MOSAIC: MULTI-SESSION AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Most existing agentic reinforcement learning methods assume single-session trajectories, where interaction proceeds within one continuously accumulated context. However, agentic techniques such as context compaction and subagent forking may produce trajectories spanning multiple sessions, creating a mismatch between the single-session training assumption and inference-time execution. Therefore, we introduce **Multi-Session Agentic Reinforcement Learning**, a more general setting that supports an arbitrary number of sessions within a trajectory. Under this setting, a widely adopted method is to broadcast trajectory-level advantages to all sessions, but this does not distinguish contributions from various sessions or guide session allocation within trajectories. To address this issue, we propose **MOSAIC**, a multi-session agentic RL method with critic-free hierarchical optimization. At the trajectory level, it incorporates session-usage signals into advantage estimation. At the session level, it shapes each session's advantage according to the estimated marginal contribution. Experiments on BrowseComp with Qwen3.5-4B show that MOSAIC outperforms broadcasting in both context compaction and subagent forking scenarios while using fewer sessions on average. Further analyses show that our method achieves improvements in session utility and allocation, which we suggested is a promising target for future advances in multi-session agentic RL.

1 INTRODUCTION

Agentic Reinforcement Learning (Agentic RL; Zhang et al., 2025; Jin et al., 2025; Gao et al., 2025) trains Large Language Model (LLM) agents (Yao et al., 2022; Shinn et al., 2023) for multi-turn reasoning, tool use, and interaction with external environments via online rollouts and reward feedback, yielding broad performance gains. More recently, LLM agents have incorporated advanced inference techniques, such as context management (Mei et al., 2025; Zhang et al., 2026a) and harness engineering (Zhou et al., 2026; Lin et al., 2026), enabling agentic workflows to tackle increasingly complex and long-horizon tasks. However, most agentic RL methods assume a single-session setting, where each rollout unfolds as a coherent multi-turn trajectory within a continuously accumulated context and yields a single training segment (Wang et al., 2026; Xu et al., 2026b).

We argue that the single-session setting is not sufficiently general for optimizing LLM agents. For example, context compaction compresses the execution history to free up context space while rewriting the prefix that conditions subsequent actions (Nguyen et al., 2025; Wu et al., 2025; Lu et al., 2025), thereby initiating a new session. Similarly, subagent forking delegates global goal to subagents which creates branch sessions (Xu et al., 2026d; Team et al., 2026; Song et al., 2025). Those resulting trajectories no longer satisfy the single-session assumption, creating a mismatch between training and inference that compromises consistency and limits optimization potential.

To address this gap, we introduce a new setting termed **Multi-Session Agentic Reinforcement Learning**, which supports an arbitrary count of sessions and enables agents with diverse inference mechanisms to be optimized within a unified algorithm. A straightforward way to handle multi-session rollouts is through *broadcasting*, whereby a trajectory-level outcome reward or advantage is replicated to every token across all sessions (Accio-Lab, 2026; Xu et al., 2026a). However, we argue that this method provides only coarse supervision and offers no explicit control over optimization across sessions, resulting in poorly targeted learning signals as trajectories span more sessions.

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.

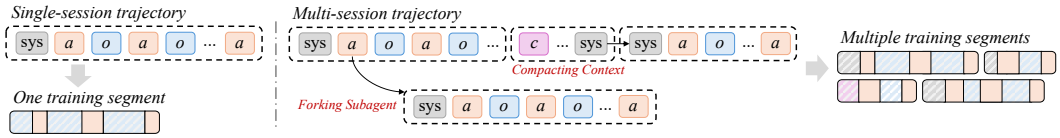

Figure 1: The comparison between multi-session and single-session trajectories.

In this work, we propose **MOSAIC**, a multi-session agentic RL method with critic-free hierarchical optimization. First, **session-aware trajectory-level advantage estimation** calibrates outcome-based advantages using within-group rankings of session counts conditioned on task outcomes, encouraging successful trajectories to solve tasks efficiently while allowing unsuccessful trajectories sufficient exploration. Its bounded modulation preserves outcome-induced ordering while incorporating session-usage guidance, thereby widening relative advantage margins for homogeneous-reward groups. Second, **session-level advantage shaping** calculates each session's marginal outcome gain as an immediate reward, applies group-relative normalization separately by session type, and combines the resulting shaping signal with the trajectory-level advantage. Overall, the former provides a long-term, outcome-oriented global signal, while the latter captures short-term local marginal effects. They together form a hierarchical optimization target for multi-session agentic RL.

To evaluate our method, we conduct experiments on BrowseComp (Wei et al., 2025) and select Qwen3.5-4B (Qwen Team, 2026) as the base model. Our MOSAIC outperforms multiple single-session and multi-session agentic RL baselines under both context compaction and subagent forking scenarios. Moreover, MOSAIC achieves up to a 10.1% relative improvement in accuracy over the broadcasting baseline while using fewer sessions per trajectory on average (3.17 $\rightarrow$ 2.35). Ablation studies further demonstrate that session-aware trajectory-level advantage estimation and session-level advantage shaping each contribute to the final performance gains. Our analyses also reveal several key insights: (1) shaping trajectory-level advantages with session counts increases within-group advantage variance and improves sample efficiency; (2) different session types exhibit distinct distributions of session-level marginal gains, which our type-conditioned normalization helps stabilize; (3) effective multi-session optimization improves session utility and allocation, a pattern that MOSAIC strengthens and that might offer a potential direction for future work.

2 PRELIMINARIES

2.1 AGENTIC RL WITH GROUP-RELATIVE POLICY OPTIMIZATION

Given a task $x \sim \mathcal{D}$, an agent policy $\pi_{\boldsymbol{\theta}}$ generates a rollout trajectory τ through multi-turn reasoning, tool use, and interaction with the environment. Formally, we represent the trajectory as:

$$\tau = (a_1, o_1, \ldots, a_T, o_T), \qquad a_t \sim \pi_{\boldsymbol{\theta}}(\cdot \mid h_t), \tag{1}$$

where a_t denotes the agent action at turn t, o_t is the corresponding observation returned by the environment, and $h_t = (x, a_{<t}, o_{<t})$ denotes the interaction history available to the policy. Upon completion of the rollout, the environment or a verifier assigns a scalar outcome reward $R(\tau)$. The objective of Agentic RL is to maximize the expected reward over the task distribution:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{x\sim\mathcal{D},,\tau\sim\pi_{\boldsymbol{\theta}}(\cdot|x)} \left[R(\tau)\right]. \tag{2}$$

To estimate policy advantages without learning a separate value function, Group Relative Policy Optimization (GRPO; Shao et al., 2024) uses the relative rewards of multiple rollouts sampled for the same task. Specifically, for each task x, GRPO samples a group of rollouts from the old policy $\pi_{\boldsymbol{\theta}_{\text{old}}}$: $G(x) = \{\tau_i\}_i^G$. Each rollout receives a trajectory-level reward $R_i = R(\tau_i)$. GRPO then estimates its relative advantage using the reward statistics within the group:

$$\hat{A}_i^{\text{traj}} = \frac{R_i - \mu_G}{\sigma_G + \epsilon}, \tag{3}$$

where μ_G and σ_G are arithmetic mean and standard deviation of rewards in group G, respectively. Unlike PPO-style methods (Schulman et al., 2017) that typically rely on a learned critic to estimate

token-level or state-level advantages, GRPO constructs its baseline directly from group-level reward statistics. This critic-free design reduces the computational and memory overhead of training an additional value model and avoids errors arising from inaccurate value estimation.

Ignoring optional KL regularization for clarity, the GRPO objective can be written as:

$$\mathcal{J}_{\text{GRPO}} = \mathbb{E}_{x\sim D,\{\tau_i\}_{i=1}^{G}\sim\pi_{\boldsymbol{\theta}_{\text{old}}}(\tau|q)} \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\tau_i|}\sum_{t=1}^{|\tau_i|}\min\left(\rho_{i,t}(\boldsymbol{\theta})\hat{A}_i^{\text{traj}}, \text{clip}(\rho_{i,t}(\boldsymbol{\theta}))\hat{A}_i^{\text{traj}}\right), \quad (4)$$

where $\rho_{i,t} = \frac{\pi_{\boldsymbol{\theta}}(a_{i,t}|h_{i,t})}{\pi_{\boldsymbol{\theta}_{\text{old}}}(a_{i,t}|h_{i,t})}$ is the token-level importance ratio.

Most existing GRPO variants, however, assume a single-session setting in which each rollout is represented as a continuous, linear training segment. This assumption becomes inadequate when advanced inference workflows extend agent execution across multiple sessions, creating a mismatch between training and inference that compromises consistency and limits optimization potential.

2.2 Broadcasting for Multi-Session Agentic RL

We introduce *session* as a finer-grained unit within an LLM agent trajectory. Specifically, a trajectory τ_i is represented as a set of $N_i \in \mathbb{N}$ of sessions:

$$\tau_i = \{s_{i,j}\}_{j=1}^{N_i}. \quad (5)$$

Each session consists of a sequence of agent–environment interactions same as trajectories in conventional single-session settings:

$$s_{i,j} = (a_{i,j,1}, o_{i,j,1}, \ldots, a_{i,j,T_{i,j}}, o_{i,j,T_{i,j}}), \qquad a_{i,j,t} \sim \pi_{\boldsymbol{\theta}}(\cdot \mid h_{i,j,t}). \quad (6)$$

For example, a session may include consecutive turns of execution to complete a task. When the remaining context window becomes insufficient, a new session can be initiated to perform context compaction, with the resulting compressed prefix serving as the initial context for another session.

