WHEN IS REUSE WORTH REPLACING EXPLORATION? QUALITY-CONSTRAINED ROLLOUT ALLOCATION FOR STATEFUL AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic reinforcement learning improves language models through multi-step environment interaction, but costly rollouts can yield little reward contrast, while restarting trajectories repeatedly discards reusable progress. Efficient training therefore requires deciding how much experience to acquire and when to generate, branch, or replay it. We propose QCAR, a quality-constrained rollout allocation framework for stateful agentic reinforcement learning. QCAR organizes observed task progress and explored transitions into a state graph, using initial Probe rollouts to estimate continuation success and identify valuable Fork boundaries. These estimates guide allocation across Fresh Generation, Fork, and Replay, with quality feedback limiting unproductive reuse. Balanced policy updates account for unequal sampling and shared prefixes, while observed outcomes recalibrate the predictor as the policy evolves. Across two models and four benchmarks, QCAR improves final accuracy over GRPO in all eight settings, averaging 2.32 percentage points (4.92% relative). QCAR yields up to 53.5% more reward-contrasting groups, with gains in all six model–environment comparisons, while fresh-token consumption decreases in five, by up to 8.35%. These results demonstrate the value of graph-informed, quality-constrained experience acquisition for stateful agentic reinforcement learning.

1 INTRODUCTION

Agentic reinforcement learning (RL) improves language models by training them to solve tasks through interaction with external environments (Wang et al., 2025; Jin et al., 2025). Rather than producing a single response, an agent repeatedly reasons, invokes tools, and incorporates observations before receiving feedback on task completion (Yao et al., 2023c; Schick et al., 2023). Learning from these interactions enables policies to improve their decisions in embodied environments, online shopping, and information-seeking tasks. Outcome-based training is particularly attractive because task success supplies supervision without requiring annotations for every intermediate action (Guo et al., 2025). Its effectiveness, however, depends on acquiring trajectories that provide useful learning signals within a limited rollout budget.

However, expensive interaction does not necessarily produce informative training data. In group-relative policy optimization (GRPO) (Shao et al., 2024), a group of entirely unsuccessful rollouts has identical rewards and therefore zero reward-based advantages; repeatedly sampling such groups consumes resources without providing the intended policy-learning signal (Yu et al., 2025; Liu et al., 2025). Generating every trajectory from the beginning also repeats prefixes and environment interactions that may already have established useful progress. Prefix branching can avoid this repetition, while experience replay can reuse completed trajectories (Li et al., 2026; Park & Chang, 2026). These options introduce a coordination problem: fresh exploration, continuation from an existing state, and reuse of past experience differ in both cost and training value. Prompt-level difficulty and textual prefix histories inform allocation (Li et al., 2025; Gao et al., 2025), but do not explicitly organize the task progress and explored state transitions revealed by tool interaction. Without this information, an allocator may repeatedly explore unproductive states or overlook valuable continuation points.

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.

Addressing this problem requires connecting rollout scheduling to the state of the interaction. Stateful agents naturally expose a graph of observed entities, relations, and action-induced transitions (Battaglia et al., 2018; Adhikari et al., 2021). This graph records satisfied requirements, unresolved subgoals, and exploration coverage, providing evidence about both whether a continuation can succeed and where alternative actions remain worth exploring. For example, an unsuccessful trajectory may already have acquired the required object, leaving a single unmet condition that can be resolved by branching from an earlier state. A trajectory cycling through unchanged states presents a different opportunity, even if its prefix has a similar length. Our motivation experiments show that graph information improves outcome-prediction separability and identifies continuation candidates with stronger reward contrast. This suggests using graph-conditioned estimates to guide both the number of rollouts assigned to a task and the choice of how to acquire them.

To address these challenges, we propose **QCAR**, a quality-constrained rollout allocation framework for adaptive training-data acquisition in stateful agent RL. QCAR coordinates *Fresh Generation*, *Fork*, and *Replay*: generating an independent trajectory, sampling a new suffix from a restorable state, and reusing completed experience, respectively. An initial Probing stage exposes task-specific outcomes and candidate states before allocation. A joint value model and MLP fork head use graph information and recent feedback to estimate continuation success and Fork benefit, supplying a common source of decision information for subsequent acquisition.

QCAR turns these estimates into an adaptive acquisition-and-learning loop. Fresh Generation and Fork compete for rollout slots using predicted value, graph coverage, and generation cost. A quality guard reduces the Fork quota when recent outcomes indicate degraded sample quality; eligible same-task Replay fills part of the released capacity subject to a staleness limit. Balanced-TreeRPO then accounts for unequal group sizes and shared prefixes during policy optimization. Finally, newly observed outcomes recalibrate the predictor, allowing acquisition decisions to adapt as the policy changes. This design connects informative state representations to budget allocation and connects the resulting experience back to future decisions.

We evaluate QCAR with Qwen3.5-4B and Qwen3-8B on ALFWorld, WebShop, NQ, and HotpotQA. Compared with GRPO, QCAR improves final accuracy by 2.32 percentage points (4.92% relative) on average, yields up to 53.5% more reward-contrasting groups, and reduces fresh-token consumption by up to 8.35%. Together, these results support coordinated state-aware acquisition: graph-conditioned decisions provide more useful training experience, while balanced optimization translates that experience into stronger policies with less fresh generation.

