TARGETING PIVOTAL DECISIONS FOR CREDIT ASSIGNMENT IN AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Group Relative Policy Optimization (GRPO) has become a promising approach for training large language model agents. However, its uniform assignment of trajectory-level advantages to all policy tokens fails to distinguish consequential decisions from less relevant ones, obscuring which intermediate decisions contributed to success. We introduce PROVER, a framework that targets potentially pivotal decisions for fine-grained credit assignment in agentic reinforcement learning. Given a rollout group, an agentic judge contrasts successful and failed trajectories to propose a segment potentially responsible for their divergent outcomes. Rather than directly trusting the judge's assessment, PROVER verifies the proposed segment by estimating its advantage from the difference in terminal success rates between current-policy continuations sampled before and after the segment. Positive estimates are then incorporated into the GRPO advantages of policy tokens within the proposed segment. By using model judgment only to select where to verify, PROVER grounds local credit in observed outcomes without exhaustively evaluating every intermediate state. Across ALFWorld, WebShop, and SearchQA, PROVER achieves the strongest average performance at both model scales, with relative improvements over GRPO of 9.91% and 7.12% for Qwen3.5-2B and Qwen3.5-4B, respectively. Further analyses demonstrate that informed segment selection improves policy training with modest additional generation overhead, even without a frontier-scale judge model, highlighting the effectiveness and efficiency of selectively targeting pivotal decisions for fine-grained credit assignment in agentic reinforcement learning.

1 INTRODUCTION

Large language model (LLM) agents have become increasingly competent at solving complex tasks through multi-turn interaction with external tools and environments, including web navigation (Zhou et al., 2024a; Wei et al., 2025), information retrieval (Chen et al., 2025; Jin et al., 2025), and tool use (Huang et al., 2024; Qian et al., 2026; Feng et al., 2026a). Recent work has therefore trained LLM agents directly through environment interaction using reinforcement learning with verifiable outcomes (Wei et al., 2025; Jin et al., 2025; Feng et al., 2026a). In particular, Group Relative Policy Optimization (GRPO) has become a widely used foundation for such training because it estimates relative advantages from multiple trajectories sampled for the same task without requiring a separately trained value model (Shao et al., 2024; Guo et al., 2025). However, when applied to long-horizon agents, its trajectory-level supervision creates a fundamental challenge for credit assignment: terminal feedback distinguishes trajectories within a group but provides little information about which intermediate decisions were consequential to the outcome. Thus, the same trajectory-level advantage is assigned to all policy tokens within a trajectory, regardless of their contributions, potentially reinforcing mistakes in successful trajectories and penalizing useful progress in failed ones.

This limitation has motivated approaches that provide finer-grained, intermediate supervision to distinguish the contributions of intermediate decisions rather than uniformly propagating the trajectory-level advantages. Prior work obtains intermediate supervision through learned value functions or critic models that estimate expected returns (Bahdanau et al., 2017; Zhou et al., 2024b), or through process reward models and LLM evaluators that directly assess intermediate behavior (Lightman et al., 2024; Zhang et al., 2026; Wang et al., 2026). Although these approaches provide finer-grained

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

supervision, their model-based evaluations may be inaccurate, unreliable as the policy distribution shifts, or susceptible to exploitation during optimization (Liu et al., 2026; Cui et al., 2026; Hou et al., 2025). Monte Carlo policy evaluation instead estimates intermediate-state values by averaging terminal returns from fresh continuations sampled from the current policy, grounding supervision in observed outcomes rather than model-predicted values or evaluator judgments (Sutton, 1988; Sutton & Barto, 2018). In long-horizon reasoning and agentic settings, this can be implemented by branching from intermediate states and sampling multiple continuations (Kazemnejad et al., 2025; Guo et al., 2026; Ji et al., 2026). However, extending this approach to long-horizon agentic tasks remains costly since evaluating every candidate state may require repeatedly restoring environment states and executing full continuations. This motivates selectively identifying potentially consequential decisions and concentrating fine-grained credit assignment on them, rather than exhaustively evaluating every intermediate state.

To address this challenge, we introduce PROVER (**Pro**pose, **Ver**ify, Credit), a framework for targeting potentially pivotal decisions for fine-grained credit assignment in agentic reinforcement learning. The key idea is to use model judgment only to identify a candidate segment, then concentrate outcome verification on its boundaries to estimate its advantage from current-policy continuations. PROVER comprises three stages: propose, verify, and credit. In the propose stage, an LLM-based agentic judge contrasts successful and failed trajectories from the same GRPO group to localize a potentially pivotal segment in a successful trajectory. In the verify stage, we restore the environment states immediately before and after the proposed segment and sample fresh continuations from the current policy. The difference in empirical terminal success rates provides an estimate of the segment's advantage under the current policy. In the credit stage, positive estimates are added to the original GRPO advantages of policy tokens within the proposed segment. By separating LLM-based proposal from Monte Carlo verification, PROVER grounds additional credit in observed outcomes rather than potentially inaccurate critic scores, while restricting evaluation to the proposed segment's boundaries.

We evaluate PROVER on ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), and SearchQA using Qwen3.5-2B and Qwen3.5-4B. PROVER achieves the strongest average performance at both model scales, yielding relative improvements over GRPO of 9.91% and 7.12%, respectively. Further analyses demonstrate that informed segment selection improves policy learning with modest additional generation overhead, even without a frontier-scale judge model. Together, these results support selective outcome verification as an effective and efficient approach to credit assignment in agentic reinforcement learning.

2 RELATED WORK

2.1 AGENTIC REINFORCEMENT LEARNING

Agentic reinforcement learning casts interaction with an environment as a sequential decision problem and optimizes policies over complete environment trajectories. This paradigm spans web navigation, information seeking, and tool use (Wei et al., 2025; Jin et al., 2025; Chen et al., 2025; Qian et al., 2026). A common foundation is Group Relative Policy Optimization (GRPO), which assigns a trajectory-level outcome advantage to all policy-generated tokens without training a value model (Shao et al., 2024; Wang et al., 2025). GiGPO (Feng et al., 2026b) refines this supervision through anchor-state grouping, comparing discounted downstream returns of actions taken from repeated environment states. This provides step-level credit without auxiliary models or additional rollouts, although its coverage depends on state recurrence within the sampled group. PROVER instead estimates local value changes through fresh continuations from judge-selected boundary states.

2.2 CRITIC-BASED PROCESS SUPERVISION FOR CREDIT ASSIGNMENT

LLM critics provide intermediate supervision by assessing reasoning steps or agent actions. Some methods use natural-language critiques to identify errors, guide revised attempts, and incorporate critique-guided improvements into the policy (Zhang et al., 2025; Lin et al., 2026). Beyond guiding revisions, LLM critics can directly supply intermediate credit. CriticSearch uses a frozen retrospective critic to evaluate search actions using completed trajectories and reference answers, converting

its judgments into turn-level feedback (Zhang et al., 2026). Related methods assign semantic roles to actions and translate them into local credit corrections (Xu et al., 2026), or weight retrieval rounds according to their judged contributions (Wang et al., 2026). In these credit-assignment methods, evaluator judgments directly shape local credit. In PROVER, the agentic judge selects a candidate segment, while fresh environment continuations determine its estimated advantage and the resulting additional credit.

