LEARNING AT THE BRANCH POINT: STATE-LOCAL POLICY OPTIMIZATION FOR SEARCH AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) has emerged as an effective approach for training large language model (LLM)-based search agents on knowledge-intensive tasks requiring iterative information seeking. However, existing RL methods typically optimize at the trajectory level, where credit assignment based primarily on whole-trajectory outcomes can obscure the individual contributions of useful and redundant search actions. To address this limitation, we propose State-Local Policy Optimization (SLPO), which shifts the optimization unit from whole trajectories to individual actions under the same search history. At each candidate state, SLPO executes and scores alternative actions, deriving local advantages from their relative rewards for action-level updates. To explore diverse search directions while limiting rollout cost, we structure exploration as a semantic tree, clustering similar actions and expanding only cluster representatives while retaining all sampled alternatives for local comparison. We use precomputed evidence-to-document mappings to reward actions that retrieve new evidence without an LLM judge during training. We further prioritize states with greater within-state reward dispersion for more informative local comparisons. Across seven QA benchmarks, SLPO improves answer accuracy (COVER-EM) over the strongest baseline by 3.54 and 3.84 percentage points for 4B- and 8B-parameter models, respectively, while using comparatively few searches. These results demonstrate the effectiveness of state-local credit assignment for training LLM-based search agents.

1 INTRODUCTION

Large language models increasingly act as search agents, interleaving reasoning and retrieval to answer knowledge-intensive questions (Yao et al., 2023). They must decide what to retrieve, how to use the evidence, and when to answer. Reinforcement learning (RL) trains these sequential decisions through interaction. Search-R1 and ReSearch demonstrate that final-answer feedback can teach models to reason with search (Jin et al., 2025; Chen et al., 2025).

Recent work extends this approach with process supervision (Zheng et al., 2025; Liu et al., 2026), state grouping, and tree-based credit assignment (Feng et al., 2025; Zhang et al., 2026a). Yet finer feedback alone does not ensure action-specific credit when the optimization unit remains a whole trajectory. Actions under different search histories can still share an outcome-derived advantage.

This matters because a successful trajectory can contain redundant searches, while an unsuccessful one can still acquire useful evidence. Moreover, the value of a search depends on history: evidence that is new at one state may already be covered at another. This motivates scoring alternatives independently under the same history and preserving their relative credit during policy updates.

We therefore propose State-Local Policy Optimization (SLPO; Figure 1), a *state-local credit-assignment formulation* that aligns rewards, relative comparisons, and policy updates at the state–action level. At each candidate state, we independently execute alternative actions and assign action-specific transition rewards for new evidence or answer quality. Within-state normalization converts these rewards into relative advantages for selected groups. Each advantage updates only the corresponding action tokens; the shared history provides context.

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.

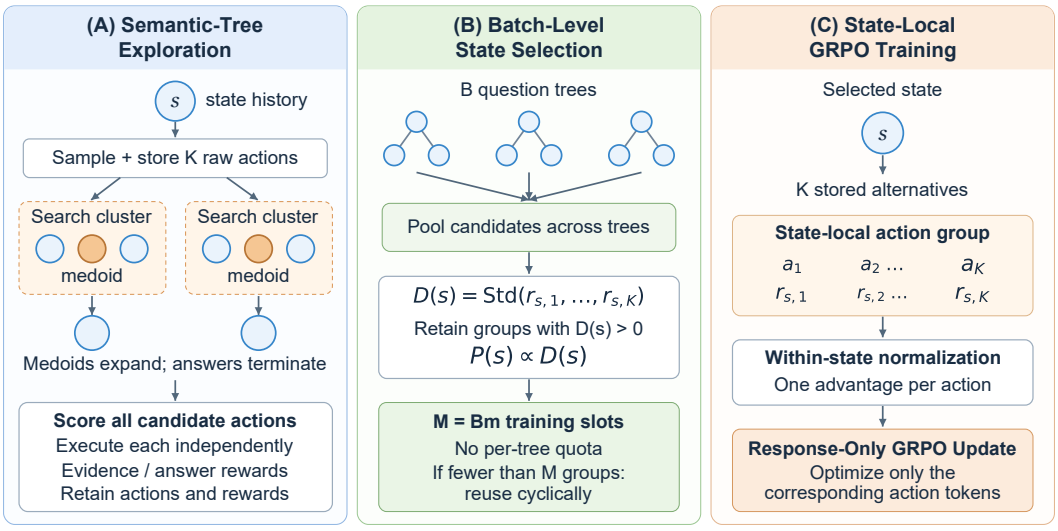


Figure 1: Overview of SLPO. (A) Semantic trees store all actions for independent scoring but expand only search-cluster medoids. (B) Reward dispersion guides batch-level state selection, with group reuse when needed. (C) Within-state advantages update the corresponding action tokens only.

To construct these comparisons, semantic trees retain same-state alternatives while expanding only representative search actions. A fixed offline evidence mapping supplies action-specific feedback without an LLM judge during training. We pool scored states across the batch and prioritize greater reward dispersion, since identical rewards yield no relative signal. On a seven-dataset validation mixture, this procedure improves weighted COVER-EM by 3.54 and 3.84 points over the strongest baseline at 4B and 8B, respectively, with fewer search calls than Search-R1 and PiCA. It also outperforms trajectory-level controls with evidence-guided credit and tree-based trajectory construction, supporting the state-local formulation.

Our contributions are:

- We introduce a state-local credit-assignment formulation that connects independently scored transition rewards, within-state relative advantages, and action-level policy updates.
- We develop a training procedure that uses semantic trees and fixed evidence mappings to construct action groups and allocates training slots by batch-level reward dispersion.
- We demonstrate accuracy and search-efficiency gains, with trajectory-level controls supporting the value of the state-local formulation.

2 RELATED WORK

Learning Search Agents. WebGPT learns browser-assisted question answering from demonstrations and human feedback (Nakano et al., 2021), while ReAct prompts models to interleave reasoning and actions (Yao et al., 2023). Search-R1, ReSearch, and R1-Searcher learn interactive retrieval through outcome-based RL (Jin et al., 2025; Chen et al., 2025; Song et al., 2025). Search-R1 and ReSearch mask retrieved text from the policy loss, separating agent-generated tokens from external observations.

Process Supervision and Evidence Credit. Process supervision uses human labels or sampled continuations to evaluate intermediate reasoning (Lightman et al., 2024; Wang et al., 2024). For search agents, approaches include state–action preferences (Xiong et al., 2026; Zhang et al., 2025a), information-gain rewards and redundancy penalties (Zheng et al., 2025), and turn-level evaluation through CriticSearch's frozen retrospective critic or PiCA's learned progress model (Zhang et al., 2026b; Liu et al., 2026). SLPO scores evidence acquisition through a fixed offline mapping, with no learned process evaluator or online judge in RL.

State-Conditioned Credit Assignment. GRPO estimates relative advantages within sampled groups (Shao et al., 2024). Finer-grained methods use Monte Carlo step or segment continuations (Kazemnejad et al., 2025; Guo et al., 2025), repeated-state groups with varying treatment of history (Feng et al., 2025; He et al., 2026), or pooled returns for state–action signatures (Wang et al., 2026). For retrieval, RICE-PO evaluates local branches with retrieval metrics and gates credit propagation to preceding reasoning spans (Li et al., 2026). The main SLPO formulation instead normalizes immediate evidence or answer rewards among stored alternatives at one tree-node history and applies each advantage to the corresponding action response.

