CREDIT-AWARE MEMORY POLICY OPTIMIZATION FOR SELF-EVOLVING BLACK-BOX AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving agents improve online by reusing experience accumulated in a memory of past user interactions. Adapting the agent itself from deployment feedback requires model parameters that the strongest API-served models do not expose, so the practical alternative learns per-memory utilities that decide which memories are retrieved for the current state. Such utilities, however, are updated with a shared trajectory-level return, which produces Memory Credit Leakage: because co-retrieved memories are indistinguishable to a single outcome signal, irrelevant and even harmful entries inherit the credit of the useful memories retrieved with them and accumulate spurious reinforcement as deployment lengthens. We argue that a memory deserves retrieval only if including it improves the return over excluding it, a contrast that no shared return can reveal. Therefore, we propose Credit-Aware Memory Policy Optimization (CMPO), an online framework that recovers this contrast from a single rollout by randomizing what the black-box agent sees rather than how it acts. CMPO exposes each memory through a stochastic gate with a recorded inclusion probability, so one rollout already contains both included and excluded memories, and reweighting the joint return by these probabilities yields a per-memory comparison that updates the gate, exposing useful memories more often and suppressing harmful ones. We prove that this credit is an unbiased estimator of the gate-local inclusion–exclusion return contrast and equals the derivative of the gate-local objective with respect to the inclusion probability. Across three benchmarks covering terminal interaction, embodied planning, and tool use, CMPO ranks first in 17 of 24 white-box and 20 of 24 black-box settings, sustaining accurate credit attribution over long-horizon deployment.

1 INTRODUCTION

Large language model (LLM) agents exhibit strong capabilities across complex tasks such as web navigation, mathematical reasoning, tool use, and question answering (Zhou et al., 2023; Liu et al., 2024), yet many deployed agents cannot learn from ongoing interactions. Self-evolving agents address this limitation by learning from the successful and failed trajectories collected during deployment (Liu et al., 2025; Suzgun et al., 2026; Jin et al., 2026). External memory provides a practical mechanism for self-evolution: agents distill prior interactions into reusable experiences and retrieve relevant memories into the model context for subsequent tasks (Shinn et al., 2023; Zhao et al., 2024; Liu et al., 2025; Luo et al., 2026). Unlike post-training, external memory permits continuous adaptation without altering model parameters or degrading base capabilities (Suzgun et al., 2026; Xiong et al., 2026).

Existing methods fall into two categories. **(i) Model-centric methods** optimize the agent policy on stored trajectories and task feedback through reinforcement learning with return- or advantage-based objectives (Zhou et al., 2024; Yu et al., 2026; Ma et al., 2026). **(ii) Memory-centric methods** optimize only the memory contents, per-memory utility estimates, and retrieval policies (Ouyang et al., 2026; Zhang et al., 2026b; Yang et al., 2026; Yan et al., 2026b). Model-centric methods directly reshape the action distribution but require access to parameters, gradients, or action log-probabilities (Qi et al., 2025; Suzgun et al., 2026), which excludes high-performance black-box models served exclusively through APIs. Memory-centric methods avoid white-box access, yet they rely solely on memory valuation, so errors in this valuation propagate and compound across interactions (Xiong et al., 2026). Fig. 1 illustrates these complementary limitations in a long-horizon interaction, where memory-centric methods must attribute a single joint return to multiple co-retrieved memories. Prevailing memory-centric methods sidestep this attribution by regressing the utility or Q-value of every co-

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.

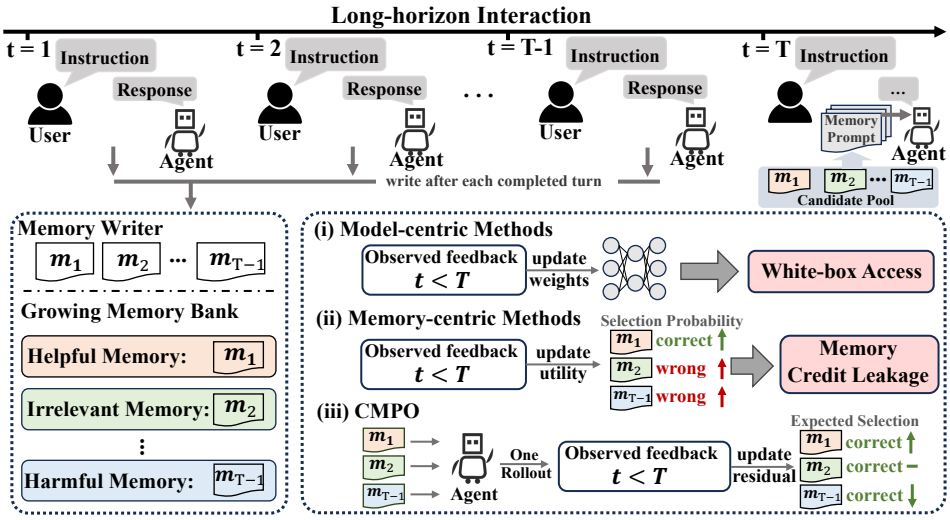


Figure 1: **Complementary limitations of self-evolving agents.** A memory writer distills interactions into a bank containing helpful (m_1), irrelevant (m_2), and harmful (m_{T-1}) memories. Model-centric methods demand white-box access, whereas memory-centric methods regress all co-retrieved memories toward a single joint return, causing memory credit leakage. In contrast, CMPO updates per-memory residuals using propensity-corrected credit derived from exactly one rollout per interaction.

retrieved memory toward the same trajectory-level return (Zhang et al., 2026b; Yang et al., 2026). Section 2.2 shows that such updates converge to the return observed whenever a memory is included, rather than to its actual contribution, a systematic error we term **memory credit misattribution**. In Fig. 1, where m_i denotes the i-th stored memory, the trajectory at round T succeeds because of the helpful memory m_1, whereas the co-retrieved memories m_2 and m_{T-1} are behaviorally inert and harmful, respectively. Nevertheless, the shared-return update raises the utilities of all three, because the joint return evaluates the entire trajectory without revealing individual contributions. In offline attribution, this error would remain a one-off estimate, but in online memory systems, the inflated utilities govern future retrieval. *Once an irrelevant or harmful memory acquires positive utility, it is retrieved more frequently, co-occurs with more successful trajectories, and receives further spurious reinforcement, turning a single attribution error into a persistent bias in future memory exposure.* This misattribution can be cast as a structural credit-assignment problem (Pignatelli et al., 2023), yet classical remedies cannot break this cycle. Counterfactual baselines (Foerster et al., 2018; Mesnard et al., 2021) require a critic that evaluates the inaccessible agent policy, while Shapley-style attribution and explicit ablation demand exponentially many subset evaluations or additional rollouts per decision (Shapley et al., 1953; Ghorbani & Zou, 2019), which is prohibitive under the per-call pricing of high-performance black-box models (e.g., the GPT 5.6, Claude 5, and Gemini 3 series).

We argue that a memory deserves retrieval only if including it improves the return over excluding it, which a shared trajectory-level return cannot reveal. Such a comparison remains feasible even for a black-box agent, because we can intervene on which memories enter its context. When each memory is included with a recorded probability, a single rollout suffices to compare the expected returns of including versus excluding that memory. Building on this observation, we propose Credit-Aware Memory Policy Optimization (CMPO), an online framework for self-evolving black-box agents that couples two components. **(a) Policy-guided memory selection** combines semantic relevance with learned per-memory residuals to sample memories under a token budget and records the inclusion probability of every gate. **(b) Propensity-corrected credit optimization** uses these recorded probabilities to convert the observed return into signed per-memory credit, which updates the residuals to refine subsequent selection. We prove that the expected credit is an unbiased estimator of the gate-local return contrast between inclusion and exclusion conditional on the pre-gate history, and that it equals the derivative of the gate-local objective with respect to the inclusion probability. CMPO thus promotes helpful memories and suppresses harmful ones without modifying agent parameters or requiring additional rollouts. In summary, the main contributions are:

- We formalize memory credit leakage, showing that updating every co-retrieved memory with the same trajectory-level return drives its learned utility toward the return observed whenever it is

included rather than toward its own contribution, which inflates irrelevant and harmful entries and biases future exposure.

- We propose CMPO, an online framework that removes this leakage for black-box agents. CMPO samples the inclusion of each memory from a stochastic gate and records the probability used, so a single rollout contains both included and excluded memories. Reweighting the joint return by these probabilities turns it into a per-memory credit that recovers the actual contribution of each memory.
- We prove that this credit is an unbiased estimator of the gate-local contrast between including and excluding a memory and equals the derivative of the gate-local objective with respect to the inclusion probability, so in expectation each update raises the inclusion probability of a memory that improves the return and lowers it otherwise.
- Experiments on three benchmarks covering terminal interaction, embodied planning, and tool using show that CMPO ranks first in 17/24 white-box and 20/24 black-box evaluation settings, confirming that removing memory credit leakage yields substantial and consistent downstream gains.

2 Problem Formulation

We formalize external-memory control for a black-box agent as a sequential decision problem (Zhou et al.; Zhang et al., 2026b), since selected memories influence the current action and, through environment transitions and memory updates, subsequent states and rewards. We first define the memory policy and its objective, then show that prevailing shared-return utility updates target an inclusion-conditioned quantity that leaks joint credit into individual memories.

2.1 Memory Policy for Self-Evolution

A session contains T task-level interactions indexed by $t \in \{1, \dots, T\}$. At step t, the agent observes a state s_t from the state space $\mathcal{S}$, and the memory bank $\mathcal{M}_t = \{m_i\}_{i=1}^{|\mathcal{M}_t|}$ collects memories m_i distilled from earlier interactions. The memory policy ρ_{θ_t} with parameters θ_t samples a possibly empty subset $Y_t \subseteq \mathcal{M}_t$, presented chronologically from oldest to newest (Park et al., 2023; Shinn et al., 2023). Conditioned on the state and the presented memories, the agent produces its action,

$$Y_t \sim \rho_{\theta_t}(\cdot \mid s_t, \mathcal{M}_t), \qquad a_t \sim \pi_\phi(\cdot \mid s_t, Y_t), \tag{1}$$

where a_t denotes the complete task-solving rollout and π_ϕ is the internal agent policy with parameters ϕ. The agent is queried only through an API, so ϕ, gradients, and action log-probabilities are inaccessible. Given the reward function $\mathcal{R}$ and the transition kernel $\mathcal{P}$, the reward and the next state satisfy $r_t = \mathcal{R}(s_t, a_t)$ and $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$. Task scores and user feedback are normalized to $[0, 1]$, which underlies the bounded credit magnitude in Section 3.2.