2.3 OUTCOME-BASED CREDIT ASSIGNMENT

Outcome-based credit assignment estimates the contribution of intermediate decisions from downstream task outcomes. Monte Carlo value estimation provides one way to obtain this signal by averaging returns from continuations sampled at intermediate states. Prior work organizes outcome-labeled continuations as implicit prefix trees or explicitly sampled branching structures (Hou et al., 2025; Ji et al., 2026; Zhao et al., 2026). VinePPO samples continuations at intermediate reasoning-step boundaries to estimate step-level advantages (Kazemnejad et al., 2025), while SPO partitions trajectories into segments and estimates segment-level advantages through chain- or tree-based sampling (Guo et al., 2026). PROVER shares their use of continuation-based value estimation, but uses an agentic judge to select one potentially consequential segment through cross-trajectory comparison, focusing evaluation on its two boundaries rather than evaluating successive steps or segments throughout a trajectory.

3 METHOD

We consider reinforcement learning for LLM agents that receive verifiable terminal rewards from an environment. We first introduce the agent–environment setting and GRPO formulation in Section 3.1, followed by an overview of PROVER in Section 3.2. We then describe its three stages in Sections 3.3–3.5.

3.1 PRELIMINARIES

Agent–environment interaction. Let x denote a task instance and π_θ an LLM agent policy. At turn t, the complete agent–environment state s_t comprises the interaction history available to the policy and the corresponding environment state. The policy samples a response $a_t \sim \pi_\theta(\cdot \mid s_t)$, which induces the next state s_{t+1}. Repeating this interaction produces a trajectory $\tau = (s_1, a_1, \ldots, s_T, a_T, s_{T+1})$. After the final turn, the environment returns a terminal reward $R(\tau)$. We focus on tasks with binary outcomes, where $R(\tau) \in \{0, 1\}$ indicates failure or success, and assume no intermediate rewards.

Agentic reinforcement learning with GRPO. For each task x, GRPO samples a group of G trajectories $\mathcal{G} = \{\tau_i\}_{i=1}^{G}$ from the same policy. Let $R_i = R(\tau_i)$. With binary rewards, we use the mean-centered group-relative advantage:

$$A_i^{\mathrm{GRPO}} = R_i - \frac{1}{G}\sum_{j=1}^{G} R_j. \tag{1}$$

GRPO assigns A_i^{GRPO} to every trainable policy token in τ_i, while environment observations and other non-policy tokens are masked from the optimization objective. This distinguishes successful from failed trajectories within the group, but it does not distinguish among the decisions that compose each trajectory.

We denote the successful and failed trajectories by $\mathcal{G}^+ = \{\tau_i \in \mathcal{G} \mid R_i = 1\}$ and $\mathcal{G}^- = \{\tau_i \in \mathcal{G} \mid R_i = 0\}$, respectively. We call $\mathcal{G}$ a *mixed-outcome group* when both subsets are nonempty.

3.2 PROVER OVERVIEW

Figure 1 illustrates the overview of PROVER. Given a mixed-outcome GRPO group, PROVER aims to localize potentially pivotal decisions within a successful trajectory and assign them fine-grained

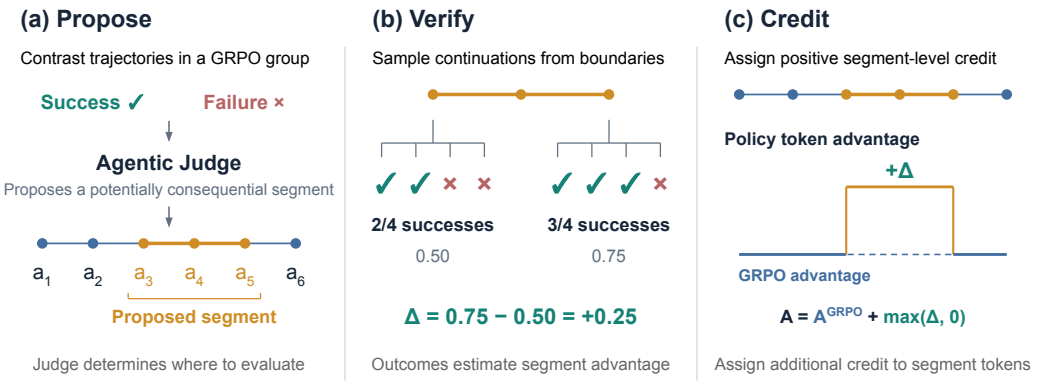


Figure 1: Overview of PROVER. **(a) Propose:** An agentic judge contrasts successful and failed trajectories to localize a potentially consequential segment. **(b) Verify:** Boundary continuations estimate the segment advantage Δ from the difference in terminal success rates. **(c) Credit:** Positive estimates augment GRPO advantages only for policy tokens within the selected segment.

credit without exhaustively evaluating every intermediate state. We operationalize such decisions as short contiguous segments of agent turns and estimate their contribution from downstream outcomes.

PROVER proceeds in three stages. In the propose stage, an LLM agent, called the agentic judge, contrasts successful and failed trajectories to select a potentially consequential segment from a successful trajectory. In the verify stage, PROVER restores the agent–environment states immediately before and after the proposed segment and samples fresh current-policy continuations from both states. The difference between the two empirical terminal success rates estimates the segment's advantage. In the credit stage, when this estimate is positive, PROVER adds it to the original GRPO advantage assigned to each trainable policy token in the selected segment. This design combines model-based guidance (Zhang et al., 2026; Wang et al., 2026) with outcome-grounded Monte Carlo estimation (Kazemnejad et al., 2025; Guo et al., 2026): the judge identifies a promising segment without directly supplying credit scores, while fresh current-policy continuations estimate its advantage without requiring exhaustive evaluation of intermediate states.

3.3 PROPOSE: AGENTIC JUDGE-GUIDED SEGMENT SELECTION

The propose stage localizes a potentially consequential decision or sequence of decisions within a successful trajectory. We represent the candidate as a contiguous segment of agent turns, which is subsequently evaluated to determine whether it warrants additional credit. Selecting such a segment requires understanding how its actions address difficulties encountered elsewhere in the rollout group, rather than assessing each action in isolation. Because the relevant evidence may be scattered across long trajectories, we use an **agentic judge** that can search for recurring failure patterns and selectively inspect the turns needed to assess a candidate segment. This allows segment selection to draw on cross-trajectory context without requiring every trajectory to be processed in full.

Agentic Judge Design. The judge is implemented as an external LLM agent and receives the task, a complete turn-indexed successful trajectory $\tau^+ \in \mathcal{G}^+$, and compact previews of the failed trajectories in $\mathcal{G}^-$, each truncated to a fixed length. It uses the failed trajectories to identify recurrent difficulties and examines how a segment in the successful trajectory avoids or resolves them. To support this inspection, we provide two tools adapted from Lee et al. (2026):

- **`search_trajectory(query, k)`**: Searches all failed trajectories using the natural-language `query` and returns the top-k bounded turn previews with their trajectory identifiers and turn indices.
- **`get_segment(traj_id, start_turn, end_turn)`**: Retrieves the inclusive turn range $[$`start_turn`, `end_turn`$]$ from the trajectory identified by `traj_id`, including the policy-visible context before the first retrieved turn.

