CREDIT FOLLOWS USE: USE-GROUPED POLICY OPTIMIZATION FOR LONG-CONTEXT MEMORY AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context memory agents have emerged as a promising paradigm for retaining information across document streams and multi-session interactions. However, optimizing these agents poses a fundamental credit-assignment challenge: the consequences of a memory operation may remain unobserved until its content is retrieved and grounded in a distant response. Existing approaches typically either propagate a trajectory-level reward to all operations or derive process-level supervision from gold evidence and branched rollouts. We argue that both paradigms overlook an observable signal intrinsic to external memory systems—each materialized memory item exhibits a traceable lifecycle from writing, through retrieval, to grounded use in downstream generation. Inspired by this observation, we propose Use-Grouped Policy Optimization (UGPO), which records Write–Retrieve–Use (WRU) traces and organizes policy updates around recurring item-use events rather than complete memory states. By modeling this lifecycle, UGPO routes task, use, and cost credit to responsible operations and propagates delayed outcomes to their originating writes. We further show that *exact, high-specificity state grouping* loses comparison coverage as free-form writes accumulate, while item-use groups remain comparable across diverged memory states. Extensive experiments on long-document and multi-session benchmarks demonstrate that UGPO consistently improves task performance and attribution quality while reducing memory pollution and serving cost. [1]

1 INTRODUCTION

Long-context memory agents have emerged as a promising paradigm for retaining information across document streams and multi-session interactions Packer et al. (2024); Park et al. (2023); Zhong et al. (2024). The key idea is to externalize interaction history into a persistent memory store, from which a compact working set can be retrieved for subsequent requests Packer et al. (2024); Park et al. (2023); Zhong et al. (2024). This paradigm supports long-document reasoning and multi-session interaction by enabling an agent to revisit earlier evidence and reuse accumulated experience over extended horizons Packer et al. (2024); Wu et al. (2025); Maharana et al. (2024).

Recent studies take a step further by replacing fixed storage heuristics with learnable policies that determine which observations should be written, revised, removed, or retrieved Yan et al. (2025); Yu et al. (2025); Zhang et al. (2026). Optimizing these memory decisions, however, gives rise to a fundamental long-horizon credit-assignment problem . A memory operation is executed at the current step, whereas its utility may remain unobserved until a distant request retrieves and uses the corresponding item Yan et al. (2025; 2026b); Ma et al. (2026). The learning objective must therefore distinguish useful operations from writes that only expand the store and retrievals that introduce irrelevant context Yan et al. (2025); Ma et al. (2026); Zhang et al. (2026). Otherwise, a successful response may reinforce unused memories, while a later reasoning failure may incorrectly penalize an earlier operation that retained essential evidence Yan et al. (2025); Ma et al. (2026); Zhang et al. (2026).

Current memory credit-assignment methods can be broadly categorized into trajectory-level optimization and process-level supervision Shao et al. (2024); Ma et al. (2026); Yan et al. (2026b).

[1] All the source code and datasets are available at `https://anonymous.4open.science/status/credit-26D8`.

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.

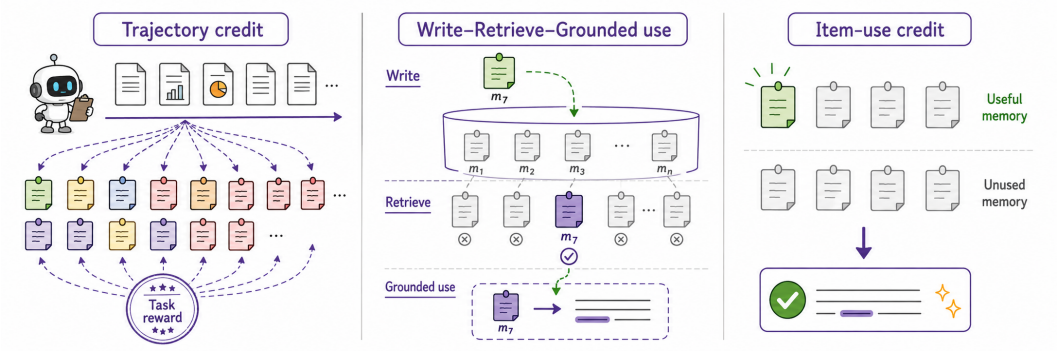


Figure 1: **Credit assignment through observable memory use.** Trajectory-level rewards propagate a single outcome to every write, including memories that do not contribute to the response. A memory service naturally records the items that are written, retrieved, and grounded in generation. UGPO routes learning signals along this observable lifecycle, distinguishes retrieval inefficiency from answer failure, and jointly optimizes memory quality and serving cost.

Trajectory-level methods, represented by GRPO, estimate a group-relative return over complete rollouts and assign the resulting signal to all generated tokens Shao et al. (2024). Although this formulation maintains an end-to-end learning objective, it provides limited temporal resolution when a trajectory contains many operations unrelated to the final request Shao et al. (2024); Yan et al. (2026b). This limitation is particularly pronounced in persistent-memory settings, because each free-form write modifies the inherited store and increases the number of heterogeneous operations coupled to the same scalar outcome Shao et al. (2024); Yan et al. (2026b).

Process-level methods seek to provide more fine-grained supervision through intermediate evidence or additional execution Ma et al. (2026); Tao et al. (2026); Yan et al. (2026b). While these approaches improve attribution granularity, they rely on annotated evidence, auxiliary supervision, counterfactual comparisons, or additional rollout branches Ma et al. (2026); Yan et al. (2026b); Yue et al. (2026); Yan et al. (2026a). Such requirements limit their applicability to long-running memory services, where historical interactions are costly to reproduce and naturally occurring requests seldom provide gold evidence Ma et al. (2026); Yan et al. (2026b); Yue et al. (2026); Yan et al. (2026a).

The above analysis suggests that effective memory attribution requires a supervision unit that is both fine-grained and directly observable from the original interaction Ma et al. (2026); Tao et al. (2026); Yan et al. (2026b). External memory provides such a structure through the lifecycle of each materialized item. A write produces a stable identifier, retrieval returns a ranked set of identifiers, and response generation records which retrieved items are grounded in the answer Yan et al. (2025). Collectively, as illustrated in Figure 1, these events form a *Write–Retrieve–Use trace*.

Inspired by this observation, we propose Use-Grouped Policy Optimization (UGPO), a group-relative optimization framework for long-context memory agents. UGPO records stable item provenance within the original rollout and constructs comparison groups from compatible item-use events rather than complete memory states. It propagates task credit along grounded-use paths, reinforces supported consumption through a use channel, and penalizes redundant storage and retrieval through a cost channel. Since these rewards exhibit different sparsity and scales, UGPO normalizes them independently before aggregation. Moreover, delayed-use replay associates future consumption events with their originating writes, providing cross-session supervision without reproducing the intervening interaction history.

Taking a step further, we identify a structural limitation of state-based local comparison Feng et al. (2025). Free-form memory generation causes trajectories initialized from the same input to rapidly evolve toward distinct memory stores. Consequently, exact, high-specificity state recurrence becomes increasingly sparse as the interaction horizon grows, causing those local groups to degenerate into singletons and eliminating relative learning signals. This coverage decay is a property of exact (high-specificity) state grouping, not of every state-relative comparison. By grouping operations according to shared item-use events, UGPO preserves comparability even after unrelated

memory contents have diverged. We establish an extensive evaluation protocol covering mainstream datasets, together with representative memory-learning and credit-assignment baselines Yang et al. (2018); Ho et al. (2020); Maharana et al. (2024); Wu et al. (2025); Shridhar et al. (2021); Yao et al. (2023a). The protocol evaluates task performance, attribution precision, group coverage, memory pollution, serving latency, delayed credit, and robustness under varying context and retrieval scales. Controlled ablations further examine grounded use, channel-wise normalization, responsibility routing, delayed-use replay, and item-use grouping. Our key contributions are summarized as follows:

- We formulate long-horizon memory attribution through WRU traces and show that exact, high-specificity state grouping loses resolution under free-form memory updates.
- We propose UGPO, which integrates item-use grouping, responsibility routing, channel-wise normalization, and delayed-use replay without requiring gold evidence labels or additional rollout branches.
- We establish an evaluation protocol across long-document, multi-session, and agentic settings, incorporating task, attribution, robustness, and systems measurements for the proposed framework.

2 Problem Formulation

Memory-Agent Formulation. Following learned memory-control agents Yan et al. (2025); Zhang et al. (2026), we consider a policy π_θ interacting for T steps with an external memory store $\mathcal{M}_t$. At step t, the state is $s_t = (x_t, S_t, \mathcal{M}_{t-1})$, where x_t is the observation and S_t the retrieved items. The policy and memory transition follow

$$(y_t, o_t, v_t) \sim \pi_\theta(\cdot \mid s_t), \qquad \mathcal{M}_t = F(\mathcal{M}_{t-1}, o_t, v_t), \tag{1}$$

where $o_t \in \{\text{ADD}, \text{UPDATE}, \text{DELETE}, \text{NOOP}\}$ and v_t denotes optional memory content. Each materialized item receives a stable identifier i; m_i denotes its current content and $w_{t,i}$ its write event.

Trajectory-Level Credit Assignment. Following group-relative outcome optimization Shao et al. (2024), a trajectory $\tau = \{(s_t, y_t, o_t, v_t)\}_{t=1}^{T}$ receives terminal reward $R_{\text{task}}(\tau)$. For a write $w_{t,i}$, the trajectory-level advantage is

$$\widehat{A}^{\text{traj}}_{t,i} = \frac{R_{\text{task}}(\tau) - \mu_{\mathcal{G}(\tau)}}{\sigma_{\mathcal{G}(\tau)}}, \tag{2}$$

where $\mu_{\mathcal{G}(\tau)}$ and $\sigma_{\mathcal{G}(\tau)}$ are computed over the same-input rollout group $\mathcal{G}(\tau)$. Thus, all writes in a trajectory receive the same credit regardless of their downstream use.

Exact-State Grouping. State-relative methods instead compare actions from recurrent states Feng et al. (2025). When the comparison key $h(\cdot)$ is an *exact, high-specificity* encoding of the inherited store, usable comparison coverage is

$$C_{\text{state}}(\mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \mathbb{I}[|\{b' \in \mathcal{B} : h(s_{b'}) = h(s_b)\}| > 1], \tag{3}$$

where $h(s_b)$ equals the full memory-store identifier of trajectory b, not a coarse environment abstraction. For free-form memory updates, assume that two matching stores remain identical after one write with conditional probability at most $\gamma < 1$. After n writes,

$$\mathbb{E}[C_{\text{state}}(\mathcal{B}, n)] \le \min\{1, (|\mathcal{B}| - 1)\gamma^n\}. \tag{4}$$

The proof is given in Appendix B. Hence, coverage under exact, high-specificity state grouping decays exponentially with the write horizon. The bound does not apply to coarser keys that ignore store contents. Diverged stores may still contain items serving comparable downstream uses, motivating comparison at the item-use level rather than under exact-store equality.