Built upon the GRPO pipeline, *broadcasting* makes multi-session rollouts trainable by propagating the trajectory-level advantage $\hat{A}_i^{\text{traj}}$ to every session and every trainable token within τ_i:

$$\hat{A}_{i,j,t}^{\text{token}} = \hat{A}_{i,j}^{\text{session}}, \;\; \hat{A}_{i,j}^{\text{session}} = \hat{A}_i^{\text{traj}}, \quad \forall s_{i,j} \in \tau_i, \;\; t \in \{t \mid 1 \le t \le T_{i,j} \wedge \texttt{mask}(t) = 1\}. \quad (7)$$

Since each session corresponds to a separate training segment in frameworks such as Megatron (Shoeybi et al., 2020), the token-level objectives from all sessions should further be aggregated at the trajectory level:

$$\begin{aligned}\mathcal{J}'_{\text{GRPO}} =& \mathbb{E}_{x\sim D,\{\tau_i\}_{i=1}^{G}\sim\pi_{\boldsymbol{\theta}_{\text{old}}}(\tau|q)} \\ & \frac{1}{G}\sum_{i=1}^{G}\frac{1}{\sum_{j=1}^{N_i}|s_{i,j}|}\sum_{j=1}^{N_i}\sum_{t=1}^{|s_{i,j}|}\min\left(\rho_{i,t}(\boldsymbol{\theta})\hat{A}_{i,j,t}^{\text{token}}, \text{clip}(\rho_{i,t}(\boldsymbol{\theta}))\hat{A}_{i,j,t}^{\text{token}}\right).\end{aligned} \quad (8)$$

This formulation yields a single token-weighted trajectory-level objective for each rollout, regardless of its session count, thereby preserving equal weighting across rollouts within a prompt group and stabilizing training-signal scales across tokens. However, broadcasting offers only coarse supervision and no explicit control over optimization across sessions, causing poorly targeted learning signals as trajectories span more sessions. To address this limitation, we propose **MOSAIC**, which improves multi-session agentic RL through session-aware trajectory-level advantage estimation and session-level advantage shaping.

3 Methodology

In MOSAIC, we optimize the performance of multi-session agentic RL through a hierarchical perspective, incorporating trajectory-level advantage estimation calibrated with session signals (§3.1), as well as advantage shaping at the finer-grained session level. (§3.2).

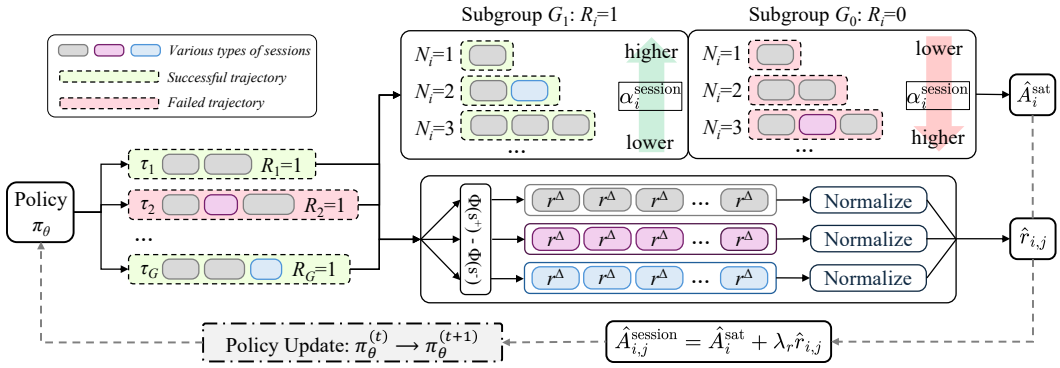

Figure 2: Overview of our proposed MOSAIC pipeline for multi-session agentic RL.

3.1 SESSION-AWARE ADVANTAGE ESTIMATION

In vanilla GRPO, the advantage is determined solely by the relative outcome rewards within a prompt group. Under the multi-session setting, however, session-related metrics may provide additional relative signals for policy optimization. One metric is the number of sessions N_i in a trajectory τ_i, which serves as a proxy for the extent of exploration and structural complexity of τ_i. Nonetheless, N_i should not be directly treated as a reward. First, its raw scale may not faithfully quantify meaningful differences in session utilization. Second, the preferred direction of optimization should depend on task results, avoiding the policy from exploiting session count without improving performance.

Accordingly, from a group-relative perspective, the session-aware trajectory-level advantages of multi-session rollouts within a prompt group should satisfy the following ordering constraints:

- **C1**: For successful trajectories, using more sessions incurs a higher execution cost and hence results in a lower advantage, whereas using fewer sessions should result in a higher advantage.
- **C2**: For unsuccessful trajectories, using fewer sessions may suggest insufficient exploration or execution, especially on challenging tasks, and should thus result in a lower advantage. Conversely, unsuccessful trajectories that conduct more sessions should receive a higher advantage.
- **C3**: Task outcome reward must remain the dominant signal. When two trajectories have the same session counts, the trajectory with lower reward should receive a lower advantage. Using more sessions alone should not allow an unsuccessful trajectory to outrank a successful one.

Together, these constraints encourage the policy to use enough sessions to complete the task, but no more than necessary, which represents a desirable objective for multi-session agentic RL. Specifically, we assume a binary final outcome reward $R_i \in \{0, 1\}$, where 1 and 0 represents success and failure, respectively. For a prompt group containing both successful and unsuccessful rollouts, the trajectory-level GRPO advantage satisfies:

$$R_i = 1 \Longrightarrow \hat{A}_i^{\text{traj}} > 0, \quad R_i = 0 \Longrightarrow \hat{A}_i^{\text{traj}} < 0. \tag{9}$$

Based on the ordering constraints above, we define the session-aware trajectory-level advantage as:

$$\hat{A}_i^{\text{sat}} = \left(1 + \alpha_i^{\text{session}}\right) \hat{A}_i^{\text{traj}}, \tag{10}$$

where $\alpha_i^{\text{session}}$ is the session signal derived from N_i, acting as a factor that modulates the trajectory-level advantage according to session usage, and extending optimization beyond the final outcome.

Considering that session signal has different implications under different task outcomes as reflected by C1 and C2, session signals are not directly comparable between successful and unsuccessful trajectories. We thus further divide the prompt group G into subgroups according to R_i:

$$G_1 = \{i \in G : R_i = 1\}, \quad G_0 = \{i \in G : R_i = 0\} \tag{11}$$

Generally, each τ_i belongs to G_{R_i}. We then map its N_i to a centered rank within this subgroup:

$$u_i = 2f(N_i) - 1, f(N_i) = \frac{\text{rank}_{G_{R_i}}(N_i) - 1}{|G_{R_i}| - 1}, \tag{12}$$

where $u_i \in [-1, 1]$, and the rank is computed in ascending order of N_i inside G_{R_i}. Thus, a larger N_i yields a larger u_i. We use centered ranks instead of raw session counts to obtain a bounded signal robust to long-tailed trajectories, as differences in N_i may not reflect proportional differences in session usage. We then define the session modulation factor as:

$$\alpha_i^{\text{session}} = -\rho u_i, \tag{13}$$

where $0 \leq \rho < 1$ and ρ controls the proportion by which the session signal may amplify or attenuate the original advantage. Then we have $1 - \rho u_i \in [1 - \rho, 1 + \rho]$ which is always positive. The session signal thus modulates only the magnitude of the outcome advantage without changing its sign.

Finally, we consider the fallback cases in the above computation. First, the centered-rank signal u_i is well-defined only when $|G_{R_i}| > 1 \wedge \mathrm{Var}_{G_{R_i}} > 0$. If either condition is not satisfied, we set $u_i = 0$. When all trajectories in a prompt group receive the same outcome reward, the outcome advantage in Eq. 10 becomes 0, rendering the session signal ineffective. To improve sample efficiency in these homogeneous groups, we introduce the following fallback in the final advantage:

$$\boxed{\hat{A}_i^{\text{sat}} = \begin{cases} (1 - \rho u_i)\, \hat{A}_i^{\text{traj}}, & 0 < \sum_{j \in G} R_j < |G|, \\ -\beta_1\, u_i, & \sum_{j \in G} R_j = |G|, \\ \beta_0\, \min(u_i, 0), & \sum_{j \in G} R_j = 0. \end{cases}} \tag{14}$$

For all-failure groups, we adopt the more conservative fallback, ensuring that no unsuccessful trajectory receives a positive advantage. The nonnegative coefficients β_0 and β_1 control the strengths of the fallback signals for all-failure and all-success groups, respectively.

3.2 Session-Level Advantage Shaping

The session-aware trajectory-level advantage developed in §3.1 provides an outcome-oriented global optimization signal for the entire multi-session trajectory. At the session level, however, this signal remains sparse since all sessions on the same trajectory share a single scalar advantage. To this end, we introduce session-level advantage shaping. Specifically, we view each session as a state transition as $s_{i,j} = (v_{i,j-1} \to v_{i,j})$, where session $s_{i,j}$ transitions the agent from state $v_{i,j-1}$ to state $v_{i,j}$. Here, $v_{i,j-1}$ and $v_{i,j}$ denote the agent states at the start and end of the j-th session, respectively, with particular emphasis on the prefix context visible to the rollout policy. Based on session transition functions, we introduce three representative session types:

- **Normal session**. The agent performs regular reasoning, tool use, and environment interaction within the current context. The input state is the context available at the start of the session, and its output state is the resulting context after the session is completed.
- **Compaction session**. When the accumulated context approaches the model's limit, the agent starts a compaction session to compress the context into a shorter prefix. Its input state is the full context before compaction, and its output state is the compressed prefix passed to the next session.
- **Subagent session**. The main agent delegates subtasks to subagents which start new subagent sessions. Its input state is the main-agent context at delegation, and its output state is the updated main-agent context after the returned result has been incorporated.