The judge can alternate between searching for supporting evidence and inspecting relevant turns to refine its hypothesis about which segment warrants evaluation. After inspection, it outputs inclusive turn boundaries (ℓ, r) in τ^+, defining the proposed segment $\tau^{\text{seg}} = (s_\ell, a_\ell, \ldots, s_r, a_r, s_{r+1})$. The selected segment therefore remains a hypothesis about which decisions contributed to success; the verify stage tests whether it warrants additional credit.

3.4 VERIFY: OUTCOME-BASED SEGMENT EVALUATION

The propose stage identifies a potentially consequential segment, but it does not determine whether the segment deserves additional credit. The verify stage makes this distinction explicit: the judge's proposal only allocates the evaluation budget, while the segment's advantage is estimated from downstream task outcomes. Following the Monte Carlo principle of estimating intermediate values from sampled continuations, we compare fresh current-policy continuations from the states immediately before and after the proposed segment. The difference between the two empirical terminal success rates provides an outcome-based estimate of whether, and by how much, executing the segment improves the policy's probability of success. We first formalize the segment advantage that this comparison seeks to estimate.

Segment advantage. The turn boundaries (ℓ, r) determine the pre- and post-segment states $s_{\text{pre}} = s_\ell$ and $s_{\text{post}} = s_{r+1}$. Let $\mathbf{a}_{\ell:r} = (a_\ell, \ldots, a_r)$ denote the recorded action sequence within τ^{seg}. Let π denote the policy that generated the source group, held fixed during verification. We write $V^\pi(s)$ for the expected discounted return when following π from state s, and $Q^\pi_{\text{seg}}(s_{\text{pre}}, \mathbf{a}_{\ell:r})$ for the expected discounted return when executing the recorded segment and then following π.

Treating the recorded segment as a temporally extended action and assuming deterministic segment execution, its current-policy advantage is given below. The final equality uses our setting of nonterminal segments with no intermediate rewards and $\gamma = 1$:

$$\begin{aligned} A^\pi_{\text{seg}}(s_{\text{pre}}, \mathbf{a}_{\ell:r}) &= Q^\pi_{\text{seg}}(s_{\text{pre}}, \mathbf{a}_{\ell:r}) - V^\pi(s_{\text{pre}}) \\ &= \sum_{j=\ell}^{r} \gamma^{j-\ell} r_j + \gamma^d V^\pi(s_{\text{post}}) - V^\pi(s_{\text{pre}}) \\ &= V^\pi(s_{\text{post}}) - V^\pi(s_{\text{pre}}), \end{aligned} \tag{2}$$

where r_j is the reward received at turn j, $d = r - \ell + 1$ is the segment duration in turns, and γ is the per-turn discount factor. With binary terminal rewards and $\gamma = 1$, the boundary values represent probabilities of eventual success. Thus, verifying a proposal reduces to estimating the current-policy values at these two boundaries.

Boundary rollouts. We estimate the two boundary values through Monte Carlo policy evaluation. For each valid proposal, we restore s_{pre} and s_{post}, including the interaction history and corresponding environment state. From each restored state, we independently sample K fresh continuations using the fixed policy π. Let R^{pre}_k and R^{post}_k denote their terminal rewards. Because these rewards are binary, their empirical means estimate the policy's success probabilities at the two boundaries:

$$\widehat{V}^\pi_{\text{pre}} = \frac{1}{K}\sum_{k=1}^{K} R^{\text{pre}}_k, \qquad \widehat{V}^\pi_{\text{post}} = \frac{1}{K}\sum_{k=1}^{K} R^{\text{post}}_k, \qquad \widehat{\Delta}_{\text{seg}} = \widehat{V}^\pi_{\text{post}} - \widehat{V}^\pi_{\text{pre}}, \tag{3}$$

where $\widehat{\Delta}_{\text{seg}}$ is the resulting segment-advantage estimate.

Under the stated assumptions, $\widehat{\Delta}_{\text{seg}}$ is conditionally unbiased for the segment advantage in Equation 2, given the selected segment and its boundary states.[1] A positive $\widehat{\Delta}_{\text{seg}}$ provides outcome-based evidence that the segment improves the policy's probability of success.

3.5 CREDIT: SEGMENT-LEVEL ADVANTAGE ASSIGNMENT

Because the boundary-value comparison estimates the advantage of the segment as a whole, we assign the same verified bonus to all of its trainable policy tokens. Let u index trainable policy

[1] Appendix B proves this result and clarifies its scope.

tokens in τ^+, and let $I_u^{\text{seg}} = \mathbb{1}[\ell \leq \text{turn}(u) \leq r]$ indicate whether token u belongs to an action turn within the proposed segment. For a valid segment-advantage estimate, we define

$$A_u^{\text{PROVER}} = \begin{cases} A^{\text{GRPO}} + \lambda \widehat{\Delta}_{\text{seg}} I_u^{\text{seg}}, & \text{if } \widehat{\Delta}_{\text{seg}} > 0, \\ A^{\text{GRPO}}, & \text{otherwise}, \end{cases} \tag{4}$$

where A^{GRPO} is the trajectory-level advantage of τ^+ and $\lambda \geq 0$ controls the strength of the segment-level bonus. Invalid proposals or measurements receive no additional credit. We substitute A_u^{PROVER} for the ordinary GRPO advantage on τ^+ while leaving the source trajectories, policy ratios, clipping rule, and other optimization terms unchanged. All other trajectories retain their original GRPO advantages. Boundary continuations are used only for advantage estimation and are not included in the policy-training batch.

Overall, the judge selects potentially pivotal segments for evaluation, while boundary-rollout outcomes determine whether and how much additional credit they receive. This separation enables PROVER to concentrate outcome-based evaluation on a small part of the trajectory and augment GRPO's trajectory-level supervision with selectively verified local credit.

4 EXPERIMENTAL SETUP

4.1 AGENT ENVIRONMENTS

We evaluate PROVER on three long-horizon agent environments: (1) **ALFWorld** (Shridhar et al., 2021): a text-based household simulator in which the agent observes its surroundings and executes natural-language actions to manipulate objects and complete a task; (2) **WebShop** (Yao et al., 2022): a simulated e-commerce website in which the agent issues searches and navigates product pages through web actions before selecting a product that satisfies a natural-language goal; and (3) **SearchQA**: a corpus-backed retrieval environment in which the agent iteratively submits search queries, examines retrieved evidence, and returns an answer to a single- or multi-hop question. Further details on the environments, interaction protocols, and prompt templates are provided in Appendix A.1.