Tree-Based Exploration and Credit. Trees support inference-time planning (Zhou et al., 2024) and training-time credit through propagated outcomes (Hou et al., 2025; Yang et al., 2025b), uncertainty-guided branching (Dong et al., 2026; Zong et al., 2026), and outcome comparisons within or across trees or shared prefixes (Ji et al., 2026; Zhao et al., 2026). TreeRPO, for example, forms relative groups from one node's child steps. TreePS-RAG prunes search branches using retrieved-passage overlap and derives step advantages from descendant outcomes (Zhang et al., 2026a). In SLPO, semantic query clusters limit expansion to medoids while all raw actions receive independent evidence scores. Reward dispersion selects states across trees for within-state reward normalization.

Trajectory Decomposition and Context Construction. Agent Lightning constructs a separate context for each model-call transition, but its reported LightningRL implementation assigns episode returns and groups transitions by task (Luo et al., 2025). EvoCUA-1.5 builds context-managed computer-use step samples with redistributed trajectory advantages (Huang et al., 2026). Unlike these uses of trajectory-derived credit, SLPO normalizes action-specific rewards within same-state groups and updates response-only transitions.

3 METHOD

3.1 AGENT STATES AND RESPONSE-ONLY TRANSITIONS

An action is an assistant response containing a search query or a final answer, optionally preceded by reasoning. For question q, let the pre-action history at turn t be

$$H_t = (q, a_0, o_0, \ldots, a_{t-1}, o_{t-1}), \qquad x_t = \mathrm{ChatRender}(H_t), \tag{1}$$

where a_i is an assistant action and o_i is a retrieved observation. This history defines semantic state s_t. On its first expansion, the policy samples actions $y_{t,k} = a_t^{(k)} \sim \pi_\theta(\cdot \mid x_t)$ and stores their content and original generated tokens. Each transition training example pairs the same rendered context x_t used for sampling with one stored action.

A continuous-trajectory representation trains multiple actions in one token stream, whose prefixes and loss masks must remain aligned with rollout. In contrast, as in transition-based agent training (Luo et al., 2025), SLPO constructs a separate context–response example for each action (Figure 2). Earlier turns are re-rendered by the chat template rather than replayed as immutable generated tokens. The current action retains its sampling context x_t and original response tokens; only these response tokens receive policy loss. This avoids cross-turn prefix matching and supports template-native handling of historical reasoning without changing the action being optimized. Appendix A.1 specifies the serialization rules.

3.2 SEMANTIC-TREE EXPLORATION

At every frontier state s, the policy samples and stores a raw action set $\mathcal{A}(s) = \{a^{(1)}, \ldots, a^{(K)}\}$. Search queries are grouped by semantic similarity, while answer actions form singleton clusters. Only the medoid—the most central sampled query—of each search cluster creates a child state for further expansion. This preserves distinct search directions without expanding every surface-form variant. Every raw action is retained for independent reward evaluation and local comparison, including those whose continuations are not expanded. Appendix A.2 gives the construction procedure.

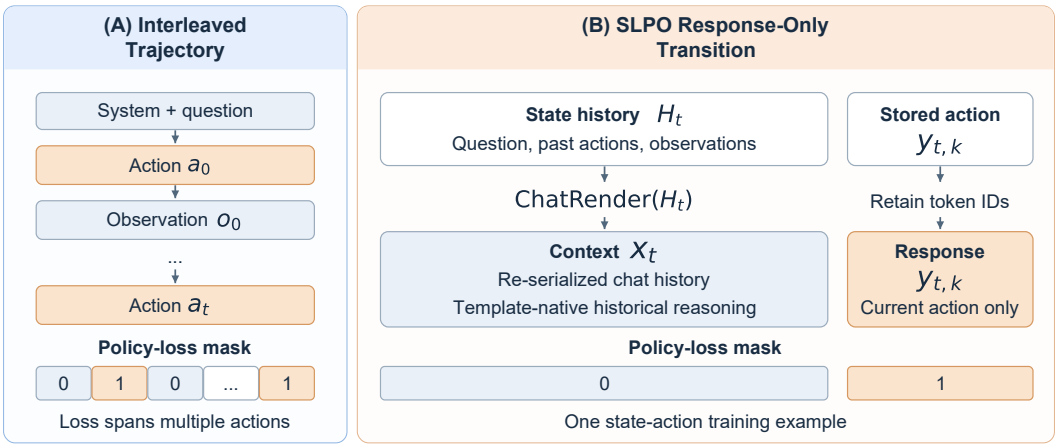


Figure 2: Training representations and policy-loss masks (1: included; 0: excluded). (A) A continuous trajectory interleaves actions with masked observations. (B) SLPO independently renders each state's history through the model's chat template and pairs it with one stored action's original tokens. Only the current response receives policy loss.

3.3 Evidence-Grounded Transition Rewards

Training questions are annotated offline with evidence items (atomic supporting facts) mapped to corpus documents. At each candidate state, stored actions are executed independently: every search query retrieves its own documents rather than inheriting the medoid's results. If $E(a)$ is the set of evidence items reached by action a and $C(s)$ is the set already covered in the history, the newly acquired evidence is $E(a) \setminus C(s)$. Search rewards count these new evidence items and penalize redundant, irrelevant, or failed retrievals. Answer rewards combine correctness with the fraction of annotated evidence covered at that state. Each action's transition reward $r_{s,k}$ is stored before state selection.

Given the fixed evidence mapping, rewards are computed deterministically from retrieved document identifiers, state history, and the parsed action. No LLM judge is needed during RL. Appendix A.4 gives the numerical schedule.

3.4 Batch-Level Reward-Dispersion State Selection

Local normalization yields no relative reward signal when every action in a group receives the same reward. We therefore select states according to the dispersion of their independently scored alternatives. For a batch of B question trees, let $\mathcal{S}_b$ contain the candidate states of tree b, and define

$$\mathcal{S} = \bigcup_{b=1}^{B} \mathcal{S}_b, \qquad D(s) = \mathrm{Std}(r_{s,1}, \ldots, r_{s,K}), \qquad \mathcal{S}^+ = \{s \in \mathcal{S} : D(s) > 0\}. \tag{2}$$

The training budget is $M = Bm$ state slots, where m is a nominal allocation per input, not a per-tree quota. We sample up to M distinct states from $\mathcal{S}^+$ with weights

$$P(s) = \frac{D(s)}{\sum_{u \in \mathcal{S}^+} D(u)}. \tag{3}$$

When fewer than M informative states are available, their stored groups are reused cyclically to fill the slots. Trees can contribute unequal numbers of states, including none. Dispersion determines sampling weights, while within-group normalization determines advantage scale.

3.5 State-Local Relative Optimization

For each selected state s, the transitions $\{(x_s, y_{s,k}, r_{s,k})\}_{k=1}^{K}$ form one comparison group. Here x_s is the rendered context, $y_{s,k}$ is a stored action, and $r_{s,k}$ is its transition reward. Using GRPO in

veRL (Sheng et al., 2025), we compute within-group advantages

$$\hat{A}_{s,k} = \frac{r_{s,k} - \bar{r}_s}{\mathrm{std}(r_s) + \epsilon}, \tag{4}$$

where $\bar{r}_s$ and $\mathrm{std}(r_s)$ are the mean and standard deviation of the group's rewards, and ϵ is a numerical stabilizer. A clipped policy objective applies to response tokens only. State-local grouping determines which actions define the relative baseline; response-only serialization determines which tokens receive loss. Optimization otherwise follows standard critic-free GRPO, with numerical settings in Appendix B.3.