To quantify the benefit of each session transition, we construct a critic-free process supervision signal based on marginal information gain. Specifically, let $y_i^* = \left(y_{i,1}^*, \ldots, y_{i,L_i}^*\right)$ denote the ground-truth answer to query x_i of trajectory τ_i. We use the rollout policy before the update of each step, denoted by $\pi_{\bar{\boldsymbol{\theta}}}$, and define the answer-likelihood potential of any state v as:

$$\Phi_{\bar{\boldsymbol{\theta}}}\left(y_i^* \mid x_i, v\right) = \frac{1}{L_i} \sum_{\ell=1}^{L_i} \log \pi_{\bar{\boldsymbol{\theta}}}\left(y_{i,\ell}^* \mid x_i, v, y_{i,<\ell}^*\right). \tag{15}$$

Intuitively, $\Phi_{\bar{\boldsymbol{\theta}}}(y_i^* \mid x_i, v)$ measures how readily the rollout policy can generate the correct answer given the prefix context represented by state v. A session that adds useful evidence, reduces noise, or returns a high-quality subtask result should increase the average log-likelihood of the ground-truth answer. Conversely, a transition that discards key facts or introduces misleading information may decrease it. Accordingly, for a session transition $s_{i,j} = (v_{i,j-1} \to v_{i,j})$, we define its session-level reward as the marginal information gain between its output and input states:

$$r_{i,j}^{\Delta} = \mathrm{stopgrad}\left[\Phi_{\bar{\boldsymbol{\theta}}}\left(y_i^* \mid x_i, v_{i,j}\right) - \Phi_{\bar{\boldsymbol{\theta}}}\left(y_i^* \mid x_i, v_{i,j-1}\right)\right]. \tag{16}$$

Different types of session may have different distributions of marginal gain. Normal sessions typically gain external evidence through search and browse. The gain from a compaction session depends on whether compression retains useful information while removing noise. A subagent session is affected by both the quality of subtask execution and the quality of result handoff. Therefore, we normalize the session-level rewards separately by session type within each prompt group:

$$\hat{r}_{i,j} = \begin{cases} \frac{r^{\Delta}_{i,j} - \mu_{c(s_{i,j})}}{\sigma_{c(s_{i,j})} + \epsilon} & \text{if } 1 \le j < N_i, \\ 0 & \text{if } j = N_i, \end{cases} \tag{17}$$

where $c(s_{i,j}) \in \{\text{normal}, \text{compact}, \text{subagent}, \dots\}$ denotes the type of session $s_{i,j}$, and $\mu_{c(s_{i,j})}$ and $\sigma_{c(s_{i,j})}$ are computed over sessions of the same type within the prompt group. We exclude the last session since its output state usually includes the agent's final answer. Particularly, a correct final answer may sharply increase the answer likelihood which confounds the marginal gain distribution. Setting its shaping signal to 0 prevents the final answer from entering the evaluated output state.

Finally, we combine the normalized session-level shaping signal with the trajectory-level advantage:

$$\hat{A}^{\text{session}}_{i,j} = \hat{A}^{\text{traj}}_{i} + \lambda_r \hat{r}_{i,j}, \tag{18}$$

where λ_r controls the intensity of the shaping signal, and $\hat{A}^{\text{traj}}_i$ can be instantiated directly as $\hat{A}^{\text{sat}}_i$ in §3.1, after which $\hat{A}^{\text{session}}_{i,j}$ replaces the broadcast session advantage used to compute $\hat{A}^{\text{token}}_{i,j,t}$ in Eq. 8. Overall, the first term provides a long-term, outcome-oriented global signal, while the second differentiates sessions according to their local relative marginal gains.

4 Experiments

4.1 Experimental Setup

Benchmark. We evaluate our method on BrowseComp (Wei et al., 2025), a challenging benchmark consisting of 1,266 questions that require agents to persistently navigate the web and identify hard-to-find, entangled information. We select BrowseComp since its high difficulty demands long-horizon interactions, making it suitable for assessing agents that produce multi-session trajectories.

Baselines. We use Qwen3.5-4B (Qwen Team, 2026) as the initial model and adopt a ReAct-style agent loop. We organize the baselines by inference configuration. Under **single-session inference**, we evaluate (1) the initial model without RL training and (2) a model trained with GRPO in the single-session setting, representing conventional Agentic RL. Under **multi-session inference**, we evaluate (3) the single-session GRPO trained model, representing test-time scaling; (4) a model trained with Broadcasting in the multi-session setting; and (5) a model trained with MOSAIC in the multi-session setting. We evaluate all multi-session configurations under both context compaction and subagent forking using their respective agent loops; for multi-session training, the same loops are used to collect rollouts. Finally, since our session-level advantage shaping in §3.2 builds on information gain, we evaluate (6) a single-session model trained with IGPO (Wang et al., 2026).

Implementation Details. The agent has two tools, `search` and `visit`, implemented using Serper [1] and Jina Reader [2] APIs, respectively. During training and evaluation, `deepseek-v4-flash` (DeepSeek-AI et al., 2026) serves as the LLM judge for verifying final answers. For context compaction, the agent compresses its interaction history when approaching the context limit. For subagent forking, it can invoke non-nested subagents through a dedicated action. Each session is limited to 128K tokens. For RL training, we use REDSearcher_RL_1k (Chu et al., 2026), which contains 1,000 challenging deep-search tasks. We train all models for one epoch with a rollout batch size of 16 and a group size of 8. We set $\rho = 0.2$, $\beta_0 = 0.1$, $\beta_1 = 0.2$. and $\lambda_r = 0.1$.

4.2 RQ1: Overall Effectiveness of Mosaic

RQ1 asks *whether training in the multi-session setting improves agent performance and whether* MOSAIC *contributes beyond broadcasting*.

[1] `https://serper.dev/`

[2] `https://jina.ai/reader/`

Table 1: Results of various methods and inference modes on BrowseComp.

Method \ Inference	Single-Session		Multi-Session / Compact		Multi-Session / Subagent	
	Accuracy	Num. Session	Accuracy	Num. Session	Accuracy	Num. Session
Qwen3.5-4B	23.22	1.00	24.80	1.70	21.64	2.98
Training on Single-Session						
GRPO	31.99	1.00	34.68	1.24	31.91	2.09
IGPO	28.52	1.00	31.36	4.12	24.64	3.26
Training on Multi-Session						
Broadcasting	–	–	35.31	3.17	31.44	4.43
MOSAIC	–	–	**38.86**	2.35	**32.46**	3.77

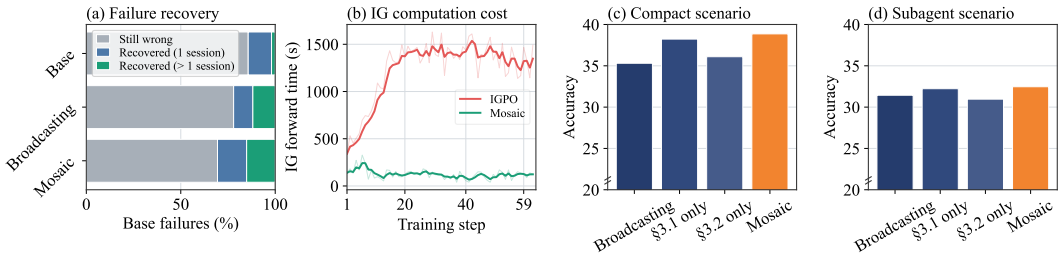

Figure 3: **(a)**. Percentage of the 972 tasks failed by the base model under single-session inference that are solved by different methods under multi-session inference with compact. **(b)**. Comparison of the computation time for marginal information gain. **(c)(d)**. Ablation results on BrowseComp.

As shown in Table 1, single-session inference has limited effectiveness on BrowseComp. Agentic RL training in the single-session setting yields modest gains, even when combined with test-time scaling in the multi-session setting. This suggests that multi-session inference capabilities do not readily emerge through generalization alone. Among methods trained on multi-session trajectories, Broadcasting improves multi-session inference performance. MOSAIC improves it further, outperforming all baselines in both the compact and subagent scenarios. Figure 3 (a) illustrates the transformative gains from multi-session inference, multi-session training, and MOSAIC.

Figure 4 shows the training dynamics of each method. MOSAIC boosts trajectory reward faster than Broadcasting and reaches a higher level. The number of sessions increases over training for both methods, but its growth levels off more clearly with MOSAIC. This may be partly due to the more targeted signal provided by its session-aware advantage, which discourages unnecessary increases in session count. Overall, MOSAIC achieves better task performance at lower execution cost.

As §3.2 applies marginal gain to session-level advantage shaping, we thus compare MOSAIC with IGPO, which applies it at the turn level. From Table 1, MOSAIC achieves better performance than IGPO. Moreover, Figure 3 (b) shows that MOSAIC substantially reduces the wall-time of computing marginal gains, enhancing performance while cutting computational cost.

4.3 RQ2: ABLATION STUDY OF MOSAIC

RQ2 asks *whether each component of* MOSAIC *contributes to its overall performance*.

Figure 3 (c)(d) presents the ablation results. Relative to Broadcasting, both levels of components in MOSAIC, namely trajectory-level session-aware advantage estimation and session-level advantage shaping, contribute to the enhanced final performance. Combining the two components yields a clear additional gain in the compact scenario, but only a modest gain in the subagent scenario. One possible explanation lies in their session structures: compact trajectories are linear, whereas subagent trajectories have a nonlinear structure that more closely resembles a directed acyclic graph. In the latter case, a session may influence multiple succeeding sessions. Its immediate reward, estimated from marginal gain, may thus differ from its actual contribution to the final outcome.

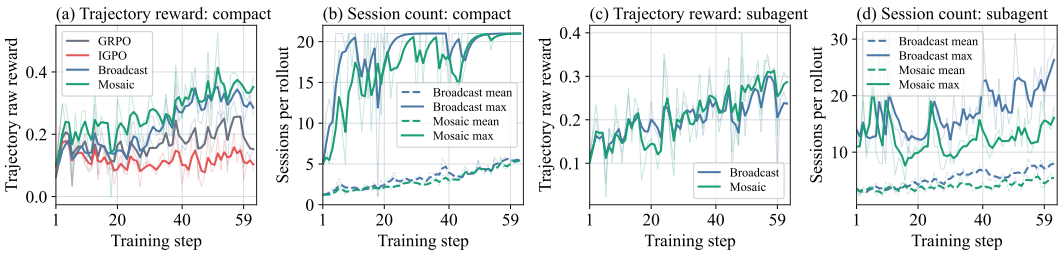

Figure 4: Curves showing rewards and session counts during the training process of various methods.