4.2 BASELINES

We compare PROVER with six baselines: (1) **GRPO** (Shao et al., 2024), the standard group-relative policy optimization baseline; (2) **Budget-Matched GRPO**, which augments GRPO with additional policy rollouts to match the rollout budget used by PROVER for segment-boundary value estimation; (3) **GiGPO** (Feng et al., 2026b), which augments GRPO with step-level advantages derived from discounted returns at repeated environment states; (4) **SPO-chain** (Guo et al., 2026), which assigns segment-level credit using differences between Monte Carlo value estimates at segment boundaries; (5) **SPO-tree** (Guo et al., 2026), which uses tree-structured sampling to share trajectory prefixes and estimate sibling-relative advantages; (6) **CriticSearch** (Zhang et al., 2026), which augments GRPO with binary action-level scores from a retrospective LLM critic. We use the same underlying LLM for this critic and our agentic judge.

These baselines examine complementary aspects of credit assignment. Budget-Matched GRPO tests whether additional policy rollouts alone account for the gains. GiGPO provides a comparison with step-level credit inferred from repeatedly visited states. SPO-chain and SPO-tree compare targeted segment selection with fixed segment boundaries and tree-structured sampling, respectively. CriticSearch compares credit derived from sampled downstream outcomes with credit derived directly from an LLM critic's action-level judgments. Complete training and method-specific implementation details are provided in Appendix A.3.

4.3 IMPLEMENTATION DETAILS

We use Prime-RL (Prime Intellect, 2025) to train Qwen3.5-2B and Qwen3.5-4B for 100 optimizer steps. The standard GRPO configuration uses 16 groups of $G = 8$ trajectories per update. By default, PROVER uses `gpt-5.4-mini` as its agentic judge, matching the underlying LLM used by the CriticSearch critic. We apply PROVER to groups with success rates strictly between 0 and 0.5,

Method	ALFWorld	WebShop	SearchQA	Avg.
Qwen3.5-2B				
GRPO	84.08 ± 3.02	42.53 ± 0.31	35.58 ± 0.38	54.07 ± 0.78
Budget-Matched GRPO	84.05 ± 4.13	39.00 ± 2.62	37.75 ± 0.25	53.60 ± 1.32
GiGPO	85.82 ± 2.24	44.60 ± 0.35	36.17 ± 1.66	55.53 ± 1.25
SPO-tree	56.96 ± 1.14	38.47 ± 0.50	31.67 ± 0.72	42.36 ± 0.27
SPO-chain	75.87 ± 1.72	**47.80** ± 2.62	27.67 ± 0.76	50.45 ± 1.20
CriticSearch	85.08 ± 1.49	39.20 ± 0.72	38.50 ± 0.90	54.26 ± 0.63
PROVER	**90.80** ± 3.02	47.67 ± 0.42	**39.83** ± 2.08	**59.43** ± 0.74
Qwen3.5-4B				
GRPO	93.78 ± 1.55	45.67 ± 0.58	40.92 ± 0.58	60.12 ± 0.50
Budget-Matched GRPO	93.28 ± 1.29	44.93 ± 0.42	41.17 ± 1.38	59.79 ± 0.94
GiGPO	97.01 ± 0.00	40.87 ± 0.81	41.83 ± 0.95	59.90 ± 0.57
SPO-tree	92.30 ± 3.12	46.00 ± 1.31	42.92 ± 0.72	60.40 ± 0.73
SPO-chain	94.28 ± 1.55	47.47 ± 0.92	39.92 ± 0.80	60.55 ± 0.71
CriticSearch	95.27 ± 0.43	**48.07** ± 1.14	43.50 ± 1.00	62.28 ± 0.35
PROVER	**98.26** ± 1.88	47.93 ± 0.61	**47.00** ± 0.90	**64.40** ± 0.92

Table 1: Performance of the baselines and PROVER with Qwen3.5-2B and Qwen3.5-4B on three benchmarks. We report the mean and sample standard deviation over three evaluation runs.

selecting one successful trajectory and proposing one nonterminal segment of at most four action turns per eligible group. Verification uses $K = 8$ continuations per boundary, and positive segment advantages are incorporated with weight $\lambda = 1$. We evaluate task success rate on ALFWorld and WebShop and semantic answer accuracy on SearchQA, using `gpt-5.4-mini` to judge answer correctness. Additional training and segment-selection details are provided in Appendix A.2.

5 EXPERIMENTAL RESULTS

5.1 MAIN RESULTS

PROVER delivers the strongest overall performance. Table 1 shows that PROVER achieves the highest average score at both model scales and the best result in four of the six model–environment settings. Relative to GRPO, PROVER improves average performance by 9.91% with Qwen3.5-2B and 7.12% with Qwen3.5-4B, with gains in all six settings. These results demonstrate the effectiveness of augmenting trajectory-level GRPO with fine-grained credit targeted toward potentially consequential decisions.

Additional group-level rollouts alone do not explain the gains. Budget-Matched GRPO matches the number of additional policy rollouts used by PROVER but retains trajectory-level credit. PROVER outperforms this control in every setting, with relative improvements ranging from 5.34% to 22.23%. Moreover, Budget-Matched GRPO does not consistently outperform standard GRPO. These results support concentrating the additional rollouts on evaluating potentially consequential segments, rather than simply collecting more trajectories for GRPO.

Targeting consequential decisions outperforms structured rollout baselines. PROVER outperforms SPO-tree across all six settings and SPO-chain in five of the six settings. The gains over SPO-chain are particularly pronounced on SearchQA, reaching 43.95% with Qwen3.5-2B and 17.74% with Qwen3.5-4B. While SPO-chain evaluates fixed segment boundaries and SPO-tree uses a predetermined branching structure, PROVER evaluates the boundaries of a segment selected by the agentic judge. These comparisons support the effectiveness of using judge-guided localization to concentrate continuation-based credit estimation on potentially consequential decisions.

Judge-guided verification outperforms direct critic-based credit. CriticSearch assigns action-level credit directly from binary labels produced by a retrospective LLM critic, whereas PROVER uses an agentic judge to select a segment and estimates its advantage from downstream outcomes.

Method	Generated Tokens / Step (K)			Judge Cost / Step (¢)			Time / Step (min)		
	ALF	WS	SQA	ALF	WS	SQA	ALF	WS	SQA
GRPO	237.0	97.1	37.6	–	–	–	3.39	0.82	1.39
SPO-tree	**153.2**	**71.0**	63.4	–	–	–	7.18	4.85	4.65
SPO-chain	421.2	172.3	72.3	–	–	–	6.73	5.07	4.77
CriticSearch	227.1	136.3	45.8	17.31	**8.54**	7.40	4.64	3.15	3.25
PROVER	242.6	108.4	**43.9**	**4.00**	10.81	**5.69**	**3.73**	**2.00**	**2.25**

Table 2: Per-step training cost for Qwen3.5-4B on ALFWorld (ALF), WebShop (WS), and SearchQA (SQA). Generated tokens count policy-rollout outputs. Judge cost is calculated from recorded `gpt-5.4-mini` API usage using OpenAI's published pricing as of August 2026. Time is the mean wall-clock time per optimizer step in minutes.

Both use the same underlying LLM for the critic or judge. PROVER achieves a higher average score at both model scales and outperforms CriticSearch in five of the six settings. These results support the proposed separation between segment selection and outcome-based credit estimation, compared with directly translating critic judgments into credit.