4 EXPERIMENTS

4.1 DATA, RETRIEVAL, MODELS, AND PROTOCOL

Training and Validation. Training uses 77,133 HotpotQA (Yang et al., 2018) questions retained after offline evidence mapping; Table 6 summarizes this training-data preprocessing. All reported scores use a fixed 51,713-example validation mixture of Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), HotpotQA, 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). This same mixture is used for checkpoint selection and reporting. We call the first three datasets single-hop and the last four multi-hop. Per-dataset validation sizes appear in Appendix B.1.

Agent and Retriever. We evaluate Qwen3-4B-Instruct and Qwen3-8B policies (Yang et al., 2025a) with the same Wiki-18 retrieval environment across methods. Validation uses greedy decoding. Retriever, index, rollout, and optimization details appear in Appendices B.2 and B.3.

Metric and Checkpoint Selection. The primary metric, COVER-EM, marks an example correct when the normalized prediction contains any normalized reference answer. Overall, single-hop, and multi-hop scores are example-weighted rather than dataset-macro averages. For main and ablation comparisons, we select each method's maximum overall validation score in the common step window 0–200; ties prefer the earlier step, and all other metrics are reported at that checkpoint. ReAct is the corresponding untrained step-0 policy. Exact normalization and aggregation rules are given in Appendix B.1.

Baselines. We compare ReAct, Search-R1, PiCA, AT^{2}PO, and GiGPO with shared training/validation data, retriever, corpus, and five-search limit. Method-specific objectives and rollouts are retained (Appendix B.4). Trajectory-level GRPO uses eight rollouts to match SLPO's eight-action comparison groups, not total computation. SLPO limits each action to 2,048 tokens and retained history to a rolling 4,096-token prompt budget; trajectory baselines mainly use a 4,096-token response budget. This provides approximate budget-scale alignment across different representations.

4.2 MAIN RESULTS

SLPO achieves 68.34% overall COVER-EM at 4B and 66.30% at 8B, outperforming the strongest baseline, AT^{2}PO, by 3.54 and 3.84 points, respectively (Table 1). It improves both single-hop and multi-hop accuracy at both scales.

The per-dataset results show the breadth of these gains (Table 2). At 4B, SLPO leads on four datasets and AT^{2}PO on three; at 8B, SLPO leads on six and GiGPO on Bamboogle. Bamboogle is the smallest validation set, with 125 examples.

For Qwen3-4B-Instruct, three training seeds evaluated once at step 100 yield $65.46 \pm 0.51\%$ overall COVER-EM (mean $\pm$ sample standard deviation; Appendix B.1). These fixed-checkpoint results are separate from the best-checkpoint comparisons above.

Table 1: Validation results at each method's best 0–200-step checkpoint. Scores are example-weighted COVER-EM percentages; Calls denotes mean search calls. †: policy before RL; –: unavailable comparable call statistics. Within each model block, best accuracy and lowest available call count are **bold and underlined**.

Model	Method	Overall	Single	Multi	Calls ↓
4B	ReAct†	15.74	20.17	10.00	–
	Search-R1	63.91	64.91	62.62	3.11
	PiCA	61.00	62.16	59.50	3.15
	AT^{2}PO	64.80	65.32	64.12	–
	GiGPO	59.40	62.32	55.62	**1.37**
	SLPO (ours)	**68.34**	**69.63**	**66.67**	2.61
8B	ReAct†	24.61	30.49	16.98	–
	Search-R1	58.44	62.64	52.99	2.20
	PiCA	59.78	62.72	55.96	3.10
	AT^{2}PO	62.47	64.66	59.63	–
	GiGPO	58.18	61.72	53.59	**1.34**
	SLPO (ours)	**66.30**	**67.39**	**64.89**	1.95

Table 2: Per-dataset validation COVER-EM (percent) at the selected checkpoints. The first three datasets are single-hop and the remaining four are multi-hop. Within each model block, every column maximum is **bold and underlined**.

Model	Method	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle
4B	ReAct†	10.83	32.38	12.86	11.74	10.44	2.77	3.20
	Search-R1	64.16	74.71	57.33	59.61	71.26	27.72	45.60
	PiCA	59.20	74.21	53.36	57.77	66.62	28.09	52.80
	AT^{2}PO	66.34	75.89	56.69	**65.73**	69.04	**34.05**	**55.20**
	GiGPO	56.38	72.13	56.05	53.11	63.66	23.10	23.39
	SLPO (ours)	**67.56**	**78.49**	**63.12**	64.28	**74.90**	31.82	54.40
8B	ReAct†	28.48	41.20	22.50	18.47	18.09	6.50	20.00
	Search-R1	58.98	73.67	54.83	52.76	58.45	25.11	56.80
	PiCA	58.73	74.61	54.31	53.61	62.99	26.60	55.20
	AT^{2}PO	61.44	75.67	56.74	58.53	65.77	31.11	58.40
	GiGPO	57.20	72.22	54.54	52.80	59.50	24.24	**73.79**
	SLPO (ours)	**64.63**	**77.42**	**60.15**	**62.24**	**72.41**	**34.42**	55.20

The validation curves show different learning dynamics across scales (Figure 3). The 4B policy peaks before the end of training and then fluctuates, whereas the 8B policy peaks at the end of the comparison window.

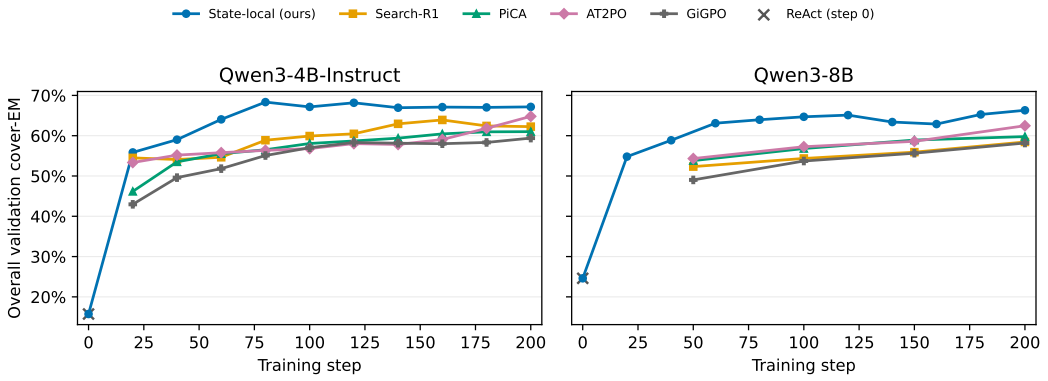


Figure 3: Overall validation COVER-EM throughout the common 0–200-step comparison window. Scores are example-weighted across the seven datasets.

4.3 Ablations: Credit Assignment and Reward

Table 3 compares trajectory-level controls, SLPO, and a reward variant on Qwen3-4B-Instruct. Evidence Credit (EC) uses evidence information for finer-grained credit assignment. The trajectory controls reweight search-token advantages according to new evidence and final-answer correctness, preserving the outcome-derived return and advantage sign. In SLPO, evidence enters action-specific rewards for state-local comparison.