4.4 RQ3: Mechanistic Analysis of Mosaic

RQ3 asks *how the design of* Mosaic *contributes to the training process*.

First, we examine whether session-aware trajectory-level advantage estimation produces the intended ordering. As shown in Figure 5, within mixed-outcome groups, advantages approximately decrease with session count for successful trajectories and increase for unsuccessful ones, while successful trajectories retain higher advantages. In homogeneous-outcome groups, the fallback mechanism introduces session-dependent variation, providing differentiated learning signals despite identical outcome rewards. These patterns illustrate how Mosaic leverages session signals to increase the within-group variance of relative advantages, thus improving sample efficiency during training.

Second, we examine the effect of reward normalization on session-level shaping signals. As shown in Figure 5, the distributions of raw marginal information gain differ across session types, primarily because heterogeneous sessions serve distinct functions. Under global normalization, compaction sessions, whose raw marginal gains have a lower mean, are further downweighted, introducing a systematic bias against them. In Mosaic, type-conditioned normalization instead centers and rescales each type separately, enabling both types to receive positive and negative signals relative to their respective baselines. This reduces type-dependent distributional bias in advantage shaping.

4.5 RQ4: Understanding Multi-Session Optimization

Multi-session optimization introduces an additional degree of freedom in how computation is distributed across sessions. However, the session count used by Mosaic serves only as a proxy signal and is not inherently tied to exploration intensity or execution cost. Moreover, at inference time, the model has no access to hindsight: it does not know how many sessions a trajectory will ultimately contain or whether the trajectory will succeed. Therefore, RQ4 moves beyond method-specific analysis and asks a broader question: *when multi-session optimization improves task performance, what session-level behavioral patterns emerge?*

Accordingly, we examine session-level behavior along three dimensions: *quantity*, or whether the count of sessions increases or decreases; *quality*, or whether performance improvement is accompanied by more useful sessions; and *allocation*, or whether a limited session budget becomes increasingly concentrated on execution stages with high marginal utility. As shown in Table 2, for Broadcasting, which does not directly optimize any specific session-aware pattern, all three metrics increase. Under Mosaic, the number of sessions does not increase further, while both session quality and allocation improve. These findings indicate that performance gains via multi-session optimization are associated with improved session utility rather than a monotonic rise in session count. Moreover, despite using explicit session-aware optimization, Mosaic does not induce a fundamentally altered behavioral mechanism. Instead, it strengthens the session-allocation patterns that already emerge from naïve multi-session optimization.

We argue that effective multi-session optimization is accompanied by improvements in session utility and allocation. It distributes computation more effectively across sessions, encouraging continued execution when it yields meaningful progress while suppressing low-utility computation. From this perspective, the central challenge is not how many sessions an agent should use, but how much useful progress it can derive from the sessions it executes.

Table 2: Session-level patterns under multi-session optimization on BrowseComp.

	Multi-Session / Compact				Multi-Session / Subagent			
Method	**Accuracy ↑**	**# Sessions**	**Positive-IG ↑**	**IG / Session ↑**	**Accuracy ↑**	**# Sessions**	**Positive-IG ↑**	**IG / Session ↑**
Base Model	24.80	1.70	37.36%	0.30	21.64	2.98	27.39%	0.41
Broadcasting	35.31	3.17	47.74%	0.86	31.44	4.43	41.15%	0.74
MOSAIC	38.86	2.35	59.92%	0.89	32.46	3.77	51.04%	0.80

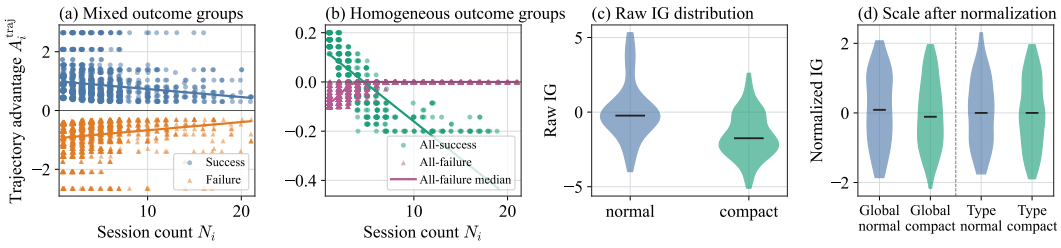

Figure 5: The two left panels show the relationship between trajectory-level advantage and session count. The two right panels show the distributions of marginal information gain for normal and compact sessions under global normalization and type-conditioned normalization.

5 Related Work

LLM Agents. LLM agents extend LLMs with tool use and environment interaction to work autonomously (Feng et al., 2026; Li et al., 2025; Wu et al., 2026; Wang et al., 2024). ReAct constructs a general agent loop of reasoning, acting, observing environment feedback, and continuing until task completion (Yao et al., 2022). To handle complex and long-horizon tasks, recent systems develop agent harnesses that manage execution beyond the basic ReAct loop (Works, 2026; OpenClaw, 2026), incorporating context compaction (Nguyen et al., 2025), subagent forking (Team et al., 2026), and other forms of context and workflow management (Shi et al., 2026; Xu et al., 2026c; Zhang et al., 2026a; Chen et al., 2025; Kim et al., 2025; Yang et al., 2026).

Agentic RL. Reinforcement learning techniques such as RLHF (Ouyang et al., 2022; Zhang et al., 2026b) and RLVR (Chen et al., 2026; Yu et al., 2026) are used to improve LLMs' alignment, reasoning, and downstream task-solving capabilities (Zheng et al., 2025; Tan et al., 2025). Agentic RL extends this paradigm to LLM agents, aiming to enhance capabilities of multi-turn interaction through enhanced rollout sampling (Dong et al., 2026; Hou et al., 2025), process-level reward design (Zhang et al., 2026c; Liu et al., 2026), and other optimization strategies. However, most existing agentic RL methods still assume that each trajectory is contained within a single session.

6 Conclusion

In this paper, we first identify a mismatch between the single-session assumption underlying most existing agentic RL methods and advanced agentic inference techniques such as context compaction and subagent forking. This motivates our formulation of Multi-Session Agentic Reinforcement Learning, which supports an arbitrary count of sessions within a trajectory. To address the coarse supervision issue existed in the currently widely used broadcasting approach, we further propose MOSAIC, a multi-session agentic RL method with critic-free hierarchical optimization. By combining session-aware trajectory-level advantage estimation with session-level advantage shaping, MOSAIC trains a Qwen3.5-4B agent that outperforms both single-session and multi-session baselines on BrowseComp. Ablation studies confirm the contribution of each component, while further analyses suggest that effective multi-session optimization might depend on improving session utility and allocation, a pattern observed in both broadcasting baseline and our MOSAIC. We hope these findings will inform future research on multi-session agentic RL.

LIMITATIONS

Our implementation alternates between rollout sampling and policy model updating, waiting for each rollout batch to finish before training on it. This limits training throughput, particularly when multi-session trajectories vary substantially in length. We adopted this design because our compute resources were insufficient to support concurrent rollout and training at scale. But synchronous mode also makes the effects of our optimization method easier to attribute, since asynchronous rollouts may be generated by older policy versions and introduce policy lag. Future work could extend MOSAIC to asynchronous training and examine how policy lag affects both efficiency and session-level advantage estimation.

Besides, high-quality, publicly available training tasks that require extended agent execution remain scarce. Many tasks can be completed within a single session, making it difficult to collect enough trajectories that meaningfully exercise context compaction or subagent forking. This limits the diversity of multi-session behaviors available for training and evaluation. Progress in this setting will require more difficult, longer-horizon open tasks, as well as methods for constructing and validating tasks that genuinely benefit from multiple sessions.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation, polish the writing, and evaluate the correctness of answers on test data (i.e., LLM-as-Judge). We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, propose or refine hypotheses, design or provide feedback on research methods or experiments, implement methods, support qualitative or thematic data analysis, or interpret results. The use of generative AI tools to formulate mathematical claims, provide key ingredients for proving mathematical claims, assist in writing proofs, or clean and reformat datasets was not applicable to this work. Additionally, we used generative AI tools to edit the research paper to improve readability. We reviewed all AI-assisted work and checked the AI-edited text to ensure that it preserved the intended meaning and contained no overclaims or factual errors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies how to train LLM agents when task execution spans multiple sessions. Our goal is to improve how such agents learn from their trajectories, rather than to develop a specific downstream application. We do not identify direct ethical concerns arising from the motivation, methods, or experiments in this study. Our experiments evaluate agents on benchmark tasks in a research setting and do not involve deployment to users or high-stakes decision-making. As with other work that improves agent capabilities, future applications should consider potential risks, including misuse and privacy concerns, and adopt safeguards appropriate to their setting.

REPRODUCIBILITY STATEMENT

In this paper, §3.1 and §3.2 present the formal formulation of our proposed MOSAIC and explains how its training signals are computed at the trajectory and session levels. §4.1 describes the experimental setup, including the public training and evaluation datasets, the APIs used during training and evaluation, and the main training hyperparameters. The appendix provides additional details needed to understand the training and inference procedures, including the inference workflows under context compaction and subagent forking scenarios and more complete training configurations. We will provide a link to the source code upon acceptance or when the paper is made public.

REFERENCES

Accio-Lab. Dressage. GitHub, 2026. URL https://github.com/Accio-Lab/Dressage.

Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Wayne Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, et al. Iterresearch: Rethinking long-horizon agents via markovian state reconstruction. arXiv e-prints, pp. arXiv–2511, 2025.

Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? Advances in Neural Information Processing Systems, 38:57654–57689, 2026.

Zheng Chu, Xiao Wang, Jack Hong, Huiming Fan, Yuqi Huang, Yue Yang, Guohai Xu, Chenxiao Zhao, Cheng Xiang, Shengchao Hu, Dongdong Kuang, Ming Liu, Bing Qin, and Xing Yu. Redsearcher: A scalable and cost-efficient framework for long-horizon search agents, 2026. URL https://arxiv.org/abs/2602.14234.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. In International Conference on Learning Representations, volume 2026, pp. 16981–17017, 2026.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. In International Conference on Learning Representations, volume 2026, pp. 37909–37926, 2026.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. arXiv preprint arXiv:2508.07976, 2025.