3 Use-Grouped Policy Optimization

As illustrated in Figure 2, UGPO converts the observable lifecycle of each memory item into fine-grained policy supervision. Specifically, we first record Write–Retrieve–Use (WRU) events during

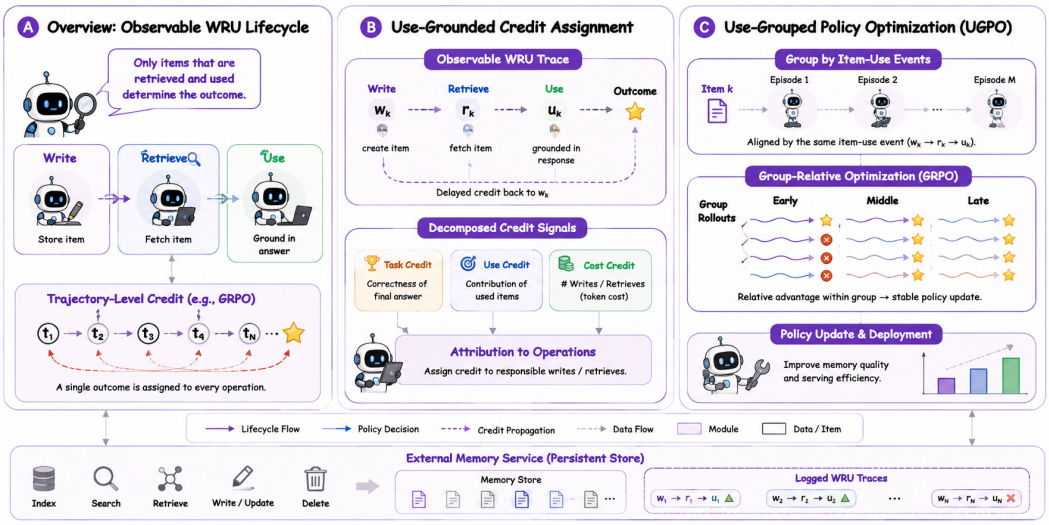


Figure 2: **Overview of UGPO.** The original rollout emits an instrumented WRU trace linking stable write identifiers to retrieved candidates, grounded citations, and task outcomes. The training-only path constructs task, use, and cost credit, normalizes each channel independently, and applies the resulting item-use advantage to the responsible policy tokens.

the original rollout and identify grounded downstream use. Based on these events, responsibility routing assigns credit to the corresponding memory operations. We then construct item-use groups to obtain operation-level relative advantages, which are finally integrated across multiple reward channels for policy optimization.

3.1 Write–Retrieve–Use Traces

To explicitly track the lifecycle of each materialized memory item, we represent every observed event as

$$e = (\texttt{type}, i, t, q, p, \ell), \quad \texttt{type} \in \{\textsc{write}, \textsc{retrieve}, \textsc{cite}, \textsc{outcome}\}, \tag{5}$$

where i is the stable item identifier, t is the step index, $q \in \mathcal{Q}(\tau)$ is a downstream request, p is an optional retrieval rank, and ℓ denotes the predecessor identifier for an update. Equation 5 therefore preserves the provenance of each item from its originating write to subsequent retrieval and use, while remaining service-side metadata outside the model context.

Based on the lifecycle events in Equation 5, we further distinguish actual memory use from retrieval exposure. Since a retrieved set may contain unused distractors, we define the grounded-use indicator as

$$u_{q,i} = \mathbb{I}[i \in S_q]\,\mathbb{I}[i \in C_q]\,\mathbb{I}[g(m_i, y_q) = 1], \tag{6}$$

where S_q and $C_q \subseteq S_q$ denote the retrieved and cited identifier sets, respectively, and $g(m_i, y_q) \in \{0, 1\}$ determines whether memory item m_i supports the cited span in answer y_q. According to Equation 6, an item is regarded as used only when it is simultaneously retrieved, cited, and grounded. We implement g using a frozen entailment verifier with the memory item as premise and the cited sentence as hypothesis. The support threshold is calibrated on generic NLI data, while invalid identifiers and citations without a localized supporting span are rejected before scoring. In ALFWorld and WebShop the cited span is replaced by the executed action; Appendix H states the corresponding use detector. The resulting grounded-use signal provides the basis for assigning responsibility to individual memory operations.

3.2 Responsibility Routing

Leveraging the grounded-use signal in Equation 6, we assign downstream outcomes only to operations that participate in the observed memory lifecycle. Specifically, we define the responsibility

masks as

$$z_i^{\text{write}} = \mathbb{I}\left[\sum_{q\in\mathcal{Q}(\tau)} \mathbb{I}[i \in S_q] > 0\right], \quad z_i^{\text{use}} = \mathbb{I}\left[\sum_{q\in\mathcal{Q}(\tau)} u_{q,i} > 0\right], \\ z_{q,i}^{\text{ret}} = \mathbb{I}[i \in S_q], \qquad\qquad z_q^{\text{ans}} = 1. \tag{7}$$

where z_i^{write}, z_i^{use}, $z_{q,i}^{\text{ret}}$, and z_q^{ans} indicate write eligibility, grounded use, retrieval, and answer generation, respectively.

Based on Equation 7, different failure modes are assigned to their corresponding operations rather than broadcast across the full trajectory. An unretrieved write receives no task credit but still incurs storage cost, whereas a retrieved but unused item assigns retrieval waste to the retrieval decision. If a grounded item participates in an unsuccessful answer, the negative task outcome remains on answer generation rather than being propagated to its upstream write. Conversely, successful grounded use routes positive task credit to the responsible write and retrieval operations. These masks therefore determine the operation-level eligibility used by the reward construction in Section 3.

3.3 Item-Use Groups

Equation 4 shows that exact, high-specificity state grouping becomes increasingly sparse as free-form memory updates accumulate. To preserve local comparison after memory states diverge, we replace that exact-store key with item-use events derived from Equations 5–6. We construct local groups from two complementary sources: listwise comparisons among items retrieved for the same request and replay comparisons across compatible item-use events.

For an event e, the item-use grouping key is defined as

$$\kappa(e) = \big(\texttt{type}(e),\, \phi(i),\, \psi(q),\, \Delta(i,q)\big), \tag{8}$$

where $\texttt{type}(e)$ denotes the lifecycle stage, $\phi(i)$ the item bucket derived from the source domain and pre-write source segment, $\psi(q)$ the task stratum of request q, and $\Delta(i,q)$ the coarsened write-to-use delay. Concretely, $\phi(i)$ is obtained from eight frozen source clusters before the write action, while $\psi(q)$ combines the benchmark, request type, context-length quartile, and reference-policy difficulty quintile. All grouping maps are fixed before policy optimization and contain neither sampled memory content nor task rewards. For items that remain unretrieved when their traces close, $\psi(q)$ and $\Delta(i,q)$ are replaced by fixed NO-USE sentinels to form cost-only groups.

Replay therefore places a write from one request in the same group as a later use of a compatible item on another request. That cross-request grouping is a surrogate for operational similarity: events share a frozen pre-action key, so delayed uses can be ranked against one another without claiming that the two requests are the same MDP state or that Equation 9 is an unbiased policy gradient. Appendix F states the corresponding order-invariance claim.

Based on the group $\mathcal{K}$ induced by Equation 8, we compute a group-relative advantage independently for each reward channel:

$$\widehat{A}_e^c = \frac{R_e^c - \mu_{\mathcal{K}}^c}{\sigma_{\mathcal{K}}^c}, \qquad e \in \mathcal{K}, \tag{9}$$

where $c \in \mathcal{C} = \{\text{task}, \text{use}, \text{cost}\}$, and $\mu_{\mathcal{K}}^c$ and $\sigma_{\mathcal{K}}^c$ denote the reward mean and standard deviation within group $\mathcal{K}$. In contrast to the trajectory-level advantage in Equation 2, Equation 9 compares memory operations according to compatible item-use events rather than assigning the same signal to all writes.

Groups containing fewer than two events or zero reward variance contribute zero local advantage. Since answer generation has no responsible memory item, answer tokens retain the same-input rollout grouping used in Equation 2. Detailed listwise assignment, replay sampling, and event masking are provided in Appendix E.

3.4 Multi-Channel Rewards and Policy Update

Based on the responsibility masks in Equation 7, we construct three complementary reward channels for task outcome, grounded use, and memory cost. For the task channel, an eligible upstream event

receives $R_{\text{task}}(\tau)$ only when it lies on a successful grounded-use path defined by Equation 6. If the grounded evidence is followed by an unsuccessful answer, its upstream task reward is set to zero, while answer generation retains the original terminal outcome.

Leveraging the grounded-use indicator in Equation 6, we define the use reward as

$$R^{\text{use}}_{q,i} = u_{q,i} - \lambda_{\text{spam}}\, \mathbb{I}[i \in C_q]\mathbb{I}[g(m_i, y_q) = 0], \tag{10}$$

where $\lambda_{\text{spam}} \geq 0$ penalizes unsupported citations. Thus, Equation 10 reinforces grounded consumption while explicitly discouraging unsupported citation behavior.

To further discourage redundant storage and retrieval, we define the event-level cost as

$$R^{\text{cost}}_e = \begin{cases} -\lambda_{\text{store}}\, \text{tok}(m_i), & e \text{ writes item } i, \\ -\lambda_{\text{ret}} - \lambda_{\text{ctx}}\, \text{tok}(m_i), & e \text{ retrieves item } i, \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where $\text{tok}(\cdot)$ counts tokens, and λ_{store}, λ_{ret}, and λ_{ctx} control storage, retrieval, and retrieved-context costs, respectively.

Since the task, use, and cost signals in Equations 10–11 exhibit different sparsity and numerical scales, directly normalizing their sum may obscure within-channel distinctions. We therefore first normalize each channel according to Equation 9 and then aggregate them as

$$\widehat{A}_e = \alpha_{\text{task}}\widehat{A}^{\text{task}}_e + \alpha_{\text{use}}\widehat{A}^{\text{use}}_e + \alpha_{\text{cost}}\widehat{A}^{\text{cost}}_e, \tag{12}$$

where $\alpha_c \geq 0$ and $\sum_{c\in\mathcal{C}} \alpha_c = 1$. By separating normalization before aggregation, Equation 12 preserves reward distinctions that may disappear under joint normalization Liu et al. (2026).

Finally, based on the event-level advantage in Equation 12, we update only the policy tokens responsible for event e. For a responsible token a_j with context h_j, we define the importance ratio as

$$\rho_j(\theta) = \frac{\pi_\theta(a_j \mid h_j)}{\pi_{b(e)}(a_j \mid h_j)}, \tag{13}$$

where $\pi_{b(e)}$ is the behavior policy that generated event e and equals the previous policy for current-rollout events. Using Equation 13, the final policy objective is

$$\mathcal{L}_{\text{UGPO}}(\theta) = -\mathbb{E}_{e,\, j\in\mathcal{T}(e)}\Big[\min\big(\rho_j(\theta)\widehat{A}_e,\, \text{clip}_{\delta_{\text{clip}}}(\rho_j(\theta))\widehat{A}_e\big)\Big] + \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}), \tag{14}$$

where $\mathcal{T}(e)$ denotes the responsible token set, $\text{clip}_{\delta_{\text{clip}}}$ bounds the importance ratio within $[1 - \delta_{\text{clip}}, 1 + \delta_{\text{clip}}]$, π_{ref} is the reference policy, and $\beta \geq 0$ controls KL regularization. Following group-relative policy optimization Shao et al. (2024), Equation 14 applies the item-use advantage only to the tokens responsible for the corresponding lifecycle event.