Semantic Tree (ST) adds tree-structured exploration to the evidence-guided trajectory control. State-Local Credit (SLC) uses the tree's alternative actions for within-state comparison and action-level updates. Both SLPO and its Subtree Reward (SR) variant use batch-level reward-dispersion selection. Appendix B.5 gives the control settings.

Table 3: Qwen3-4B-Instruct ablations at the best validation checkpoint in the common 0–200-step window. Checkmarks indicate evidence credit (EC), semantic-tree exploration (ST), state-local credit (SLC), and subtree reward (SR). Both state-local variants select groups using their respective reward dispersion. Scores are COVER-EM percentages; column maxima are **bold and underlined**.

Variant	EC	ST	SLC	SR	Overall	Single	Multi
Trajectory GRPO					63.91	64.91	62.62
+Evidence Credit	✓				65.88	65.88	65.88
+Semantic Tree	✓	✓			63.50	63.36	63.67
SLPO (full)	✓	✓	✓		**68.34**	**69.63**	**66.67**
+Subtree Reward	✓	✓	✓	✓	65.38	67.48	62.65

Evidence-guided trajectory credit improves Trajectory GRPO by 1.97 points overall, with larger gains on multi-hop (3.27) than single-hop (0.97) data. Intermediate evidence thus helps refine trajectory-level credit, but adding semantic-tree trajectory construction removes this gain: the ST control falls 0.41 points below Trajectory GRPO. Its trajectory objective still does not compare the same-state alternatives exposed by the tree, and shared prefixes may correlate the resulting trajectories.

SLPO outperforms the EC and ST trajectory-level controls by 2.46 and 4.84 points overall, respectively. This pattern supports directly using informative action groups for state-local comparison, rather than using the tree only to construct trajectories.

Subtree Reward. SR adds a cluster-shared downstream outcome value to each action's local reward *before* computing reward dispersion and selecting states. It therefore shapes both within-state advantages and state selection. Under the tested scaling, overall accuracy falls by 2.96 points relative to SLPO, with a larger decline on multi-hop than single-hop data.

Training Cost. Over updates 1–200 on Qwen3-4B-Instruct, SLPO averages 147.1 seconds per training step versus 70.1 for *+Evidence Credit*, excluding validation: $2.10\times$ the cost. Most of the extra time comes from constructing and scoring local groups and from actor updates (Appendix C.1).

With comparable learning-rate and PPO update settings, SLPO reaches 68.34% validation accuracy at step 80, while the control peaks at 65.88% at step 200. Cumulative training time at these checkpoints is 2.63 and 3.89 hours, respectively. Thus, the higher per-step cost is offset by faster optimization progress: SLPO reaches higher accuracy in fewer updates and less training time in this comparison.

4.4 Search Efficiency

At both model scales, SLPO attains higher validation accuracy with fewer search calls than Search-R1 and PiCA (Figure 4). GiGPO uses fewer calls, while its accuracy is 8.94 and 8.12 points below SLPO at 4B and 8B. AT^{2}PO is omitted because comparable search-call statistics are unavailable.

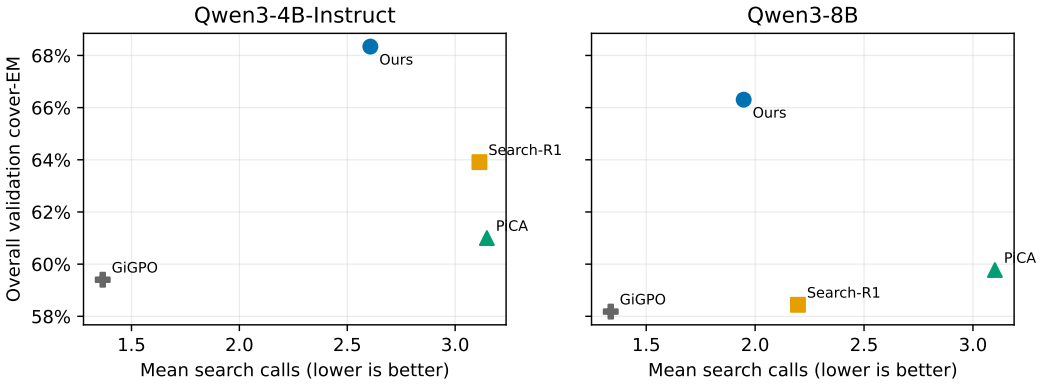


Figure 4: Overall validation COVER-EM versus mean search calls at each selected checkpoint; upper-left is better.

5 DISCUSSION AND LIMITATIONS

The ablations support the state-local formulation as a whole. Local rewards, within-state normalization, response-only updates, and state selection are coupled, so the controls do not isolate their individual contributions. Subtree Reward likewise changes rewards and selection together; its result applies to the tested relative scale.

The three-seed 4B check at step 100 complements the cross-scale, multi-dataset results and validation curves, but does not measure seed variability over the full 0–200-step model-selection protocol. Training cost (Appendix C.1) limits this check to a single validation per seed.

State discovery leaves substantial room for efficiency gains: 79.4% of 4B candidate groups have zero reward dispersion. Clustering reduces branch expansion, but all candidate actions still require scoring. More aggressive merging, adaptive expansion, or early stopping could reduce this cost at the expense of exploration diversity. Dispersion measures reward differentiation rather than policy improvement, and batch-level selection can concentrate updates on a subset of question histories.

Actor updates take 38.3% of 4B training time, while reused groups occupy 69.4% of training slots. Reducing the number of state slots could lower actor-update cost without changing the exploration procedure, although its effects on learning and state coverage remain untested (Appendix C.2).

The method depends on usable evidence mappings and stable corpus identifiers. Preprocessing excludes questions without mapped evidence (Table 6), and automatic mappings may miss valid alternative evidence. Extending state-local credit to dynamic corpora, tasks without gold evidence, or other tools requires reproducible process signals for those settings.

ETHICS AND BROADER IMPACT

Search-agent training can improve access to external knowledge, but it can also amplify corpus errors, expose memorized or retrieved sensitive content, and create answers that appear supported when retrieval is misleading. This work uses public QA benchmarks and a fixed Wikipedia-derived corpus. The reward checks document coverage, not truth or source quality; deployment would require provenance display, privacy filtering, and safeguards against prompt injection and malicious retrieval content. We do not evaluate demographic fairness or high-stakes use and make no claim of readiness for medical, legal, or financial decision making.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 present the method and experimental protocol. Appendices A and B provide implementation and reproduction details, respectively.

AI USE STATEMENT

An OpenAI language model accessed through Codex assisted with code comprehension, literature search, quantitative analysis, interpretation of results, research feedback, figure preparation, and manuscript drafting and editing. DeepSeek-V4-Flash was used for offline evidence annotation and document mapping (Appendix A.3). Reported experimental results were computed from the logs by deterministic scripts. The authors reviewed the manuscript, citations, implementation descriptions, and empirical claims and remain responsible for the submitted work.

REFERENCES

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. ReSearch: Learning to reason with search for LLMs via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2858.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. In *International Conference on Learning Representations*, pp. 16981–17017, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1544.