All compared methods share a fixed memory writer $\mathcal{W}$ (Shinn et al., 2023; Zhang et al., 2026b), which distills a successful interaction into a reusable procedure and a failed one into a reflection on unsuccessful actions, yielding $\mathcal{M}_{t+1} = \mathcal{W}(\mathcal{M}_t, s_t, a_t, r_t)$. The writer is not optimized, and this work studies only which memories to expose. Because selection affects subsequent interactions through both $\mathcal{P}$ and $\mathcal{W}$, while feedback must arrive within the session to support online updates, we evaluate each selection with the finite-horizon return $G_t^{(H)} = \sum_{\tau=t}^{\min(t+H-1,T)} \gamma^{\tau-t} r_\tau$ (Sutton et al., 1998), where τ indexes the summed steps, $\gamma \in (0, 1]$ is the discount factor, and $H \geq 1$ is the return horizon truncated at the session boundary. The memory policy maximizes the expected return:

$$J_{\mathrm{Mem},t}(\theta_t) = \mathbb{E}_{\rho_{\theta_t}, \pi_\phi, \mathcal{P}} \left[G_t^{(H)} \mid s_t, \mathcal{M}_t \right], \tag{2}$$

where the expectation is taken over the current memory selection Y_t, the black-box agent, environment transitions, memory writing, and the continuation induced by scheduled online updates. Here ρ_{θ_t} denotes the rollout distribution initialized at the current parameters θ_t, rather than a policy whose parameters stay fixed across each overlapping H-interaction window.

2.2 Memory Credit Leakage

Prevailing memory-centric methods instantiate the memory policy with per-memory scores, setting $\theta_t = \{u_{t,i}\}_{i=1}^{|\mathcal{M}_t|}$, where $u_{t,i}$ is the utility or Q-value of memory m_i before step t, and select Y_t by ranking or sampling these utilities. The utilities are updated with a shared episode-level signal such as a task return, bootstrapped return, or contrastive performance gap (Liao et al., 2026; Zhang et al., 2026b;d;c), so co-selected memories receive no separate environmental feedback, which instantiates the structural credit-assignment problem. We isolate this pattern through its Monte Carlo form (Sutton et al., 1998), which moves the utility of every selected memory toward the same return:

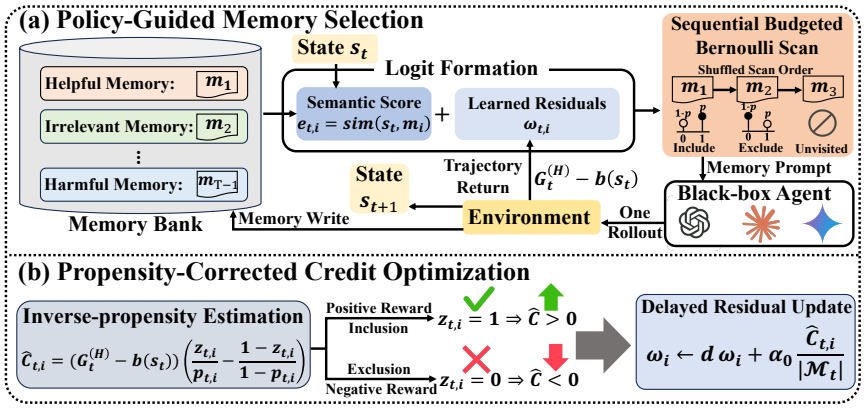


Figure 2: **Overview of CMPO. (a) Policy-guided memory selection** scores each memory by semantic relevance plus a learned residual and samples inclusion via a Bernoulli gate, producing one rollout from the black-box agent. **(b) Propensity-corrected credit optimization** turns the observed return into signed per-memory credit that drives a delayed residual update.

$$u_{t+1,i} = u_{t,i} + \eta_u \mathbb{I}[m_i \in Y_t] \left(G_t^{(H)} - u_{t,i}\right), \tag{3}$$

where $\eta_u \in (0, 1]$ is the update step size and $\mathbb{I}[m_i \in Y_t]$ indicates whether m_i is selected. The index $t + 1$ denotes the value after incorporating the feedback for step t, which may arrive only after H steps. Let $\mu_{t,i}^{\text{raw}} = \mathbb{E}[G_t^{(H)} \mid m_i \in Y_t, s_t, \mathcal{M}_t]$ denote the inclusion-conditioned return and $\bar{p}_{t,i} = \Pr(m_i \in Y_t \mid s_t, \mathcal{M}_t)$ the marginal inclusion probability of m_i. Taking the conditional expectation of Eq. 3 gives

$$\mathbb{E}\left[u_{t+1,i} - u_{t,i} \mid s_t, \mathcal{M}_t\right] = \eta_u \bar{p}_{t,i} \left(\mu_{t,i}^{\text{raw}} - u_{t,i}\right). \tag{4}$$

The actual contribution of m_i is instead measured by its marginal incremental value, the difference between the inclusion- and exclusion-conditioned returns:

$$\Delta_{t,i} = \mathbb{E}\left[G_t^{(H)} \mid m_i \in Y_t, s_t, \mathcal{M}_t\right] - \mathbb{E}\left[G_t^{(H)} \mid m_i \notin Y_t, s_t, \mathcal{M}_t\right]. \tag{5}$$

Writing $\mu_{t,i}^{\text{excl}} = \mathbb{E}[G_t^{(H)} \mid m_i \notin Y_t, s_t, \mathcal{M}_t]$ for the exclusion-conditioned return yields the decomposition $\mu_{t,i}^{\text{raw}} = \Delta_{t,i} + \mu_{t,i}^{\text{excl}}$. The following proposition shows that the shared-return update always targets this sum rather than $\Delta_{t,i}$. **See Appendix C.1 for the detailed proof.**

Proposition 1 (Biased target of shared-return updates)**.** *At every step t and for every memory m_i with $\bar{p}_{t,i} > 0$, the expected update in Eq. 4 moves $u_{t,i}$ toward $\mu_{t,i}^{\text{raw}}$ and vanishes if and only if*

$$u_{t,i} = \mu_{t,i}^{\text{raw}} = \Delta_{t,i} + \mu_{t,i}^{\text{excl}}. \tag{6}$$

The learned utility therefore overstates the contribution of m_i by $\mu_{t,i}^{\text{excl}}$, a term aggregating the effects of co-selected memories. **This decomposition is the formal content of memory credit leakage.** An inert memory with $\Delta_{t,i} = 0$ still acquires utility $\mu_{t,i}^{\text{excl}}$, and a harmful memory with $\Delta_{t,i} < 0$ remains positively valued whenever $\mu_{t,i}^{\text{excl}} > -\Delta_{t,i}$. Preventing leakage requires an update target that reflects the value of each inclusion decision rather than the aggregate return of co-selected memories. **The marginal contrast $\Delta_{t,i}$ captures this value but marginalizes over the full subset-selection distribution, so it cannot be identified from a single rollout without additional structure.**

3 Methodology

This section details CMPO, whose two components are illustrated in Fig. 2: **(a) policy-guided memory selection** samples memories and records the inclusion probability of every gate, and **(b) propensity-corrected credit optimization** converts the observed return into signed per-memory credit for online policy updates. We prove that the expected credit recovers the gate-local incremental value of each memory and aligns residual updates with the policy gradient in expectation.

3.1 Policy-Guided Memory Selection

Design Motivation. Section 2.2 shows that preventing leakage requires removing the leaked term $\mu_{t,i}^{\text{excl}}$ from the update target, whereas the marginal contrast $\Delta_{t,i}$ cannot be identified from one rollout

because it marginalizes over the full subset-selection distribution. Inspired by inverse-propensity estimation under bandit feedback (Swaminathan & Joachims, 2015), CMPO samples each memory through a stochastic binary gate with a known probability, so one rollout already supplies the randomization needed to estimate a gate-local inclusion–exclusion contrast without bias.

As shown in Fig. 2(a), at step t, CMPO uses the state s_t, memory bank $\mathcal{M}_t$, and policy parameters θ_t from Section 2, where each memory m_i has token length $c_i = \mathrm{len}(m_i)$. Following prior memory-retrieval methods (Park et al., 2023; Zhang et al., 2026b; Liao et al., 2026), the semantic relevance of m_i is $e_{t,i} = \mathrm{sim}(s_t, m_i)$, where $\mathrm{sim}(\cdot,\cdot)$ computes the cosine similarity between the dense state embedding $\mathrm{emb}(s_t) \in \mathbb{R}^D$ and the memory embedding $\mathrm{emb}(m_i) \in \mathbb{R}^D$ of dimension D. This score is combined with a learnable residual $\omega_{t,i}$ to form the memory-policy logit

$$h_{t,i} = e_{t,i} + \omega_{t,i} - \log|\mathcal{M}_t|. \tag{7}$$

The semantic score $e_{t,i}$ captures state-dependent relevance, whereas $\omega_{t,i}$ accumulates corrections from interaction feedback as an intrinsic reliability prior, and CMPO instantiates the policy parameters in Eq. 1 as $\theta_t = \{\omega_{t,i}\}_{i=1}^{|\mathcal{M}_t|}$. Each new or rewritten memory receives a new identifier with $\omega_{t,i} = 0$, deleting a memory removes its residual, and $Y_t = \emptyset$ when $\mathcal{M}_t = \emptyset$. The correction $-\log|\mathcal{M}_t|$ keeps the expected exposure of neutral memories below one. CMPO allocates B_{mem} prompt tokens to external memories and denotes by $B_{t,j}$ the remaining budget after scan position j, with $B_{t,0} = B_{\mathrm{mem}}$. Before selection, CMPO samples a uniformly random permutation $\varpi_t = \mathrm{Shuffle}(1, \ldots, |\mathcal{M}_t|)$ and scans the bank in this order, preventing tight budgets from systematically favoring memories placed early in a fixed order. At position j, the gate of the visited memory $i = \varpi_t(j)$ is executed only if $c_i \le B_{t,j-1}$, and the indices of all executed gates form the set $\mathcal{V}_t$. For each $i \in \mathcal{V}_t$, CMPO samples