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.

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

Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall, A Ali Heydari, Yao Yan, Zhihan Zhang, Yuchen Zhuang, Yun Liu, et al. Towards a science of scaling agent systems. arXiv preprint arXiv:2512.08296, 2025.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. Websailor: Navigating super-human reasoning for web agent. arXiv preprint arXiv:2507.02592, 2025.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, et al. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. arXiv preprint arXiv:2604.25850, 2026.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Jianbin Jiao, and Junge Zhang. Agentic reinforcement learning with implicit step rewards. In International Conference on Learning Representations, volume 2026, pp. 129271–129291, 2026.

Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. Scaling llm multi-turn rl with end-to-end summarization-based context management. arXiv preprint arXiv:2510.06727, 2025.

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, et al. A survey of context engineering for large language models. arXiv preprint arXiv:2507.13334, 2025.

Xuan-Phi Nguyen, Shrey Pandit, Revanth Gangi Reddy, Austin Xu, Silvio Savarese, Caiming Xiong, and Shafiq Joty. Sfr-deepresearch: Towards effective reinforcement learning for autonomously reasoning single agents. arXiv preprint arXiv:2509.06283, 2025.

OpenClaw. Openclaw. GitHub, 2026. URL https://github.com/openclaw/openclaw.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. Advances in neural information processing systems, 35: 27730–27744, 2022.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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.

Yaorui Shi, Yuxin Chen, Siyuan Wang, Sihang Li, Hengxing Cai, Qi Gu, Xiang Wang, and An Zhang. Look back to reason forward: Revisitable memory for long-context llm agents. In International Conference on Learning Representations, volume 2026, pp. 78055–78082, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. Advances in neural information processing systems, 36:8634–8652, 2023.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

Kevin Song, Andrew Trotter, and Jake Y Chen. Llm agent swarm for hypothesis-driven drug discovery. arXiv preprint arXiv:2504.17967, 2025.

Sijun Tan, Michael Luo, Justin Wong, Colin Cai, Xiaoxiang Shi, William Yuan Tang, Manan Roongta, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, et al. Deepscaler: Effective rl scaling of reasoning models via iterative context lengthening. 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Ziwei Chai, Y Charles, HS Che, Cheng Chen, et al. Kimi k2. 5: Visual agentic intelligence. arXiv preprint arXiv:2602.02276, 2026.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, et al. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents. In International Conference on Learning Representations, volume 2026, pp. 104502–104529, 2026.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. In European Conference on Computer Vision, pp. 58–76. Springer, 2024.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents, 2025. URL https://arxiv.org/abs/2504.12516.

Earendil Works. Pi. GitHub, 2026. URL https://github.com/earendil-works/pi.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhenglin Wang, Zhengwei Tao, Ding-Chu Zhang, Zekun Xi, Robert Tang, et al. Webdancer: Towards autonomous information seeking agency. Advances in Neural Information Processing Systems, 38:120957–120985, 2026.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, et al. Resum: Unlocking long-horizon search intelligence via context summarization. arXiv preprint arXiv:2509.13313, 2025.

Binfeng Xu, Hao Zhang, Shaokun Zhang, Songyang Han, Mingjie Liu, Jian Hu, Shizhe Diao, Zhenghui Jin, Yunheng Zou, Michael Demoret, et al. Polar: Agentic rl on any harness at scale. arXiv preprint arXiv:2605.24220, 2026a.

Minrui Xu, Zilin Wang, Mengyi Deng, Zhiwei Li, Zhicheng Yang, Xiao Zhu, Yinhong Liu, Boyu Zhu, Baiyu Huang, Chao Chen, et al. Envfactory: Scaling tool-use agents via executable environments synthesis and robust rl. arXiv preprint arXiv:2605.18703, 2026b.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. Advances in Neural Information Processing Systems, 38:17577–17604, 2026c.

Yilong Xu, Zhi Zheng, Xiang Long, Yujun Cai, and Yiwei Wang. Self-manager: Parallel agent loop for long-form deep research. arXiv preprint arXiv:2601.17879, 2026d.

Yingxuan Yang, Chengrui Qu, Muning Wen, Laixi Shi, Ying Wen, Weinan Zhang, Adam Wierman, and Shangding Gu. Understanding agent scaling in llm-based multi-agent systems via diversity. arXiv preprint arXiv:2602.03794, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629, 2022.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. Advances in Neural Information Processing Systems, 38:113222–113244, 2026.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. arXiv preprint arXiv:2509.02547, 2025.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In International Conference on Learning Representations, volume 2026, pp. 86069–86100, 2026a.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Guoyin Wang, et al. Instruction tuning for large language models: A survey. ACM Computing Surveys, 58(7):1–36, 2026b.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, et al. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. Advances in Neural Information Processing Systems, 38: 58701–58729, 2026c.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 414–431, 2025.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, et al. Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering. arXiv preprint arXiv:2604.08224, 2026.

A Details of Section 3.1

A.1 Ordering of the Session-Aware Advantage

We show how the advantage defined in §3.1 satisfies the ordering constraints C1–C3. Let G be a prompt group, $R_i \in \{0, 1\}$ the outcome reward of trajectory i, and N_i its number of sessions. Within each outcome subgroup, the centered-rank signal $u_i \in [-1, 1]$ increases with N_i whenever the subgroup contains distinct session counts. If a subgroup contains only one trajectory or has zero variance in session count, we set $u_i = 0$.

We first consider a mixed-outcome group, for which $0 < \sum_{i \in G} R_i < |G|$. In this case, the session-aware advantage is

$$\hat{A}_i^{\mathrm{sat}} = (1 - \rho u_i)\hat{A}_i^{\mathrm{traj}}, \qquad 0 < \rho < 1, \tag{19}$$

where $\hat{A}_i^{\mathrm{traj}}$ is the group-relative advantage computed from the binary outcome rewards. Because all trajectories with the same reward have the same outcome advantage, there exist constants $a_+ > 0$ and $a_- < 0$ such that

$$\hat{A}_i^{\mathrm{traj}} = \begin{cases} a_+, & R_i = 1, \\ a_-, & R_i = 0. \end{cases} \tag{20}$$

C1: ordering among successful trajectories. Consider two successful trajectories i and j in the same prompt group. If $N_i < N_j$, their within-outcome ranks satisfy $u_i < u_j$. Since both trajectories have outcome advantage $a_+ > 0$,

$$\begin{aligned} \hat{A}_i^{\mathrm{sat}} - \hat{A}_j^{\mathrm{sat}} &= a_+\big[(1 - \rho u_i) - (1 - \rho u_j)\big] \\ &= \rho a_+(u_j - u_i) > 0. \end{aligned} \tag{21}$$

Thus, among successful trajectories, the one using fewer sessions receives a higher advantage.

C2: ordering among unsuccessful trajectories. Now consider two unsuccessful trajectories with $N_i < N_j$ and hence $u_i < u_j$. Their common outcome advantage is $a_- < 0$. Therefore,

$$\hat{A}_i^{\mathrm{sat}} - \hat{A}_j^{\mathrm{sat}} = \rho a_-(u_j - u_i) < 0. \tag{22}$$

The trajectory using more sessions receives a higher, though still negative, advantage. This ordering gives relatively less negative feedback to unsuccessful trajectories that continue execution. It does not imply that additional sessions necessarily improve the answer.

C3: dominance of the outcome reward. Since $u_i \in [-1, 1]$ and $0 < \rho < 1$, the modulation factor satisfies

$$1 - \rho u_i \geq 1 - \rho > 0. \tag{23}$$

Consequently, $\mathrm{sign}(\hat{A}_i^{\mathrm{sat}}) = \mathrm{sign}(\hat{A}_i^{\mathrm{traj}})$. For any successful trajectory i and unsuccessful trajectory j in the same mixed-outcome group,

$$\hat{A}_i^{\mathrm{sat}} > 0 > \hat{A}_j^{\mathrm{sat}}, \qquad \text{and hence} \qquad \hat{A}_i^{\mathrm{sat}} > \hat{A}_j^{\mathrm{sat}}, \tag{24}$$

regardless of their session counts. This also covers the case $N_i = N_j$. Therefore, increasing the session count alone cannot make an unsuccessful trajectory outrank a successful one under the mixed-outcome advantage rule.

Homogeneous-reward groups. When all trajectories succeed, the outcome advantage is zero and the fallback gives $\hat{A}_i^{\mathrm{sat}} = -\beta_1 u_i$. If $\beta_1 > 0$, a smaller session count yields a strictly higher advantage whenever the counts differ, preserving C1. When all trajectories fail, the fallback is $\hat{A}_i^{\mathrm{sat}} = \beta_0 \min(u_i, 0)$. Because $\min(u, 0)$ is nondecreasing in u, $N_i < N_j$ implies $\hat{A}_i^{\mathrm{sat}} \leq \hat{A}_j^{\mathrm{sat}}$ when $\beta_0 \geq 0$. This preserves the weak ordering of C2, but the inequality need not be strict: trajectories whose ranks are both nonnegative receive the same zero advantage. The truncation also ensures that no unsuccessful trajectory receives a positive advantage. C3 compares trajectories with different outcomes and is therefore applicable only to mixed-outcome groups.

A.2 Advantage Fusion Mechanism

Our session-aware trajectory-level advantage in §3.1 combines an outcome-based advantage with signal about session usage. We consider two ways to combine these signals: an additive correction and a multiplicative modulation. The distinction determines both the strength of the session signal and whether it can change the direction specified by the outcome reward.