In summary, this paper makes the following contributions:

- We use state graphs to expose task progress and exploration coverage for predicting continuation outcomes and identifying states worth further exploration.
- We introduce quality-constrained allocation across Fresh Generation, Fork, and Replay, combining graph-conditioned value estimates with generation cost and quality feedback.
- We integrate balanced policy optimization and online predictor recalibration into an adaptive acquisition loop that accounts for shared prefixes and an evolving policy.
- Across two models and four benchmarks, QCAR improves accuracy over GRPO by 2.32 points (4.92% relative) on average, increases effective groups by up to 53.5%, and reduces fresh tokens by up to 8.35%.

2 Preliminaries and Motivation

2.1 Stateful rollouts and group-relative learning

Agentic interaction as a state graph. Given task x, an agent chooses action $a_t \sim \pi_\theta(\cdot \mid h_t)$ from its visible history $h_t = (x, o_0, a_0, \ldots, o_t)$. A tool call changes the environment and returns observation o_{t+1}; a completed trajectory τ receives terminal reward $R_x(\tau)$. Unlike a single-turn completion, a tool turn exposes an intermediate decision state. We represent the observable part of this state by a typed graph $\mathcal{G}_{x,t} = (V_{x,t}, E_{x,t})$: nodes describe observed entities, locations, constraints, or evidence, and edges record actions and relations revealed by tool feedback. Only information available by turn t enters the graph. A separate lineage tree retains exact histories

and verified restoration boundaries; graph similarity alone does not make two environment states interchangeable.

Grouped rollout optimization. GRPO samples K complete trajectories $\mathcal{T}_x = \{\tau_{x,j}\}_{j=1}^{K}$ for task x and centers each return against its group, for example

$$A_{x,j}^{\text{grp}} = \frac{R_x(\tau_{x,j}) - \overline{R}_x}{\text{std}(R_x(\tau_{x,1:K})) + \epsilon}. \tag{1}$$

A group is *reward-contrasting* if $C(\mathcal{T}_x) = \mathbb{1}[\max_j R_x(\tau_{x,j}) > \min_j R_x(\tau_{x,j})]$. For a homogeneous group, its reward-driven group-relative advantages vanish. Fixed-K sampling nevertheless spends the same number of rollouts on every task. In a stateful environment, it also regenerates prefixes whose intermediate states have already been observed.

2.2 State graphs expose continuation opportunities

Prior allocation uses task estimates, prompt difficulty, or prefix histories. State graphs additionally expose task progress and explored transitions, helping distinguish unproductive continuations from promising states.

Graph state and continuation quality. Figure 1 illustrates two continuations from a shared state s_8: cleaning the acquired utensil opens paths to successful placement, whereas premature placement leads to cycles that leave it uncleaned. The graph exposes both the unresolved requirement and a branching state from which further exploration can be redirected.
Micro-experiments on ALFWorld and WebShop show that adding graph information to current-state and history features improves success-prediction separability (Figure 2a).

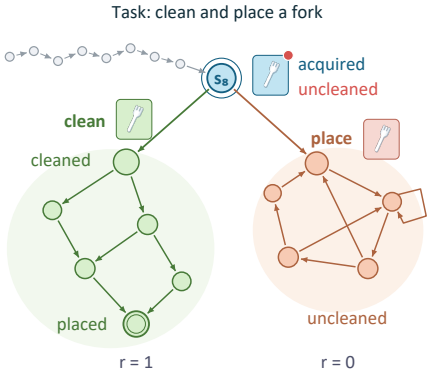


Figure 1: Graph-conditioned continuations: successful paths (green) and unresolved cycles (orange).

Error-loop cues are also associated with lower subsequent success (Figure 2b). These trends suggest that graph state provides useful evidence for deciding whether further continuation is worthwhile.

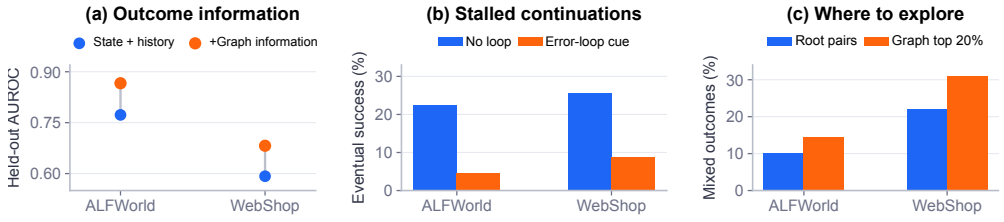


Figure 2: Graph information supports (a) outcome prediction, (b) low-progress detection, and (c) contrasting continuation selection.

Graph coverage and state-directed exploration. Across a rollout group, graph coverage reveals repeated exploration and unresolved relations, helping identify states where another continuation may produce a different outcome. In our continuation-selection benchmark, graph-ranked shared-prefix pairs exhibit a higher reward-contrast rate than ordinary independent Root pairs on both ALFWorld and WebShop (Figure 2c). This trend motivates selecting promising continuation states while reusing their prefixes, connecting graph-based state assessment to more effective exploration.

2.3 ROLLOUT REUSE NEEDS STATE-AWARE ALLOCATION

Prefix branching and experience replay reduce generation costs (Li et al., 2026; Hu et al., 2026; Park & Chang, 2026), but their value depends on the task and state. After an initial *Probe* phase reveals outcomes and graph states, allocation chooses among *Fresh Generation* (an independent rollout), *Fork* (a suffix from a restorable state), and *Replay* (a completed trajectory).