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. In *Advances in Neural Information Processing Systems*, volume 38, pp. 126866–126898, 2025. doi: 10.52202/085713-3815.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *International Conference on Learning Representations*, pp. 27572–27593, 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. doi: 10.18653/v1/2020.coling-main.580.

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. doi: 10.18653/v1/2025.acl-long.604.

Mianqiu Huang, Taofeng Xue, Chong Peng, Jinrui Ding, Jie Yang, Sicheng Fan, Jiale Hong, Yufei Gao, Xiaocheng Zhang, Linsen Guo, Xin Yang, Dengchang Zhao, Xiandi Ma, Yuchen Xie, Peng Pei, Xunliang Cai, and Xipeng Qiu. EvoCUA-1.5: Online reinforcement learning for multi-turn computer-use agents. *arXiv preprint arXiv:2607.09773v2*, 2026.

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*, pp. 87362–87388, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Ö. Arık, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Conference on Language Modeling*, 2025.

Mandar Joshi, Eunsol Choi, Daniel 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. doi: 10.18653/v1/P17-1147.

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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29557–29590, 2025.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019. doi: 10.1162/tacl_a_00276.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Mingchen Li, Hansi Zeng, Zhuo Qian, Jiatan Huang, Sunjae Kwon, Hamed Zamani, and Hong Yu. RICE-PO: Turning retrieval interactions into credit signals for reasoning agents. *arXiv preprint arXiv:2605.26352*, 2026.

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*, 2024.

Dongyi Liu, Yifan Niu, Qinwen Wang, Han Xiao, and Jia Li. PiCA: Pivot-based credit assignment for search agentic reinforcement learning. *arXiv preprint arXiv:2605.09287*, 2026.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent Lightning: Train ANY AI agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025.

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. doi: 10.18653/v1/2023.acl-long.546.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

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. doi: 10.18653/v1/2023.findings-emnlp.378.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025. doi: 10.1145/3689031.3696075.

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.

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. doi: 10.1162/tacl_a_00475.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024. doi: 10.18653/v1/2024.acl-long.510.

Tao Wang, Suhang Zheng, and Xiaoxiao Xu. RTMC: Step-level credit assignment via rollout trees. *arXiv preprint arXiv:2604.11037*, 2026.

Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, Zhiyong Lu, and Aidong Zhang. Supervising the search process produces reliable and generalizable information-seeking agents. *arXiv preprint arXiv:2502.13957v3*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Yongxin Wang, Yiwei Wang, Xiaodan Liang, and Jing Tang. TreeRPO: Tree relative policy optimization. *arXiv preprint arXiv:2506.05183v2*, 2025b.

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. doi: 10.18653/v1/D18-1259.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Tianhua Zhang, Kun Li, Junan Li, Yunxiang Li, Hongyin Luo, Xixin Wu, James Glass, and Helen Meng. TreePS-RAG: Tree-based process supervision for reinforcement learning in agentic RAG. *arXiv preprint arXiv:2601.06922*, 2026a.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. Process vs. outcome reward: Which is better for agentic RAG reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. doi: 10.52202/085713-1965.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

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, 2026b. doi: 10.18653/v1/2026.findings-acl.596.

Yubao Zhao, Weiquan Huang, Sudong Wang, Ruochen Zhao, Chen Chen, Yao Shu, and Chengwei Qin. BranPO: Scalable contrastive branch sampling for long-horizon agentic reinforcement learning. *arXiv preprint arXiv:2602.03719*, 2026.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. doi: 10.18653/v1/2025.emnlp-main.1106.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160, 2024.

Zefang Zong, Dingwei Chen, Yang Li, Qi Yi, Bo Zhou, Chengming Li, Bo Qian, Peng Chen, and Jie Jiang. AT^2PO: Agentic turn-based policy optimization via tree search. *arXiv preprint arXiv:2601.04767*, 2026.

A Method and Implementation Details

This appendix specifies the interaction, tree construction, state selection, and evidence rewards used by SLPO.

A.1 Agent Prompt, Interaction, and Transition Serialization

Agent Prompt and Action Grammar. Tree construction and validation use the following system prompt (line breaks preserved):

> You are a helpful assistant that solves questions.
> Use the search tool when factual information is needed, and you may use `<search>` multiple times if needed.
> For each question, proceed through assistant turns using exactly one of these formats:
> `<think>concise reasoning</think>`
> `<search>one concise search query</search>`
> or:
> `<think>concise reasoning</think>`
> `<answer>brief answer only, no explanation</answer>`
> Answer only when ready.

The prompt requests one action. The parser permits one leading `<think>` segment; among consecutive well-formed action tags, it selects the first nonempty answer, or otherwise the first nonempty search. Malformed tags, extra untagged text, or no nonempty action are invalid. Questions are prefixed with `Question:`. Each retrieval observation contains the top three passages in one `<result>` block, with ranked titles and text but no retrieval scores.

The interaction budget is five searches. At the limit, a final user instruction forbids further searches and requests an answer in `<answer>...</answer>` format using the available evidence. A further search is not sent to the retriever and receives the search-limit penalty. Tree construction uses stochastic decoding (Table 10), whereas validation is greedy. Both use a 2,048-token generation cap per action; action tags do not terminate generation.

State Re-Serialization. The state stores the system instruction, question, parsed prior actions, rendered observations, search count, and covered evidence. Sampling and training use the same state renderer, chat template, and context-budget rules. The resulting context is paired with the current action's original generated token IDs, without re-tokenizing that response. Table 4 specifies message roles and loss masks.

Table 4: State-to-transition serialization. "Context" tokens participate in attention but have zero response loss.

Semantic record	Serialized role	Content and policy loss
System instruction	system	Agent prompt above; context only.
Question	user	`Question:` followed by the question; context only.
Historical action	assistant	Reconstructed reasoning and action, subject to the chat template's history rules; context only.
Historical retrieval	user	One `<result>` block containing ranked passages after the associated search; context only.
Current stored action	separate response	Original generated tokens follow the generation prefix; only these tokens contribute to the policy loss.

Historical reasoning is template-dependent. Qwen3-8B removes reasoning before the latest ordinary user message, while retaining it within native multi-step tool calls. Our user-role `<result>` observations therefore trigger removal of earlier reasoning. Qwen3-4B-Instruct-2507 instead preserves assistant content. Neither behavior changes the stored current-action tokens. Separate transitions simplify cross-turn alignment but can repeat shared-context computation.

The 1,024-token observation budget includes the `<result>` delimiters; only the enclosed body is head-truncated. Within the 4,096-token rolling-prompt budget, the system instruction and question are preserved, and the most recent complete assistant–observation turn groups are retained while they fit. If no complete group fits, the latest assistant action is retained when possible; otherwise only the fixed prefix and any forced-answer message remain. The fixed prefix is never token-sliced, and transition construction applies no further prompt truncation.

A.2 Semantic-Tree Construction and State Selection

Algorithm 1 constructs state-local GRPO groups. Invalid actions remain stored for local training and receive the format penalty, but create no tree children. Answers create terminal leaves; search clusters expand through their medoids. At the search limit, a further search terminates the branch without retrieval and receives the search-limit penalty.