5.2 TRAINING COST ANALYSIS

Having established the performance gains of PROVER, we next examine the additional computation required to obtain them. Table 2 reports generated policy tokens, judge API cost, and optimizer-step wall time for Qwen3.5-4B training.

Selective evaluation of potentially consequential decisions limits additional policy generation. PROVER selects one short segment from one successful trajectory per eligible group and evaluates only its two boundary states. SPO-chain instead evaluates multiple boundaries for every eligible successful trajectory, while CriticSearch scores every scorable turn in all eight trajectories. Relative to GRPO, PROVER increases generated policy tokens per step by 2.4% on ALFWorld, 11.6% on WebShop, and 16.8% on SearchQA. Together with the strongest average performance in Table 1, these results show that selectively evaluating one potentially consequential segment provides fine-grained credit with modest additional policy-generation overhead.

PROVER achieves the lowest per-step wall time among the evaluated fine-grained methods. Table 2 shows that PROVER has lower per-step wall time than SPO-tree, SPO-chain, and CriticSearch across all three environments. This advantage is not explained by generation volume alone: SPO-tree generates fewer policy tokens than GRPO on ALFWorld and WebShop but takes substantially longer per optimizer step, consistent with additional execution overhead from sequential branching and environment-state restoration. SPO-chain incurs additional rollout cost by evaluating multiple segment boundaries in every eligible successful trajectory, whereas PROVER evaluates only two boundaries of one selected segment. Similarly, CriticSearch scores every scorable action across all eight trajectories in an eligible group, whereas the agentic judge in PROVER selectively inspects trajectory evidence to propose one segment, with 38.4% lower judge API cost on average across the three environments. Together, these comparisons show that concentrating evaluation on a single proposed segment can provide segment-level credit with lower per-step wall time than the evaluated fine-grained alternatives.

5.3 EFFECT OF JUDGE MODEL CHOICE

A natural question is whether PROVER requires an expensive frontier-scale model as its external judge. We compare three models from the GPT-5.4 family and a locally served Qwen3.5-9B judge while holding the Qwen3.5-4B policy backbone, training procedure, and outcome-verification mechanism fixed. We also include a `Random` baseline that replaces the judge with random sampling of a valid nonterminal segment from the same selected successful trajectory, while retaining the verification budget and credit-assignment rule.[2] Table 3 reports policy performance across the three

[2]The random sampling procedure is detailed in Appendix A.3.

Agentic Judge	Policy Performance (%)			SearchQA Proposal Diagnostics	
	ALFWorld	WebShop	SearchQA	Accept (%)	Mean Len.
`Random`	94.0	42.4	42.8	19.5	2.38
`Qwen3.5-9B`	**98.5**	44.2	45.8	71.2	1.45
`gpt-5.4-nano`	94.8	46.2	43.5	72.4	1.25
`gpt-5.4-mini`	98.3	**47.9**	**47.0**	62.5	1.18
`gpt-5.4`	96.3	47.2	43.8	58.5	1.03

Table 3: Effect of segment selection and judge model choice on Qwen3.5-4B policy training. `Random` replaces judge-guided selection with random segment sampling while retaining the same outcome-verification and credit-assignment procedure. We additionally report proposal diagnostics on SearchQA: acceptance rate ($\widehat{\Delta}_{\text{seg}} > 0$) and mean segment length in agent turns.

environments, together with proposal acceptance rate ($\hat{\Delta}_{\text{seg}} > 0$) and mean segment length on SearchQA.

Informed segment selection concentrates evaluation on more promising candidate segments. All four agentic judges outperform random selection across all three environments. On SearchQA, random selection yields an acceptance rate of 19.5%, compared with 58.5–72.4% for the agentic judges, despite proposing substantially longer segments on average (2.38 vs. 1.03–1.45 turns). This suggests that the judges more precisely localize potentially consequential segments, rather than obtaining positive estimates simply by covering more actions. Together with the policy results, this supports the benefit of judge-guided localization over allocating fine-grained credit to arbitrarily selected segments.

Within the GPT-5.4 family, the lower acceptance rates of `gpt-5.4-mini` and `gpt-5.4` accompany shorter proposals than those of `gpt-5.4-nano`. A plausible explanation is that these models follow the minimal-segment instruction more closely, which requires more precise localization, whereas longer segments span more environment interactions and observations and may be more likely to yield a positive $\hat{\Delta}_{\text{seg}}$. However, lower acceptance does not necessarily imply less effective selection, as `gpt-5.4-mini` produces stronger downstream policies than `gpt-5.4-nano` across all three environments.

Effective segment proposal does not require a frontier-scale judge. Although informed selection matters, using a frontier-scale model does not necessarily improve the resulting policy. Both `gpt-5.4-mini` and `gpt-5.4-nano` improve over standard GRPO across all three environments, and `gpt-5.4-mini` achieves the highest scores on WebShop and SearchQA. The locally served Qwen3.5-9B judge achieves the highest reported ALFWorld score and exceeds GRPO on SearchQA, although it falls below GRPO on WebShop. In contrast, `gpt-5.4` does not achieve the highest score on any environment. Together, these findings support the practical design of PROVER: smaller or locally served models can identify useful candidate segments, while environment rollouts supply the numerical credit signal. Judge choice remains task-dependent, but frontier-scale capability is unnecessary for strong performance in the evaluated settings.

6 CONCLUSION

We present PROVER, a framework for targeting potentially pivotal decisions for fine-grained credit assignment in agentic reinforcement learning. An agentic judge localizes a candidate segment, while outcome verification estimates its advantage from fresh current-policy continuations. Incorporating positive segment-advantage estimates into GRPO yields the strongest average performance among the evaluated baselines at both model scales across three agent environments, with modest additional generation overhead. Our analyses show that informed segment selection contributes to these gains without requiring a frontier-scale judge. Together, these findings support a practical role for LLM judgment in agentic reinforcement learning: identifying where fine-grained credit is most useful, while leaving the numerical credit signal to downstream environment outcomes.

AI USE STATEMENT

We used generative AI tools to assist with experiment implementation and debugging, quantitative result analysis, and manuscript revision. In particular, these tools helped implement and test training and evaluation infrastructure, inspect experimental artifacts, and improve the clarity and structure of the paper. All AI-assisted code was reviewed and tested against the intended behavior, and all reported results and numerical claims were checked against the underlying experiment artifacts or independently recomputed by the authors. We did not use generative AI to create the benchmark datasets or ground-truth task outcomes. The authors reviewed all AI-assisted work and take responsibility for the final text, claims, code, and artifacts presented in this paper.

REFERENCES

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJDaqqveg.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=9SkkifLopZ.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

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. In *International Conference on Learning Representations*, volume 2026, pp. 37909–37926, 2026a.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026b.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *Advances in Neural Information Processing Systems*, 38:114399–114431, 2026.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. Treerl: Llm reinforcement learning with on-policy tree search. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12355–12369, 2025.