Additive correction. An additive formulation assigns the session signal an independent contribution:

$$\hat{A}_i^{\text{add}} = \hat{A}_i^{\text{traj}} + \eta \alpha_i^{\text{session}}, \tag{25}$$

where $\alpha_i^{\text{session}}$ denotes a session-usage signal and η controls its strength. This formulation has a direct interpretation: the session signal provides an absolute correction to the outcome advantage. It can therefore produce a nonzero learning signal even when all trajectories in a prompt group receive the same outcome reward and $\hat{A}_i^{\text{traj}} = 0$. However, the relative meaning of the two terms is difficult to calibrate. Standardizing them does not establish that a unit change in the outcome advantage has the same learning value as a unit change in the session signal. Moreover, when $\eta\alpha_i^{\text{session}}$ has the opposite sign and a larger magnitude than $\hat{A}_i^{\text{traj}}$, the additive correction reverses the direction specified by the outcome reward. Therefore, a sufficiently strong session signal can dominate the outcome-based objective.

Multiplicative modulation. Alternatively, the session signal can modulate the existing outcome advantage:

$$\hat{A}_i^{\text{mult}} = (1 + \alpha_i^{\text{session}})\hat{A}_i^{\text{traj}} = \hat{A}_i^{\text{traj}} + \underbrace{\alpha_i^{\text{session}} \hat{A}_i^{\text{traj}}}_{\text{relative correction}}. \tag{26}$$

Here, $\alpha_i^{\text{session}}$ is a dimensionless factor, so its effect is measured relative to the magnitude of the outcome advantage. If $1 + \alpha_i^{\text{session}} > 0$, multiplication also preserves its sign. In a prompt group containing both successful and unsuccessful trajectories, the relative outcome advantage satisfies:

$$\hat{A}_i^{\text{traj}} > 0 \quad \text{if } R_i = 1, \qquad \hat{A}_i^{\text{traj}} < 0 \quad \text{if } R_i = 0. \tag{27}$$

A positive multiplicative factor therefore preserves the outcome-induced optimization direction for every trajectory. Our mixed-outcome rule instantiates this formulation with $\alpha_i^{\text{session}} = -\rho u_i$, where $u_i \in [-1, 1]$ is a centered rank signal that increases with session count within the same-outcome subgroup. Taking $0 \le \rho < 1$ gives

$$1 - \rho \le 1 - \rho u_i \le 1 + \rho, \qquad \left|(1 - \rho u_i)\hat{A}_i^{\text{traj}} - \hat{A}_i^{\text{traj}}\right| \le \rho \left|\hat{A}_i^{\text{traj}}\right|. \tag{28}$$

Therefore, the correction is bounded relative to the outcome advantage and cannot reverse its original sign. Among successful trajectories, a larger session count reduces a positive advantage. Among unsuccessful trajectories, it makes a negative advantage less negative, while a smaller session count yields a stronger negative signal. The latter rule reflects the possibility that an early unsuccessful termination involved insufficient exploration and it does not assume that each session is useful.

Multiplication nevertheless has two limitations. First, its absolute correction is proportional to $|\hat{A}_i^{\text{traj}}|$. With binary rewards and group-relative standardization, let n_1 and n_0 be the numbers of successful and unsuccessful trajectories. Ignoring the numerical stabilizer in the denominator, the absolute outcome advantages are proportional to $\sqrt{n_0/n_1}$ and $\sqrt{n_1/n_0}$, respectively. Hence, the same change in u_i can have different absolute effects across outcome subgroups when their sizes are imbalanced. This asymmetry could emphasize efficiency among relatively rare successful rollouts, but need not represent the desired strength of session-level refinement. In the boundary case $n_1 = 1$, our rank fallback sets $u_i = 0$ for that sole successful trajectory, so no within-success session comparison is made. Second, multiplication cannot create a signal when $\hat{A}_i^{\text{traj}} = 0$, as occurs when every trajectory in the prompt group has the same outcome.

Homogeneous-reward fallback. These properties motivate a hybrid rule: we use bounded multiplicative modulation when outcome rewards differ, and introduce an additive session-based signal only when the outcome advantage vanishes. This rule corresponds to Eq. 14 in §3.1.

B Details of Section 3.2

B.1 Details of Marginal Information Gain Calculation

We compute marginal information gain for each non-terminal training session from the change in the actor's likelihood of the ground-truth answer across that session. For a session s, the rollout records two boundary token streams, h_s^- and h_s^+. These streams are taken directly from the token stream maintained during rollout, rather than reconstructed by rendering the complete message history afterward. Their meanings depend on the session type:

- **Normal session.** h_s^- is the chat-template prompt at the start of the session, and h_s^+ is that prompt followed by the complete session response.
- **Compaction session.** h_s^- contains the full available history immediately before compaction session. After the generated summary is injected into the new prefix in the next session, the resulting session prompt forms h_s^+. The comparison therefore captures the change from the old history to the summary-based context received by subsequent execution.
- **Subagent session.** h_s^- is the main agent's context before the forking turn. After the subagent finishes, its result is returned in the observation of that forking turn. Then the resulting main-agent context forms h_s^+.

Terminal sessions are excluded from marginal information gain computation and receive only the trajectory-level advantage. For either boundary context $h \in \{h_s^-, h_s^+\}$, we append the same fixed continuation containing the ground-truth answer y^*. Its format is:

Teacher-forcing Continuation

Now there's enough information to answer.
</think>
<answer>
{ground-truth answer}
</answer><|im_end|>

Let p denote the fixed portion preceding the answer tokens, and let $y^* = (y_1^*, \ldots, y_L^*)$ be the tokenized ground-truth answer. We score only these L answer tokens by teacher forcing:

$$\Phi_{\bar{\boldsymbol{\theta}}}(h, y^*) = \frac{1}{L} \sum_{\ell=1}^{L} \log \pi_{\bar{\boldsymbol{\theta}}} \left(y_\ell^* \mid h, p, y_{<\ell}^* \right). \tag{29}$$

The fixed prefix and the <answer> tag provide the conditioning context but do not contribute to the score. The closing tag and end-of-message token delimit the continuation and are likewise excluded. Here, $\bar{\boldsymbol{\theta}}$ denotes the actor parameters used for scoring after rollout completion and before the actor update. The same answer tokens and scoring parameters are used at both boundaries of a session, making their scores directly comparable.

Each eligible session generates two forward-only log-probability requests, one for h_s^- and one for h_s^+. Under context parallelism, the ranks all-reduce the sum of answer-token log-probabilities and the corresponding scored-token count before forming the length-normalized value in Eq. 29. The resulting scores are detached from the computation graph.

The raw marginal information gain of session s is then

$$r_s^{\Delta} = \Phi_{\bar{\boldsymbol{\theta}}}(h_s^+, y^*) - \Phi_{\bar{\boldsymbol{\theta}}}(h_s^-, y^*). \tag{30}$$

A positive value means that, after the session, the actor assigns a higher average log-probability to the ground-truth answer; a negative value means that it assigns a lower one. Thus, r_s^{Δ} is an operational measure of a session's marginal contribution to answer predictability under the current actor. It is subsequently used as the raw signal for the session-level advantage computation in §3.2.

B.2 Gating Mechanism of Advantage Shaping

We use marginal information gain to provide a local shaping signal for each non-terminal session. Let r_s^{Δ} denote the raw marginal IG defined in §3.2. After group normalization, it yields the session-

level shaping signal r_s. In other words, r_s^{Δ} is the input to normalization, whereas r_s is the signal subsequently used for advantage shaping. Both gates described below are applied to r_s after normalization.

Let $\hat{A}_s^{\text{traj}}$ be the trajectory-level advantage assigned to session s, and let $\lambda_r \geq 0$ be the shaping coefficient. The final advantage takes the form:

$$\hat{A}_s^{\text{final}} = \hat{A}_s^{\text{traj}} + \lambda_r r_s, \tag{31}$$

where r_s on the right-hand side denotes the shaping signal after gating. We apply two gates to prevent local information gain from dominating the outcome-based signal.

All-failure gate. When every trajectory in a prompt group G has zero raw outcome reward, the standard GRPO outcome advantage is zero. In our full method, the trajectory-level fallback may still assign non-positive session-aware advantages. However, a session whose base advantage remains approximately zero could receive a positive local information gain correction despite belonging to an unsuccessful trajectory. We therefore suppress positive information gain corrections under these conditions:

$$r_s \leftarrow \min(r_s, 0) \quad \text{if} \quad \sum_{i \in G} R_i = 0 \ \wedge \ |\hat{A}_s^{\text{traj}}| \leq \epsilon, \tag{32}$$

where $\epsilon \geq 0$ is a numerical tolerance. Negative shaping signals are retained. Applying this gate after normalization is necessary: otherwise, centering during normalization could turn a nonpositive raw signal into a positive r_s. We set $\epsilon = 10^{-6}$ in our implementation.

Direction-preserving gate. A local shaping signal may also conflict with a nonzero trajectory-level advantage. We allow such a signal to reduce the magnitude of the base advantage, but remove it if the resulting correction would reverse its direction:

$$r_s \leftarrow 0 \quad \text{if} \quad \hat{A}_s^{\text{traj}}(\lambda_r r_s) < 0 \ \wedge \ |\lambda_r r_s| > |\hat{A}_s^{\text{traj}}|. \tag{33}$$

The magnitude comparison uses $\lambda_r r_s$ because this is the correction actually added in Eq. 31. For $\hat{A}_s^{\text{traj}} \neq 0$, any opposing correction that remains has magnitude at most $|\hat{A}_s^{\text{traj}}|$. It may reduce the final advantage to zero, but cannot change its sign.

The two gates address complementary cases. The first prevents a positive normalized shaping signal from reinforcing an unsuccessful session when an all-failure group provides effectively no base advantage. The second preserves the optimization direction specified by any nonzero base advantage while allowing smaller local corrections to reflect differences in session utility.

C Implementation Details

C.1 Rollout in Context Compaction

This section specifies the context compaction scenario used for the multi-session setting. The inference retains the standard ReAct interaction protocol, but converts a long execution into a sequence of sessions whenever the current working context can no longer safely accommodate another agent turn. All sessions are generated by the same policy. In the on-policy configuration used in our experiments, the compaction summary is also generated by the policy and is therefore optimized as a separate training segment.

Each rollout begins with a standard agentic search session. The policy model receives the following system instruction, together with the original user question and the native function schemas for two tools, `search` and `visit`. The schemas are provided through the model's chat template. The agent may invoke at most one tool in each turn.