Algorithm 1 Semantic-tree exploration and batch-level local-group selection

Require: batch of B questions, policy π_θ, $K = 8$, target $M = Bm$
initialize one root per question and put all roots in frontier F
while F contains an expandable state **do**
  for all $s \in F$ **do**
    sample and store $\{(a_k, \text{tokens}_k)\}_{k=1}^{K}$
    cluster search actions; make answers singleton clusters
    materialize answer leaves and one search-medoid child per cluster
  end for
  set F to the newly materialized nonterminal children
end while
pool all nonterminal states with stored actions into $\mathcal{S}$
for all $s \in \mathcal{S}$ **do**
  independently execute and score each stored action from s
  store group $\mathcal{G}(s) = \{(x_s, a_k, r_{s,k})\}_{k=1}^{K}$ and compute $D(s)$
end for
form the informative pool $\mathcal{S}^+ = \{s : D(s) > 0\}$
from the nonempty pool, sample up to M distinct states with weights $D(s)$
if fewer than M states are available, reuse their groups cyclically
emit M groups, preserving separate within-state normalization for each slot

Clustering and Medoids. Search queries use Qwen3-Embedding-4B (Zhang et al., 2025b); the first 768 dimensions are ℓ_2-normalized. For query embeddings z_i, z_j, the distance is $d_{ij} = 1 - z_i^\top z_j$. Agglomerative complete linkage merges the closest pair of clusters only when their maximum cross-cluster distance is at most 0.4, corresponding to cosine similarity at least 0.6. For cluster C, the medoid is

$$k^*(C) = \arg\min_{k \in C} \frac{1}{|C|} \sum_{j \in C} (1 - z_k^\top z_j). \tag{5}$$

Clustering applies only to search actions at the same state. The medoid's retrieved observation enters the child history; all other raw actions receive independent local rewards. A failed medoid retrieval terminates that branch.

Candidate Evaluation and Group Reuse. All candidate groups are scored before selection using the rewards in Appendix A.4. A candidate is informative when its action rewards have nonzero standard deviation. Selection pools candidates from all $B = 64$ trees and targets $M = 128$ slots ($m = 2$), each containing $K = 8$ actions, without per-tree quotas. Sampling weights are proportional to $D(s)$. If the informative pool is smaller than the target, it is traversed cyclically to fill the remaining slots. Each reused slot retains the stored actions and rewards and is normalized as a separate group. Reuse increases their optimization weight without additional exploration.

A.3 Offline Evidence Annotation and Document Mapping

The reward map is constructed offline from each HotpotQA training question, its accepted gold answers, and the same Wiki-18 retrieval corpus used by the agent. Annotation uses stochastic sampling from DeepSeek-V4-Flash with temperature 1.0, top-p 1.0, and a 2,048-token output cap. The resulting mapping is held fixed during RL, where reward computation is deterministic. Algorithm 2 summarizes the pipeline; Table 5 gives the prompt requirements.

Table 5: Offline annotation outputs and acceptance rules. Each response is a single JSON object.

Role	Required output	Acceptance rule
Annotator	`{"action":"search", "query":"..."}` or `{"action":"final", "status":"ok", "evidence":[ {"claim":"...", "text_id":"..."}]}`	Search rather than guess from memory. Each item is one minimal fact independently supported by exactly one retrieved identifier. Multi-document facts must be split. `evidence_id`, copied evidence text, and multiple identifiers in one item are rejected. Allowed non-ok final states are `corpus_missing` and `insufficient_evidence`.
Verifier	`{"status":"pass", "items":[{"index":0, "supported":true, "atomic":true}], "missing":[]}`	Every claim must be supported by its cited text alone and be atomic; the complete set must suffice for the gold answer. Missing facts produce `insufficient_evidence`; absent corpus support produces `corpus_missing`.
Support judge	`{"supported":true, "notes":""}`	One claim–document pair is judged independently; outside knowledge and other documents may not be used.

The annotator sees the question, gold answers, and retrieved document identifiers, titles, and text. After eight turns without a final answer, it is instructed to finalize using only retrieved identifiers. Candidate evidence must be nonempty and satisfy Table 5's claim–document requirements. The verifier sees each claim paired with its cited text. Failed attempts provide feedback for relabeling, with at most three attempts in total.

To broaden mapping coverage, each claim and its verifier-approved seed text separately retrieve five candidate documents. The seed is retained; additional documents are included only when independently judged to support the claim. Duplicate document identifiers are removed, and all supporting

documents map to the same evidence item. Only questions with at least one resulting evidence item enter RL training.

Algorithm 2 Offline evidence-to-document mapping

Require: question q, gold answers G, Wiki-18 corpus

for $attempt = 1, 2, 3$ **do**
  annotator alternates JSON search actions with top-5 retrieval (at most 8 turns)
  parse final atomic pairs $(\mathrm{claim}, \mathrm{text_id})$
  if schema fails or an identifier was not retrieved **then**
    relabel with feedback; **continue**
  end if
  verifier checks single-document support, atomicity, and joint sufficiency
  if verification fails **then**
    relabel with verifier feedback; **continue**
  end if
  for all verified pairs (c, d_0) **do**
    retrieve top-5 with c and with the full text of d_0
    independently judge whether each candidate document supports c
    map c to d_0 plus every additional supported identifier
  end for
  if the resulting map is nonempty **then**
    save it and **return**
  end if
end for
discard the question from RL training

A.4 Transition Rewards

For question q, let $\mathcal{M}_q(d)$ be the set of offline evidence identifiers supported by retrieved document identifier d, and let $C(s)$ be the set covered before the current action. For a successful search action a returning document identifiers $\mathcal{D}(a)$, define

$$E(a) = \bigcup_{d \in \mathcal{D}(a)} \mathcal{M}_q(d), \qquad N(a, s) = E(a) \setminus C(s). \tag{6}$$

The new state coverage is $C(s) \cup N(a, s)$. The search reward is

$$r_{\mathrm{search}}(s, a) = \begin{cases} -0.5, & \text{search emitted after the interaction limit,} \\ -0.1, & \text{retrieval fails,} \\ |N(a, s)|, & |N(a, s)| > 0, \\ -0.3, & |N(a, s)| = 0 \text{ and } |E(a)| > 0, \\ -0.5, & |E(a)| = 0. \end{cases} \tag{7}$$

Table 6: Offline evidence preprocessing outcome. The four failure categories partition all 13,314 filtered questions; percentages use all 90,447 source questions as denominator.

Quantity	Value
Source / retained / filtered questions	90,447 / 77,133 / 13,314
Retained-question rate	85.28%
Filtered (zero-final-evidence) rate	14.72%
Corpus-missing failures	3,255 (3.60%)
Insufficient-evidence failures	2,561 (2.83%)
Format-error failures	7,141 (7.90%)
Annotation-service failures	357 (0.39%)
Evidence items / mapped document IDs	145,429 / 428,288
Evidence items per retained question	1.885
Document IDs per evidence item	2.945

Search reward counts newly covered evidence items, not documents. One document can cover multiple items, so a search can earn more than $+1$. "Repeated" means that retrieved evidence is nonempty but already covered by $C(s)$.

For an answer action, correctness is COVER-EM and evidence coverage is $\rho(s) = |C(s) \cap E_q|/|E_q|$, where E_q is the question's annotated evidence set. The answer reward is

$$r_{\text{answer}}(s, a) = \begin{cases} -0.5, & \text{answer is incorrect,} \\ +1.2, & \text{answer is correct and } \rho(s) = 1, \\ +0.7, & \text{answer is correct and } 0 < \rho(s) < 1, \\ +0.3, & \text{answer is correct and } \rho(s) = 0. \end{cases} \tag{8}$$

Any parse or format error receives -0.5 and leaves coverage unchanged. In the base SLPO configuration, these immediate transition rewards determine both candidate-state dispersion and within-state advantages. The Subtree Reward ablation adds a downstream component before both operations (Appendix B.5).