$$p_{t,i} = \mathrm{clip}\left(\sigma(h_{t,i}), \frac{\epsilon_0}{|\mathcal{M}_t|}, 1 - \frac{\epsilon_0}{|\mathcal{M}_t|}\right), \qquad z_{t,i} \sim \mathrm{Bernoulli}(p_{t,i}), \tag{8}$$

where $p_{t,i}$ is the conditional inclusion probability, $z_{t,i} \in \{0,1\}$ is the binary gate outcome, $\sigma(\cdot)$ is the sigmoid function, $\mathrm{clip}(x, l, u)$ restricts x to $[l, u]$, and $\epsilon_0 \in (0, \frac{1}{2})$ controls exploration. The gate assigns probabilities $p_{t,i}$ and $1 - p_{t,i}$ to the inclusion and exclusion branches used for propensity correction. The remaining budget then becomes $B_{t,j} = B_{t,j-1} - z_{t,i}c_i$, so $z_{t,i} = 1$ includes m_i and deducts c_i tokens, whereas $z_{t,i} = 0$ leaves the budget unchanged. Clipping keeps both branch probabilities strictly positive and the inverse-propensity weights finite, and scaling the margin by $\frac{1}{|\mathcal{M}_t|}$ caps the total exploration mass at ϵ_0 rather than letting it grow linearly with the memory bank size. After the scan, $\mathrm{Order}_{\mathrm{time}}(\cdot)$ arranges the included memories in the chronological order defined in Section 2, forming the prompt sent to the black-box agent in Fig. 2(a),

$$Y_t = \mathrm{Order}_{\mathrm{time}}\left(\{m_i \mid i \in \mathcal{V}_t,\ z_{t,i} = 1\}\right). \tag{9}$$

3.2 Propensity-Corrected Credit Optimization

Design Motivation. Above selection produces the inclusion–exclusion variation required for credit estimation, but it does not translate the joint return into distinct updates for included and excluded memories. As shown in Fig. 2(b), CMPO therefore assigns each executed gate a signed credit that is corrected by its recorded propensity. In expectation, this credit raises the residual, and hence the inclusion probability, of a memory whose inclusion improves the return and lowers it otherwise.

To suppress estimator variance without introducing bias, CMPO maintains an online state-conditioned value baseline $b(s_t) = \mathbf{w}_t^\top \mathrm{emb}(s_t) + b_t$, where $\mathbf{w}_t \in \mathbb{R}^D$ is a weight vector and $b_t \in \mathbb{R}$ is a scalar offset, initialized to $\mathbf{w}_1 = \mathbf{0}$ and $b_1 = H/2$ as an uninformative prior. Reusing the state embedding $\mathrm{emb}(s_t)$ computed during retrieval, without additional API queries, CMPO updates these parameters online via single-step gradient descent on the squared loss $(b(s_t) - G_t^{(H)})^2$ with step size η_b:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_b \left(b(s_t) - G_t^{(H)}\right) \mathrm{emb}(s_t), \qquad b_{t+1} \leftarrow b_t - \eta_b \left(b(s_t) - G_t^{(H)}\right). \tag{10}$$

Each rollout realizes either inclusion or exclusion of a gate, but not both. To estimate their expected-return difference from this single observed outcome, CMPO weights the baseline-subtracted return by the inverse probability of the realized branch and assigns opposite signs to the two branches, which connects the inverse-propensity formulation to the score-function policy gradient (Williams, 1992):

$$\widehat{C}_{t,i} = \left(G_t^{(H)} - b(s_t)\right)\left(\frac{z_{t,i}}{p_{t,i}} - \frac{1 - z_{t,i}}{1 - p_{t,i}}\right) = \left(G_t^{(H)} - b(s_t)\right) \frac{\partial}{\partial p_{t,i}} \log \mathrm{Bern}(z_{t,i}; p_{t,i}), \quad i \in \mathcal{V}_t, \tag{11}$$

where $\mathrm{Bern}(z;p) = p^z(1-p)^{1-z}$ and $b(s_t)$ is clipped to $[0, H]$. The credit reduces to $\frac{G_t^{(H)} - b(s_t)}{p_{t,i}}$ when $z_{t,i} = 1$ and to $-\frac{G_t^{(H)} - b(s_t)}{1-p_{t,i}}$ when $z_{t,i} = 0$, with $\widehat{C}_{t,i} = 0$ if $i \notin \mathcal{V}_t$. Unlike Eq. 3, this estimator credits both gate outcomes with signed, variance-reduced values instead of attributing the full trajectory return only to retrieved memories. The return $G_t^{(H)}$ becomes available after interaction $\min\{t + H - 1, T\}$, with its horizon truncated near the end of the session, so CMPO caches the memory identifier, $z_{t,i}$, $p_{t,i}$, $b(s_t)$, and $|\mathcal{M}_t|$ for each executed gate until that interaction. The delayed residual update for credit originating from interaction t is applied when $G_t^{(H)}$ becomes available:

$$\omega_i \leftarrow d\,\omega_i + \alpha_0 \frac{\widehat{C}_{t,i}}{|\mathcal{M}_t|}, \qquad i \in \mathcal{V}_t, \tag{12}$$

where ω_i is the residual stored for m_i, α_0 is the step size, and d is the retention factor. CMPO uses $d = 1$ by default to retain historical credit, whereas $d \in (0, 1)$ discounts older credit when decay is desired. If several truncated returns become available at the final interactions, their credit increments are accumulated after a single retention step. Because Eq. 8 ensures $p_{t,i}, 1 - p_{t,i} \geq \frac{\epsilon_0}{|\mathcal{M}_t|}$, the normalized credit satisfies $\frac{|\widehat{C}_{t,i}|}{|\mathcal{M}_t|} \leq \frac{|G_t^{(H)} - b(s_t)|}{\epsilon_0}$, so the worst-case update magnitude remains independent of the memory bank size. **Appendix B provides the complete algorithm.**

3.3 Properties of Propensity-Corrected Credit

We analyze gates executed under the token budget. For $i \in \mathcal{V}_t$, let $\mathcal{H}_{t,i}$ denote all information fixed before sampling $z_{t,i}$, including s_t, $\mathcal{M}_t$, the permutation prefix, earlier gate outcomes, the baseline $b(s_t)$, and the remaining budget. Conditional on $\mathcal{H}_{t,i}$, the propensity $p_{t,i}$ is fixed and $z_{t,i} \sim \mathrm{Bernoulli}(p_{t,i})$. The gate-local counterpart of the incremental value in Eq. 5 is

$$\Delta_{t,i}^{\mathrm{gate}} = \mathbb{E}\left[G_t^{(H)} \mid \mathcal{H}_{t,i}, z_{t,i} = 1\right] - \mathbb{E}\left[G_t^{(H)} \mid \mathcal{H}_{t,i}, z_{t,i} = 0\right], \tag{13}$$

which contrasts inclusion with exclusion under the same pre-gate history $\mathcal{H}_{t,i}$ rather than the marginal selection distribution. This gate-local contrast is the operative decision margin of the policy, measuring the value of including m_i under the actual budget. When the budget rarely binds, $\Delta_{t,i}^{\mathrm{gate}}$ averaged over permutations coincides with $\Delta_{t,i}$, and otherwise it additionally captures competitive effects that $\Delta_{t,i}$ abstracts away (Appendix C, Eq. 26). The following proposition shows that the credit in Eq. 11 estimates this quantity and aligns with the direction of objective improvement.

Proposition 2 (Unbiased and score-aligned credit)**.** *For every executed gate $i \in \mathcal{V}_t$ and any baseline $b(s_t)$ measurable with respect to $\mathcal{H}_{t,i}$,*

$$\mathbb{E}\left[\widehat{C}_{t,i} \mid \mathcal{H}_{t,i}\right] = \Delta_{t,i}^{\mathrm{gate}} = \frac{\partial}{\partial p_{t,i}} \mathbb{E}\left[G_t^{(H)} \mid \mathcal{H}_{t,i}\right]. \tag{14}$$

Proposition 2 confirms that baseline subtraction preserves exact unbiasedness because $\mathbb{E}[\nabla_{p_{t,i}} \log \mathrm{Bern}(z_{t,i}; p_{t,i}) \mid \mathcal{H}_{t,i}] = 0$. The result does not require future residuals to remain numerically fixed: scheduled updates may depend on the realized gate outcome $z_{t,i}$ and intervening rewards, provided the continuation algorithm depends on the propensity $p_{t,i}$ only through $z_{t,i}$ before $G_t^{(H)}$ is completed. Expanding the expected return as $\mathbb{E}[G_t^{(H)} \mid \mathcal{H}_{t,i}] = p_{t,i}\,\mathbb{E}[G_t^{(H)} \mid \mathcal{H}_{t,i}, z_{t,i} = 1] + (1 - p_{t,i})\,\mathbb{E}[G_t^{(H)} \mid \mathcal{H}_{t,i}, z_{t,i} = 0]$ shows that the two branch returns are independent of $p_{t,i}$, so differentiating with respect to $p_{t,i}$ yields exactly $\Delta_{t,i}^{\mathrm{gate}}$. The proposition thus serves as the constructive counterpart of Proposition 1, replacing the leaked target $\Delta_{t,i} + \mu_{t,i}^{\mathrm{excl}}$ with the gate-local incremental value alone, and remains valid under memory interactions because both conditional returns absorb downstream randomness. **The proof and variance derivation are provided in Appendix C.2.**

4 Experiments

This section evaluates CMPO on three datasets spanning long-horizon interaction, planning, and tool-use tasks, addressing the following research questions:

- **RQ1:** How does CMPO compare with state-of-the-art methods in overall performance?
- **RQ2:** Does propensity-corrected credit recover the inclusion–exclusion value of individual memories and prevent credit leakage when helpful, irrelevant, and harmful memories are jointly exposed?

Table 1: **Overall performance (%).** The upper block evaluates CMPO against model-centric baselines on two white-box backbones (Llama-3.1-8B, Qwen3-8B), and the lower block evaluates CMPO against memory-centric baselines on two black-box LLMs (GPT-5.6-Luna, Claude-Sonnet-5). CMPO accesses only model inputs and outputs in both settings. LifelongAgentBench reports last-epoch success rate and cumulative success rate (Last/CSR), ALFWorld reports per-type success rate, and AppWorld reports Task Goal Completion (TGC) and Scenario Goal Completion (SGC).