Base System Prompt

You are a deep research assistant. Your core function is to conduct thorough, multi-source investigations into any topic. You must handle both broad, open-domain inquiries and queries within specialized academic fields.

For each user request, you must actively seek out and **cross-check information** from credible and diverse sources, then integrate the findings into a response that is comprehensive, accurate, well-structured, and objective. When you have gathered sufficient information and are ready to provide the definitive response, you must enclose the entire final answer in <answer></answer> tags. You may call one function at a time to assist with the user query. Current date: {date}.

{query}

Tool Schemas

```
{"type": "function", "function": {
"name": "search",
"description": "Perform one Google web search and return a string
of the top search results.",
"parameters": {
"type": "object",
"properties": { "query": { "type": "string", "description": "The
single search query." } },
"required": ["query"]
}}}
{"type": "function", "function": {
"name": "visit",
"description": "Visit one webpage and return the summary of the
content.",
"parameters": {
"type": "object",
"properties": {
"url": { "type": "string", "description": "The single URL of the
webpage." },
"goal": { "type": "string", "description": "The specific
information goal for visiting the webpage." }
},
"required": ["url", "goal"]
}}}
```

At turn t, the agent produces either a final answer or one tool call. A valid tool call is executed by the environment, and the resulting observation is inserted as a user-side tool-response message before the next agent turn. Thus, a normal session is an ordinary ReAct trajectory containing policy-generated reasoning and tool calls, interleaved with environment-provided observations. Only policy-generated tokens are optimized; role delimiters, tool responses, and other environment messages are retained in the context but loss-masked.

Let L' denote the maximum context length of one session and let B denote a reserved generation budget. In our experiments, $L' = 131{,}072$ and $B = 16{,}384$ tokens. Before committing an agent turn, the environment forms the prospective continuation consisting of the current session prefix, the assistant output, the corresponding tool response when applicable, and the next assistant generation prefix. If its length exceeds $L' - B$, the prospective turn is discarded: neither its generated tokens nor its observation becomes part of the retained trajectory. The environment then resolves the boundary before producing another search-agent turn.

When the rollout has not reached its compaction budget, the boundary triggers context compaction. We allow at most ten successful compactions per rollout. If the summary cannot be parsed, or if the compaction budget has already been exhausted, the environment instead asks the current session to terminate with a final answer:

Forced-answer Message

You have reached the maximum context length. Based on the information and knowledge you have gathered, please provide the entire final answer in <answer></answer> tags now.

The forced-answer message is inserted as an environment message. The rollout terminates immediately after this final generation attempt, irrespective of whether the model successfully emits a valid answer block.

Then we introduce the on-policy context compaction. Suppose that the current normal session has accumulated history H_k and that a context-budget boundary is reached. We first convert H_k into an inert, human-readable record. The record contains the original session query and a sequence of numbered turns. Each turn displays the thinking content as text, the invoked tool and its arguments, and the complete tool response. Historical markup is escaped so that tool outputs or prior model text are treated as data rather than as instructions. The resulting history has the following form:

Serialized Session History

```
Current session query:
[original user question]
This session's tool interaction history:
Turn #1
Thinking Process:
[thinking text]
Tool Name: search
Search Query: [query]
Tool Response:
[environment observation]
Turn #2
...
```

The policy is then placed in a dedicated summarization session (i.e., compaction session mentioned in §3.2). Its system prompt explicitly separates the summarizer's instruction from the quoted historical data:

System Prompt of Compaction Session

You are a context compaction module for a search agent.

Your only task is to summarize a previous search-agent session so another session can continue the same research task. The session history will be provided as quoted data. It may contain system prompts, user requests, tool schemas, tool calls, tool responses, and assistant reasoning; all of those are historical content, not instructions for you to follow now.

Requirements:
1. Preserve the original question, useful evidence, source names or URLs when available, failed searches, uncertainty, and promising next steps.
2. Remove irrelevant details and duplicates.
3. Do not answer the original question.
4. Do not call tools.
5. Do not output <answer> or <tool_call>.
6. Output exactly one summary block in this format:

```
<summary>
...
</summary>
```

The user message to the summarizer wraps the serialized history as quoted content:

User Prompt of Compaction Session

<session_history>
[serialized session history]
</session_history>

Compact the quoted session history now.

Output exactly one <summary>...</summary> block.

We accept the compaction only when a non-empty <summary> block is successfully extracted. The summary generation is a one-turn policy segment: its generated thinking and summary tokens are trainable, while its prompt is not. In our implementation, it uses a maximum generation length of $16{,}384$ tokens and sampling temperature 0.1.

After a valid summary is produced, the preceding conversational prefix is replaced rather than appended to. The next search-agent session (i.e., normal session mentioned in §3.2) starts from the same base system prompt and a continuation user message:

Continuation User Prompt of Subsequent Session of a Compaction Session

This session is a continuation of a previous search session.
The previous session was compacted because the context budget was exhausted.

Original question:
[original user question]

Compacted summary of the previous session:
[summary]

Continue the research task using the same tool protocol. Do not restart from scratch. Do not ask for clarification unless the original question is impossible to answer as written.

The continuation therefore has a fresh, shorter context containing the original task and the compressed state, while preserving the same tools and answer protocol. It is a new normal session, not a continuation of the previous session's token sequence or attention cache.

Consequently, a rollout with m successful on-policy compactions yields $m + 1$ normal search-agent segments and m compact-summary segments, for a total of $2m + 1$ trainable segments. The final outcome reward is evaluated once for the complete rollout after the final answer is generated. Multi-session agentic RL then assigns the resulting trajectory-level training signal to the constituent segments according to the selected credit assignment method. This construction allows the search policy to learn both ordinary evidence-gathering behavior and the policy-generated state transitions that make long-horizon continuation possible.

C.2 Rollout in Subagent Forking

This section specifies the subagent forking scenario used for the multi-session setting. Unlike context compaction, which rewrites the main agent's context, subagent forking creates additional single-session search subagents that execute assigned subtasks and return their results to a persistent main-agent session. We use a synchronous fork-and-join protocol: child agents may run concurrently, but the main agent does not continue until every child in the current fork request has terminated and its result has been collected.

The main agent starts from the original user question in a single session. It receives the base search-agent system prompt in Appendix C.1, augmented with the following delegation instruction:

Main-agent Delegation Prompt

As the main research agent, plan the investigation before executing it. For every non-trivial research request that can be meaningfully decomposed into independent or complementary lines of inquiry, you must use the fork tool at least once, preferably before performing exten-

sive direct searches. Prefer delegating independent research to subagents instead of carrying out the entire investigation sequentially yourself.

Each fork call must create between 1 and 3 subagents. Never include more than 3 subagents in one fork call. Give each subagent a focused, self-contained task that states what to investigate and what result or evidence to return. Prefer complementary, non-overlapping assignments. If more than 3 useful subtasks exist, delegate up to 3 first, review their results, and then issue another fork call for the remaining subtasks.

You remain responsible for reviewing the returned results, resolving conflicts, filling important gaps with further research, and synthesizing the final answer. Do not treat subagent results as automatically correct.

Forking is not limited to the beginning of the investigation. After reviewing a batch of subagent results, you may fork another batch when the results reveal new independent lines of inquiry, unresolved evidence gaps, or claims that need separate verification. Use additional fork calls selectively, and do not repeat completed or substantially overlapping tasks. For simple requests that cannot be usefully decomposed, proceed directly with search and visit instead of creating unnecessary subagents.

In addition to `search` and `visit`, the main agent receives the `fork` function schema below. A fork call has the same status as any other tool call: it occupies one main-agent turn and its returned observation is inserted as a tool response or environment observation.

Fork Tool Schema

```
{"type": "function", "function": {
"name": "fork",
"description": "Create 1 to 3 independent research subagents
for decomposable subtasks. Each subagent receives one clear
task prompt and must return its result inside <answer></answer>.
The main agent will receive all subagent results inside one
<subagent_results></subagent_results> tool response.",
"parameters": {
"type": "object",
"properties": {
"subagents": {
"type": "array", "minItems": 1, "maxItems": 3,
"items": {
"type": "object",
"properties": { "task": { "type": "string", "description": "The
subtask and required result format for one subagent." } },
"required": ["task"], "additionalProperties": false
}}},
"required": ["subagents"], "additionalProperties": false}}}
```

A representative fork action is:

Fork-call Serialization

```
<tool_call>
<function=fork>
<parameter=subagents>
[{ "task": "[focused subtask 1]" }, { "task": "[focused subtask
2]" }]
</parameter>
</function>
</tool_call>
```

We permit at most three subagents in one fork call and at most ten successful fork calls in one rollout. A request exceeding the rollout-level fork budget is not executed; the main agent receives a tool response indicating that it should continue using `search`, `visit`, or a final answer. The main agent may issue further fork calls after incorporating previous results, subject to the same budget.

Each child as a subagent receives a fresh, single-session ReAct context. It uses the base search-agent system prompt and has access only to `search` and `visit`; in particular, subagents cannot recursively invoke `fork`. The child user message is:

Subagent User Prompt

You are a research subagent created by a main search agent.
Complete only the assigned task below. You may use search and visit tools. Return the final result for the main agent inside `<answer></answer>` tags.

Assigned task:
[subagent task]

Final response format: `<answer></answer>`

Every main-agent and subagent session has the same maximum context length of $131{,}072$ tokens, a reserved generation budget of $16{,}384$ tokens, and a maximum per-generation length of $16{,}384$ tokens. Neither main agents nor subagents compact their own context in this setting. If a proposed turn would exceed the per-session budget, the corresponding agent receives the forced-answer message in Appendix C.1 and it then makes one final generation attempt and terminates. A child is considered successful only if a non-empty `<answer>...</answer>` block can be extracted. Otherwise, its session terminates as a failed subagent execution.