Table 7: Evidence-grounded transition reward schedule.

Action outcome	Reward
Search: each newly covered evidence item	$+1.0$
Search: repeated evidence	-0.3
Search: no mapped evidence	-0.5
Search: retrieval failure	-0.1
Search after interaction limit	-0.5
Correct answer, full evidence coverage	$+1.2$
Correct answer, partial evidence coverage	$+0.7$
Correct answer, no evidence coverage	$+0.3$
Incorrect answer or format error	-0.5

B EXPERIMENTAL REPRODUCTION

The following settings supplement the experimental protocol in Section 4.

B.1 DATA, EVALUATION, AND MODEL SELECTION

Training starts from the 90,447-question HotpotQA training split; offline evidence processing retains 77,133 questions, with filtering outcomes reported in Table 6. The fixed 51,713-example validation mixture is used for both model selection and reporting; Table 8 gives its composition.

Table 8: Validation-mixture composition used for checkpoint selection and reporting. Family and overall scores pool examples rather than macro-averaging dataset scores.

Dataset	Family	Examples
Natural Questions	Single-hop	3,610
TriviaQA	Single-hop	11,313
PopQA	Single-hop	14,267
HotpotQA	Multi-hop	7,405
2WikiMultiHopQA	Multi-hop	12,576
MuSiQue	Multi-hop	2,417
Bamboogle	Multi-hop	125
Total	–	51,713

For COVER-EM, predictions and references are lowercased; punctuation and the English articles *a*, *an*, and *the* are removed; and whitespace is collapsed. An example is correct when any normalized reference is a substring of the normalized prediction. Overall and family scores are means over

all examples in the relevant mixture, equivalently weighting dataset means by the counts in Table 8. Mean search calls divide the number of emitted search actions by the number of validation examples.

Main and ablation comparisons use the checkpoint selected by the common 0–200-step rule in Section 4; ReAct uses step 0.

Fixed-Step Seed Check. Three Qwen3-4B-Instruct seeds are trained for 100 updates and validated once on the same 51,713 examples, without checkpoint selection. Their overall scores are 65.42%, 65.98%, and 64.97%, giving $65.46 \pm 0.51\%$. Single-hop and multi-hop scores are $67.63 \pm 0.16\%$ and $62.63 \pm 1.15\%$, respectively. Summaries report the mean and sample standard deviation across seeds and supplement the main comparisons.

B.2 Retriever, Corpus, and Index

Training, validation, and all baselines share the same fixed retriever and corpus. Corpus text is embedded directly. Search and semantic-clustering queries use the instruction "Given a web search query, retrieve relevant passages that answer the query" in the format `Instruct:` instruction, newline, `Query:` query. Documents and queries use the first 768 dimensions of Qwen3-Embedding-4B with ℓ_2 normalization. Table 9 lists the retrieval settings.

Table 9: Retrieval settings.

Component	Configuration
Corpus	Wiki-18 (`PeterJinGo/wiki-18-corpus`); document titles and text
Embedding	Qwen3-Embedding-4B; first 768 dimensions; ℓ_2 normalization
Index build	Inner product; unsharded cuVS CAGRA; IVF-PQ construction; graph degree 64; intermediate degree 128
CAGRA search	Intermediate top-k 128; width 2
Depth	Top 3 for agents/baselines; top 5 for offline annotation and mapping expansion

B.3 SLPO Generation and Optimization Settings

Table 10 reports the main generation and optimization settings. Rollouts use vLLM (Kwon et al., 2023); rewards are deterministic, with no learned reward model. KL regularization against the reference policy is applied only in the actor loss.

Table 10: SLPO generation, optimization, and validation settings for the main runs.

Component	Value
Policy backbones	Qwen3-4B-Instruct-2507 and Qwen3-8B
Tree/local-group shape	8 actions/node; complete-linkage cosine similarity threshold 0.6; batch-level sampling weighted by reward dispersion
Training batch	64 questions; 128 state slots ($m = 2$); 1,024 transition samples including reused groups
Action / interaction	2,048 tokens/action; 5 searches; observation / rolling-prompt budgets 1,024 / 4,096
Advantage	GRPO; within-state standard-deviation normalization; no critic or GAE
Actor	LR 10^{-6}; transition-level PPO mini-batch size 128; 1 PPO epoch; clip 0.2; entropy coefficient 0
KL regularization	Low-variance actor KL coefficient 0.001; no reward KL
Rollout	$T = 1$; top-$p = 0.85$; model context limit 8,192 tokens
Validation	Greedy ($T = 0$, $n = 1$); before training and every 20 steps

B.4 Baseline Reproduction and Shared Environment

Shared Controls. Baselines share the data, evaluation, and retrieval settings in Appendices B.1–B.2, the same Qwen3 model scale, a five-search limit, a 4,096-token prompt budget, and a training batch of 64 questions. ReAct evaluates the corresponding step-0 policy with the same interaction interface. The baseline PPO mini-batch setting is 8 before rollout expansion; SLPO uses a transition-level PPO mini-batch size of 128 for its 1,024 transition samples. Both yield eight mini-batches per PPO epoch under their respective expansion rules, despite counting different units.

Algorithm-specific estimators, rollout construction, reward shaping, and critic use are retained. Table 11 distinguishes their optimization and generation settings, including PiCA's unequal actor/critic learning rates; Table 12 lists the remaining method-specific settings.

Table 11: Optimization settings and rollout units. Rollouts are per question, except SLPO's eight actions per selected state. Response caps are per action for GiGPO and SLPO, and per trajectory otherwise. PPO mini-batch units are distinguished in the text.

Method	Estimator / loss unit	Actor LR	Critic LR	Rollouts	PPO MB setting	Response cap
Search-R1	GRPO / trajectory	10^{-6}	–	8	8	4,096
GiGPO	GiGPO / episode + micro	10^{-6}	–	8	8	512/action
PiCA	GAE / trajectory	7×10^{-7}	7×10^{-6}	32	8	4,096
AT^{2}PO	GRPO / turn	10^{-6}	–	11	8	4,096
SLPO	GRPO / state–action	10^{-6}	–	8/state	128	2,048/action

Table 12: Algorithm-specific baseline settings.

Method	Retained design and settings
Search-R1 (Jin et al., 2025)	Outcome-only rule reward and trajectory-level GRPO.
GiGPO (Feng et al., 2025)	Episode plus step micro-advantages; step-advantage weight 1.0; mean/std normalization; similarity grouping enabled at threshold 0.9; invalid-action penalty enabled with coefficient 0.01.
PiCA (Liu et al., 2026)	Actor–critic GAE; critic warmup ratio 0.015; step-reward scale 0.3; baseline step reward 0.55; outcome-reward scale 1.5; step reward enabled.
AT^{2}PO (Zong et al., 2026)	Turn policy loss `gspo_turn`; 5 initial rollouts; entropy expansion for 2 iterations; beam size 3; 11 samples/tree; branch probability 0.5; entropy weight 0.2; leaf-value normalization; `child_softmax` node values and `node_value` advantages.