Tenghao Huang, Dongwon Jung, Vaibhav Kumar, Mohammad Kachuee, Xiang Li, Puyang Xu, and Muhao Chen. Planning and editing what you retrieve for enhanced tool learning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 975–988, 2024.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 87362–87388, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Myx2kJFzAn.

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*, 7:452–466, 2019.

Yoonsang Lee, Howard Yen, Xi Ye, and Danqi Chen. Agentic aggregation for parallel scaling of long-horizon agentic tasks. In *Second Workshop on Agents in the Wild: Safety, Security, and Beyond*, 2026. URL https://openreview.net/forum?id=hXzAocijH7.

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 *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Jianbo Lin, Xiaomin Yu, Yi Xin, Yifu Guo, Zhuosong Jiang, Zhongqi Yue, Weishi Wang, Heqing Zou, Chengwei Qin, and Hui Xiong. Icrl: Learning to internalize self-critique with reinforcement learning. *arXiv preprint arXiv:2605.15224*, 2026.

Haolin Liu, Dian Yu, Sidi Lu, Yujun Zhou, Rui Liu, Zhenwen Liang, Haitao Mi, Chen-Yu Wei, and Dong Yu. Save the good prefix: Precise error penalization via process-supervised rl to enhance llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 35450–35477, 2026.

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 *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 9802–9822, 2023.

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*, pp. 5687–5711, 2023.

Prime Intellect. prime-rl, 2025. URL https://github.com/PrimeIntellect-ai/prime-rl.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *Advances in Neural Information Processing Systems*, 38:105523–105553, 2026.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Junzhe Wang, Zhiheng Xi, Yajie Yang, Hao Luo, Shihan Dou, Tao Gui, and Qi Zhang. Enhancing llm-based search agents via contribution weighted group relative policy optimization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31704–31718, 2026.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7920–7939, 2025.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, et al. Agentgym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27914–27961, 2025.

Yuanda Xu, Zhengze Zhou, Hejian Sang, Xiaomin Li, Jiaxin Zhang, Xinchen Du, Sen Na, Zhipeng Wang, and Alborz Geramifard. Triage: Role-typed credit assignment for agentic reinforcement learning. *arXiv preprint arXiv:2606.32017*, 2026.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Xiaoying Zhang, Yipeng Zhang, Hao Sun, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. Critique-grpo: Advancing llm reasoning with natural language and numerical feedback. *arXiv preprint arXiv:2506.03106*, 2025.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Zijie Zhao, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Criticsearch: Fine-grained credit assignment for search agents via a retrospective critic. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12272–12290, 2026.

Ziqi Zhao, Zhaochun Ren, Jiahong Zou, Liu Yang, Zhiwei Xu, Xuri Ge, Zhumin Chen, Xinyu Ma, Daiting Shi, Shuaiqiang Wang, et al. Reinforced efficient reasoning via semantically diverse exploration. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 47994–48007, 2026.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024a.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*, 2024b.

A Experiment Details

A.1 Agent Environment Details

We use the AgentGym implementations of ALFWorld, WebShop, and SearchQA (Xi et al., 2025). The system and user prompts below are reproduced verbatim from our implementation; brace-delimited fields indicate dynamic content in the environment-observation templates. The policy responds through native tool calling with exactly one `act(action=...)` call per turn.

ALFWorld. ALFWorld (Shridhar et al., 2021) is a text-based embodied environment for household tasks. The initial observation contains the task description, current location, visible objects, and exact available actions. Each subsequent observation reports the result of the previous action and an updated action set. The agent must choose an exact available action string, which may navigate to a location or manipulate an object through operations such as taking, opening, placing, heating, cooling, or cleaning. We use the text-world version with a maximum of 50 turns. We train on the canonical 2,620 training games and evaluate on all 134 valid-unseen games. The evaluation metric is task success accuracy.

ALFWorld Prompt Template

```
SYSTEM
You are controlling an agent in the ALFWorld household environment.
The initial task, current room, and exact AVAILABLE ACTIONS are
provided before your first turn.
Call act with exactly one currently available action on every turn.
Do not invent actions.  Continue until the environment reports success
or the turn budget ends.

USER
Complete the ALFWorld task.

ENVIRONMENT OBSERVATION TEMPLATE
OBSERVATION:
{observation}
AVAILABLE ACTIONS:
- {action_1}
- {action_2}
...
DONE: {done}
REWARD: {reward}
```

WebShop. WebShop (Yao et al., 2022) presents a natural-language shopping instruction and a rendered e-commerce page. Each observation contains the visible page text and the currently available interaction affordances. The agent may issue `search[query]` when a search bar is available or select an exact visible target with `click[target]`. Through these actions, it navigates search results and product pages, chooses product options, and completes a purchase. We allow at most 15 turns. We train on the 10,587 canonical training goals using binary task success as the reward. We evaluate on all 500 canonical test goals and report exact binary task success metrics.

WebShop Prompt Template

```
SYSTEM
You are controlling an agent in the WebShop environment.
The shopping instruction, current page, and available search/click
actions are provided before your first turn.
Call act exactly once on every turn.  Use search[non-empty query] only
when search is available, or click[target] with an exact current
target.
Continue until a purchase ends the episode or the turn budget ends.

USER
Complete the WebShop task.
```

```
ENVIRONMENT OBSERVATION TEMPLATE
OBSERVATION:
{observation}
AVAILABLE ACTIONS:
- {visible_action_1}
- {visible_action_2}
...
DONE: {done}
SUCCESS: {success}
SCORE: {native_reward}
```

SearchQA. SearchQA presents a question and allows the agent to alternate between evidence retrieval and answer submission. A <search>query</search> action returns retrieved passages inside the next observation, which the agent can use to formulate later queries. A terminal <answer>answer</answer> action submits a concise final answer. We utilize a FAISS index (Douze et al., 2024) over the 2018 Wikipedia corpus with E5-base-v2 query embeddings used by Jin et al. (2025). Agents are allowed at most 30 turns to complete a task.

The training pool contains 169,615 questions from the training splits of (Jin et al., 2025) consisting of NQ (Kwiatkowski et al., 2019) and HotpotQA (Yang et al., 2018). Evaluation uses a fixed stratified manifest of 400 held-out questions used by Lin et al. (2026): 50 each from NQ, TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), HotpotQA, 2WikiMultiHopQA (Ho et al., 2020), and Bamboogle (Press et al., 2023), and 100 from MuSiQue (Trivedi et al., 2022). Training uses a binary normalized exact-match reward. For evaluation, we report semantic-equivalence accuracy using GPT-5.4-mini as an LLM judge to recognize correct answer variants that exact matching may miss.