How many additional rollouts does a task need? The group-relative advantage in Equation (1) vanishes when all returns are equal. With binary rewards, k independent fresh rollouts produce contrast with probability $1 - p_x^k - (1 - p_x)^k$, where p_x is their success probability. Figure 3(a) confirms that increasing k raises the contrasting-group rate. After k_0 Probes, however, the value of m more independent rollouts depends on the observed number s_x of successes:

$$\Pr(C = 1 \mid s_x, m) = \begin{cases} 1 - (1 - p_x)^m, & s_x = 0, \\ 1, & 0 < s_x < k_0, \\ 1 - p_x^m, & s_x = k_0. \end{cases} \tag{2}$$

A mixed group already has contrast. For an all-failure group, the next rollout's marginal conversion probability is $p_x(1 - p_x)^m$; for an all-success group, it is $(1 - p_x)p_x^m$. Both decrease with additional attempts, but at different rates across tasks. Predicting p_x from the observed graph state and recent outcomes therefore helps determine *how many* rollouts each task should receive, rather than increasing every group uniformly.

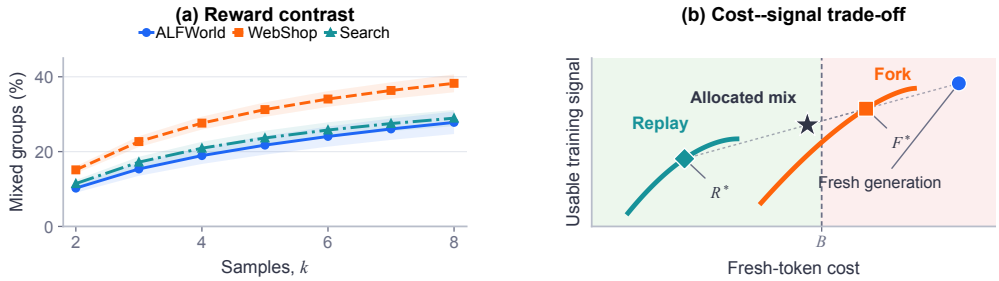


Figure 3: Rollout allocation: (a) measured group-size effect on reward contrast; (b) conceptual cost–signal trade-offs of reuse.

Which experience should fill the remaining budget? Figure 3(b) compares the three post-Probe choices. Each Fresh Generation costs roughly $\mathbb{E}[\ell_{\text{full}}]$ new tokens; a Fork at boundary b costs only $\mathbb{E}[\ell_{\text{suffix}}(b)]$; Replay generates none. Yet fewer tokens need not mean more useful training data. If a homogeneous Probe group receives m conditionally independent Fork suffixes with success probability $p_v(b)$, then

$$\Pr\big(C(\mathcal{T}_x \cup \mathcal{F}_b^m) = 1 \mid \mathcal{T}_x\big) = \begin{cases} 1 - (1 - p_v(b))^m, & \text{if all Probe rewards are } 0, \\ 1 - p_v(b)^m, & \text{if all Probe rewards are } 1. \end{cases} \tag{3}$$

Thus a late Fork can be cheap yet unlikely to change the group's reward; graph state is needed to estimate $p_v(b)$ and locate more promising boundaries. Replay may add a stored opposite-outcome trajectory to a homogeneous group without generating new tokens, but its training signal can become stale. If P_v and P_u are current- and collection-policy trajectory distributions, $\|g_v(\tau)\| \le G$, and fraction α of an update uses uncorrected Replay, then

$$\|\bar{g}_{\text{mix}} - \bar{g}_v\| \le 2\alpha G \,\mathrm{TV}(P_u, P_v), \tag{4}$$

where $\bar{g}_v = \mathbb{E}_{P_v} g_v(\tau)$ and $\bar{g}_{\text{mix}} = (1 - \alpha)\bar{g}_v + \alpha \mathbb{E}_{P_u} g_v(\tau)$. The allocation problem is therefore to preserve reward contrast and outcome quality while reducing fresh-token use. Graph progress and coverage help predict p_x and $p_v(b)$; recent policy feedback limits Replay. Together, they inform group size, Fork location, and the Fresh Generation/Fork/Replay mixture.

3 METHOD

QCAR uses graph state and recent feedback to allocate training experience (Figure 4). Probe rollouts expose outcomes and candidate states for graph-conditioned valuation (Steps 1–2). Quality-constrained allocation then coordinates Fresh Generation, Fork, and eligible Replay (Steps 3–4). Balanced policy updates consume the resulting groups, while observed outcomes recalibrate subsequent allocation (Step 5).

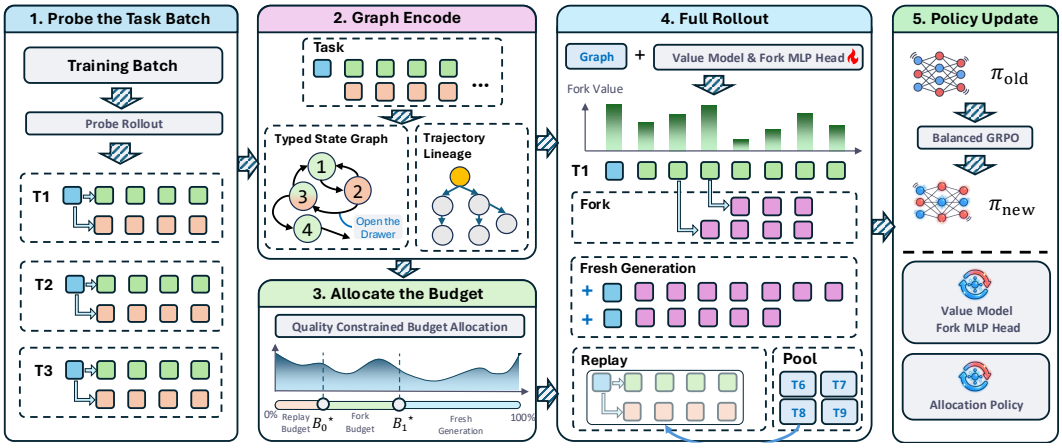


Figure 4: QCAR: (1–2) Probing and graph construction; (3–4) quality-constrained allocation; (5) balanced updates and feedback.

