background Shabby Clock Photo Backdrop Old Clock Punk Grunge Culture Adults Brave Boy Portraits Photo Studio Props' [SEP] '$24.59' [SEP] 'B09J95S478' [SEP] 'My Sanity Question Giraffe Christmas Pattern Black Ugly Wool Christmas Sweater Pullover Long Sleeve Sweater for Men Women, Couple Matching, Friends' [SEP] '$39.99' [SEP] 'B09N9T673Q' [SEP] 'Women's Heeled Sandals Sexy Minimalistic Cut out Rear Zipper Wedge Roman Sandals Casual Solid Stiletto Pump for Night Party Comfy Lightweight Summer Shoes' [SEP] '$100.0' [SEP] 'B07TN3Y9H1' [SEP] 'Maxim 32478SWSBRBP Finn Mid-Century Modern Satin White Glass Ball Bath Vanity Wall Mount, 5-Light 200 Total Watts, 7"H x 44"W, Satin Brass/Brushed Platinum' [SEP] '$219.99' [SEP] 'B09QKJHF4Q' [SEP] 'LIUEONG Lady Metal Buckle Platform Shoes Booties Comfortable High Top Boots Casual Shoes Comfort Women's Closed Toe Ankle Strap' [SEP] '$37.99' [SEP] 'B0912FQSMT' [SEP] '450 Bluetooth Marine Gauge Receiver and 6.5-Inch Speaker Package' [SEP] '$224.02' [SEP] 'B088TWNTRB' [SEP] 'MIA Delena' [SEP] '$21.89 to $55.08'
Node 7: 'Back to Search' [SEP] 'Page 1 (Total results: 50)' [SEP] 'Next >' [SEP] 'B07RPMCNKD' [SEP] 'ULTRAIDEAS Women's Comfy Lightweight Slippers Non-Slip House Shoes for Indoor & Outdoor' [SEP] '$19.99 to $21.99' [SEP] 'B07HP6LVRS' [SEP] 'MCICI Mens Loafers Moccasin Driving Shoes Premium Genuine Leather Casual Slip On Flats Fashion Slipper Breathable Big Size' [SEP] '$15.99 to $32.99' [SEP] 'B07S7HDC88' [SEP] 'VanciLin Mens Casual Leather Fashion Slip-on Loafers' [SEP] '$33.99 to $35.99' [SEP] 'B09HMCKZQW' [SEP] 'Copercn Dress Ankle Booties For Women Ladies Fashion Casual Chunky Block High Heels Dress Pump Thermal Short Boots Winter Fall Dressy Short Boots For Business Work Wedding Party Dresses' [SEP] '$17.99 to $23.99' [SEP] 'B09F2MFG1Q' [SEP] 'Womens Mens house slippers Memory Foam Garden Plush Lining Slip On Rubber sole Indoor Outdoor House Casual Shoes (Black adult-10.5-11)' [SEP] '$100.0' [SEP] 'B09S6VN97V' [SEP] 'Skechers Women's, Relaxed Fit: Arch Fit - Commute Clog Taupe 11 M' [SEP] '$74.95' [SEP] 'B008QYOGZ2' [SEP] 'OluKai MEA Ola - Men's Supportive Sandal Charcoal/Dkjava - 14' [SEP] '$100.0' [SEP] 'B07XDRVVYM' [SEP] 'Clarks Women's Un Adorn Sling Sandal' [SEP] '$34.99 to $99.95' [SEP] 'B01LXTLLVG' [SEP] 'PUMA V1.08 Tricks Top Trainer Mens Soccer Sneakers/Boots-Grey-5.5' [SEP] '$55.5' [SEP] 'B09QXF3V3X' [SEP] 'DEUVOUM Summer Trend Mesh Shoes Men's Sports Shoes Solid Color Lace-Up Sneakers Fashion All-Match Walking Shoes Outdoor Hiking Shoes Non-Slip Shock-Absorbing Casual Sports Shoes' [SEP] '$100.0'