Token Budgets. Search-R1, PiCA, and AT^{2}PO primarily limit trajectory length through a 4,096-token response budget. In SLPO, each action has a 2,048-token generation cap, while state re-serialization places previous interactions in a rolling 4,096-token prompt budget, excluding the current action (Appendix A.1). Thus, retained interaction length is controlled mainly through context retention rather than a cumulative response cap. The 4,096-token limits provide approximate budget-scale alignment, not identical sequence lengths or total generated-token budgets. GiGPO also uses per-action generation, with a 512-token cap. Tree expansion and independent candidate scoring further make SLPO's total rollout cost data-dependent.

B.5 Ablation Controls

Trajectory-Level Controls. *Trajectory GRPO* is the reproduced Search-R1 configuration: eight flat full-trajectory rollouts per question, the original outcome reward, and trajectory-level GRPO. *+Evidence Credit* computes the trajectory advantage A from the same outcome-only return and adjusts its magnitude on search-action tokens. With $\alpha = 0.3$, on a correct trajectory, searches that retrieve new mapped evidence receive $(1+\alpha)A$ and other searches receive $(1-\alpha)A$; on an incorrect

trajectory the two factors are reversed. Answer tokens retain A, and the reweighting preserves its sign.

+Semantic Tree uses the same outcome reward and evidence-guided reweighting, with complete trajectories constructed through semantic-tree exploration. Optimization remains trajectory-level. Full SLPO uses batch-level selection and local optimization from Sections 3.4–3.5, with the rewards in Appendix A.4. All rows follow the common 0–200-step model-selection rule in Appendix B.1.

Subtree Reward. For an action in semantic cluster j, let s_j be the cluster's materialized child state and $v(s_j)$ its downstream outcome value, obtained by aggregating final-answer outcomes in its subtree using semantic-cluster probabilities. The variant uses $\widetilde{r}_{s,k} = r_{s,k} + v(s_j)$ with unit weight on the subtree component. Actions in the same cluster share $v(s_j)$ but retain their independently computed local rewards $r_{s,k}$. These rewards are computed for the full candidate pool before computing $D(s) = \mathrm{Std}(\widetilde{r}_{s,1}, \ldots, \widetilde{r}_{s,K})$. The resulting dispersions define the informative pool and sampling weights; selected groups also use $\widetilde{r}$ for within-state advantages. The variant retains the 128-slot budget, eight actions per group, and transition-level PPO mini-batch size of 128.

C Training Cost and Local Learning Signals

C.1 Training Cost and Optimization Progress

Table 13 reports mean times over all 200 updates, excluding validation.

Table 13: Training costs on Qwen3-4B-Instruct over updates 1–200, excluding validation. Values are seconds per update unless marked otherwise. Indented rows are components of rollout/data construction; lower total costs are **bold and underlined**.

Quantity	+Evidence Credit	SLPO
Rollout / training-data construction	21.11	61.41
Tree construction	–	49.39
Candidate-action scoring	–	10.62
Policy / reference log-probability computation	15.32	24.89
Actor update	29.31	56.32
Remaining training operations	4.35	4.49
Total per update	**70.09**	147.11
Total over 200 updates (hours)	**3.89**	8.17

Both runs use a constant actor learning rate of 10^{-6} and eight mini-batches per PPO epoch (Appendix B.4). Timing compares complete pipelines with different token and rollout budgets, not isolated tree overhead. Cumulative times sum training steps through the reported checkpoints, excluding validation; they are not interpolated times to a shared accuracy threshold.

C.2 Candidate Pools and Group Reuse

Every selected training group has nonzero reward dispersion throughout both runs, as required by the selection rule. Table 14 distinguishes candidate pools before selection from training slots after selection. Evidence rates are computed over selected transitions, weighting reused groups by their slot counts.

Across 200 updates, 79.44% of 4B candidates and 67.23% of 8B candidates have zero dispersion, pooling candidate counts across updates. The informative pools average 39.18 and 74.63 candidates, respectively; reused slots account for 69.39% and 41.93% of the 128-slot budgets. The 4B policy reuses groups in every update, and the 8B policy in 197 of 200 updates (Figure 5).

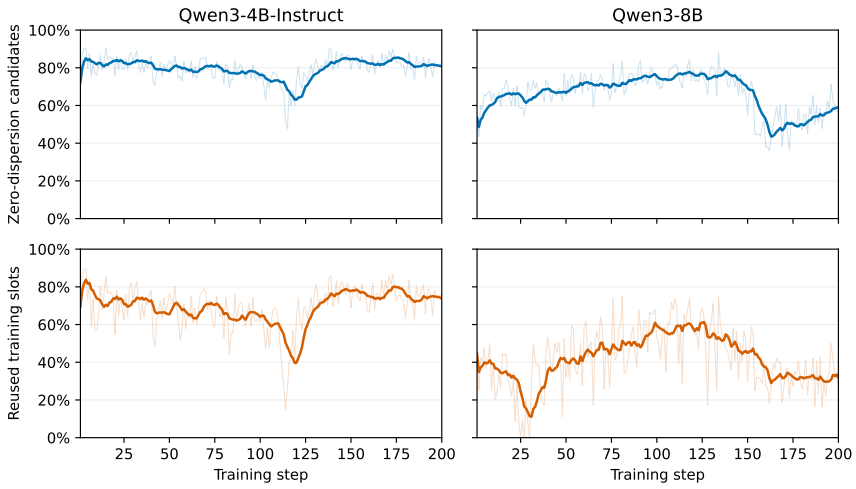


Figure 5: Candidate-pool sparsity and group reuse over training steps 1–200. Faint traces are per-update values; solid curves are 10-step trailing means. Top: zero-dispersion candidates divided by all candidate states in each update. Bottom: reused slots divided by 128 training slots.

Table 14: Training-batch diagnostics at the updates corresponding to the selected SLPO checkpoints. Rates are percentages. Each batch contains 64 trees and 128 eight-action slots; reused groups count separately in slot and transition statistics.

Metric	Definition and denominator	4B	8B
Mean tree nodes	Total nodes, averaged over 64 question trees	13.859	12.469
Mean maximum depth	Maximum root-to-node depth, averaged over 64 trees	3.109	3.109
Candidate states	Nonterminal states with stored actions in the batch	211	213
Informative candidates	Candidate groups with nonzero reward dispersion	60	92
Zero-dispersion candidates	Non-informative candidates divided by all candidate states	71.56	56.81
Contributing trees	Distinct question trees represented in the 128 training slots	36	50
Reused slots	Slots beyond the distinct selected states, divided by 128	53.12	28.12
Mean local reward std.	Population standard deviation across 8 rewards, averaged over 128 slots	0.630	0.606
Zero-variance training groups	Groups with reward std. below 10^{-12} divided by 128 slots	0.00	0.00
Search actions finding new evidence	Search transitions newly covering evidence divided by search transitions in selected groups	55.21	72.74
Coverage when answering	Covered reference-evidence fraction, averaged over answer transitions in selected groups	59.56	79.67

Reducing the nominal state allocation m from 2 to 1 would halve transition volume from 1,024 to 512 per batch, without changing tree construction or candidate scoring. At fixed mini-batch size and epoch count, this would also reduce optimizer updates per iteration. It could lower actor-update cost, but its effects on learning and state coverage remain untested.