3.1 PROBING AND GRAPH-CONDITIONED CONTINUATION PREDICTION

Probing and state construction. At iteration v, QCAR generates k_0 independent Probe rollouts per task from π_v, recording terminal rewards and intermediate tool interactions. Probes count toward the budget and enter training; their rewards identify homogeneous or contrasting groups. Domain adapters encode the observed interactions as typed state relations, with coverage, uncertainty, and recent feedback summarizing exploration progress. A separate trajectory lineage stores exact prefixes, parent identities, and restoration contexts. QCAR retains only Fork boundaries with valid prefixes and expected state hashes, forming a candidate frontier $\mathcal{P}_x$. To preserve diverse continuation opportunities, it selects distinct early, middle, and late states, favoring relation changes, under-covered continuations, and successful–unsuccessful probe divergence. The graph thus supports candidate valuation, while lineage enables verified restoration.

Value estimation. For candidate $c = (x, a, p)$, $a \in \{\text{fresh}, \text{fork}\}$ identifies the acquisition type and p its prefix, empty for Fresh Generation. The representation $\phi_v(c)$ encodes historical state transitions, Probe rewards, task information, prefix position, relation coverage, and uncertainty. The value model and MLP fork head jointly form F_{ψ_v}, which estimates continuation success and Fork benefit:

$$
\begin{aligned}
(\widehat{p}_v(c), \widehat{f}_v(c)) &= F_{\psi_v}(\phi_v(c)), \\
\widehat{p}_v(c) \approx \Pr(Y_c = 1 \mid \phi_v(c)), &\qquad \widehat{f}_v(c) \approx \Pr(Y_c \neq Y_p \mid \phi_v(c)) \quad \text{for Fork.}
\end{aligned}
\tag{5}
$$

Here Y_c and Y_p denote continuation success and the factual parent outcome; $\widehat{f}_v$ measures the chance of introducing reward contrast with that parent. The MLP fork head uses two hidden layers with SiLU activations and sigmoid outputs. The predictor is initialized from several training rollout samples. Its outputs are blended with probe statistics and a decaying training-derived prior to estimate candidate sampling value, which is combined with graph coverage, uncertainty, and generation cost for allocation (Section 3.2). Replay eligibility depends on task identity and recency.

3.2 QUALITY-CONSTRAINED ALLOCATION AND ROLLOUT EXECUTION

QCAR first allocates new-sampling slots between Fresh Generation and Fork using the estimates in Section 3.1, then applies quality control and fills part of the released Fork budget with eligible Replay. This procedure implements Steps 3–4 of Figure 4.

Budget allocation. Let u_x, f_x, r_x count additional Fresh Generation, Fork, and Replay trajectories. With k_0 probes per task and total trajectory budget B,

$$n_x = k_0 + u_x + f_x + r_x, \qquad \sum_x n_x = B, \qquad n_{\min} \le n_x \le n_{\max}. \tag{6}$$

Allocation proceeds in three stages. First, after reserving minimum independent fresh sampling, Fresh Generation and recoverable Forks compete by predicted value and graph coverage. Task and prefix caps prevent concentration; the selected candidates determine each task's group size. Second, if recent matched-task success averages satisfy $\bar{q}_{\text{fork}} + \epsilon_q < \bar{q}_{\text{fresh}}$, quality control removes lower-ranked Fork assignments. Third, same-task Replay fills part of the released budget, admitting trajectories collected at policy version v_τ only if $v - v_\tau \le S_{\max}$. Replay replaces low-success Fork opportunities without additional generation while limiting policy lag. Unfilled slots return to Fresh Generation, preserving B.

Fork boundary selection. Fork placement combines model-estimated sampling value, graph coverage, and generation cost. Let $\widehat{V}_v(c)$ denote the expected benefit of an additional continuation, computed from the joint predictor's success and Fork-benefit estimates before sampling. Fresh Generation uses the increment of $1 - p^n - (1-p)^n$ under the blended success estimate; Fork uses $\widehat{f}_v(1-\widehat{f}_v)^\ell$ after ℓ branches have been assigned to that boundary. The allocation score combines this estimated value with graph coverage:

$$\begin{aligned} \Gamma_v(c) &= \sum_e w_{e,v} \log \frac{1 + z_e + \delta_e(c)}{1 + z_e}, \\ S_v(c) &= \max\{\rho \widehat{V}_v(c), \widehat{V}_v(c) - \lambda \sigma_v(c)\}[1 + \Gamma_v(c)] + b_v(c). \end{aligned} \tag{7}$$

Here z_e is accumulated discovery and training coverage, $\delta_e(c)$ the candidate contribution, and $w_{e,v}$ the priority of an uncertain, under-covered relation. The logarithm discounts repeated exploration. The score discounts estimated value by its uncertainty $\sigma_v(c)$, retains a floor $\rho\widehat{V}_v(c)$, and adds an exploration bonus $b_v(c)$.