For memory items whose use is observed only after their original collection batch, we further exploit the stable provenance in Equation 5. Specifically, delayed-use replay joins the future grounded-use event in Equation 6 with its originating write through $(i, t_{\text{write}}, t_{\text{use}})$ and reuses the stored behavior-policy log-probabilities in Equation 13. The resulting write event is inserted into a later optimization batch and truncated according to replay age and importance-ratio range. Thus, delayed downstream outcomes can supervise their originating writes without regenerating the intervening interaction history. The additional bookkeeping complexity is analyzed in Appendix C.

4 Experiments and Discussion

In this section, we aim to address the following research questions (RQs). RQ 1 examines end-to-end task quality; RQ 2 tests whether item-use groups remain selective after free-form writes; RQ 3 isolates the contribution of each component; and RQ 4 measures rollout and serving overhead. Appendix I reports training stability, robustness, agentic transfer, residual errors, citation gaming, and group-key misspecification.

Table 1: **Task performance under matched settings with Qwen2.5-7B-Instruct.** All methods use the same context, memory, retrieval, and optimization budgets. HotpotQA and 2Wiki report F1, while LoCoMo and LongMemEval use their official scores.

Method	HotpotQA ↑	2Wiki ↑	LoCoMo ↑	LongMemEval ↑	Average ↑
Outcome GRPO (2024)	68.4±0.7	65.0±0.8	55.7±0.9	62.8±0.7	63.0±0.8
GiGPO (2025)	69.1±0.6	65.8±0.7	56.5±0.8	63.6±0.6	63.8±0.7
ReMemR1 (2025)	71.3±0.4	68.1±0.5	58.6±0.7	65.1±0.5	65.8±0.5
Memory-R1 (2025)	69.8±0.6	66.7±0.6	59.4±0.7	67.1±0.6	65.8±0.6
Fine-Mem (2026)	71.8±0.5	68.6±0.5	60.1±0.6	66.4±0.5	66.7±0.5
Mem-T (2026)	72.1±0.5	68.9±0.5	60.5±0.6	66.8±0.5	67.1±0.5
Memory-R2 (2026)	72.4±0.4	69.3±0.5	61.0±0.5	67.3±0.5	67.5±0.5
HiMPO (2026)	72.7±0.4	69.5±0.4	61.1±0.5	67.5±0.5	67.7±0.5
Retrieve-gated (Control)	71.0±0.5	67.7±0.5	59.2±0.6	66.4±0.5	66.1±0.5
UGPO	**73.6±0.3**	**70.2±0.4**	**61.8±0.5**	**68.7±0.4**	**68.6±0.4**

4.1 Experimental Setup

Tasks and baselines. We evaluate streaming long-document reasoning on HotpotQA and 2WikiMultiHopQA Yang et al. (2018); Ho et al. (2020) and multi-session memory on LoCoMo and LongMemEval Maharana et al. (2024); Wu et al. (2025). Documents arrive as a stream rather than a packed context, so a write helps only if a later request retrieves and uses it; a multi-session request may consume an item written many sessions earlier. Baselines cover outcome-only GRPO, state-relative GiGPO, ReMemR1, Memory-R1, Fine-Mem, Mem-T, Memory-R2, HiMPO, and a retrieve-gated control Shao et al. (2024); Feng et al. (2025); Yu et al. (2025); Yan et al. (2025); Ma et al. (2026); Yue et al. (2026); Yan et al. (2026b;a). Matched runs use Qwen2.5-7B-Instruct, a 128K read context, an 8K-token memory budget, top-8 retrieval, groups of eight rollouts, and the same verifier and training-token budget. Appendix G provides the complete configuration.

Metrics and implementations. HotpotQA and 2WikiMultiHopQA report answer F1; LoCoMo and LongMemEval use their official scores. The four-task macro-average is the unweighted mean of these four numbers. Attribution precision is the fraction of positively credited writes whose identifiers lie in the reference set P_q constructed in Appendix H; that set is unused during training. Pollution counts tokens retained in items that are never grounded in a response. All comparisons use five random seeds and report mean $\pm$ standard deviation. Implementations share Qwen2.5-Instruct backbones, AdamW at 1×10^{-6}, an 8K-token store, top-8 retrieval, and groups of eight rollouts. UGPO scores grounded use with a frozen DeBERTaV3-large NLI verifier and forms item buckets from eight pre-write E5-large source clusters. Channel weights are $(\alpha_{\text{task}}, \alpha_{\text{use}}, \alpha_{\text{cost}}) = (0.60, 0.25, 0.15)$; Table 5 lists the remaining matched settings.

4.2 Comparison on memory-agent tasks (RQ1)

Table 1 and Figure 3(a)–(b) report the main results. UGPO improves the four-task macro-average by 0.9 over HiMPO and 5.6 over Outcome GRPO, including gains of 0.9 F1 on HotpotQA and 1.2 on LongMemEval. It also exceeds retrieve-gated credit by 2.5 on average, supporting grounded use over retrieval exposure. Figure 3(b) places these gains at fewer retrieved tokens per request, improving quality and context efficiency together. Table 2 reports the same ranking at 14B and 32B, with macro-average gains of 0.9 over HiMPO in each size.

4.3 Credit selectivity of item-use grouping (RQ2)

Figure 3(c) shows coverage under exact, high-specificity state grouping decaying toward singleton groups with preceding writes, while item-use coverage remains usable. Panel (d) places UGPO at 78.6% attribution precision, 6.9 points above HiMPO and 13.4 above retrieve-gated credit. Lifecycle tables in Appendix I exhibit the same trend: write utilization rises to 71.3% and pollution falls to 980 tokens. These results imply that the relevant comparison unit is the consumed item rather than the complete store.

Table 2: **Scaling results across Qwen2.5 model sizes.** We compare representative credit-assignment methods using Qwen2.5-7B/14B/32B-Instruct under matched memory, retrieval, and optimization settings. Average denotes the macro-average over the four benchmarks.

Backbone	Method	HotpotQA ↑	2Wiki ↑	LoCoMo ↑	LongMemEval ↑	Average ↑
Qwen2.5-7B	Outcome GRPO	68.4±0.7	65.0±0.8	55.7±0.9	62.8±0.7	63.0±0.8
	HiMPO	72.7±0.4	69.5±0.4	61.1±0.5	67.5±0.5	67.7±0.5
	Retrieve-gated	71.0±0.5	67.7±0.5	59.2±0.6	66.4±0.5	66.1±0.5
	UGPO	**73.6±0.3**	**70.2±0.4**	**61.8±0.5**	**68.7±0.4**	**68.6±0.4**
Qwen2.5-14B	Outcome GRPO	71.0±0.6	68.1±0.7	59.4±0.8	66.8±0.6	66.3±0.7
	HiMPO	75.1±0.4	72.3±0.4	64.8±0.5	71.5±0.4	70.9±0.4
	Retrieve-gated	73.2±0.5	70.4±0.5	62.6±0.6	69.2±0.5	68.9±0.5
	UGPO	**76.0±0.3**	**73.1±0.3**	**65.7±0.4**	**72.5±0.4**	**71.8±0.4**
Qwen2.5-32B	Outcome GRPO	73.8±0.5	71.0±0.6	62.8±0.7	70.0±0.6	69.4±0.6
	HiMPO	77.2±0.3	74.5±0.4	67.5±0.5	74.1±0.4	73.3±0.4
	Retrieve-gated	75.4±0.4	72.8±0.5	65.2±0.5	72.1±0.4	71.4±0.5
	UGPO	**78.0±0.3**	**75.4±0.3**	**68.3±0.4**	**75.0±0.3**	**74.2±0.3**

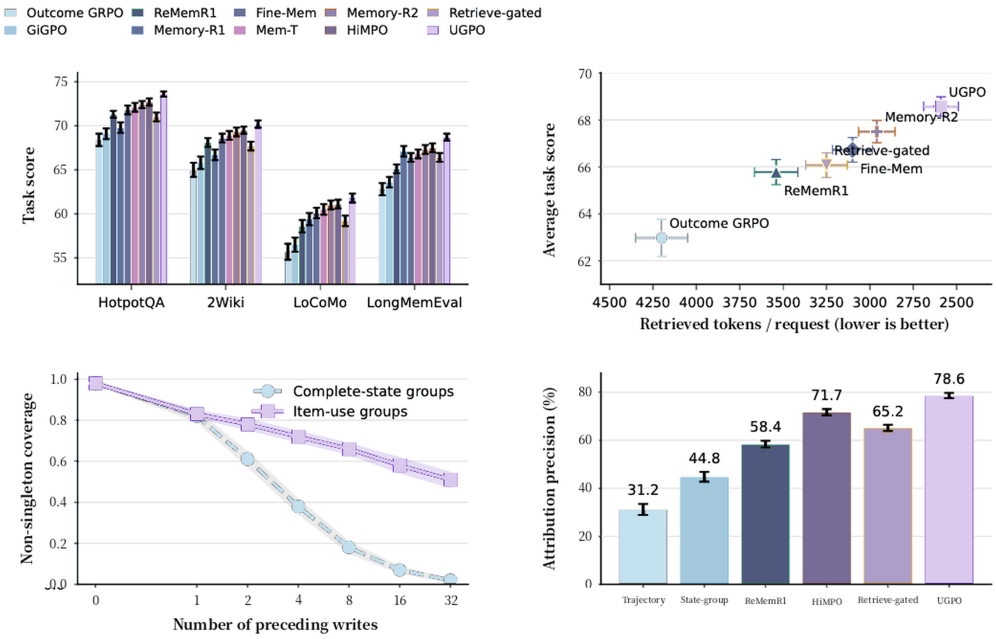


Figure 3: **Task effectiveness and credit-quality dashboard.** (a) Task scores on four benchmarks. (b) Macro-average task score versus retrieved tokens per request. (c) Non-singleton coverage for exact-state and item-use groups. (d) Controlled-provenance precision of positively credited writes.

4.4 Ablation results (RQ3)

Table 3 and Figure 4 isolate grounded use, responsibility routing, channel-wise normalization, delayed-use replay, and item-use grouping. Every removal degrades at least one target metric. Replacing item-use groups with exact-state keys drops average task from 68.6 to 65.8 and attribution precision from 78.6% to 44.8%. Removing the use channel reduces precision to 54.0%, while removing the cost channel leaves task nearly unchanged (68.0) but more than doubles pollution (2180 versus 980). Figure 4 further places the full model in the high-precision, low-pollution region and shows delayed-use replay retaining quality through sixteen sessions.