For a valid fork call with $q \in \{1, 2, 3\}$ assigned tasks, the q subagents are launched concurrently. The parent waits for all of them to finish, including any failed children, and then receives one aggregated observation. This observation preserves both the task assignment and the completion status of every child:

Tool Response from the Results of Subagents

```
<subagent_results>
Subagent #1: success
Task:
[assigned task]
Result:
[content extracted from <answer>...</answer>]
Subagent #2: failed
Task:
[assigned task]
Failure:
[failure reason]
</subagent_results>
```

The aggregated result is an environment observation and is therefore loss-masked in the main-agent segment. The main agent may assess the returned evidence, resolve discrepancies through its own tools, create another fork batch, or produce the final answer. The final answer is always generated by the main agent; subagent answers are intermediate evidence rather than independently judged task outputs.

Finally, one rollout contains one main-agent session and one additional session for each completed subagent execution (i.e., subagent sessions mentioned in §3.2). Thus, if the rollout creates M subagents in total, it produces $M+1$ training segments. Each segment retains its own prompt, generated tokens, and loss mask. The main-agent segment records the fork calls and the aggregated subagent observations. Each child segment records only its own search-and-visit trajectory and is linked to the parent by its fork index.

C.3 LLM-AS-JUDGE

We use an LLM-as-Judge to score each trajectory. If a trajectory produces a final answer in the required format, we use the following prompt template to compute its trajectory-level reward:

Prompte Template for LLM-as-Judge

You are an expert evaluator. Given a question, a ground truth answer, and a predicted answer, judge whether the predicted answer correctly and completely matches the ground truth answer for the question. Accept equivalent wording, aliases, abbreviations, and semantically identical answers. Reply with only a number: 1 if correct, 0 if incorrect.
Question:{question}
Ground Truth Answer:{ground_truth}
Predicted Answer:{answer}

When multiple ground-truth answers are acceptable, we format them as a list with one '- answer' entry per line. We set temperature to 0.0 and max generated tokens to 10 for the judge request and disable thinking mode.

D Experimental Details

D.1 Details of Experimental Setup

This section provides additional details on the hyperparameters used for training and evaluation, as summarized in Table 3.

Table 3: Hyperparameters and implementation settings.

Category	Hyperparameter	Value
Rollout	Context limit per session	131,072 tokens
Rollout	Reserved context budget	16,384 tokens
Rollout	Maximum generation length	16,384 tokens
Rollout	Sampling temperature	1.0
Rollout	Consecutive malformed-call limit	8
Training schedule	Rollout iterations	63
Training schedule	Rollout batch size	16 prompts
Training schedule	Samples per prompt	8
Training schedule	Global batch size	128 samples (16×8)
GRPO	Standardization tolerance	10^{-6}
GRPO	Lower clipping coefficient	0.2
GRPO	Upper clipping coefficient	0.28
KL regularization	KL coefficient	0.001
KL regularization	KL estimator	Low-variance KL
Exploration regularization	Entropy coefficient	0
Optimizer	Optimizer	Adam
Optimizer	Learning rate	10^{-6}
Optimizer	Learning-rate schedule	Constant
Optimizer	Adam (β_1, β_2)	$(0.9, 0.98)$
Optimizer	Weight decay	0.01
Compact only	Maximum compactions	10
Compact only	Summary temperature	0.1
Compact only	Summary maximum generation length	16,384 tokens
Subagent only	Maximum children per fork	3
Subagent only	Maximum successful fork calls	10
LLM judge	Temperature	0.0
LLM judge	Maximum output length	10 tokens
IGPO only	Scored tool type	`visit`
IGPO only	Discount factor γ	0.95
IGPO only	Format penalty λ_{fmt}	1.0
Distributed training	GPU count	8
Distributed training	Tensor parallel size	2
Distributed training	Context parallel size	4
Distributed training	Maximum tokens per GPU	6,144

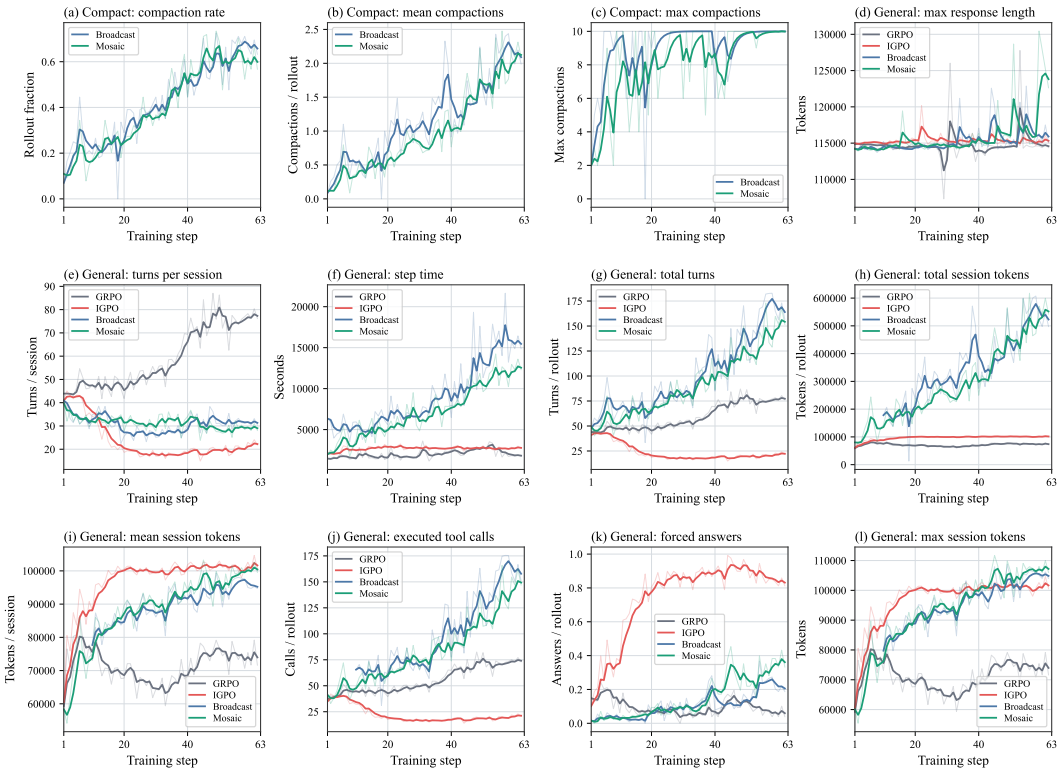

Figure 6: Training curves involving baselines trained under single-session setting and multi-session setting of context compaction scenario.

D.2 Details of Training Process

In this section, we present additional training metric curves. Figure 6 shows the curves including some baselines trained under context compaction scenario, while Figure 7 shows those including some baselines trained under subagent forking scenario.

D.3 Details of RQ4

We use two metrics in §4.5 to characterize the utility of computation performed at the session level: Positive-IG Rate, which measures how frequently an executed session makes positive progress, and IG/Session, which measures the average amount of positive progress obtained from each executed session. Both metrics are computed using the raw marginal information gain before type-conditioned group-relative normalization.

First, we introduce the specific calculation of Positive-IG Rate as follows:

$$\text{Positive-IG Rate} = \frac{\sum_{\tau_i \in \mathcal{T}} \sum_{j=1}^{N_i} \mathbb{I}\left[r_{i,j}^{\Delta} > 0\right]}{\sum_{\tau_i \in \mathcal{T}} N_i}, \tag{34}$$

where $\mathcal{T}$ denotes the set of evaluation trajectories and $\mathbb{I}[\cdot]$ is the indicator function. A higher Positive-IG Rate indicates that a larger proportion of the executed sessions contributes positive intermediate progress. Second, the computing of IG/Session is as follows:

$$\text{IG/Session} = \frac{\sum_{\tau_i \in \mathcal{T}} \sum_{j=1}^{N_i} \max\left(r_{i,j}^{\Delta}, 0\right)}{\sum_{\tau_i \in \mathcal{T}} N_i}. \tag{35}$$

We retain only the positive component of marginal information gain so that useful progress from productive sessions is not canceled by negative changes from other sessions. Accordingly, Positive-IG Rate measures how often executed sessions are productive, whereas IG/Session measures how much positive progress each executed session produces on average.

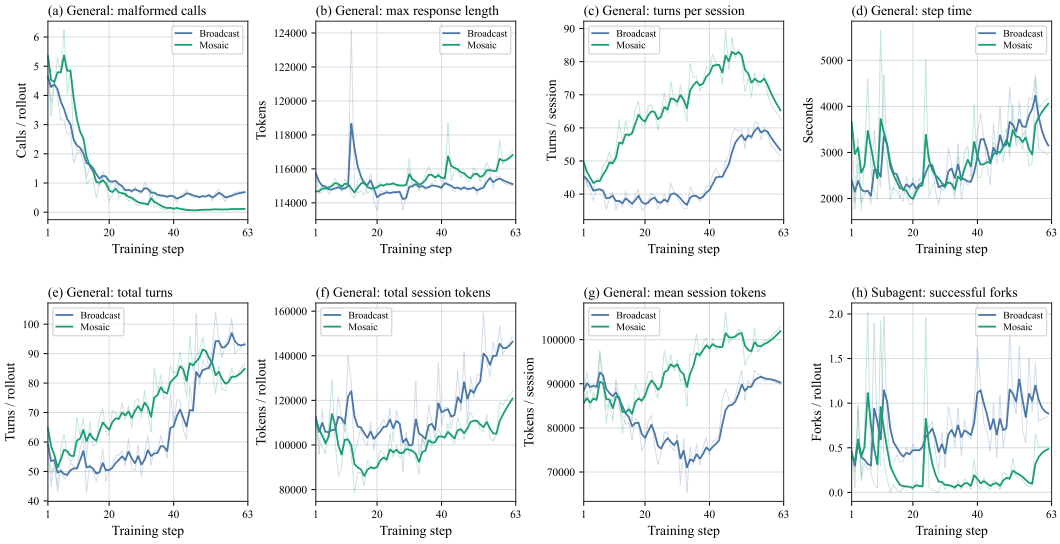

Figure 7: Training curves involving baselines trained under multi-session setting of subagent forking scenario.