After satisfying fresh-sampling minima, Fork boundaries are ranked by $S_v(c)/\widehat{C}(c)^\gamma$, with suffix-based cost estimates bounded away from zero. The best Fork displaces Fresh Generation only if its undiscounted score clears a selection margin. Task and prefix caps limit concentration, and coverage is updated after each selection. Execution verifies the restored state before generating a suffix; restoration failure falls back to Fresh Generation and is charged accordingly.

3.3 BALANCED POLICY OPTIMIZATION AND ONLINE RECALIBRATION

Step 5 updates both the agent and the information used to allocate its next batch. Adaptive group sizes and shared prefixes create unequal, correlated training samples. QCAR therefore uses Balanced-TreeRPO, the balanced GRPO update in Figure 4, to retain equal total weight for each original task.

Let G_x be the completed task group and $\mathcal{A}_x$ its represented acquisition types. For a trajectory j of type a_j, its within-task weight is

$$\bar{w}_{x,j} = \frac{\omega_{a_j}}{\sum_{a \in \mathcal{A}_x} \omega_a} \frac{1}{|G_x^{a_j}|}, \qquad A_{x,j} = A_{x,j}^{\text{grp}} + \eta A_{x,j}^{\text{sib}}, \qquad \sum_{j \in G_x} \bar{w}_{x,j} = 1. \tag{8}$$

Fresh trajectories, including probes, have reliability weight $\omega_{\text{fresh}} = 1$; Fork and Replay use smaller weights. The group advantage is defined in Equation (1). The sibling term normalizes rewards among a factual parent and its Fork descendants and is zero when these outcomes agree. It adds a local comparison of alternative continuations to the task-level learning signal.

For Fork trajectories, the mask $M_{x,j,t}$ excludes the historical prefix and trains only newly generated suffix tokens. Before optimization, replayed trajectories are rescored under the current policy π_v, with the resulting token log-probabilities held fixed throughout the update. For all acquisition types, the policy ratio is $r_{x,j,t}(\theta) = \pi_\theta(a_t \mid h_t)/\pi_v(a_t \mid h_t)$, and the clipped objective is

$$L(\theta) = -\frac{1}{N} \sum_x \sum_{j \in G_x} \bar{w}_{x,j} \frac{\sum_t M_{x,j,t} \min\{r_{x,j,t}(\theta) A_{x,j}, \mathrm{clip}(r_{x,j,t}(\theta), 1-\epsilon, 1+\epsilon) A_{x,j}\}}{\sum_t M_{x,j,t} + \varepsilon}. \quad (9)$$

The policy objective also includes KL regularization, with its coefficient reported in Appendix B. Thus more allocated trajectories improve a task's available evidence without automatically increasing its total objective weight, and reused prefixes receive no duplicate Fork loss.

Closing the allocation loop. After each policy update, terminal success and Fork–parent outcome differences supervise the joint predictor through binary cross-entropy, using pre-allocation representations. Recent success rates update the next batch's Fork quota, with eligible Replay filling released slots. Thus newly acquired experience improves both the policy and subsequent allocation decisions.

4 Experiments

4.1 Experimental Setup

Models and Datasets. We evaluate Qwen3.5-4B and Qwen3-8B (Yang et al., 2025) on ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2023a), Natural Questions (NQ) (Kwiatkowski et al., 2019), and HotpotQA (Yang et al., 2018). NQ and HotpotQA share the Search agent environment. All comparisons use the first 100 optimizer steps. Appendix A details graph construction; Appendix B summarizes the experimental settings.

Baselines. We compare QCAR with GRPO and Prompt Curriculum Learning (PCL) (Gao et al., 2025). GRPO samples a fixed-size group of independent trajectories per task and learns from group-relative rewards (Shao et al., 2024). PCL is a pure chain-of-thought prompt-selection baseline that uses a learned difficulty predictor to select training prompts; each selected prompt receives a fixed-size group of fresh rollouts.

Metrics. We evaluate held-out tasks every two optimizer steps with one sampled trajectory per task. ALFWorld success requires completing the specified task. WebShop accuracy counts purchases with a task score of one as correct; partial purchase scores are incorrect. For NQ and HotpotQA, correctness is exact answer matching after normalizing case, punctuation, articles, and whitespace. We report validation accuracy, reward-contrasting training groups, and cumulative fresh rollout tokens. Absolute accuracy gains are in percentage points; relative accuracy gains and token savings are measured against GRPO.

4.2 Policy Learning Performance

Figure 5 presents the training curves, and Table 1 summarizes accuracy at step 100 by model and task. QCAR improves over GRPO in all eight settings, with an average gain of 2.32 percentage points (4.92% relative). Gains span interactive task completion and evidence-based question answering: Qwen3.5-4B gains 5.94% on WebShop and 9.38% on HotpotQA, while Qwen3-8B gains 3.10% on NQ and 6.18% on WebShop.

The comparison with PCL distinguishes adaptive experience acquisition from prompt selection alone. At matching final checkpoints, QCAR outperforms PCL in seven of eight settings; PCL performs better on Qwen3.5-4B NQ. Across tasks, the curves show how the benefit of allocation develops as the policy changes, while the endpoint table captures the resulting policy quality.