Node 8: 'Back to Search' [SEP] '< Prev' [SEP] 'size' [SEP] '6.5' [SEP] '7' [SEP] '8' [SEP] '8.5' [SEP] '9.5' [SEP] '10' [SEP] '10.5' [SEP] '11' [SEP] '11.5' [SEP] 'color' [SEP] 'black' [SEP] 'blue' [SEP] 'brown' [SEP] 'white' [SEP] 'yellow' [SEP] 'black-b' [SEP] 'blue-a' [SEP] 'yellow-a' [SEP] 'MCICI Mens Loafers Moccasin Driving Shoes Premium Genuine Leather Casual Slip On Flats Fashion Slipper Breathable Big Size' [SEP] 'Price: $15.99 to $32.99' [SEP] 'Rating: N.A.' [SEP] 'Description' [SEP] 'Features' [SEP] 'Reviews' [SEP] 'Buy Now'
Node 9: 'Back to Search' [SEP] '< Prev' [SEP] 'size' [SEP] '6.5' [SEP] '7' [SEP] '8' [SEP] '8.5' [SEP] '9.5' [SEP] '10' [SEP] '10.5' [SEP] '11' [SEP] '11.5' [SEP] 'color' [SEP] 'black' [SEP] 'blue' [SEP] 'brown' [SEP] 'white' [SEP] 'yellow' [SEP] 'black-b' [SEP] 'blue-a' [SEP] 'yellow-a' [SEP] 'MCICI Mens Loafers Moccasin Driving Shoes Premium Genuine Leather Casual Slip On Flats Fashion Slipper Breathable Big Size' [SEP] 'Price: $15.99 to $32.99' [SEP] 'Rating: N.A.' [SEP] 'Description' [SEP] 'Features' [SEP] 'Reviews' [SEP] 'Buy Now' clicked attributes: [brown]
Node 10: 'Back to Search' [SEP] '< Prev' [SEP] 'size' [SEP] '6.5' [SEP] '7' [SEP] '8' [SEP] '8.5' [SEP] '9.5' [SEP] '10' [SEP] '10.5' [SEP] '11' [SEP] '11.5' [SEP] 'color' [SEP] 'black' [SEP] 'blue' [SEP] 'brown' [SEP] 'white' [SEP] 'yellow' [SEP] 'black-b' [SEP] 'blue-a' [SEP] 'yellow-a' [SEP] 'MCICI Mens Loafers Moccasin Driving Shoes Premium Genuine Leather Casual Slip On Flats Fashion Slipper Breathable Big Size' [SEP] 'Price: $15.99 to $32.99' [SEP] 'Rating: N.A.' [SEP] 'Description' [SEP] 'Features' [SEP] 'Reviews' [SEP] 'Buy Now' clicked attributes: [brown, 6.5]
Node 11: Thank you for shopping with us! [SEP] Your code: [SEP] None [SEP] (Paste it in your MTurk interface.) [SEP] Purchased [SEP] asin [SEP] B07HP6LVRS [SEP] options [SEP] "color": "brown", "size": "6.5" [SEP] attrs [SEP] None [SEP] category [SEP] None [SEP] query [SEP] None [SEP] product category [SEP] None [SEP] Target [SEP] asin [SEP] options [SEP] attrs [SEP] price upper [SEP] instuction text [SEP] category [SEP] product category [SEP] query [SEP] Goal [SEP] None [SEP] Reward [SEP] Your score (min 0.0, max 1.0) [SEP] 0.7142857142857143 [SEP] Reward Details [SEP] None