4.5 Systems efficiency (RQ4)

Figure 5 reports training and serving cost, with the full execution breakdown in Table 4 of Appendix C. UGPO keeps a 1.0× rollout multiplier, matching Outcome GRPO and GiGPO, whereas

Table 3: **RQ3: Component ablations.** Task is the four-benchmark macro-average; Attr. Prec. is controlled-provenance precision; Pollution counts tokens in never-used items.

Variant	Average task (%) ↑	Attr. Prec. (%) ↑	Pollution (tokens) ↓
UGPO	**68.6±0.3**	**78.6±1.1**	**980±80**
w/o Grounded use	67.2±0.5	65.2±1.3	1530±105
w/o Grounding check	66.9±0.5	59.4±1.8	1420±120
w/o Channel-wise norm.	67.5±0.4	73.1±1.5	1210±95
w/o Responsibility routing	66.6±0.6	62.5±1.7	1600±130
w/o Delayed-use replay	67.7±0.5	70.2±1.4	1320±110
w/o Item-use groups	65.8±0.7	44.8±2.0	2490±145
w/o Use channel	66.8±0.6	54.0±1.9	1900±135
w/o Cost channel	68.0±0.4	77.9±1.2	2180±150
w/o Replay groups	67.8±0.5	72.4±1.5	1280±100
w/o Listwise groups	67.4±0.5	69.8±1.6	1390±110

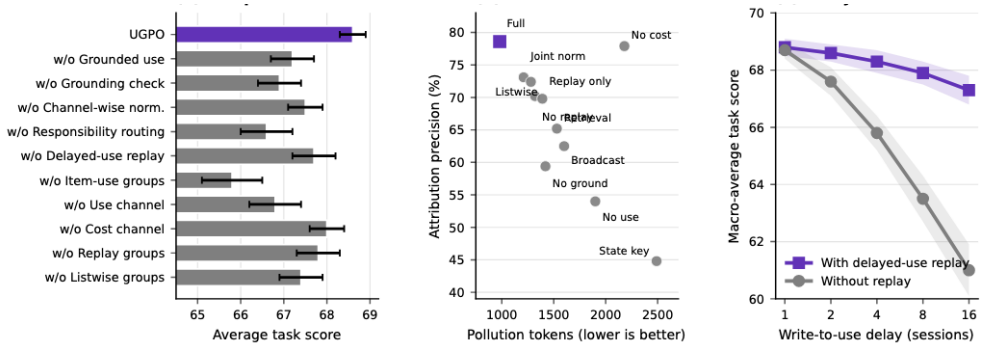


Figure 4: **Mechanism diagnostics.** (a) Component ablations; (b) attribution precision versus pollution; (c) task score over write-to-use delay with and without replay.

Memory-R2 and Mem-T require $3.8\times$ and $4.2\times$ continuations. Generated tokens remain close to ReMemR1 (8450 versus 8600), while serving latency decreases to 875 ms. Trace processing grows with logged events but remains bookkeeping, demonstrating lifecycle credit without branching the original rollout.

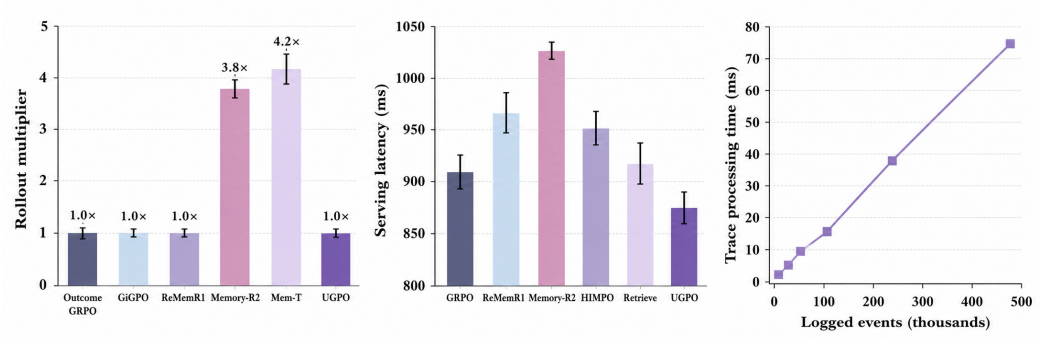


Figure 5: **Systems-efficiency analysis.** (a) Environment rollout multiplier; (b) end-to-end serving latency; (c) UGPO trace-processing time over logged WRU events.

5 Conclusion

Long-horizon memory interactions make trajectory- and state-level credit increasingly noisy. UGPO instead groups write, retrieval, and grounded-use events, turning memory-service telemetry into operation-level task, use, and cost supervision without extra rollouts or counterfactual execution. This yields cleaner credit assignment, supports delayed-use attribution, and scales naturally to long-horizon memory agents. Experiments show that UGPO improves attribution fidelity, task performance, robustness, and serving efficiency.

USE OF GENERATIVE AI

We used generative AI tools for language polishing of the manuscript. We have reviewed and verified all AI-assisted text and take full responsibility for the final content of this work, including claims, equations, and experimental descriptions.

ETHICS STATEMENT

This work studies credit assignment for long-context memory agents on public academic benchmarks and does not involve human subjects or personally identifiable information. The evaluation uses HotpotQA, 2WikiMultiHopQA, LoCoMo, LongMemEval, ALFWorld, and WebShop under their standard protocols.

REPRODUCIBILITY STATEMENT

Anonymous source code and datasets are available at `https://anonymous.4open.science/status/credit-26D8`. Training and memory settings appear in Appendix G; dataset and metric protocols in Appendix H; the training procedure in Algorithm 1; the estimator in Appendix E; the group-conditioned surrogate claim in Appendix F; and the exact-state coverage bound in Appendix B.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding, 2024. URL https://arxiv.org/abs/2308.14508.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *CoRR*, abs/2505.10978, 2025. doi: 10.48550/ARXIV.2505.10978. URL https://doi.org/10.48550/arXiv.2505.10978.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=sE7-XhLxHA.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps, 2020. URL https://arxiv.org/abs/2011.01060.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization, 2026. URL https://arxiv.org/abs/2601.05242.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Weitao Ma, Xiaocheng Feng, Lei Huang, Xiachong Feng, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Bing Qin. Fine-mem: Fine-grained feedback alignment for long-horizon memory management. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19666–19684, San Diego, California, United States,

July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.900. URL https://aclanthology.org/2026.acl-long.900/.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Marina Mao, Alexandr Liu, Pengbo Li, Siheng Li, Bo Zhou, and Xiang Wang. Tree-based credit assignment for multi-agent memory system, 2026. URL https://arxiv.org/abs/2605.04811.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In Sean Follmer, Jeff Han, Jürgen Steimle, and Nathalie Henry Riche (eds.), *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pp. 2:1–2:22. ACM, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

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, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.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.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. Trace: Turn-level reward assignment via credit estimation for long-horizon agents, 2026. URL https://arxiv.org/abs/2607.13988.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *CoRR*, abs/2212.03533, 2022. doi: 10.48550/ARXIV.2212.03533. URL https://doi.org/10.48550/arXiv.2212.03533.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101/.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Jiangze Yan, Yi Shen, Wenjing Zhang, Jieyun Huang, Zhaoxiang Liu, Ning Wang, Kai Wang, and Shiguo Lian. Himpo: Hindsight-informed memory policy optimization for less-entangled credit in long-horizon agents, 2026a. URL https://arxiv.org/abs/2606.16285.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *CoRR*, abs/2508.19828, 2025. doi: 10.48550/ARXIV.2508.19828. URL https://doi.org/10.48550/arXiv.2508.19828.

Sikuan Yan, Ahmed Bahloul, Ercong Nie, Susanna Schwarzmann, Riccardo Trivisonno, Volker Tresp, and Yunpu Ma. Memory-r2: Fair credit assignment for long-horizon memory-augmented llm agents, 2026b. URL https://arxiv.org/abs/2605.21768.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024. doi: 10.48550/ARXIV.2412.15115. URL https://doi.org/10.48550/arXiv.2412.15115.

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

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context LLM with multi-conv rl-based memory agent. *CoRR*, abs/2507.02259, 2025. doi: 10.48550/ARXIV.2507.02259. URL https://doi.org/10.48550/arXiv.2507.02259.

Yanwei Yue, Boci Peng, Xuanbo Fan, Jiaxin Guo, Qiankun Li, and Yan Zhang. Mem-t: Densifying rewards for long-horizon memory agents, 2026. URL https://arxiv.org/abs/2601.23014.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *CoRR*, abs/2601.03192, 2026. doi: 10.48550/ARXIV.2601.03192. URL https://doi.org/10.48550/arXiv.2601.03192.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19724–19731. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

APPENDIX: TABLE OF CONTENT

A Related Work **13**

B Coverage Bound for Exact-State Grouping **14**

C Complexity **14**

D Algorithmic Summary **15**

E Complete Estimator Specification **15**

F Group-Conditioned Surrogate Objective **16**

G Implementation and Reproducibility Details **16**

H Datasets and Evaluation Protocol **17**

I Extended Results **18**

I.1 Extended analyses for RQ2 and RQ3 18

I.2 RQ5: How stable is UGPO throughout optimization? 19

I.3 RQ6: How robust is UGPO to workload and configuration changes? 19

I.4 RQ7: Does lifecycle credit transfer beyond question answering? 20

I.5 RQ8: Where do residual attribution errors arise? 21

I.6 RQ9: Is grounded use robust to citation gaming? 21

I.7 RQ10: Does the grouping key require a tuned oracle? 22

I.8 Statistical reliability of the main comparison 23

I.9 Same reward, different assignment 23

I.10 Grounded but harmful memory 23

A RELATED WORK

Long-context and persistent memory agents. External memory lets a language agent retain evidence beyond the context window. Operating-system-style paging and write policies manage a hierarchical working store Packer et al. (2024), while structured notes and evolving knowledge graphs support multi-session interaction Park et al. (2023); Gutiérrez et al. (2024). Complementary designs keep an episodic diary Park et al. (2023); Zhong et al. (2024), recursively summarize documents into a retrieval tree , or paginate long inputs with gist memory . Long-context benchmarks further isolate whether models can reuse distant evidence without an explicit store Bai et al. (2024). Learned controllers then replace these heuristics: MemAgent and ReMemR1 train compression policies for streaming documents, Memory-R1 learns add, update, delete, and no-op operations, and MemRL adapts the store from episodic outcomes Yu et al. (2025); Yan et al. (2025); Zhang et al. (2026). These systems show that memory control can be optimized, but they still couple delayed utility to a trajectory-level or evidence-labeled signal.