SearchQA Prompt Template

```
SYSTEM
You are answering a question with the SearchQA environment.
The exact question is provided before your first turn.
Call act exactly once on every turn with exactly one argument named
action.
The action value must be one complete string with both tags:  use
<search>non-empty query</search> to retrieve evidence or
<answer>concise final answer</answer> to finish.
Never omit </search> or </answer>, and never send query or answer as a
separate argument.
Base the final answer on retrieved evidence and do not invent search
results.

USER
Answer the SearchQA question.

ENVIRONMENT OBSERVATION TEMPLATE
OBSERVATION:
{observation}
AVAILABLE ACTIONS:
- <search>non-empty query</search>
- <answer>concise final answer</answer>
DONE: {done}
REWARD: {reward}
```

A.2 Additional Implementation Details

Training configuration. The standard GRPO configuration has a batch size of 128 trajectories, organized into 16 groups of eight trajectories. We use AdamW with a learning rate of 10^{-6} and weight decay of 0 on ALFWorld and WebShop and 0.1 on SearchQA. Training uses a sampling temperature of 1 and a maximum completion length per turn of 512 tokens on ALFWorld and WebShop and 256 tokens on SearchQA. Each run uses four A100 SXM GPUs for policy inference and four

for training. The baselines use the same policy backbones and shared optimization settings, with method-specific changes to credit assignment and rollout allocation detailed in Appendix A.3.

Segment selection and verification. For $G = 8$, the eligibility criterion selects groups containing one, two, or three successful trajectories. From each eligible group, we select the successful trajectory with the shortest horizon and ask the judge to identify a contiguous nonterminal segment of at most four action turns. We restore the exact agent–environment states immediately before and after the segment and sample $K = 8$ current-policy continuations from each boundary. The difference between their mean terminal rewards estimates the segment advantage. When this estimate is positive, we add it to the GRPO advantage of each trainable policy token within the segment, using $\lambda = 1$ in Equation 4. All other source tokens retain their GRPO advantages, and ineligible groups retain standard GRPO updates. Boundary continuations are used only for estimation and are not included in the policy-training batch.

A.3 Baseline Implementation

GRPO. GRPO assigns Equation 1 uniformly to all trainable assistant tokens in a trajectory. It uses neither localized credit nor auxiliary policy rollouts and therefore provides the outcome-only reference.

Budget-Matched GRPO. Budget-Matched GRPO uses the same outcome-only credit in Equation 1 but replaces PROVER's boundary continuations with additional rollouts. Each update contains 16 groups, and we use two phase-specific group sizes to match the corresponding PROVER run's realized rollout budget over 100 steps. Table 4 reports the number of trajectories sampled per group in each phase. Thus, Budget-Matched GRPO controls for additional policy sampling without using boundary restoration or localized credit.

Model	Benchmark	Phase 1 steps	Rollouts/group	Phase 2 steps	Rollouts/group
Qwen3.5-2B	ALFWorld	1–84	11	85–100	12
Qwen3.5-2B	WebShop	1–94	10	95–100	11
Qwen3.5-2B	SearchQA	1–70	10	71–100	11
Qwen3.5-4B	ALFWorld	1–26	8	27–100	9
Qwen3.5-4B	WebShop	1–35	10	36–100	11
Qwen3.5-4B	SearchQA	1–36	9	37–100	10

Table 4: Number of trajectories sampled per group in each phase of Budget-Matched GRPO. Every update contains 16 groups.

GiGPO. GiGPO augments trajectory-level GRPO advantages with step-level advantages obtained by grouping action turns that share the same environment state and comparing their discounted downstream returns. This provides local credit from state recurrence within the sampled rollout group, without auxiliary continuation rollouts or an external judge. We use the same policy backbones and shared training configuration as the other baselines.

SPO-chain. SPO-chain assigns segment j the advantage $A_j = \gamma V_j - V_{j-1}$, where V_{j-1} and V_j are Monte Carlo values at the segment's start and end boundaries and γ is the discount factor. It masks tokens with generating-policy probability at least 0.9. Applying the original method to every trajectory in every group is impractical for long, stateful agent trajectories. We therefore apply it only to successful trajectories in groups containing one, two, or three successes following PROVER's setup and divide their action turns into up to three contiguous, approximately equal segments; all other trajectories retain GRPO credit. Let V_0 be the original group's mean reward, V_1 and V_2 the internal-boundary values, and $V_3 = R_i$ the trajectory reward. With no temporal discounting ($\gamma = 1$), the three advantages are $A_1 = V_1 - V_0$, $A_2 = V_2 - V_1$, and $A_3 = R_i - V_2$. Each internal value uses $K = 8$ exact-state continuations. Fixed thirds keep the budget comparable to PROVER, although SPO-chain remains more expensive because it measures every successful trajectory rather than one selected success.

SPO-tree. SPO-tree constructs a three-level binary tree with branching factors 2–2–2, giving 14 sampled branches and eight terminal leaves per training group. For each child c of parent p, it assigns the branch advantage $A(c) = V(c) - V(p)$ with no temporal discounting, where a leaf value is its binary terminal reward and each internal value is the mean reward of its descendant leaves. The first two levels advance by four environment actions per level in ALFWorld and one action per level in WebShop and SearchQA, while the final level continues to the terminal condition or episode horizon. Each child resumes from the parent's exact environment state. We pack the 14 branches into eight leaf containers and assign each branch to exactly one container, preventing duplicated credit for shared prefixes. As in SPO-chain, tokens with generating-policy probability at least 0.9 are masked from the loss. If tree collection fails, we discard the incomplete tree and sample a fresh group of eight standard GRPO trajectories.

CriticSearch. CriticSearch uses the same frozen `gpt-5.4-mini` as its retrospective critic. Because labeling every trajectory in every group would require many costly critic calls, we apply CriticSearch only to groups containing one, two, or three successes, matching PROVER's setup; all other groups use GRPO. For each eligible group, the critic labels every trajectory, and CriticSearch uses

$$A_t = 0.75A_i^{\mathrm{GRPO}} + 0.25A_t^{\mathrm{critic}}. \tag{5}$$

Because CriticSearch labels all eight trajectories in each eligible group, it uses eight critic calls per group, whereas PROVER sends only one selected successful trajectory to its agentic judge. SearchQA critics receive reference answers and label valid search actions. ALFWorld and WebShop critics receive only policy-visible trajectories and terminal success, never hidden environment state. Invalid critic outputs or token alignment produce a whole-group GRPO fallback.

Random segment selection. For the random-selection baseline introduced in Section 5.3, we replace the agentic judge with random selection of a starting turn and a segment length between one and four turns from the selected successful trajectory. Intervals that violate the proposal-validity constraints are discarded and receive no additional credit. All other settings, including outcome verification and credit assignment, remain identical to PROVER.

A.4 AGENTIC JUDGE PROMPT

The agentic judge receives the complete turn-indexed successful trajectory and compact indexes of the failed trajectories from the same GRPO group. The prompt asks it to diagnose a recurrent failure mode and return the shortest non-terminal expert segment that solves or avoids that failure. The judge is given trajectory search and bounded segment-inspection tools. Its output contains only the inclusive segment bounds and a rationale. We deterministically validate the output schema, turn bounds, action validity, and nonterminal endpoint, allowing up to three additional turns to correct any validation errors.