Figure 19: The node list of cases in WebShop in Figures 16, 17, and 18.

Edge list of examples in WebShop

Node 0 $->$ **Node 6:** search[228r.brown-a 6.5]
Node 0 $->$ **Node 1:** search[228r.brown-a, size 6.5, price lower than 70.00]
Node 0 $->$ **Node 7:** search[228r.brown-a 6.5 rubber outsole men slipper non slip]
Node 1 $->$ **Node 2:** click[b07s7hdc88]
Node 2 $->$ **Node 3:** click[228r.brown-a]
Node 3 $->$ **Node 4:** click[6.5]
Node 4 $->$ **Node ★:** click[buy now]
Node 6 $->$ **Node 2:** click[b07s7hdc88]
Node 7 $->$ **Node 8:** click[b07hp6lvrs]
Node 8 $->$ **Node 9:** click[brown]
Node 9 $->$ **Node 10:** click[6.5]
Node 10 $->$ **Node 11:** click[buy now]

Figure 20: The edge list of cases in WebShop in Figures 16, 17, and 18.

Sokoban. Repeated board configurations connect movement sequences, including detours that return to a state on the route to a solved board.

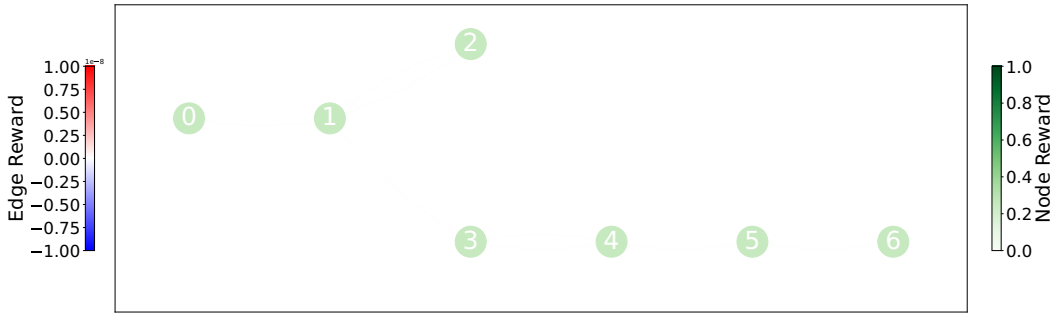


Figure 21: The original state graph constructed from 1 sampled rollout in Sokoban. Darker nodes indicate higher values on the source plot's color scale. Node 0 is the initial state, and ★ is the successful terminal state. Node and action descriptions appear in Figures 24 and 25.

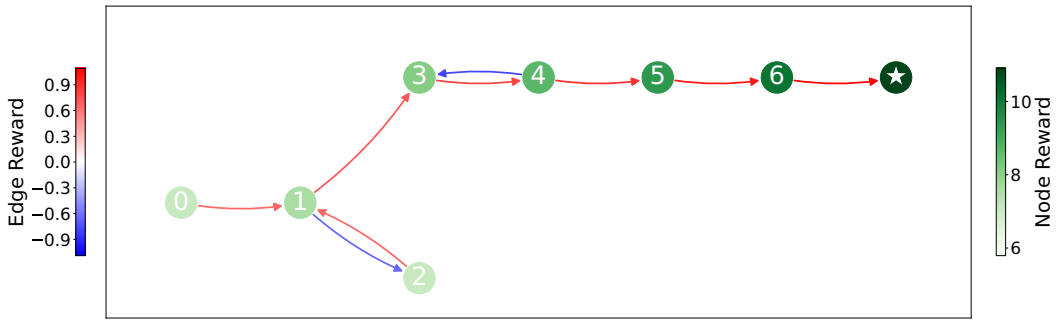


Figure 22: The original state graph constructed from 2 sampled rollouts in Sokoban. Darker nodes indicate higher values on the source plot's color scale. Node 0 is the initial state, and ★ is the successful terminal state. Node and action descriptions appear in Figures 24 and 25.

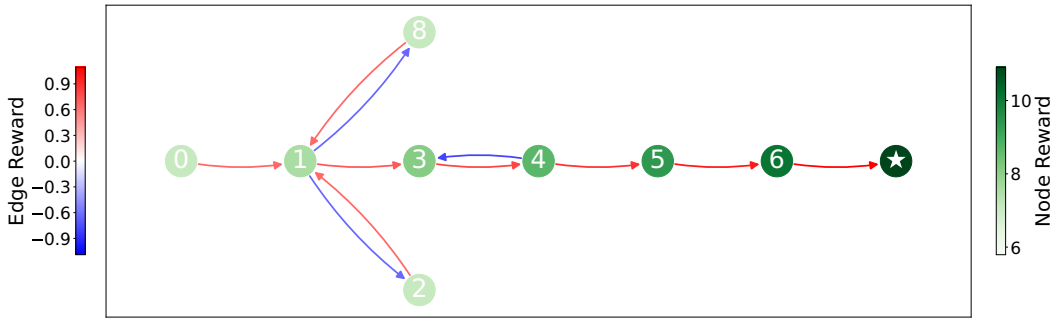
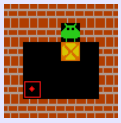
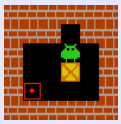
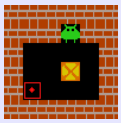


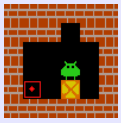
Figure 23: The original state graph constructed from 3 sampled rollouts in Sokoban. Darker nodes indicate higher values on the source plot's color scale. Node 0 is the initial state, and ★ is the successful terminal state. Node and action descriptions appear in Figures 24 and 25.