Method	LifelongAgentBench		ALFWorld						AppWorld		LifelongAgentBench		ALFWorld						AppWorld	
	OS Last/CSR	DB Last/CSR	P&P	Examine	Clean	Heat	Cool	Pick-2	TGC	SGC	OS Last/CSR	DB Last/CSR	P&P	Examine	Clean	Heat	Cool	Pick-2	TGC	SGC
White-box	Llama-3.1-8B										Qwen3-8B									
Model-Centric Baselines																				
ETO	63.6/74.4	50.1/81.0	**86.1**	70.4	78.5	**79.7**	74.6	66.7	12.2	6.7	66.1/76.4	54.8/84.2	**86.1**	72.2	78.5	**81.2**	76.2	68.6	14.1	7.8
MEM1	65.5/76.8	53.1/83.5	83.3	74.1	79.6	76.8	76.2	**70.6**	**15.2**	8.9	67.4/78.8	57.6/85.9	81.9	77.8	80.6	79.7	77.8	**72.5**	16.7	10.0
JitRL	62.8/73.9	55.3/84.8	84.7	72.2	**80.6**	76.8	74.6	68.6	14.4	7.8	65.6/76.9	61.4/88.2	84.7	75.9	80.6	76.8	77.8	70.6	15.9	8.9
CMPO (Ours)	**67.0/78.2**	**57.0/86.0**	83.3	**75.9**	78.5	78.3	**77.8**	68.6	14.8	**10.0**	**70.5/82.3**	**62.1/90.8**	83.3	**79.6**	**81.7**	79.7	**79.4**	**72.5**	**18.5**	**12.2**
Black-box	GPT-5.6-Luna										Claude-Sonnet-5									
Memory-Centric Baselines																				
No Memory	65.2/–	55.6/–	88.9	79.6	86.0	84.1	85.7	78.4	24.1	17.8	67.1/–	54.4/–	90.3	81.5	87.1	82.6	84.1	76.5	25.9	20.0
RAG	68.4/72.9	56.1/81.8	87.5	85.2	87.1	85.5	85.7	74.5	18.9	12.2	70.0/74.8	55.0/81.0	88.9	83.3	86.0	84.1	85.7	78.4	21.1	14.4
Mem0	68.9/73.8	57.6/84.2	90.3	90.7	88.2	87.0	87.3	80.4	21.9	15.6	70.3/74.9	56.7/83.7	91.7	85.2	88.2	85.5	87.3	80.4	23.7	17.8
MemP	73.8/78.2	61.6/90.5	91.7	88.9	88.2	85.5	87.3	78.4	24.4	18.9	75.1/79.4	60.7/89.4	91.7	83.3	88.2	87.0	88.9	82.4	26.3	21.1
MemRL	77.9/80.8	62.9/91.8	94.4	83.3	89.2	**89.9**	87.3	84.3	29.6	23.3	79.2/82.0	61.8/90.8	93.1	85.2	89.2	85.5	87.3	84.3	31.1	24.4
MemQ	77.1/80.2	64.0/92.7	88.9	87.0	89.2	87.0	90.5	82.4	33.0	25.6	78.5/81.5	63.3/92.0	90.3	88.9	89.2	84.1	90.5	80.4	34.1	26.7
RoMeRL	80.7/83.1	66.2/94.0	**97.2**	92.6	90.3	88.4	88.9	82.4	30.0	26.7	81.5/83.8	65.7/93.5	**97.2**	92.6	**91.4**	87.0	90.5	82.4	32.2	27.8
CMPO (Ours)	**86.3/89.1**	**72.4/97.8**	95.8	**96.3**	**93.5**	88.4	**93.7**	**88.2**	**36.3**	**34.4**	**86.1/89.0**	**70.5/96.7**	95.8	**94.4**	90.3	**88.4**	**92.1**	**86.3**	**36.7**	**33.3**

- **RQ3:** Does CMPO adapt faster and reach a given performance level with fewer black-box rollouts?
- **RQ4:** What is the contribution of each component in CMPO?
- **RQ5:** How robust is CMPO to its core hyperparameters and operating conditions?
- **RQ6:** Can CMPO generalize to held-out tasks and unseen environments?

4.1 Experimental Setup

Tasks and Datasets. We evaluate CMPO across three interactive domains: (i) terminal interaction, using the Operating System (OS) and Database (DB) environments of LifelongAgentBench (Zheng et al., 2025); (ii) embodied household planning, using the six task types of ALFWorld (Shridhar et al., 2020b); and (iii) compositional tool use, using AppWorld (Trivedi et al., 2024), in which agents complete multi-step workflows over interconnected applications. Appendices D and L detail the benchmark environments and prompt templates.

Baselines. We compare CMPO against two categories of methods. Memory-centric baselines: (i) **No Memory**; (ii) **RAG**; (iii) **Mem0** (Chhikara et al., 2025); (iv) **MemP** (Fang et al., 2026); (v) **MemRL** (Zhang et al., 2026b); (vi) **MemQ** (Liao et al., 2026); and (vii) **RoMeRL** (Yang et al., 2026). Model-centric baselines: (i) **ETO** (Song et al., 2024); (ii) **MEM1** (Zhou et al., 2026); and (iii) **JitRL** (Li et al., 2026b). Further details are provided in Appendix E.

Evaluation Metrics. We report task success rate as the primary metric. For OS and DB, we report last-epoch success rate (Last) and cumulative success rate (CSR). For AppWorld, we include the benchmark-defined goal-condition accuracy. Appendix D provides further details on the evaluation.

Implementation Protocol. An interaction denotes a single task-level rollout, which may comprise multiple agent steps and model calls. Additional implementation details are provided in Appendix F.

4.2 Overall Performance (*RQ1*)

Table 1 compares CMPO with model-centric baselines on two white-box backbones and with memory-centric baselines on two black-box LLMs, **where all black-box comparisons follow a single-rollout-per-interaction protocol.** In the white-box setting, CMPO accesses only model inputs and outputs yet remains competitive with, and often surpasses, methods that require gradient access or logit manipulation. On Qwen3-8B, CMPO reaches 70.5/82.3 (Last/CSR) on OS and 62.1/90.8 on DB, and averages 1.0 point higher than MEM1 across ALFWorld task types, showing that optimizing memory exposure alone can rival parameter-level self-evolution. In the black-box setting, CMPO achieves the best aggregate performance on all three benchmarks under both LLMs. The gains are most pronounced on tasks involving memory co-retrieval (Examine +3.7, Pick-2 +3.9) and multi-step tool composition (TGC +3.3, SGC +7.8) under GPT-5.6-Luna, both of which are especially susceptible to credit leakage under trajectory-level reward sharing. These results indicate that propensity-corrected per-memory credit provides a consistent advantage over shared-scalar utility

Table 2: **Per-memory credit accuracy.** $\widehat{C}^{+}$, $\widehat{C}^{0}$, and $\widehat{C}^{-}$ denote the credit assigned to helpful, irrelevant, and harmful memories, respectively. MAE is computed against the factorial ground truth obtained from all eight subset combinations. Lower MAE indicates more accurate credit assignment.

Benchmark	Method	$\widehat{C}^{+}$	$\widehat{C}^{0}$	$\widehat{C}^{-}$	MAE
LAB	Ground Truth	24.8	-0.6	-18.9	–
	MemRL	63.5	52.6	39.8	50.2
	MemQ	66.4	43.7	29.5	44.8
	CMPO	28.1	1.7	-14.9	3.2
ALFWorld	Ground Truth	11.6	-1.2	-14.7	–
	MemRL	86.0	78.4	62.9	77.2
	MemQ	87.8	69.1	52.4	71.2
	CMPO	8.2	-3.8	-18.1	3.1
AppWorld	Ground Truth	18.9	-0.8	-8.4	–
	MemRL	31.7	25.4	18.9	22.1
	MemQ	35.3	21.8	14.2	20.5
	CMPO	23.5	2.9	-3.2	4.5

Table 3: **Memory selection.** E^{+}, E^{0}, and E^{-} denote the exposure rates of helpful, irrelevant, and harmful memories in the final 20% of interactions. Sep. measures the gap between helpful exposure and the average exposure of the other two types. Higher E^{+} and Sep., together with lower E^{0} and E^{-}, indicate cleaner selection.

Benchmark	Method	E^{+}	E^{0}	E^{-}	Sep.
LAB	MemRL	79.4	49.4	36.7	36.4
	MemQ	81.1	41.7	30.6	45.0
	RoMeRL	86.1	29.4	20.6	61.1
	CMPO	89.4	21.7	12.2	72.5
ALFWorld	MemRL	85.6	58.3	43.9	34.4
	MemQ	87.8	50.6	37.2	43.9
	RoMeRL	91.1	35.6	22.8	61.9
	CMPO	92.8	26.7	18.3	70.3
AppWorld	MemRL	69.4	59.4	48.3	15.6
	MemQ	75.0	51.7	41.1	28.6
	RoMeRL	79.4	40.6	27.8	45.3
	CMPO	84.4	30.6	20.0	59.2

updates across diverse task domains. **All reported numbers are averaged over three random seeds, and paired confidence intervals are reported in Appendix G.**

4.3 Credit Assignment and Leakage Prevention (*RQ2*)

We build a controlled memory-probing testbed shared by the two experiments below. For each task instance (15 OS, 15 DB, 30 ALFWorld, 30 AppWorld), we compose a length- and similarity-matched triplet, consisting of a *helpful* memory containing the correct key step, an *irrelevant* memory addressing a superficially similar but distinct problem, and a *harmful* memory introducing a critical error. All methods share the same task sequences and black-box backbone (GPT-5.6-Luna) and may update internal scores but cannot rewrite or generate entries. We abbreviate the LifelongAgentBench OS and DB environments jointly as LAB.