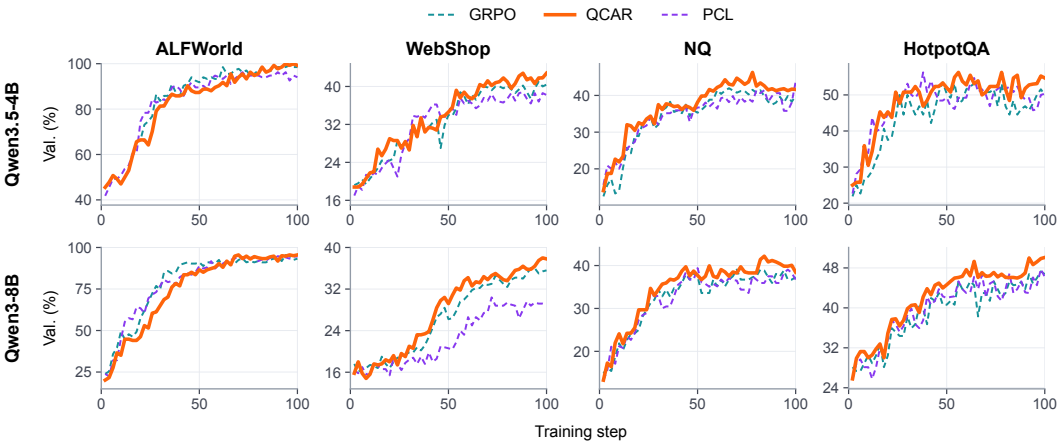


Figure 5: Validation accuracy over 100 training steps for Qwen3.5-4B and Qwen3-8B.

4.3 Reward-Contrasting Training Groups

Figure 6 compares effective groups under the same experimental protocol and counting criteria as GRPO. Over 100 steps, QCAR increases their cumulative number in all six settings: by 9.7%, 35.0%, and 35.6% on Qwen3.5-4B ALFWorld, Search, and WebShop, and by 7.1%, 20.5%, and 53.5% on Qwen3-8B ALFWorld, Search, and WebShop, respectively. The per-step differences and trailing means show how these gains accumulate during training.

Effective groups supply the reward contrast required for nonzero group-relative advantages. Their increased availability, together with the accuracy gains in Table 1, supports state-aware acquisition as a way to obtain more useful training experience. The consistent gains across models and environments connect adaptive rollout allocation to a greater supply of reward-contrasting training experience.

Table 1: Validation accuracy (%) and gains over GRPO. Best results in bold.

Task	Accuracy (%) ↑			QCAR gain ↑	
	GRPO	PCL	QCAR	Abs. (pp)	Rel. (%)
Qwen3.5-4B					
ALFWorld	98.51	94.03	**99.25**	+0.75	+0.76
WebShop	40.40	38.20	**42.80**	+2.40	+5.94
NQ	39.84	**43.75**	41.63	+1.78	+4.47
HotpotQA	50.00	50.34	**54.69**	+4.69	+9.38
Mean	57.19	56.58	59.59	+2.40	+5.14
Qwen3-8B					
ALFWorld	93.28	94.52	**95.52**	+2.24	+2.40
WebShop	35.60	29.00	**37.80**	+2.20	+6.18
NQ	37.28	36.72	**38.44**	+1.16	+3.10
HotpotQA	46.75	46.19	**50.09**	+3.34	+7.15
Mean	53.23	51.61	55.46	+2.23	+4.71



Figure 6: QCAR minus GRPO effective-group counts. Bars: per-step differences; lines: trailing 10-step means. Vertical scales vary.

Table 2: Fresh tokens and savings.

Task	Fresh tokens (M) ↓		Saving ↑
	GRPO	QCAR	(%)
Qwen3.5-4B			
ALFWorld	20.05	**19.24**	+4.04
WebShop	19.65	**19.04**	+3.10
Search	7.01	**7.00**	+0.23
Qwen3-8B			
ALFWorld	4.91	**4.50**	+8.35
WebShop	8.80	**8.24**	+6.36
Search	**2.40**	2.50	-4.17

Table 3: Ablations with Qwen3.5-4B. EGR denotes the mean effective-group rate.

Variant	Acc. (%)	Mean EGR (%)
Qwen3.5-4B		
QCAR (full)	42.80	51.88
w/o graph information	37.62	37.40
Fixed rollout allocation	41.35	38.50
w/o online recalibration	39.40	38.54
w/o balanced update	35.70	42.30

4.4 Fresh-Token Efficiency

Table 2 compares cumulative fresh-token consumption over the same training steps, using the GRPO token-counting procedure for both methods. Under this matched accounting, QCAR uses fewer fresh tokens in five of six model–environment settings. Reductions reach 4.04% on Qwen3.5-4B ALFWorld and 8.35% on Qwen3-8B ALFWorld, with savings on WebShop for both models as well. Together with the accuracy gains in Table 1, these results show that adaptive experience acquisition can improve training outcomes while reducing fresh generation.

Qwen3.5-4B Search shows a small reduction, whereas Qwen3-8B Search uses 4.17% more fresh tokens while achieving higher accuracy, reflecting the trade-off between reuse and continued exploration. This metric covers fresh rollout tokens, excluding policy optimization and environment execution.