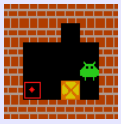
Node list of examples in Sokoban

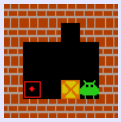
Node 0: 

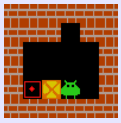
Node 1: 

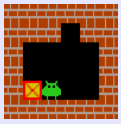
Node 2: 

Node 3: 

Node 4: 

Node 5: 

Node 6: 

Node ★: 

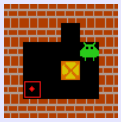
Node 8: 

Figure 24: The node list of cases in Sokoban in Figures 21, 22, and 23.

Edge list of examples in Sokoban

Node 0 — > **Node 1:** down
Node 1 — > **Node 3:** down
Node 1 — > **Node 2:** up
Node 1 — > **Node 8:** right
Node 2 — > **Node 1:** down
Node 3 — > **Node 4:** right
Node 4 — > **Node 5:** down
Node 4 — > **Node 3:** left
Node 5 — > **Node 6:** left
Node 6 — > **Node ★:** left
Node 8 — > **Node 1:** left

Figure 25: The edge list of cases in Sokoban in Figures 21, 22, and 23.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Sec. 1 describe topology-based action-level reward estimation and its use with trajectory-level supervision. Sections 3 and 4 define the method and theoretical results, and Sec. 5 evaluates performance, robustness, and computational overhead.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 4.4 states the state-identification and graph-coverage conditions for the analysis, while Appendices D.4 and E.6 discuss exploration and state-grouping sensitivity. Section 6 identifies more reliable state identification and broader open-ended evaluation as future directions.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Section 4.4 states the assumptions and results for graph approximation, state potentials, action-level reward approximation, and trajectory-level reward ordering. Appendix C provides the formal definitions and proofs, including the cumulative-sampling conditions.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sections 3 and 4 specify the algorithm; Appendices B.1, B.2, and B.3 describe state interfaces, retained state transitions, and reward and advantage notation. Appendix E.1 and the supplementary experimental subsections provide training configurations and evaluation protocols, complemented by the public implementation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our implementation is publicly available at `https://github.com/tmlr-group/RewardFlow`, with installation instructions, environment and data setup, and training scripts for RewardFlow and RL baselines. Appendix E.1 specifies the configurations used in the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 identifies the benchmarks, models, baselines, and metrics, while Appendix E.1 and Tab. 9 provide the training budgets, batch sizes, interaction horizons, sampling temperature, learning rate, and algorithm hyperparameters. Supplementary comparisons specify their differing settings in the corresponding subsections.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Tabs. 1 and 3 report means and standard deviations over three inference seeds at fixed checkpoints, as specified in Appendix E.1. The preprocessing control additionally specifies the inference seeds and population-standard-deviation calculation in Appendix E.5.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Tab. 9 reports GPU types and counts and training-step budgets; Fig. 1 and Appendix E.13 provide per-step runtime measurements. GPU memory and complete per-run and project-wide compute accounting are not reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the NeurIPS Code of Ethics and evaluate on public benchmarks using automated metrics, without new human-subject data collection. Potential deployment risks and mitigation measures are discussed in the Broader impacts item below.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: RewardFlow can improve the efficiency of training interactive agents, supporting useful automation but also potentially enabling harmful automation or tool misuse when goals, retrieved information, or success criteria are inappropriate. Our evaluation uses benchmark tasks; real-world deployment would require task-specific permission controls, monitoring, and safety evaluation.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We release the RewardFlow implementation for benchmark-based training, rather than a new pretrained model or scraped dataset requiring controlled release. Deployment considerations are addressed in the Broader impacts item above.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper credits the benchmarks, models, and training framework used in the experiments, but does not enumerate the license and terms of use for every asset.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released RewardFlow implementation includes installation instructions, environment setup, dependencies, and example training commands under the Apache-2.0 license at `https://github.com/tmlr-group/RewardFlow`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This research does not involve crowdsourcing or human-subject experiments; evaluation uses automated benchmark environments and public datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human-subject experiments were conducted, so participant risk disclosures and IRB approval are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Section 5 identifies the LLM policies and PRM baselines used in the experiments. Appendices E.14 and E.15 describe LLM-based state-normalizer synthesis and the frozen-model state-normalization variant, respectively.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.