For **credit recovery** (Table 2), we remove the token budget and apply independent $\mathrm{Bernoulli}(0.5)$ gates, so all $2^K = 8$ subsets of $K{=}3$ memories are visited uniformly at a constant propensity of 0.5, and all methods share the same exposure logs and returns, differing only in credit computation. The factorial ground truth is defined as

$$C_i^{\star} = \frac{1}{2^{K-1}} \sum_{S \subseteq \mathcal{M} \setminus \{m_i\}} \big[\bar{G}(S \cup \{m_i\}) - \bar{G}(S)\big], \tag{15}$$

where $\bar{G}(S)$ is the mean return over 20 rollouts on subset S, used only for offline evaluation, and accuracy is measured by $\mathrm{MAE} = \frac{1}{K}\sum_{i=1}^{K} |\widehat{C}_i - C_i^{\star}|$ with $\widehat{C}_i$ the credit estimated for memory m_i. For **leakage prevention** (Table 3), each method runs its native selection and update mechanism for 10 epochs with one rollout per interaction, under a token budget that accommodates any pair but not the full triplet, so co-retrieval is frequent yet harmful entries remain excludable. We report exposure rates over the final 20% of interactions and the separation $\mathrm{Sep.} = E^{+} - (E^{0} + E^{-})/2$, averaged over instances, epochs, and three seeds, where E^{+}, E^{0}, and E^{-} denote the exposure rates of helpful, irrelevant, and harmful memories. **Credit Recovery Results.** In Table 2, MemRL and MemQ assign large positive credit to all three memory types regardless of causal utility, confirming the shared-return pathology in Section 2.2. On ALFWorld, MemRL yields $(86.0, 78.4, 62.9)$ against the oracle $(11.6, -1.2, -14.7)$ with an MAE of 77.2, which MemQ reduces only to 71.2. CMPO instead tracks the oracle closely, with MAEs of 3.2 on LAB (estimating $(28.1, 1.7, -14.9)$ vs. $(24.8, -0.6, -18.9)$), 3.1 on ALFWorld, and 4.5 on AppWorld, cutting MAE by 92.9%, 95.6%, and 78.0% relative to the strongest baseline on each benchmark. **Leakage Prevention Results.** Table 3 shows that accurate credit translates into cleaner long-term selection. After 10 epochs, CMPO attains the highest E^{+} and lowest E^{0}, E^{-} on all benchmarks, with separations of 72.5, 70.3, and 59.2 on LAB, ALFWorld, and AppWorld. MemRL exhibits the weakest separation (15.6–36.4) due to shared-return score inflation, and RoMeRL, despite improving separation via reduced-order states, leaves harmful exposure at 20.6–27.8 through its shared trajectory-level scalar, whereas CMPO suppresses E^{-} to 12.2–20.0 by crediting each memory with its own inclusion–exclusion contrast.

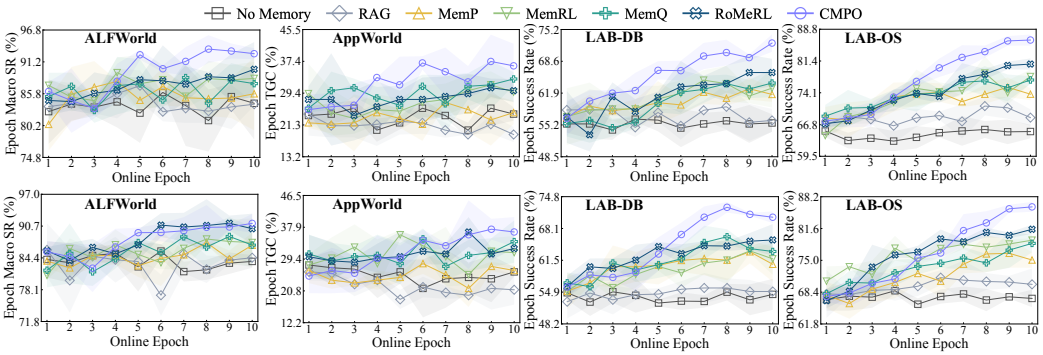


Figure 3: **Online adaptation curves.** Top row: GPT-5.6-Luna; bottom row: Claude-Sonnet-5. Each epoch contains one rollout per task, with the memory state carried across epochs. Curves show mean performance over three seeds, and shaded regions indicate one standard deviation.

4.4 Online Adaptation and Rollout Efficiency (*RQ3*)

To evaluate adaptation as the interaction budget grows, we run all memory-centric methods on GPT-5.6-Luna and Claude-Sonnet-5 across LAB-OS, LAB-DB, ALFWorld, and AppWorld for 10 epochs, where each epoch attempts every task instance exactly once (one rollout per task) and the learned memory state carries across epochs. Within each of three random seeds, all methods share identical task orders, prompt templates, retrieval budgets, and decoding settings. LAB is evaluated by per-epoch success rate, ALFWorld by macro-average success rate over six task types, and AppWorld by TGC, and the cost–performance trade-off is summarized by the normalized area under the learning curve, $\mathrm{AULC} = \frac{1}{9}\sum_{e=1}^{9}(P_e + P_{e+1})/2$, where P_e is the performance at epoch e.

Fig. 3 shows learning trajectories over ten epochs. During the early epochs, $b(s_t)$ may not yet have converged, potentially inflating credit variance (Prop. 3), and residuals may remain small, so selection relies more heavily on semantic scores. As feedback accumulates, CMPO generally improves, although the onset and persistence of its lead vary across backbones and benchmarks. From epoch 9 to 10, the largest CMPO gain is 3.07 percentage points under GPT-5.6-Luna and 0.72 under Claude-Sonnet-5. At epoch 10, CMPO exceeds the strongest baseline by 2.7–6.2 and 1.0–4.7 points across the four benchmarks under the two backbones. The AULC results indicate a favorable performance–interaction trade-off under matched task-level rollout budgets. CMPO attains the highest AULC on all four benchmarks under GPT-5.6-Luna, exceeding the strongest baseline by 2.3–4.4 points, and leads on two of four benchmarks under Claude-Sonnet-5. These results indicate that CMPO is not consistently superior under limited budgets but achieves higher mean final-epoch performance across the evaluated settings, supporting its potential for long-horizon deployment.

4.5 Related Work, Ablations, Generalization, Sensitivity, and Limitations

Due to space constraints, additional empirical analyses and discussions are deferred to the appendix. Appendix A surveys related literature. Appendix H presents component-wise ablations (RQ4). Appendix I evaluates parameter robustness across the exploration rate ϵ_0, step size α_0, retention factor d, memory budget B_{mem}, horizon H and discount factor γ (RQ5). Appendix J assesses generalization on held-out tasks (RQ6), and Appendix K discusses limitations and future directions.

5 Conclusion

In this paper, we identified memory credit leakage as a fundamental obstacle to agents that self-evolve through memory. Shared-return updates value each memory by the return observed whenever it is included rather than by its own contribution, so irrelevant and harmful entries are inflated and retrieval bias compounds over deployment. A memory instead deserves retrieval only if including it improves the return over excluding it, and this comparison stays obtainable for a black-box agent. We therefore proposed CMPO, which gates each memory with a recorded inclusion probability, turns a single rollout into signed per-memory credit, and provably yields an unbiased estimate of the gate-local inclusion–exclusion return contrast that equals the derivative of the gate-local objective with respect to the inclusion probability. Across three benchmarks covering terminal interaction, embodied planning, and tool use, CMPO ranks first in 17 of 24 white-box and 20 of 24 black-box settings and reduces the credit-estimation error of the strongest baseline by 78.0–95.6%.

AI USE STATEMENT

In this work, we used generative AI tools solely for language polishing, including grammar correction, sentence-level rewriting for clarity and conciseness, and LaTeX formatting suggestions on author-written drafts. We did not use generative AI tools for research ideation, problem formulation, theoretical derivation, algorithm design, experimental design, data analysis, or the generation of experimental results and figures. All research contributions, including the identification of memory credit leakage, the design of CMPO, the theoretical propositions and their proofs, and all reported empirical findings, were conceived and produced by the authors. Retrieval or literature discovery through generative AI tools is not applicable to this work, as related work was surveyed manually by the authors. We also used generative AI tools in a limited capacity for minor coding assistance, such as autocompletion of standard boilerplate and suggestions for plotting utilities. All AI-assisted text was reviewed and revised by the authors to ensure accuracy of technical claims and consistency with the reported results. We take full responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI tools.

REFERENCES

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Workshop on computer games*, pp. 41–75. Springer, 2018.

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.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, 2026.

Xinshun Feng, Xinhao Song, Lijun Li, Gongshen Liu, and Jing Shao. Searl: Joint optimization of policy and tool graph memory for self-evolving agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24518–24535, 2026.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pp. 2242–2251. PMLR, 2019.

Guangya Hao, Yunbo Long, and Zhuokai Zhao. Self-evolving multi-agent systems via decentralized memory, 2026. URL `https://arxiv.org/abs/2605.22721`.

Yuxin Jin, Siyuan Zhang, Hanchen Wang, Lu Qin, Ying Zhang, and Wenjie Zhang. Exg: Self-evolving agents with experience graphs. *arXiv preprint arXiv:2605.17721*, 2026.

Ruoran Li, Xinghua Zhang, Haiyang Yu, Shitong Duan, Xiang Li, Wenxin Xiang, Chonghua Liao, Xudong Guo, Yongbin Li, and Jinli Suo. Mempo: Self-memory policy optimization for long-horizon agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23286–23301, 2026a.

Yibo Li, Zijie Lin, Ailin Deng, Xuan Zhang, Yufei He, Shuo Ji, Tri Cao, and Bryan Hooi. Just-in-time reinforcement learning: Continual learning in llm agents without gradient updates. *arXiv preprint arXiv:2601.18510*, 2026b.

Junwei Liao, Haoting Shi, Ruiwen Zhou, Jiaqian Wang, Shengtao Zhang, Wei Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Bo Tang, et al. Memq: Integrating q-learning into self-evolving memory agents over provenance dags. *arXiv preprint arXiv:2605.08374*, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Yitao Liu, Chenglei Si, Karthik R Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14179–14198, 2025.

Ziyan Liu, Zhezheng Hao, Yeqiu Chen, Hong Wang, Jingren Hou, Ruiyi Ding, Yongkang Yang, Wence Ji, Wei Xia, and Feng Liu. Meta-cognitive memory policy optimization for long-horizon llm agents. *arXiv preprint arXiv:2605.30159*, 2026.

Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. From storage to experience: A survey on the evolution of llm agent memory mechanisms. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 41622–41652, 2026.

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 *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19666–19684, 2026.