Fine-grained credit for agentic reinforcement learning. Group-relative optimization estimates a critic-free advantage from complete responses Shao et al. (2024), following the clipped importance-ratio form of proximal policy optimization Schulman et al. (2017). GiGPO instead compares actions that revisit the same environment state, which reuses collected trajectories when states recur Feng et al. (2025). Free-form memory writes make that recurrence rare: each add or update changes the store, so later operations no longer share a comparison key. When several reward channels are present, jointly normalizing their sum can hide a sparse channel behind a dense one; GDPO therefore normalizes each channel before aggregation Liu et al. (2026). UGPO keeps that channel-wise step and changes the comparison unit from exact-store equality to observed item use.

Credit assignment for learned memory. Process-level memory rewards typically require extra labels or extra execution. ReMemR1 and Fine-Mem score intermediate memories against gold answers or auxiliary questions Yu et al. (2025); Ma et al. (2026); Tao et al. (2026); TRACE estimates local credit from changes in gold-answer predictability Yu et al. (2025); Ma et al. (2026); Tao et al. (2026). Tree and rerollout methods expand the original trajectory: Mem-T and TreeMem propagate value through operation trees, while Memory-R2 resamples local alternatives from a shared store Yue et al. (2026); Mao et al. (2026); Yan et al. (2026b). HiMPO instead contrasts the store immediately before and after a write Yan et al. (2026a). UGPO does not introduce gold evidence, counterfactual branches, or a second memory state. It reads the write–retrieve–use events that a persistent store already logs, and it evaluates that choice on long-document and multi-session workloads together with attribution, pollution, and serving-cost measurements.

B Coverage Bound for Exact-State Grouping

The argument applies only when $h(\cdot)$ is an exact, high-specificity store key, as stated with Equation 4 in Section 2: while two stores still match, one additional free-form write preserves the match with conditional probability at most $\gamma < 1$. Consider a fixed trajectory b after n such writes. For any other trajectory b', the chain rule then gives

$$\Pr\Big[h(s_{b'}^{(n)}) = h(s_b^{(n)})\Big] \le \prod_{r=1}^{n} \gamma = \gamma^n. \tag{15}$$

where $s_b^{(n)}$ is the state of trajectory b after n writes, r indexes writes, and $h(\cdot)$ is the exact-state comparison key. The bound is pairwise: it does not require the remaining trajectories to be independent of one another, only that each successive write is still constrained by γ while the stores match.

A union bound over the remaining $|\mathcal{B}| - 1$ trajectories in the rollout batch yields

$$\Pr\Big[\exists b' \neq b : h(s_{b'}^{(n)}) = h(s_b^{(n)})\Big] \le (|\mathcal{B}| - 1)\gamma^n. \tag{16}$$

where the event on the left is exactly the event that trajectory b belongs to a non-singleton exact-state group. Averaging this indicator over $b \in \mathcal{B}$ and capping a probability at one proves Equation 4. The same bound shows why enlarging a fixed rollout batch cannot restore coverage at long write horizons: holding $\mathbb{E}[C_{\text{state}}]$ constant requires $|\mathcal{B}|$ to grow exponentially in n. Item-use groups are not covered by this argument, because they compare identifiers and request strata rather than exact store equality.

C Complexity

UGPO keeps the environment rollout count unchanged relative to outcome-level GRPO Shao et al. (2024). Trace construction is linear in the number of recorded memory events. The write–retrieve–use joins themselves are maintained by an identifier-indexed map, so the extra work grows with logged events rather than with sampled alternatives. For E events and K retrieved items, bookkeeping costs $O(E + K)$ time and $O(E)$ trace storage before retention policies. Grounded-citation formatting is the only extra model computation, and only when the base agent does not already emit citations. Local rerollout and tree methods instead multiply environment and model execution by the number of sampled alternatives Yan et al. (2026b); Yue et al. (2026); Mao et al. (2026).

Table 4 reports the corresponding wall-clock profile under the matched training budget of Appendix G. Rollout multiplier counts continuations relative to one Outcome GRPO trajectory; generated tokens are counted per training instance.

UGPO retains a $1.0\times$ rollout multiplier, matching Outcome GRPO and GiGPO, while Memory-R2 and Mem-T require $3.8\times$ and $4.2\times$ continuations. Generated tokens stay close to ReMemR1 (8450 versus 8600), and the extra 10 accelerator hours relative to ReMemR1 come from identifier joins and the frozen entailment check rather than additional environment steps. Peak memory rises by 2 GB over ReMemR1, which is consistent with storing behavior-policy log-probabilities for delayed-use replay.

Table 4: **RQ4: Environment-execution comparison.** Rollout multiplier counts continuations relative to one Outcome GRPO rollout; generated tokens are per training instance.

Method	Rollout multiplier ↓	Generated tokens per instance ↓	Accelerator hours ↓	Peak memory (GB) ↓	Extra branches
Outcome GRPO	1.0±0.0	8200±210	220±8	68.0±1.2	0
GiGPO	1.0±0.0	8300±220	230±9	70.0±1.3	0
ReMemR1	1.0±0.0	8600±240	245±10	72.0±1.5	0
Memory-R2	3.8±0.2	31400±1200	810±34	81.0±1.8	Local rerollouts
Mem-T	4.2±0.3	34500±1450	920±41	84.0±2.0	Operation tree
UGPO	1.0±0.0	8450±230	255±11	74.0±1.5	0

D Algorithmic Summary

Algorithm 1 states one optimization step of UGPO. The original rollout is unchanged: the policy emits memory operations and an answer, while the service records write, retrieval, citation, and outcome events with stable identifiers. Citation validation applies the grounded-use rule in Equation 6 before any reward is routed. Delayed-use pairs are inserted into a replay queue rather than regenerated, and grouping uses the item-use key in Equation 8 rather than exact-store equality.

Algorithm 1 UGPO Training Procedure

Input: Memory agent π_θ, reference policy π_{ref}, task instance x
Parameter: Clipping coefficient ϵ, KL coefficient β
Output: Updated policy parameters θ

function TrainingStep($\pi_\theta, \pi_{\text{ref}}, x$)
$\tau \leftarrow$ Rollout(π_θ, x) {Assign stable IDs to materialized memory items}
$\mathcal{E} \leftarrow$ RecordEvents(τ) {Record write, retrieval, citation, and outcome events}
$\mathcal{E}_{\text{valid}} \leftarrow$ ValidateCitations($\mathcal{E}$) {Apply the grounded-use rule}
$\mathcal{R} \leftarrow$ RouteRewards($\mathcal{E}_{\text{valid}}$) {Route task, use, and cost rewards to eligible tokens}
for each delayed-use event $e_u \in \mathcal{E}_{\text{valid}}$ **do**
  $e_w \leftarrow$ MatchOriginWrite(e_u)
  $\mathcal{Q}_{\text{delay}} \leftarrow \mathcal{Q}_{\text{delay}} \cup \{(e_w, e_u)\}$ {Insert write–use pair into replay queue}
end for
$\mathcal{G} \leftarrow$ GroupByItemUse($\mathcal{E}_{\text{valid}}, \mathcal{Q}_{\text{delay}}$)
$\hat{\mathcal{R}} \leftarrow$ NormalizeRewards($\mathcal{R}, \mathcal{G}$)
$\hat{A} \leftarrow$ CombineAdvantages($\hat{\mathcal{R}}$) {Aggregate normalized reward channels}
$\mathcal{L}_{\text{UGPO}} \leftarrow \mathcal{L}_{\text{clip}}(\theta, \hat{A}) + \beta D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big)$
$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{UGPO}}$
return θ
end function

The clipping term and KL regularizer follow proximal policy optimization Schulman et al. (2017). Channel-wise normalization is applied inside NormalizeRewards before CombineAdvantages, so a dense cost channel cannot rescale a sparse task channel Liu et al. (2026). Replay entries reuse stored behavior-policy log-probabilities, which is why Algorithm 1 never rerolls the intervening history.

E Complete Estimator Specification

Event-to-group assignment. Current-rollout answer events use the same-input group $\mathcal{G}(\tau)$. When a request q retrieves at least two items, each retrieval event enters one listwise group that contains every item event in S_q. A grounded use joins its stable identifier to the responsible retrieval and to the most recent write in its predecessor chain, so credit can follow updates. The resulting write and use events enter replay groups sharing $\kappa(e)$; update and delete operations keep the source features of the affected lineage. Writes that remain unretrieved at trace closure enter cost-only replay groups under the NO-USE request and delay sentinels, and no event is compared both listwise and through replay in the same update.

Responsible token masks. For a write event, $\mathcal{T}(e)$ contains the emitted operation and content span. For retrieval, it contains the query and ranking-decision span; for grounded use, it contains the identifier-bearing citation span; and for an answer event, it contains all answer tokens. Lineage

follows stable identifiers through updates and assigns a delete only to the latest live predecessor. Unsupported citations are eligible for the use-channel spam penalty in Equation 10 but do not open a task-credit path to the originating write.

Replay construction. Each optimization batch pairs current events with an equal number of replay events sampled uniformly within the same key. Groups contain between two and eight events. Replay entries expire after 32 sessions, and an entry is discarded when any responsible-token behavior ratio falls outside $[0.5, 2.0]$ before objective clipping. The stored behavior-policy log-probability defines $\pi_{b(e)}$ in Equation 14. If a key has fewer than two eligible events or a channel has zero within-group variance, its advantage for that channel is zero.

Difficulty control and invariance. The reference-policy difficulty quintile in $\psi(q)$ is computed once before optimization from mean next-token entropy along a fixed greedy response of the frozen initial policy, without task labels. It prevents cross-request replay from comparing evidently different difficulty levels while preserving useful coverage. For a group-constant offset $d^c_{\mathcal{K}}$, channel normalization satisfies

$$\frac{(R^c_e + d^c_{\mathcal{K}}) - (\mu^c_{\mathcal{K}} + d^c_{\mathcal{K}})}{\sigma^c_{\mathcal{K}}} = \frac{R^c_e - \mu^c_{\mathcal{K}}}{\sigma^c_{\mathcal{K}}}. \tag{17}$$

where $d^c_{\mathcal{K}}$ is a nuisance shared by group $\mathcal{K}$ in channel c, and the remaining symbols follow Equation 9. Exact group-level difficulty offsets therefore do not alter the relative advantage. The identity is an invariance of within-group comparison. Appendix F states the corresponding surrogate-objective claim: Equation 9 preserves action order inside a stratum and is not an unbiased estimator of the policy gradient of expected task return.

F Group-Conditioned Surrogate Objective

Equation 9 subtracts a group mean and rescales by a group standard deviation before the clipped update in Equation 14. That centering is not an unbiased estimator of $\nabla_\theta \mathbb{E}[R_{\text{task}}(\tau)]$. Events that share a high-mean stratum can receive a negative advantage even when their raw reward exceeds that of events in a weaker stratum. We therefore treat Equation 14 as a *group-conditioned surrogate policy objective*: it optimizes relative operational return inside the pre-action key of Equation 8, not a globally unbiased policy gradient.

The useful guarantee is narrower and concerns order inside a stratum.