4.5 Component Ablations

Table 3 reports Qwen3.5-4B WebShop ablations. Removing graph information lowers accuracy from 42.80% to 37.62% and mean EGR from 51.88% to 37.40%, supporting graph-informed acquisition. Fixed allocation yields 41.35% accuracy and 38.50% EGR; removing online recalibration yields 39.40% and 38.54%, supporting adaptive allocation. Without balanced updates, accuracy drops to 35.70% despite 42.30% EGR, showing that reward contrast also requires appropriate weighting during optimization.

5 Related Work

Graph representations organize agent observations (Battaglia et al., 2018), while tree search and state revisitation guide exploration (Yao et al., 2023b; Zhou et al., 2024; Ecoffet et al., 2021). Knapsack RL allocates task budgets (Li et al., 2025); Prefix Sampling controls outcome balance (Zhu et al., 2026); and InfoTree values tree expansion (Hu et al., 2026). QCAR combines graph-conditioned valuation with quality-constrained Fresh Generation/Fork/Replay allocation.

Coverage-based exploration (Bellemare et al., 2016) and prioritized replay (Schaul et al., 2016) target informative experience, while off-policy methods address policy mismatch. ExGRPO and Headroom-Drift Replay extend reuse to GRPO (Zhan et al., 2026; Park & Chang, 2026). QCAR combines acquisition-quality control and Replay rescoring with task-balanced updates, prefix masking, and online recalibration.

6 Conclusion

QCAR combines graph-conditioned valuation, quality-constrained allocation, and balanced updates. Across two models and four benchmarks, it improves accuracy over GRPO by 4.92% on average in relative terms, increases effective groups in all six model–environment settings, and reduces fresh tokens in five. Evaluation at larger training scales and across a broader range of models and tasks remains future work.

AI USE STATEMENT

The manuscript was written by the authors, with GPT used for language polishing. AI tools also assisted in developing the experimental code. The core ideas were developed by the authors through their own research and literature review. All AI-assisted content was reviewed and verified by the authors.

REFERENCES

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and William L. Hamilton. Learning dynamic belief graphs to generalize on text-based games, 2021. URL https://arxiv.org/abs/2002.09127.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, 2018. URL https://arxiv.org/abs/1806.01261.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation, 2016. URL https://arxiv.org/abs/1606.01868.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems, 2021. URL https://arxiv.org/abs/1901.10995.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient llm post-training, 2025. URL https://arxiv.org/abs/2510.01135.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Yuelin Hu, Zhenbo Yu, Zhengxue Cheng, Wei Liu, and Li Song. Maximizing rollout informativeness under a fixed budget: A submodular view of tree search for tool-use agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2605.05262.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

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*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Xintong Li, Sha Li, Yuwei Zhang, Changlong Yu, Rongmei Lin, Hongye Jin, Shuyi Guan, Xin Liu, Linwei Li, Qingyu Yin, and Jingbo Shang. Process reward informed tree rollout for effective multi-turn rl, 2026. URL https://arxiv.org/abs/2607.15610.

Ziniu Li, Congliang Chen, Tianyun Yang, Tian Ding, Ruoyu Sun, Ge Zhang, Wenhao Huang, and Zhi-Quan Luo. Knapsack rl: Unlocking exploration of llms via optimizing budget allocation, 2025. URL https://arxiv.org/abs/2509.25849.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

Hyun Bin Park and Du-Seong Chang. Headroom-drift replay: A primitive for principled replay control in grpo, 2026. URL https://arxiv.org/abs/2609.03941.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016. URL https://arxiv.org/abs/1511.05952.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

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, 2024. URL https://arxiv.org/abs/2402.03300.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning, 2021. URL https://arxiv.org/abs/2010.03768.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

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, 2025. URL https://arxiv.org/abs/2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018. URL https://arxiv.org/abs/1809.09600.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023a. URL https://arxiv.org/abs/2207.01206.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023b. URL https://arxiv.org/abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023c. URL https://arxiv.org/abs/2210.03629.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F. Wong, and Yu Cheng. Exgrpo: Learning to reason from experience, 2026. URL https://arxiv.org/abs/2510.02245.

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, 2024. URL https://arxiv.org/abs/2310.04406.

Tianshu Zhu, Wenyu Zhang, Xiaoying Zuo, Lun Tian, Haotian Zhao, Yucheng Zeng, Jingnan Gu, Daxiang Dong, Jianmin Wu, Dawei Yin, and Dou Shen. Rollout pass-rate control: Steering binary-reward rl toward its most informative regime, 2026. URL https://arxiv.org/abs/2605.05112.

A State-Graph Construction

A.1 Observable graph and rollout lineage

At turn t, the proposed adapter constructs $\mathcal{G}_{x,t} = (V_{x,t}, E_{x,t})$ only from task x and its visible action–observation prefix. Nodes have domain types and canonical identities; edges record relations, supporting turns, and provenance as requirement, observed fact, or attempted action. An attempt alone does not establish success. The graph retains unresolved constraints and excludes future observations and terminal reward. A separate lineage tree stores parents, exact prefixes, policy versions, restoration credentials, and siblings. Neither graph similarity nor an equal text prefix proves that environment states can be restored interchangeably.

Table 4: Domain-specific graph schema. Relations are admitted only when supported by visible instructions or observations.

Domain	Node types	Typed edge examples
ALFWorld	Goals, rooms, receptacles, objects, inventory	Object located in receptacle; held by agent; requires cleaning; cleaned with appliance; placed at target.
WebShop	Queries, pages, products, options, requested attributes and price	Product returned by query; has attribute; option selected; purchase completed.
Search	Questions, queries, evidence, candidate answers	Evidence retrieved by query; mentions entity; supports answer candidate.