Thomas Mesnard, Theophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Thomas S Stepleton, Nicolas Heess, Arthur Guez, Eric Moulines, Marcus Hutter, Lars Buesing, and Remi Munos. Counterfactual credit assignment in model-free reinforcement learning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7654–7664. PMLR, 18–24 Jul 2021.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, Olivier Pietquin, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *arXiv preprint arXiv:2312.01072*, 2023.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 79791–79821, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Tao Ren, Weiyao Luo, Hui Yang, Rongzhi Zhu, Xiang Huang, Yuchuan Wu, Bingxue Chou, Jieping Ye, Jiafeng Liang, Yongbin Li, and Yijie Peng. Scaling self-evolving agents via parametric memory, 2026. URL https://arxiv.org/abs/2606.04536.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Lloyd S Shapley et al. A value for n-person games. 1953.

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.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10737–10746. IEEE, 2020a.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020b.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for llm agents. 2024.

Richard S Sutton, Andrew G Barto, and Andrew Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, 2026.

Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *International conference on machine learning*, pp. 814–823. PMLR, 2015.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024.

Zi-Han Wang, Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Jie Wu, Zhengzhou Cai, Yueqing Sun, Ziang Ye, Linji Hao, Qi Gu, et al. Agentopsd: Recursive self-distillation for agentic reinforcement learning. *arXiv preprint arXiv:2608.05987*, 2026.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Han Xiao, Guozhi Wang, Hao Wang, Shilong Liu, Yuxiang Chai, Yue Pan, Yufeng Zhou, Xiaoxin Chen, Yafei Wen, and Hongsheng Li. Ui-mem: Self-evolving experience memory for online reinforcement learning in mobile gui agents. *arXiv preprint arXiv:2602.05832*, 2026.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts llm agents: An empirical study of experience-following behavior. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 623–645, 2026.

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. *arXiv preprint arXiv:2606.16285*, 2026a.

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. *arXiv preprint arXiv:2605.21768*, 2026b.

Yi Yang, Zhennan Chen, Yihong Zhuang, Tiehan Fan, Yinan Chen, Jian Li, Jian Yang, and Ying Tai. Romerl: Balancing feedback coverage and the memory-reward trap in self-evolving agent memory via reduced-order utility states. *arXiv preprint arXiv:2608.02508*, 2026.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent, 2025. *URL https://arxiv. org/abs/2507.02259*, 2259, 2025.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, 2026.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026a.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026b.

Xiaoying Zhang, Zichen Liu, Yipeng Zhang, Xia Hu, and Wenqi Shao. Retroagent: From solving to evolving via retrospective dual intrinsic feedback. *arXiv preprint arXiv:2603.08561*, 2026c.

Yaolun Zhang, Yiran Wu, Yijiong Yu, Qingyun Wu, and Huazheng Wang. Live-evo: Online evolution of agentic memory from continuous feedback. *arXiv preprint arXiv:2602.02369*, 2026d.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating llm agents as lifelong learners. *arXiv preprint arXiv:2505.11942*, 2025.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, et al. Memento: Fine-tuning llm agents without fine-tuning llms, 2025. *URL https://arxiv. org/abs/2508.16153*.

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

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *International Conference on Learning Representations*, volume 2026, pp. 58413–58438, 2026.

A Related Work

Model-Centric Self-Evolving Agents. Model-centric methods improve an agent by updating its policy parameters with deployment feedback. Early work optimizes the policy over stored trajectories through exploration-based trajectory optimization (Song et al., 2024) and hierarchical multi-turn reinforcement learning (Zhou et al., 2024), and online curriculum reinforcement learning scales this recipe to web agents (Qi et al., 2025). A parallel line trains the model itself to manage memory, either consolidating memory and reasoning into a compact internal state (Zhou et al., 2026) or learning an overwriting memory policy with RL (Yu et al., 2025). Recent methods interleave parameter updates with deployment (Suzgun et al., 2026), refine the policy with return- or advantage-based objectives over accumulated trajectories (Yu et al., 2026), assign fine-grained credit across turns (Ma et al., 2026), derive critic-free turn-level credit through recursive self-distillation (Wang et al., 2026; Liu et al., 2026), perform advantage attribution with adaptive rollouts in multi-turn tool use (Dong et al., 2026), or absorb distilled experience into adapter weights as parametric memory (Ren et al., 2026). All of these methods require access to parameters, gradients, or action log-probabilities, and repeated fine-tuning risks degrading existing capabilities, which excludes high-performance models served only through APIs. CMPO instead optimizes only external memory exposure, achieving online self-evolution for fully black-box agents with one rollout per interaction.

Memory-Centric Self-Evolving Agents. Memory-centric methods keep the agent frozen and accumulate reusable experience in an external memory, ranging from verbal reflections and distilled cross-task insights to relevance-based retrieval (Shinn et al., 2023; Zhao et al., 2024; Park et al., 2023; Zhang et al., 2026a). On the construction side, Mem0 builds production-scale long-term memory through extraction and consolidation (Chhikara et al., 2025), and Memp distills trajectories into learnable procedural memory (Fang et al., 2026). Later systems learn what to store and retrieve, including contextual experience reuse (Liu et al., 2025), storage-efficient banks (Luo et al., 2026), experience-guided exploration (Jin et al., 2026), memory-based learning without fine-tuning (Zhou et al.), strategy-level memory (Ouyang et al., 2026), self-evolving experience memory for online GUI agent learning (Xiao et al., 2026), and joint optimization of the policy and a tool-graph memory under sparse long-horizon rewards (Feng et al., 2026). To prioritize useful memories, recent methods attach utilities or Q-values updated from task returns (Zhang et al., 2026b; Liao et al., 2026; Li et al., 2026a), rewrite memories live or retrospectively (Zhang et al., 2026d;c), and reweight decentralized memory pools from judge feedback (Hao et al., 2026). Most related to our work, three concurrent methods explicitly target credit assignment across co-retrieved memories. RoMeRL identifies the memory-reward trap, in which co-retrieved memories share one outcome-level reward so that irrelevant entries are reinforced together with useful ones, and mitigates it by compressing utilities into reduced-order per-task states (Yang et al., 2026). Memory-R2 addresses the same difficulty by issuing local rerollouts from a common intermediate memory state and comparing the outcomes of alternative memory operations (Yan et al., 2026b). HiMPO reduces causally entangled outcome-level rewards through hindsight-informed local counterfactual utility (Yan et al., 2026a). We adopt this problem formulation rather than restate it, and CMPO differs in the optimization target, the access it assumes, and its rollout cost. CMPO optimizes a continuous exposure residual over a frozen external memory instead of discrete memory operations or per-task utility states, and it requires no parameter, gradient, or log-probability access. It randomizes memory exposure with recorded propensities, so an unbiased estimate of the inclusion–exclusion return contrast of each memory is obtained from the same single rollout, without local rerollouts, auxiliary trajectories, or hindsight relevance derived from a completed trajectory. In RoMeRL the signal that updates each reduced-order coordinate is still a trajectory-level scalar shared by the co-retrieved memories mapped to it, whereas in CMPO each memory receives its own randomization-defined contrast.

B Complete Algorithm of CMPO

Algorithm 1 summarizes the complete online procedure of CMPO, where one interaction t denotes a complete task-level rollout that may comprise multiple model generations, tool calls, and environment turns. Lines 3–17 perform memory selection and rollout: CMPO computes the baseline $b(s_t)$ from $(\mathbf{w}_t, b_t)$, forms a logit per memory (Line 6), scans the bank in random order ϖ_t (Lines 8–14) with Bernoulli gates that deduct tokens from the remaining budget B, and passes the included set Y_t to the black-box agent (Line 17). Line 18 caches the tuple $(t, i, z_{t,i}, p_{t,i}, b(s_t), \mathrm{emb}(s_t), |\mathcal{M}_t|)$ for every executed gate $i \in \mathcal{V}_t$ until the return $G_t^{(H)}$ matures at interaction $\min\{t{+}H{-}1,\ T\}$. Line 19 invokes the writer $\mathcal{W}$ to produce $\mathcal{M}_{t+1}$, initializing $\omega_i = 0$ for new memories and discarding cached entries

Algorithm 1 Credit-Aware Memory Policy Optimization (CMPO)

Require: black-box agent π_ϕ, memory writer $\mathcal{W}$, similarity sim, budget B_{mem}, exploration ϵ_0, step size α_0, baseline step size η_b, retention d, horizon H, discount γ

1: Initialize $\mathcal{M}_1$, $\omega_i \leftarrow 0$ for all $m_i \in \mathcal{M}_1$, $\mathbf{w}_1 \leftarrow \mathbf{0}$, $b_1 \leftarrow H/2$, and pending queue $\mathcal{Q} \leftarrow \emptyset$
2: **for** $t = 1$ **to** T **do**
3: Observe s_t, compute $\mathrm{emb}(s_t) \in \mathbb{R}^D$ and baseline $b(s_t) \leftarrow \mathbf{w}_t^\top \mathrm{emb}(s_t) + b_t$
4: $Y_t \leftarrow \emptyset$, $\mathcal{V}_t \leftarrow \emptyset$, $B \leftarrow B_{\mathrm{mem}}$
5: **if** $\mathcal{M}_t \neq \emptyset$ **then**
6: $h_{t,i} \leftarrow \mathrm{sim}(s_t, m_i) + \omega_i - \log|\mathcal{M}_t|, \forall m_i \in \mathcal{M}_t$ ▷ Eq. 7
7: $\varpi_t \leftarrow \mathrm{Shuffle}(1, \ldots, |\mathcal{M}_t|)$
8: **for** $j = 1$ **to** $|\mathcal{M}_t|$ **do**
9: $i \leftarrow \varpi_t(j)$
10: **if** $c_i \leq B$ **then**
11: $p_{t,i} \leftarrow \mathrm{clip}\Big(\sigma(h_{t,i}), \frac{\epsilon_0}{|\mathcal{M}_t|}, 1 - \frac{\epsilon_0}{|\mathcal{M}_t|}\Big)$; $z_{t,i} \sim \mathrm{Bernoulli}(p_{t,i})$ ▷ Eq. 8
12: $\mathcal{V}_t \leftarrow \mathcal{V}_t \cup \{i\}$; $B \leftarrow B - z_{t,i} c_i$
13: **end if**
14: **end for**
15: $Y_t \leftarrow \mathrm{Order}_{\mathrm{time}}(\{m_i \mid i \in \mathcal{V}_t,\ z_{t,i} = 1\})$ ▷ Eq. 9
16: **end if**
17: Execute one task-level rollout $a_t \sim \pi_\phi(\cdot \mid s_t, Y_t)$ and observe the task-level reward r_t
18: Add $(t, i, z_{t,i}, p_{t,i}, b(s_t), \mathrm{emb}(s_t), |\mathcal{M}_t|)$ to $\mathcal{Q}$ for all $i \in \mathcal{V}_t$
19: $\mathcal{M}_{t+1} \leftarrow \mathcal{W}(\mathcal{M}_t, s_t, a_t, r_t)$; initialize $\omega_i \leftarrow 0$ for newly inserted memories and remove from $\mathcal{Q}$ all entries associated with deleted memories
20: $\Delta\omega_i \leftarrow 0$ for all i; $\mathcal{U}_t \leftarrow \emptyset$; $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$; $b_{t+1} \leftarrow b_t$
21: **for** each origin time t' satisfying $\min\{t' + H - 1, T\} = t$ **do**
22: $G_{t'}^{(H)} \leftarrow \sum_{\tau=t'}^{t} \gamma^{\tau - t'} r_\tau$; $\delta \leftarrow b(s_{t'}) - G_{t'}^{(H)}$
23: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_{t+1} - \eta_b\, \delta\, \mathrm{emb}(s_{t'})$; $b_{t+1} \leftarrow b_{t+1} - \eta_b\, \delta$ ▷ Eq. 10
24: **for** each i whose cached entry in $\mathcal{Q}$ has origin time t' **do**
25: $\widehat{C}_{t',i} \leftarrow \Big(G_{t'}^{(H)} - \mathrm{clip}(b(s_{t'}), 0, H)\Big)\Big(\frac{z_{t',i}}{p_{t',i}} - \frac{1 - z_{t',i}}{1 - p_{t',i}}\Big)$ ▷ Eq. 11
26: $\Delta\omega_i \leftarrow \Delta\omega_i + \frac{\alpha_0 \widehat{C}_{t',i}}{|\mathcal{M}_{t'}|}$; $\mathcal{U}_t \leftarrow \mathcal{U}_t \cup \{i\}$; remove the entry from $\mathcal{Q}$
27: **end for**
28: **end for**
29: **for** each $i \in \mathcal{U}_t$ **do**
30: $\omega_i \leftarrow d\, \omega_i + \Delta\omega_i$ ▷ Eq. 12
31: **end for**
32: **end for**