Proposition 1 (Within-stratum order)**.** *Fix a channel c and a group $\mathcal{K}$ with $|\mathcal{K}| \geq 2$ and $\sigma^c_{\mathcal{K}} > 0$. For any $e, e' \in \mathcal{K}$,*

$$\widehat{A}^c_e > \widehat{A}^c_{e'} \quad \textit{if and only if} \quad R^c_e > R^c_{e'}. \tag{18}$$

Consequently, $\arg\max_{e\in\mathcal{K}} \widehat{A}^c_e = \arg\max_{e\in\mathcal{K}} R^c_e$*. If $\sigma^c_{\mathcal{K}} = 0$, then $\widehat{A}^c_e = 0$ for every $e \in \mathcal{K}$, and the surrogate contributes no local ranking.*

The map $R \mapsto (R - \mu^c_{\mathcal{K}})/\sigma^c_{\mathcal{K}}$ is strictly increasing on $\mathcal{K}$ whenever $\sigma^c_{\mathcal{K}} > 0$, which is exactly the comparison used by listwise retrieval groups and by replay groups that share $\kappa(e)$. The identity does not extend across different keys: replacing $\mu^c_{\mathcal{K}}$ with a global mean can reverse the sign of an event that is locally below its stratum but globally above the batch. Equation 17 further shows that a group-constant offset cancels, so a shared difficulty or context shift inside $\mathcal{K}$ does not change the within-stratum order.

The surrogate is therefore the right object when the decision under training is a choice among alternatives that already share a key, such as which retrieved item to cite for a fixed request, or which delayed write to reinforce among uses in the same (ϕ, ψ, Δ) bin. Cross-request replay is reasonable on that operational criterion: the key is frozen metadata, so two uses can be compared as the same kind of event without treating the two requests as one MDP state. It is the wrong object if one interprets $\widehat{A}_e$ as a causal effect of a write or as an unbiased increment to expected task return.

G Implementation and Reproducibility Details

All methods use Qwen2.5-Instruct backbones with a 128K read context Yang et al. (2024) and AdamW at learning rate 1×10^{-6} Loshchilov & Hutter (2019). The memory interface exposes add,

update, and delete operations with stable item identifiers, matching the learned-control setting of Memory-R1 Yan et al. (2025). Grounded use is scored by a frozen DeBERTaV3-large NLI verifier He et al. (2023), with the support threshold calibrated on MultiNLI Williams et al. (2018) and then held fixed. Item buckets come from eight pre-write source clusters of E5-large embeddings Wang et al. (2022). Table 5 records the remaining matched settings; method-specific rows apply only to UGPO.

Table 5: **Training and memory configuration.** Every baseline receives the same backbone, token budget, memory capacity, retrieval depth, and rollout group size; method-specific settings apply only to UGPO.

Component	Setting	Scope
Policy backbone	Qwen2.5-7B-Instruct; 128K context	All methods
Memory interface	Add, update, delete; stable item identifiers	All methods
Memory / retrieval budget	8K retained tokens; top-8 items per request	All methods
Optimization	AdamW; learning rate 1×10^{-6}; 500 updates	All methods
Sampling	8 rollouts per input; effective batch size 64; five seeds	All methods
Policy regularization	$\delta_{\text{clip}} = 0.2$; $\beta = 0.01$	All policy-gradient methods
Advantage weights	$(\alpha_{\text{task}}, \alpha_{\text{use}}, \alpha_{\text{cost}}) = (0.60, 0.25, 0.15)$	UGPO
Cost weights	$\lambda_{\text{spam}} = 0.5$, $\lambda_{\text{store}} = 10^{-5}$, $\lambda_{\text{ret}} = 0.02$, $\lambda_{\text{ctx}} = 10^{-5}$	UGPO
Grounding	`cross-encoder/nli-deberta-v3-large`; threshold 0.8	UGPO and matched controls
Item buckets	Source domain; 8 pre-write source clusters from `intfloat/e5-large-v2`	UGPO
Task strata	Dataset, request type, context quartile, reference-policy difficulty quintile	UGPO
Delay / replay	Five logarithmic bins; 1:1 current/replay; age 32; ratio cutoff $[0.5, 2.0]$	UGPO
Compute	8 NVIDIA A100 80GB accelerators	All methods

Grouping maps are frozen before the first policy update and are never refit on sampled memory content or task reward. The same verifier checkpoint, retrieval depth, and token budget are shared with retrieve-gated and citation-only controls, so differences in Table 3 isolate credit routing rather than a stronger reader.

H Datasets and Evaluation Protocol

Streaming multi-hop reasoning uses HotpotQA and 2WikiMultiHopQA Yang et al. (2018); Ho et al. (2020). Documents arrive as a stream rather than a single packed context, so a write that is never later retrieved cannot help the answer. Multi-session memory uses LoCoMo and LongMemEval Maharana et al. (2024); Wu et al. (2025), where a later request may consume an item written many sessions earlier. The four-task macro-average in Table 1 is the unweighted mean of HotpotQA F1, 2Wiki F1, the official LoCoMo score, and the official LongMemEval score.

Agentic transfer in Table 9 keeps the same memory interface and replaces answer F1 with environment success. ALFWorld pairs text actions with an embodied household simulator Shridhar et al. (2021); WebShop requires grounded web shopping Yao et al. (2023a). Both environments use a ReAct-style think–act loop Yao et al. (2023b). Use detection is the same three-way conjunction as Equation 6, with the cited answer span replaced by the executed action. At step t with retrieved set S_t, identifier set C_t referenced in the thought–action, and executed action a_t,

$$u^{\text{env}}_{t,i} = \mathbb{I}[i \in S_t]\,\mathbb{I}[i \in C_t]\,\mathbb{I}[g_{\text{env}}(m_i, a_t) = 1], \tag{19}$$

where g_{env} is the same frozen NLI verifier, the memory item is the premise, and the hypothesis is the linearized action string (ALFWorld object–receptacle commands; WebShop click and purchase strings). An identifier that appears only in the retrieved context and is never referenced in the thought–action is not a use. Routing then applies Equation 7 with $u^{\text{env}}_{t,i}$ in place of $u_{q,i}$.

Controlled provenance. Attribution precision and recall in Tables 3 and 10 use a reference dependency set that is never an input to training, grouping, or reward. For each evaluation request q, gold supporting units are taken from the dataset annotation: supporting sentences on HotpotQA, evidence facts on 2WikiMultiHopQA, the session turns required by the official metric on LoCoMo and LongMemEval, and the objects or product attributes consumed by a successful trajectory on ALFWorld and WebShop. Each gold unit is aligned to the unique memory identifier of the write that first materialized that span, using the source document identifier and character offset recorded at write time. The resulting identifier set P_q is the reference provenance of q. A positively credited

write is a true positive if its identifier lies in P_q, and a reference write is recalled if it received positive task or use credit. Invalid identifiers, writes of unrelated spans, and items that were retrieved but never used are negatives. This construction is computed after rollout, stored separately from the WRU trace, and discarded from the optimization batch.

Long-context stress tests in Figure 7 additionally vary input length over 32K–256K tokens, which overlaps the long-document regime studied by LongBench Bai et al. (2024), while the store budget remains the 8K-token cap of Table 5.

I Extended Results

Results below use the same five-seed protocol and matched budgets as Section 4. Task scores follow Table 1; attribution precision and pollution follow Table 3.

I.1 Extended analyses for RQ2 and RQ3

Table 6 reports the serving-side counterpart of the main task comparison. Write utilization is the fraction of materialized items that later support a response span; retrieval precision is the fraction of retrieved items that are both cited and grounded; pollution counts tokens that remain in never-used items.

Table 6: **RQ2: Lifecycle and serving comparison.** Write utilization is the percentage of materialized items subsequently grounded in a response; retrieval precision is the percentage of retrieved items that are both cited and support the associated response span; unsupported citations are cited identifiers that fail this support check. Pollution counts tokens retained in items that are never grounded, store size counts all retained tokens at evaluation time, retrieved tokens are averaged per request, and latency is end-to-end serving time in milliseconds.

Method	Write utilization (%) ↑	Retrieval precision (%) ↑	Unsupported citations (%) ↓	Pollution (tokens) ↓	Store size (tokens) ↓	Retrieved tokens per request ↓	Latency (ms) ↓
Outcome GRPO	28.4±1.9	42.1±1.8	18.7±1.5	2860±140	9800±320	4200±150	910±20
GiGPO	33.1±1.8	46.7±1.6	16.3±1.4	2550±130	9100±300	3980±140	935±21
ReMemR1	49.2±1.5	58.4±1.4	11.2±1.1	2100±115	7600±260	3540±125	970±24
Memory-R1	53.0±1.5	60.8±1.5	10.4±1.1	1940±120	7200±240	3380±130	930±22
Fine-Mem	58.8±1.4	67.1±1.3	8.8±0.9	1560±105	6500±220	3100±115	995±24
Mem-T	60.1±1.4	69.4±1.3	8.1±0.9	1490±100	6400±210	3040±110	1010±26
Memory-R2	62.4±1.3	70.8±1.2	7.6±0.8	1380±95	6200±200	2960±105	1080±28
HiMPO	64.2±1.3	72.6±1.2	7.2±0.8	1290±90	5900±190	2860±105	955±23
Retrieve-gated	55.6±1.5	65.2±1.3	14.9±1.2	1730±105	6800±230	3250±120	920±19
UGPO	**71.3±1.2**	**78.6±1.1**	**5.6±0.6**	**980±80**	**5200±180**	**2590±100**	**875±16**

UGPO raises write utilization from 64.2% (HiMPO) to 71.3% and retrieval precision from 72.6% to 78.6%, while unsupported citations fall to 5.6%. Pollution drops to 980 tokens and retrieved tokens per request to 2590, with serving latency at 875 ms. Retrieve-gated credit improves utilization over Outcome GRPO but still cites unsupported items at 14.9%, which is the gap closed by the grounding check.

Table 7 asks whether those gains still have a usable comparison set after free-form writes. Non-singleton coverage is the fraction of events assigned to groups with at least two samples; effective group size is the mean number of usable events after variance masking.

Table 7: **RQ2: Group-formation diagnostics.** Non-singleton coverage is the fraction of events assigned to groups containing at least two samples; effective group size is the mean number of statistically usable events after masking. Attribution precision and average task follow the definitions in Tables 3 and 1, respectively.

Grouping strategy	Non-singleton coverage ↑	Effective group size ↑	Attribution precision (%) ↑	Average task (%) ↑
Exact-state key	0.18±0.03	1.3±0.2	44.8±2.0	65.8±0.7
Listwise groups only	0.59±0.04	4.8±0.4	72.4±1.5	67.8±0.5
Replay groups only	0.54±0.04	5.2±0.5	69.8±1.6	67.4±0.5
UGPO	**0.66±0.03**	**7.1±0.6**	**78.6±1.1**	**68.6±0.3**

Exact-state keys decay to coverage 0.18 and effective size 1.3, matching Equation 4. Listwise groups and replay groups each recover coverage above 0.50, and combining them reaches 0.66 coverage

with effective size 7.1. The corresponding attribution precision is 78.6%, against 44.8% for exact-state grouping.