Agentic Judge Prompt Template

```
SYSTEM
You are an agentic judge identifying a short segment of a verified
successful EXPERT trajectory that will receive additional fine-grained
positive credit on top of the GRPO terminal reward.

Diagnose one recurrent failure mode from the failed trajectories, then
select the shortest EXPERT segment that solves or avoids that
diagnosed failure mode.

You receive the complete EXPERT trajectory directly in the user prompt
and compact indexes for all failed trajectories from the same task.
Each failed index gives its stable traj_id and inclusive
start_turn/end_turn bounds for get_segment.

Diagnose a recurrent failure mode:
The search_trajectory and get_segment tools are optional and available
if useful.  Identify a failure mode that recurs in multiple failed
trajectories.  Treat different actions as equivalent when they reach
equally useful states, and do not group failures that require
different fixes.
```

```
Select the EXPERT segment that solves the failure mode:
From the complete EXPERT, find the shortest contiguous segment whose
actions solve or avoid the diagnosed failure mode.  A failure that
already performs the proposed EXPERT transition is not valid
contrastive evidence merely because it fails elsewhere.

The policy tokens in every selected turn will themselves receive
additional positive credit.  Select only actions whose own behavior
should be reinforced.  Judge each action together with the transition
it causes, not merely by whether a later state is useful.  A
malformed, unavailable, unsuccessful, or error-producing action must
not be selected merely because recovering from its error later helped
the EXPERT succeed.  A valid corrective action after an error may be
selected when that corrective action itself is useful, but the
preceding bad action must be excluded.

Segment-selection rules:
1.  Select the shortest contiguous segment that directly solves or
avoids the diagnosed failure mode.  Every selected turn must be
necessary to that transition and worth reinforcing; exclude merely
contextual or adjacent turns.
2.  Select behavior that materially advances the task, not behavior
that is merely late in a successful trajectory.  Do not select
continued retrieval or repeated activity when the prior state is
already sufficient unless it produces a necessary new state change.
3.  Treat actions as equivalent when they reach equally useful states,
even if their routes, queries, object identities, or wording differ.
4.  The state after segment_end_turn must be nonterminal, and the
terminal action must not be selected.

You do not write a hint, propose new actions, or assign reward.  You
only select an existing EXPERT segment.  Use only recorded
observations and affordances visible to the rollout policy.
Deterministic validation separately checks the output schema, bounds,
action grammar and affordances, recorded action outcomes, terminality,
and sampled tokens; your responsibility is the semantic and causal
judgment.

If you use search previews, treat them as bounded evidence rather than
complete local behavior; use get_segment when more context would help.
Failed trajectories are contrastive evidence, not demonstrations.

Return only one JSON object with exactly these fields:
{
"segment_start_turn":  <inclusive integer>,
"segment_end_turn":  <inclusive integer strictly before the EXPERT's
final recorded turn, whose resulting state is nonterminal>,
"rationale":  "<the recurrent contrast, visible state before and after
the selected segment, why each selected action itself deserves
additional positive credit, and why this is the minimal complete
causal transition>"
}
Do not include markdown or any text outside the JSON object.

USER
{
"task":  {task},
"expert":  {complete_turn_indexed_expert_trajectory},
"failed_trajectory_indexes":  {compact_failed_trajectory_indexes},
"instructions":  "The expert is provided in full turn order above.
Select one expert segment satisfying the system prompt.  The
trajectory tools are optional."
}

TOOLS
search_trajectory(query, k=10):  Globally rank bounded turn previews
across all failed trajectories.
```

```
get_segment(traj_id, start_turn, end_turn): Inspect a bounded
inclusive turn interval from the expert or a failed trajectory.
```

B CONDITIONAL UNBIASEDNESS OF THE SEGMENT ESTIMATOR

This section proves the conditional unbiasedness claim in Section 3.4 and characterizes the finite-sample variance of the raw pre/post estimator. The result does not apply to the positive part $[\widehat{\Delta}_{\text{seg}}]_+$ or to the complete judge-selected training update.

Conditional unbiasedness. Let $\mathcal{F}$ contain the source rollout group and all information used to select τ^{seg}. Under exact boundary reconstruction, a fixed policy π, and conditionally independent continuation samples, the selected segment and its boundary states are fixed after conditioning on $\mathcal{F}$. Each empirical boundary value is an unbiased sample mean:

$$\mathbb{E}\Big[\widehat{V}^{\pi}_{\text{pre}} \mid \mathcal{F}\Big] = V^{\pi}(s_{\text{pre}}), \qquad \mathbb{E}\Big[\widehat{V}^{\pi}_{\text{post}} \mid \mathcal{F}\Big] = V^{\pi}(s_{\text{post}}).$$

Linearity of expectation and Equation 2 therefore give

$$\begin{aligned}\mathbb{E}\Big[\widehat{\Delta}_{\text{seg}} \mid \mathcal{F}\Big] &= V^{\pi}(s_{\text{post}}) - V^{\pi}(s_{\text{pre}}) \\ &= A^{\pi}_{\text{seg}}(s_{\text{pre}}, \mathbf{a}_{\ell:r}).\end{aligned}$$

Thus, the raw $\widehat{\Delta}_{\text{seg}}$ is conditionally unbiased for the selected segment's advantage. □

Finite-sample variance. For binary terminal outcomes, let $p_{\text{pre}} = V^{\pi}(s_{\text{pre}})$ and $p_{\text{post}} = V^{\pi}(s_{\text{post}})$. The two sample means have conditional variances $p_{\text{pre}}(1-p_{\text{pre}})/K$ and $p_{\text{post}}(1-p_{\text{post}})/K$. Their conditional independence therefore gives

$$\text{Var}(\widehat{\Delta}_{\text{seg}} \mid \mathcal{F}) = \frac{p_{\text{post}}(1-p_{\text{post}}) + p_{\text{pre}}(1-p_{\text{pre}})}{K}.$$

Thus, the estimator becomes more precise as K increases, with variance decreasing at rate $1/K$. For finite K, however, sampling noise can change the sign of $\widehat{\Delta}_{\text{seg}}$, so a positive estimate provides empirical evidence of benefit rather than a conclusive guarantee.

Scope of the unbiasedness guarantee. The conditional-unbiasedness guarantee applies only to the raw estimator for a fixed segment after conditioning on the judge's selection. PROVER instead uses its positive part,

$$[\widehat{\Delta}_{\text{seg}}]_+ = \max(\widehat{\Delta}_{\text{seg}}, 0),$$

which is generally biased because negative sampling errors are replaced by zero while positive sampling errors are retained. Consequently,

$$\mathbb{E}\Big[[\widehat{\Delta}_{\text{seg}}]_+ \mid \mathcal{F}\Big] \neq [A^{\pi}_{\text{seg}}(s_{\text{pre}}, \mathbf{a}_{\ell:r})]_+$$

in general. For example, even when the true segment advantage is zero, finite-sample estimates can be positive or negative; positive filtering discards the negative estimates but retains the positive ones, yielding a positive expected credit bonus. The complete PROVER update additionally includes judge-guided segment selection, positive filtering, assigning the resulting bonus only to tokens within the selected segment, scaling by λ, and adding the bonus to the GRPO trajectory advantage. Therefore, conditional unbiasedness of the raw Monte Carlo estimator does not imply that the complete PROVER update is an unbiased estimator of the standard policy gradient. Nor does the boundary comparison identify separate causal effects for individual actions or tokens within the segment.