of deleted ones. Lines 21–29 perform the delayed credit update: for every origin time t' whose return matures at t, the baseline parameters are updated by single-step gradient descent on the cached prediction error δ (Line 23), and each entry cached at that same t' yields a propensity-corrected credit accumulated into $\Delta\omega_i$ (Lines 25–26). Restricting the inner loop to entries of origin time t' ensures that each gate is credited with its own matured return. The retention factor d is applied once per affected memory (Line 30), so residual updates follow the actual adaptive continuation rather than non-overlapping block synchronization. Each memory embedding is computed once upon creation and cached, so an interaction requires only one encoder pass for the state and $\mathcal{O}(|\mathcal{M}_t|)$ scalar operations for gate sampling.

C Theoretical Analysis and Complete Proofs

This section restates and proves the theoretical properties of CMPO. Throughout, expectations conditional on $(s_t, \mathcal{M}_t)$ also condition on the memory-policy parameters available before step t, namely the utilities $\{u_{t,j}\}_{j=1}^{|\mathcal{M}_t|}$ for Proposition 1 and the residuals $\{\omega_{t,j}\}_{j=1}^{|\mathcal{M}_t|}$ together with the baseline parameters $(\mathbf{w}_t, b_t)$ for Proposition 2. Rewards are normalized to $r_\tau \in [0, 1]$ (Section 2), which with $\gamma \in (0, 1]$ implies $G_t^{(H)} \in [0, \sum_{k=0}^{H-1} \gamma^k] \subseteq [0, H]$, so all conditional expectations and baseline predictions remain finite.

C.1 Proof of Proposition 1

We first derive Eq. 4 from Eq. 3. Conditional on $(s_t, \mathcal{M}_t)$ and the current utilities, $u_{t,i}$ is a fixed constant, whereas the selection Y_t and the return $G_t^{(H)}$ remain random. Recall from Section 2.2 that $\bar{p}_{t,i} = \Pr(m_i \in Y_t \mid s_t, \mathcal{M}_t)$ and $\mu_{t,i}^{\mathrm{raw}} = \mathbb{E}[G_t^{(H)} \mid m_i \in Y_t, s_t, \mathcal{M}_t]$. Taking the conditional expectation of Eq. 3 and decomposing over the binary outcomes of the event $\{m_i \in Y_t\}$ yields

$$\begin{aligned}\mathbb{E}[u_{t+1,i} - u_{t,i} \mid s_t, \mathcal{M}_t] &= \eta_u \, \mathbb{E}\Big[\mathbb{I}[m_i \in Y_t] \left(G_t^{(H)} - u_{t,i}\right) \mid s_t, \mathcal{M}_t\Big] \\ &= \eta_u \, \bar{p}_{t,i} \, \mathbb{E}\Big[G_t^{(H)} - u_{t,i} \mid m_i \in Y_t, s_t, \mathcal{M}_t\Big] \\ &= \eta_u \bar{p}_{t,i} \left(\mu_{t,i}^{\mathrm{raw}} - u_{t,i}\right),\end{aligned} \tag{16}$$

where the second line applies the tower property with the indicator vanishing on the complement event $\{m_i \notin Y_t\}$, and the third line follows because $u_{t,i}$ is deterministic given the conditioning set. This establishes Eq. 4. Since $\eta_u > 0$ and $\bar{p}_{t,i} > 0$, the expected update matches the sign of $\mu_{t,i}^{\mathrm{raw}} - u_{t,i}$ and vanishes if and only if $u_{t,i} = \mu_{t,i}^{\mathrm{raw}}$, making $\mu_{t,i}^{\mathrm{raw}}$ the unique stationary point of the step-t update. Adding and subtracting the exclusion-conditioned return $\mu_{t,i}^{\mathrm{excl}} = \mathbb{E}[G_t^{(H)} \mid m_i \notin Y_t, s_t, \mathcal{M}_t]$ yields the decomposition $\mu_{t,i}^{\mathrm{raw}} = \Delta_{t,i} + \mu_{t,i}^{\mathrm{excl}}$, completing the proof. □

C.2 Proof of Proposition 2

Fix an executed gate $i \in \mathcal{V}_t$ and condition on the pre-gate history $\mathcal{H}_{t,i}$ defined in Section 3.3. The history $\mathcal{H}_{t,i}$ fully determines the logit $h_{t,i}$, the propensity $p_{t,i} \in [\frac{\epsilon_0}{|\mathcal{M}_t|}, 1 - \frac{\epsilon_0}{|\mathcal{M}_t|}]$, and the baseline value $b(s_t)$, and the binary outcome satisfies $z_{t,i} \sim \mathrm{Bernoulli}(p_{t,i})$. Clipped propensities bound both inverse weights by $\frac{|\mathcal{M}_t|}{\epsilon_0}$, ensuring that the credit $\widehat{C}_{t,i}$ in Eq. 11 is integrable. Let

$$G_{t,i}^{+} = \mathbb{E}\Big[G_t^{(H)} \mid \mathcal{H}_{t,i}, z_{t,i} = 1\Big], \qquad G_{t,i}^{-} = \mathbb{E}\Big[G_t^{(H)} \mid \mathcal{H}_{t,i}, z_{t,i} = 0\Big] \tag{17}$$

denote the conditional expectations of the return under inclusion and exclusion, so that $\Delta_{t,i}^{\mathrm{gate}} = G_{t,i}^{+} - G_{t,i}^{-}$ by Eq. 13.

Step 1 (unbiasedness and baseline invariance). Expanding the expectation of $\widehat{C}_{t,i}$ from Eq. 11 via the tower property over $z_{t,i} \in \{0, 1\}$:

$$\begin{aligned}\mathbb{E}\Big[\widehat{C}_{t,i} \mid \mathcal{H}_{t,i}\Big] &= \Pr(z_{t,i} = 1 \mid \mathcal{H}_{t,i}) \, \mathbb{E}\left[\frac{G_t^{(H)} - b(s_t)}{p_{t,i}} \,\middle|\, \mathcal{H}_{t,i}, z_{t,i} = 1\right] \\ &\quad + \Pr(z_{t,i} = 0 \mid \mathcal{H}_{t,i}) \, \mathbb{E}\left[\frac{-\left(G_t^{(H)} - b(s_t)\right)}{1 - p_{t,i}} \,\middle|\, \mathcal{H}_{t,i}, z_{t,i} = 0\right] \\ &= p_{t,i} \frac{G_{t,i}^{+} - b(s_t)}{p_{t,i}} - (1 - p_{t,i}) \frac{G_{t,i}^{-} - b(s_t)}{1 - p_{t,i}} \\ &= \big(G_{t,i}^{+} - b(s_t)\big) - \big(G_{t,i}^{-} - b(s_t)\big) \\ &= G_{t,i}^{+} - G_{t,i}^{-} = \Delta_{t,i}^{\mathrm{gate}},\end{aligned} \tag{18}$$

where $p_{t,i}$ and $b(s_t)$ factor out of the conditional expectations because both are measurable with respect to $\mathcal{H}_{t,i}$. Equivalently, this identity follows from the score-function formulation:

$$\begin{aligned}\mathbb{E}\Big[\Big(G_t^{(H)} - b(s_t)\Big) \nabla_{p_{t,i}} \log \mathrm{Bern}(z_{t,i}; p_{t,i}) \mid \mathcal{H}_{t,i}\Big] &= \mathbb{E}\Big[G_t^{(H)} \nabla_{p_{t,i}} \log \mathrm{Bern}(z_{t,i}; p_{t,i}) \mid \mathcal{H}_{t,i}\Big] \\ &= \Delta_{t,i}^{\mathrm{gate}},\end{aligned} \tag{19}$$

since $\mathbb{E}_{z \sim \mathrm{Bernoulli}(p)}[\nabla_p \log \mathrm{Bern}(z; p)] = p\frac{1}{p} + (1 - p)\frac{-1}{1-p} = 0$. This confirms the first equality of Eq. 14.