Table 8 isolates the normalization choice in Equation 12. Each channel standard deviation is measured after normalization over eligible events; values near one mean that the channel still has unit-scale variation.

Table 8: **RQ3: Reward-normalization diagnostics.** Each reward-channel standard deviation is measured after normalization over eligible events; values near one indicate that the channel retains unit-scale variation. Average task and pollution follow Tables 1 and 3.

Normalization	Task-channel s.d.	Use-channel s.d.	Cost-channel s.d.	Average task (%) ↑	Pollution (tokens) ↓
Joint normalization	0.96±0.03	0.42±0.05	0.21±0.04	67.5±0.4	1210±95
Channel-wise normalization	**1.00±0.02**	**1.00±0.03**	**1.00±0.03**	**68.6±0.3**	**980±80**

Joint normalization leaves the use and cost channels at 0.42 and 0.21, so the dense task channel dominates the sum. Channel-wise normalization restores unit scale on all three channels, raises average task from 67.5 to 68.6, and reduces pollution from 1210 to 980 tokens, consistent with GDPO Liu et al. (2026).

I.2 RQ5: How stable is UGPO throughout optimization?

A final checkpoint can hide unstable learning, store growth, or disappearing comparison groups. Figure 6 therefore tracks macro-average validation score, pollution tokens, and non-singleton group coverage at matched optimization steps.

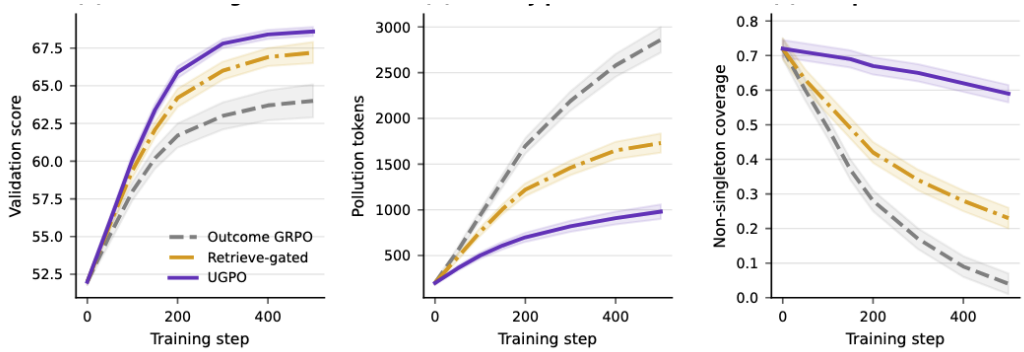


Figure 6: **RQ5: Training dynamics at matched optimization steps.** (a) Macro-average validation score (%) over the four primary tasks. (b) Tokens retained in memory items that are never grounded in a response. (c) Non-singleton comparison-group coverage for Outcome GRPO, Retrieve-gated, and UGPO. Lines show seed means and shaded regions show standard deviations.

Task score rises monotonically for UGPO while pollution falls, whereas Outcome GRPO accumulates unused tokens as training proceeds. Panel (c) shows item-use coverage remaining above the exact-state and retrieve-gated curves throughout the 500 updates, so the local groups used in Equation 9 do not shrink to singletons as the store grows.

I.3 RQ6: How robust is UGPO to workload and configuration changes?

Figure 7 varies input length, memory budget, retrieval depth, distractor rate, item-bucket count, and grounding threshold. The grouping panels expose a precision–coverage trade-off: a coarse key mixes unrelated events, whereas an excessively fine key recreates singleton groups.

The gap over Outcome GRPO persists from 32K to 256K inputs. Task score is stable between 4K and 16K memory tokens and across retrieval depths 4–16, with a drop only at depth 2 where the retrieved set is too small to form listwise groups. Attribution precision remains above retrieve-gated credit as distractors increase. The selected operating point (eight item buckets, grounding threshold 0.8) lies in a broad high-scoring region of panel (f) rather than at an isolated peak.

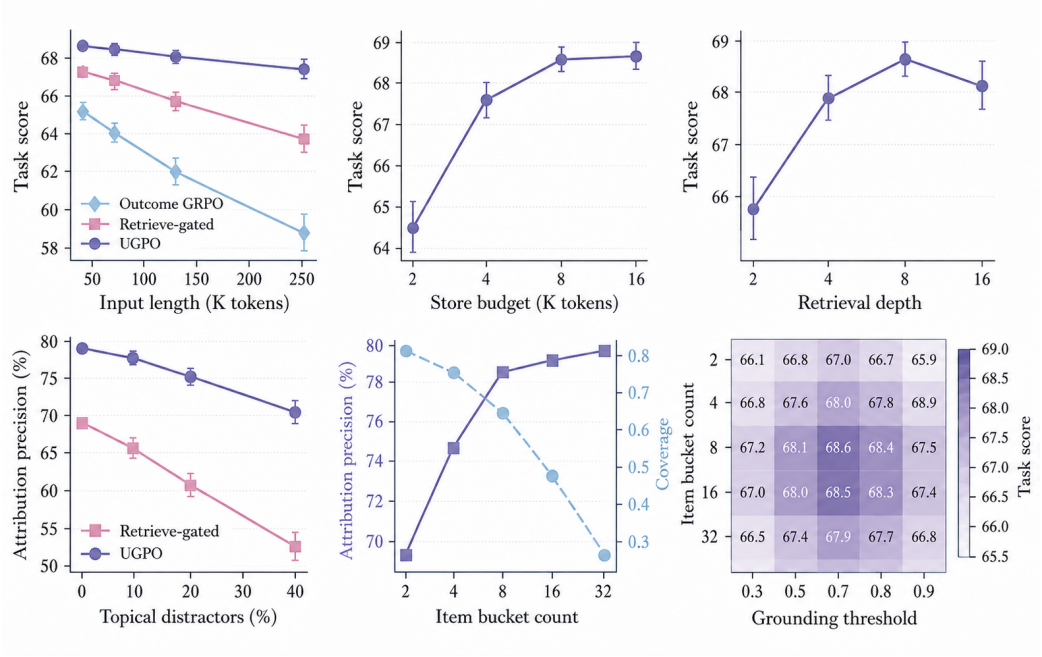


Figure 7: **RQ6: Robustness and sensitivity analysis.** (a) Task score (%) under 32K–256K input lengths for Outcome GRPO, Retrieve-gated, and UGPO. (b) UGPO task score under 2K–16K-token memory budgets. (c) UGPO task score as retrieval depth varies from 2 to 16 items. (d) Attribution precision (%) for Retrieve-gated and UGPO under increasing rates of topically similar distractors. (e) Attribution precision (purple, left axis) and non-singleton coverage (green, right axis) as the item-bucket count changes. (f) Macro-average task score (%, color scale) across item-bucket counts and grounding thresholds. Error bars denote standard deviation over five seeds.

I.4 RQ7: Does lifecycle credit transfer beyond question answering?

ALFWorld and WebShop keep the same memory interface and replace answer F1 with environment success Shridhar et al. (2021); Yao et al. (2023a). Both use a ReAct-style think–act loop Yao et al. (2023b). Use is detected by Equation 19: an item counts only if it is retrieved, referenced in the thought–action, and accepted by g_{env} as supporting the executed command. Table 9 reports official success rates under that detector.

Table 9: **RQ7: Transfer to agentic environments.** ALFWorld and WebShop report official task success rates (%) under a shared ReAct-style interface. Use follows Equation 19.

Method	ALFWorld ↑	WebShop ↑
Outcome GRPO	58.1±0.9	54.0±0.8
GiGPO	59.0±0.8	54.8±0.8
ReMemR1	60.2±0.7	56.0±0.7
Memory-R1	60.8±0.8	56.4±0.7
Fine-Mem	61.7±0.7	57.2±0.6
Mem-T	62.1±0.7	57.8±0.6
Memory-R2	62.5±0.6	58.2±0.6
HiMPO	62.7±0.6	58.5±0.6
Retrieve-gated	61.4±0.7	57.2±0.6
UGPO	**63.8±0.6**	**59.4±0.6**

UGPO reaches 63.8% on ALFWorld and 59.4% on WebShop, a 1.1 and 0.9 point gain over HiMPO. The ranking among memory-credit methods matches Table 1: outcome-only GRPO is weakest, process-level and branched methods occupy the middle, and item-use grouping remains strongest. Lifecycle credit therefore transfers to settings where the downstream event is an environment action rather than a cited answer span.

I.5 RQ8: Where do residual attribution errors arise?

Trajectory credit can obtain high recall by assigning positive credit to every write. Table 10 therefore reports precision and recall separately against the reference set P_q of Appendix H. That set is constructed after rollout from dataset gold units aligned to write identifiers and is never used as a training target.

Table 10: **RQ8: Attribution fidelity on controlled provenance.** Precision is the percentage of positively credited writes whose identifiers lie in the reference set P_q; recall is the percentage of identifiers in P_q that receive positive credit. P_q is constructed as in Appendix H and is unused during training; entries report mean±s.d. over five seeds.

Credit unit	Precision ↑	Recall ↑
Trajectory	31.2±2.3	**88.3±1.0**
Exact-state	44.8±2.0	51.5±2.2
ReMemR1	58.4±1.4	76.8±1.6
HiMPO	71.7±1.3	72.5±1.5
Retrieve-gated	65.2±1.3	83.0±1.4
UGPO	**78.6±1.1**	81.7±1.2

Trajectory credit recalls 88.3% of reference evidence writes but at 31.2% precision. Exact-state credit raises precision to 44.8% and loses recall because high-specificity store groups vanish. UGPO reaches 78.6% precision at 81.7% recall, above HiMPO on both axes and above retrieve-gated credit on precision.

Figure 8 decomposes remaining failures into missing writes, missed retrieval, unsupported use, downstream reasoning, and stale conflicts, then measures add, update, and delete precision separately.

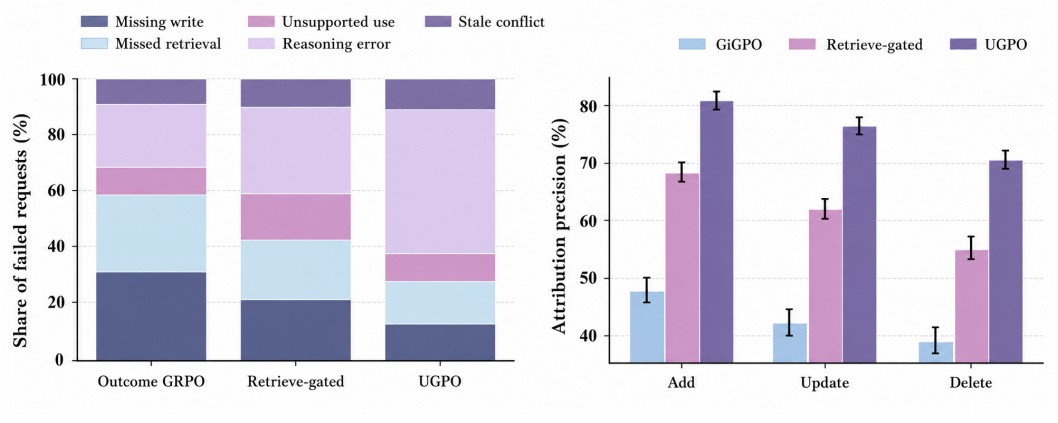


Figure 8: **RQ8: Error and operation analysis.** (a) Percentage composition of unsuccessful requests across missing-write, missed-retrieval, unsupported-use, reasoning, and stale-conflict errors for Outcome GRPO, Retrieve-gated, and UGPO. (b) Attribution precision (%) for Add, Update, and Delete operations under GiGPO, Retrieve-gated, and UGPO. Error bars denote standard deviation over five seeds.