Graph information and candidate prediction. Relation identities enter the coverage term in Equation (7); visit counts and policy age remain separate features. For each candidate continuation, the encoder summarizes historical state transitions, observed task progress, unresolved requirements, relation coverage, and recent outcomes. These features form the representation $\phi_v(c)$ in Section 3.1. A value model and an MLP fork head jointly predict continuation success and disagreement with the factual parent. The graph describes exploration opportunities; the lineage supplies the exact prefix and restoration context needed to execute a Fork.

Worked ALFWorld example. For "clean some fork and put it in diningtable," two logged GRPO trajectories share eight actions through `take fork 2 from drawer 3` (Figure 1). The graph can contain the fork, observed drawer, and still-unsatisfied cleaning and placement goals. It records a pickup attempt, adding `holding`(agent, fork 2) only if confirmed by the observation. One suffix cleans at a sink basin and then places the fork (reward 1); the other attempts placement without cleaning (reward 0). The successful branch adds `cleaned_with`(fork 2, sinkbasin 1) only after observing the effect. These independent trajectories were aligned retrospectively, not generated by an online Fork. WebShop can likewise connect a requested black attribute to an observed selected option; Search can link an issued query to visible evidence and a candidate answer.

B Experimental Settings

B.1 Tasks and evaluation

We evaluate Qwen3.5-4B and Qwen3-8B on ALFWorld, WebShop, and Search, with NQ and HotpotQA reported separately for Search. GRPO and PCL are the accuracy baselines; fresh-token comparisons use GRPO. Experiments follow the GRPO evaluation and accounting protocol and report the first 100 optimizer steps. Evaluation is performed every two optimizer steps. ALFWorld success requires completing the task; WebShop requires a purchase score of one. Search uses normalized exact answer matching. Table 1 reports the step-100 checkpoint, while training curves show the unsmoothed validation measurements.

B.2 Training and allocation configuration

Table 5 summarizes policy optimization and the reference acquisition configuration. Each batch contains 16 tasks and 128 training trajectories, matching a nominal GRPO group size of eight.

Probe trajectories are included in this budget. Dynamic allocation varies the group size between four and twelve while preserving the batch total. We use stochastic rollout generation, disable the model's thinking mode, and optimize the policy with one update epoch per batch.

Table 5: Policy optimization and reference acquisition settings.

Setting	Value
Policy optimization	
Models	Qwen3.5-4B; Qwen3-8B
Training duration / evaluation interval	100 optimizer steps / every 2 steps
Optimizer / learning rate	AdamW / 10^{-6}, constant schedule, no warmup
Adam coefficients / weight decay	$(0.9, 0.999)$ / 0.01
Gradient-norm clipping / update epochs	1.0 / 1
Task batch size / nominal group size	16 / 8
Sampling temperature / top-p / top-k	1.0 / 1.0 / disabled
Policy-ratio clipping / KL coefficient	0.2 / 10^{-3}
Entropy coefficient / numerical precision	0 / bfloat16
Graph-conditioned acquisition	
Probe count k_0 / trajectory budget B	2 per task / 128 per batch
Task-group bounds $(n_{\min}, n_{\max})$	$(4, 12)$
Minimum fresh trajectories per task	4, including Probes
Candidate Fork boundaries per task	6
Maximum Fork share / task cap / prefix cap	0.125 / 1 / 1
Exploration floor ρ / cost exponent γ	0.30 / 0.45
Fork selection margin	1.40
Maximum Replay staleness $S_{\max}$	4 policy updates
Quality tolerance ϵ_q / EMA coefficient	0.03 / 0.50
Quota decay / recovery / minimum scale	0.50 / 0.04 / 0.10
Fresh / Fork / Replay reliability weights	1.0 / 0.25 / 0.25
Sibling-advantage weight	0.25

Interaction limits. The maximum initial prompt length is 2,048 tokens. ALFWorld, WebShop, and Search use response-token budgets of 32,768, 16,384, and 4,096, respectively, with at most 50, 15, and 4 interaction steps. Each action-generation call is capped at 512 tokens, and the agent retains three recent interaction turns in its local history. The state graph accumulates information across the interaction rather than being restricted to this local text window.

B.3 Training-signal and cost measurements

A reward-contrasting group contains unequal terminal rewards. We compare the accumulated effective groups over 100 steps and show per-step differences with trailing 10-step means. Fresh-token consumption follows the same counting procedure used for GRPO: it includes newly generated rollout tokens and excludes environment execution, restoration, Replay rescoring, predictor fitting, and policy optimization. Absolute accuracy differences are reported in percentage points; relative accuracy gains and token savings are measured against GRPO.

B.4 Motivation diagnostics

The motivation analysis uses saved Qwen3.5-4B GRPO trajectories from steps 1–100 on ALFWorld, WebShop, and Search. The diagnostic collection contains 16 tasks with eight Root rollouts per step and environment, totaling 38,400 trajectories. Prediction splits hold out entire task instructions. Figure 2 examines outcome prediction, error-loop cues, and a continuation-selection benchmark comparing graph-ranked shared-prefix pairs with independent Root pairs. Figure 3(a) measures reward contrast as group size increases, whereas panel (b) illustrates the conceptual cost–signal trade-off of rollout reuse.