Relative to Outcome GRPO, UGPO reduces missing-write and unsupported-use mass; residual errors concentrate in downstream reasoning, which the memory policy does not control. Panel (b) shows that identifier lineage carries credit through updates and deletes, not only first-time adds, matching the write, update, and delete eligibility used by responsibility routing.

I.6 RQ9: Is grounded use robust to citation gaming?

Grounded use is an operational attribution signal: a citation is credited only when the frozen verifier accepts the memory item as supporting the cited span. Table 11 compares that rule with retrieve-gated credit, citation-only credit, and a token-overlap verifier on sentence-level support annotations.

The NLI verifier reaches 91.8% precision and 89.6% recall, and the resulting credit precision is 13.4 points above retrieve-gated credit. Citation-only scoring is weaker than retrieve-gated scoring, because unsupported identifiers still receive use credit. Figure 9(a) then injects valid identifiers

Table 11: **RQ9: Grounding validation on controlled provenance.** Verifier precision and recall compare the predicted support relation against sentence-level annotations. Credit precision follows Table 10; unsupported citations follow Table 6.

Attribution rule	Verifier precision (%) ↑	Verifier recall (%) ↑	Credit precision (%) ↑	Unsupported citations (%) ↓
Retrieve-gated	–	–	65.2±1.3	14.9±1.2
Citation-only	–	–	59.4±1.8	18.1±1.6
Token-overlap verifier	84.7±1.2	88.9±1.1	74.2±1.4	7.9±0.8
Frozen NLI (UGPO)	**91.8±0.9**	**89.6±1.0**	**78.6±1.1**	**5.6±0.6**

whose items are topically similar but do not support the answer: false-positive credit stays below 17% for the full grounded-use rule when 40% of answer citations are injected.

I.7 RQ10: Does the grouping key require a tuned oracle?

The item-use key uses pre-action source metadata, frozen E5 clusters, request descriptors, and timestamps Wang et al. (2022). None of these fields contains sampled memory content or training reward. Table 12 perturbs the cluster or request-type field after the key has been fixed.

Table 12: **RQ10: Sensitivity to grouping-key misspecification.** Key perturbation randomly replaces the semantic-cluster or request-type field for the stated percentage of events. Coverage and attribution precision follow Table 7; task is the four-benchmark macro-average.

Grouping key	Non-singleton coverage ↑	Attribution precision (%) ↑	Average task (%) ↑
Domain-only coarse key	0.82±0.03	68.1±1.5	66.9±0.5
Same-request groups only	0.39±0.03	79.1±1.1	66.7±0.5
w/o Difficulty stratum	0.73±0.03	73.5±1.4	67.4±0.4
Fixed item-use key (UGPO)	**0.66±0.03**	**78.6±1.1**	**68.6±0.3**
10% perturbed fields	0.64±0.03	77.8±1.2	68.3±0.3
20% perturbed fields	0.61±0.04	75.9±1.4	67.9±0.4
40% perturbed fields	0.54±0.04	71.2±1.7	66.8±0.5
Random hash key	0.21±0.03	47.3±1.9	65.9±0.6

At 20% corruption, attribution precision remains 75.9% and average task 67.9. Removing the difficulty stratum increases coverage to 0.73 but lowers precision by 5.1 points. Same-request grouping preserves precision (79.1%) and loses delayed-write coverage (0.39). A random hash recreates state-like singleton behavior (coverage 0.21, precision 47.3%).

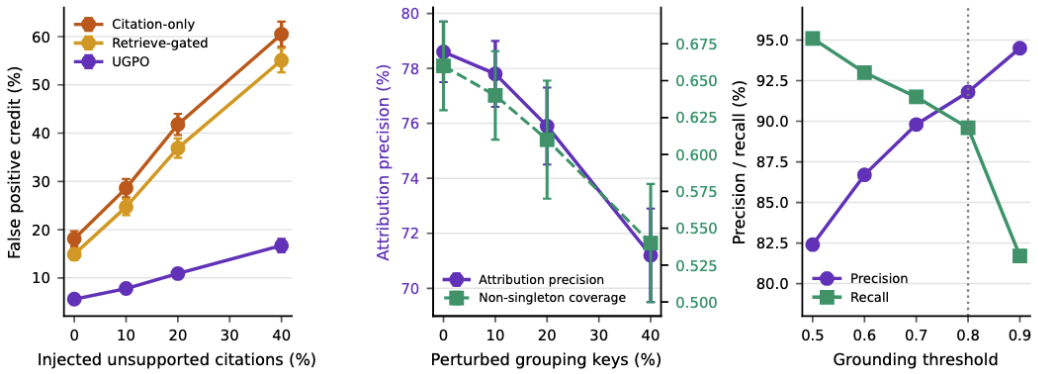


Figure 9: **RQ9–RQ10: Stress tests for the attribution proxy and grouping key.** (a) False positive credit under injection of topically similar but unsupported citations. (b) Attribution precision and non-singleton coverage as grouping-key fields are perturbed. (c) Controlled-provenance precision and recall across grounding thresholds; the dotted line marks the selected threshold. Error bars denote standard deviation over five seeds.

Figure 9(b) shows a gradual rather than cliff-shaped drop under field perturbation. Panel (c) places the selected grounding threshold on a broad precision–recall plateau, so the Table 11 result is not tied to a single cutoff.

I.8 STATISTICAL RELIABILITY OF THE MAIN COMPARISON

Table 13 reports hierarchical paired-bootstrap intervals for UGPO against HiMPO, the strongest baseline in Table 1. Resampling preserves matched examples within each seed, and the reported p-values use a Holm correction across the four benchmarks.

Table 13: **Paired significance analysis for RQ1.** Δ denotes the score difference between UGPO and HiMPO. Intervals contain 95% of the hierarchical paired-bootstrap distribution; corrected p-values test a zero median paired difference.

Benchmark	Score difference $\Delta \uparrow$	95% confidence interval	Corrected p-value
HotpotQA	$+0.9$	$[+0.6, +1.2]$	0.006
2Wiki	$+0.7$	$[+0.3, +1.1]$	0.014
LoCoMo	$+0.7$	$[+0.2, +1.2]$	0.021
LongMemEval	$+1.2$	$[+0.8, +1.6]$	0.004
Macro-average	$+0.9$	$[+0.6, +1.2]$	0.003

Every interval lies strictly above zero, including the smaller LoCoMo and 2Wiki margins ($+0.7$). The macro-average gain of $+0.9$ has corrected $p = 0.003$.

I.9 SAME REWARD, DIFFERENT ASSIGNMENT

Native Outcome GRPO and HiMPO differ from UGPO in both the reward they optimize and the operator that assigns that reward to tokens. Table 14 freezes the three channels of Equations 10–11 on the same WRU traces, then trains only the assignment operator: trajectory broadcast as in Outcome GRPO, write-local contrast as in HiMPO, exact-state groups, or item-use groups. Task, use, and cost weights remain $(0.60, 0.25, 0.15)$; backbone, token budget, retrieval depth, and seeds match Section 4.

Table 14: **Shared task, use, and cost rewards with different assignment operators.** All rows consume the same recorded R^{task}, R^{use}, and R^{cost} tensors. Trajectory broadcast copies the trajectory-level combination to every generated token. Write-local contrast assigns the item-level combination to the write that changed the store. Exact-state and item-use groups reuse Equations 3 and 8. Average task and attribution precision follow Tables 1 and 3.

Assignment operator	Average task (%) $\uparrow$	Attr. Prec. (%) $\uparrow$	Pollution (tokens) $\downarrow$
Trajectory broadcast	64.8 ± 0.6	40.9 ± 1.9	2260 ± 140
Write-local contrast	67.3 ± 0.4	73.9 ± 1.3	1160 ± 95
Exact-state groups	65.8 ± 0.7	44.8 ± 2.0	2490 ± 145
Item-use groups (UGPO)	$\mathbf{68.6 \pm 0.3}$	$\mathbf{78.6 \pm 1.1}$	$\mathbf{980 \pm 80}$

Giving Outcome GRPO the three UGPO channels raises average task from 63.0 (Table 1) to 64.8, which is still 3.8 points below item-use grouping. Write-local contrast with the same rewards reaches 67.3 and 73.9% attribution, close to native HiMPO in Tables 1 and 10, yet remains 1.3 task points and 4.7 attribution points behind item-use groups. Exact-state grouping of the same tensor reproduces the exact-state row of Table 3 (65.8 task, 44.8% attribution). The remaining gap is therefore the assignment operator, not a richer reward.

I.10 GROUNDED BUT HARMFUL MEMORY

Responsibility routing in Equation 7 withholds negative task credit from a write when the downstream answer fails. That rule protects a correct item that is grounded in a later mistake: the write receives no task penalty, and the unsuccessful outcome stays on answer tokens. It does not inspect whether a grounded item is stale or inconsistent with other stored items. Table 15 records four constructed traces that isolate this distinction.

Table 15: **Constructed traces for grounded but harmful credit.** $g = 1$ means the frozen verifier accepts the cited span. Task-to-write is the sign of R^{task} on the originating write after Equation 7; use-channel is the sign of Equation 10.

Constructed case	$g = 1$	Task to write	Use channel	Write penalized
Grounded evidence, failed answer	yes	0	$+$	no
Stale fact, cited span still entailed	yes	$+$	$+$	no
Two conflicting items, both entailed	yes	$+$	$+$	no
Unsupported citation of a live item	no	0	$-$	no

In the first row, a later reasoning error does not send negative task credit to the write that supplied the cited evidence. In the last row, an unsupported citation is charged on the use channel and never opens a task path to the write. The middle two rows are the complementary failure: a stale address or a pair of contradictory facts can both satisfy $g(m_i, y_q) = 1$ if the cited sentence is still locally entailed. They then receive positive use credit, and they receive positive task credit when the answer happens to match the cited span. Figure 8 already places residual mass on stale-conflict errors; those errors are exactly this class.

The limitation is the support predicate. Equation 6 tests whether a materialized item supports a cited span, not whether the item is current or consistent with the rest of the store. Routing therefore does not treat a harmful grounded item as a write-level fault, and a freshness or conflict check would have to enter as an additional channel rather than as a reinterpretation of g.