Step 2 (score gradient alignment). Conditioned on $\mathcal{H}_{t,i}$ and $z_{t,i}$, all subsequent trajectory mechanisms (including subsequent gates, environment dynamics, memory writing, and continuation updates) depend on $z_{t,i}$ rather than $p_{t,i}$. This condition is guaranteed by the delayed update schedule in Algorithm 1: the cached propensity $p_{t,i}$ enters no computation before $G_t^{(H)}$ matures, and within interaction t the budget recursion depends only on $z_{t,i}$. Collecting the remaining stochasticity into a random variable U whose conditional law given $(\mathcal{H}_{t,i}, z_{t,i})$ does not depend on $p_{t,i}$, there exists a measurable function g such that $G_t^{(H)} = g(\mathcal{H}_{t,i}, z_{t,i}, U)$. Defining the gate-local expected return as a function of the inclusion parameter $q \in (0,1)$, with U drawn from its conditional law given $(\mathcal{H}_{t,i}, z)$:

$$J_{t,i}(q) = \mathbb{E}_{z\sim\mathrm{Bernoulli}(q)}[g(\mathcal{H}_{t,i}, z, U) \mid \mathcal{H}_{t,i}] = q\, G_{t,i}^{+} + (1-q)\, G_{t,i}^{-}, \tag{20}$$

where $G_{t,i}^{+} = \mathbb{E}[g(\mathcal{H}_{t,i}, 1, U) \mid \mathcal{H}_{t,i}, z_{t,i}{=}1]$ and $G_{t,i}^{-} = \mathbb{E}[g(\mathcal{H}_{t,i}, 0, U) \mid \mathcal{H}_{t,i}, z_{t,i}{=}0]$ are determined by $\mathcal{H}_{t,i}$ alone and do not depend on q. Because $p_{t,i}$ is itself determined by $\mathcal{H}_{t,i}$, the derivative in Eq. 14 is defined as $J'_{t,i}(q)$ evaluated at $q = p_{t,i}$, which yields

$$\frac{\partial}{\partial p_{t,i}} \mathbb{E}\Big[G_t^{(H)} \mid \mathcal{H}_{t,i}\Big] = J'_{t,i}(q)\big|_{q=p_{t,i}} = G_{t,i}^{+} - G_{t,i}^{-} = \Delta_{t,i}^{\mathrm{gate}}. \tag{21}$$

Combining Eqs. 18 and 21 completes the proof of Eq. 14. □

C.3 Variance Analysis of the Score-Function Credit Estimator

Let $\sigma_{+,t,i}^2 = \mathrm{Var}(G_t^{(H)} \mid \mathcal{H}_{t,i}, z_{t,i} = 1)$ and $\sigma_{-,t,i}^2 = \mathrm{Var}(G_t^{(H)} \mid \mathcal{H}_{t,i}, z_{t,i} = 0)$ denote the conditional return variances within the inclusion and exclusion branches.

Proposition 3 (Variance of Propensity Credit)**.** *For any baseline $b(s_t)$ measurable with respect to $\mathcal{H}_{t,i}$, the conditional variance of $\widehat{C}_{t,i}$ is*

$$\mathrm{Var}\Big(\widehat{C}_{t,i} \mid \mathcal{H}_{t,i}\Big) = \frac{\big(G_{t,i}^{+} - b(s_t)\big)^2 + \sigma_{+,t,i}^2}{p_{t,i}} + \frac{\big(G_{t,i}^{-} - b(s_t)\big)^2 + \sigma_{-,t,i}^2}{1-p_{t,i}} - \big(\Delta_{t,i}^{\mathrm{gate}}\big)^2. \tag{22}$$

Proof. Using $\mathrm{Var}(\widehat{C}_{t,i} \mid \mathcal{H}_{t,i}) = \mathbb{E}[\widehat{C}_{t,i}^2 \mid \mathcal{H}_{t,i}] - (\mathbb{E}[\widehat{C}_{t,i} \mid \mathcal{H}_{t,i}])^2$:

$$\begin{aligned}
\mathbb{E}\Big[\widehat{C}_{t,i}^2 \mid \mathcal{H}_{t,i}\Big] &= p_{t,i}\, \mathbb{E}\left[\left(\frac{G_t^{(H)} - b(s_t)}{p_{t,i}}\right)^2 \,\middle|\, \mathcal{H}_{t,i}, z_{t,i} = 1\right] \\
&\quad + (1-p_{t,i})\, \mathbb{E}\left[\left(\frac{G_t^{(H)} - b(s_t)}{1-p_{t,i}}\right)^2 \,\middle|\, \mathcal{H}_{t,i}, z_{t,i} = 0\right] \\
&= \frac{\big(G_{t,i}^{+} - b(s_t)\big)^2 + \sigma_{+,t,i}^2}{p_{t,i}} + \frac{\big(G_{t,i}^{-} - b(s_t)\big)^2 + \sigma_{-,t,i}^2}{1-p_{t,i}}.
\end{aligned} \tag{23}$$

Subtracting $(\mathbb{E}[\widehat{C}_{t,i} \mid \mathcal{H}_{t,i}])^2 = (\Delta_{t,i}^{\mathrm{gate}})^2$ yields Eq. 22. □

Remark (Variance Reduction via Baseline). Differentiating Eq. 22 with respect to the scalar baseline $b(s_t)$ shows that the variance-minimizing baseline is the propensity-weighted state return

$$b^*(s_t) = (1-p_{t,i})G_{t,i}^{+} + p_{t,i}G_{t,i}^{-}, \tag{24}$$

which is infeasible because the branch means $G_{t,i}^{+}$ and $G_{t,i}^{-}$ of each gate are unknown. Substituting the gate-local mixture $\mathbb{E}[G_t^{(H)} \mid \mathcal{H}_{t,i}] = p_{t,i}G_{t,i}^{+} + (1-p_{t,i})G_{t,i}^{-}$ instead replaces the squared terms $(G_{t,i}^{+}-b)^2$ and $(G_{t,i}^{-}-b)^2$ with $(1-p_{t,i})^2(\Delta_{t,i}^{\mathrm{gate}})^2$ and $p_{t,i}^2(\Delta_{t,i}^{\mathrm{gate}})^2$, shrinking the residual scale from the return magnitude to the contrast magnitude $|\Delta_{t,i}^{\mathrm{gate}}|$, which is typically far smaller. The feasible surrogate $b(s_t) = \mathbf{w}_t^{\top}\,\mathrm{emb}(s_t) + b_t$ tracks the state-level average $\mathbb{E}[G_t^{(H)} \mid s_t] = \mathbb{E}_{\mathcal{H}_{t,i}|s_t}[\mathbb{E}[G_t^{(H)} \mid \mathcal{H}_{t,i}]]$ and therefore additionally leaves the dispersion of the gate-local mixture around the state mean. It removes the common return level of the state, and the remaining terms of order $(\Delta_{t,i}^{\mathrm{gate}})^2/p_{t,i}$ are controlled by the propensity clipping and the update normalization, as quantified next.

Remark (Bounded Update Variance). Because $r_\tau \in [0,1]$ we have $G_t^{(H)} \in [0,H]$, and the baseline prediction is clipped to $b(s_t) \in [0,H]$, so $|G_t^{(H)} - b(s_t)| \le H$. Combining this with propensity clipping $p_{t,i}, 1-p_{t,i} \ge \frac{\epsilon_0}{|\mathcal{M}_t|}$ yields $|\widehat{C}_{t,i}| \le \frac{|\mathcal{M}_t|H}{\epsilon_0}$. With the update normalization in Eq. 12, the effective parameter increment satisfies the deterministic uniform bound

$$\left|\frac{\alpha_0 \widehat{C}_{t,i}}{|\mathcal{M}_t|}\right| \le \frac{\alpha_0 H}{\epsilon_0}, \qquad \mathrm{Var}\left(\frac{\alpha_0 \widehat{C}_{t,i}}{|\mathcal{M}_t|}\,\middle|\,\mathcal{H}_{t,i}\right) \le \frac{\alpha_0^2 H^2}{\epsilon_0^2}, \tag{25}$$

confirming that the variance of individual residual updates remains strictly bounded and independent of the memory bank size $|\mathcal{M}_t|$.

Remark (Local Sign Alignment vs. Global Optimization). The sign alignment in Proposition 2 holds conditional on the pre-gate history $\mathcal{H}_{t,i}$, so each scalar update moves $\omega_{t,i}$ along a sign-consistent ascent direction of the gate-local objective $J_{t,i}(q)$ whenever $p_{t,i}$ lies in the interior of the clipped interval. At the clipping boundaries, the update accumulates evidence in $\omega_{t,i}$ without changing $p_{t,i}$ until the logit re-enters the interior. Because sequential token budgets couple gate outcomes and residuals are reused across future states, this coordinate-wise property does not imply unconstrained monotonic ascent of the global trajectory return $J_{\mathrm{Mem},t}$. Instead, the randomized permutation $\varpi_t = \mathrm{Shuffle}(1,\dots,|\mathcal{M}_t|)$ marginalizes gate executions over varied budget contexts, preventing systematic positional bias across the memory bank.

Remark (Relation to Marginal Contrast). Proposition 2 targets $\Delta_{t,i}^{\mathrm{gate}}$ rather than the marginal contrast $\Delta_{t,i}$ in Eq. 5, which averages over the full subset-selection distribution. Conditional on the gate for m_i being executed, the law of total expectation gives

$$\mathbb{E}\Big[G_t^{(H)} \mid z_{t,i}=1\Big] = \frac{\mathbb{E}_{\mathcal{H}_{t,i}}\big[p_{t,i}\, G_{t,i}^{+}\big]}{\mathbb{E}_{\mathcal{H}_{t,i}}[p_{t,i}]}, \qquad \mathbb{E}\Big[G_t^{(H)} \mid z_{t,i}=0\Big] = \frac{\mathbb{E}_{\mathcal{H}_{t,i}}\big[(1-p_{t,i})\, G_{t,i}^{-}\big]}{\mathbb{E}_{\mathcal{H}_{t,i}}[1-p_{t,i}]}, \tag{26}$$

where all expectations condition on $(s_t, \mathcal{M}_t)$, the current policy parameters, and $i \in \mathcal{V}_t$, and the outer expectations average over $\mathcal{H}_{t,i}$. Under CMPO, $p_{t,i}$ is constant across these histories, so the two ratios reduce to $\mathbb{E}_{\mathcal{H}_{t,i}}[G_{t,i}^{+}]$ and $\mathbb{E}_{\mathcal{H}_{t,i}}[G_{t,i}^{-}]$. Thus, $\mathbb{E}_{\mathcal{H}_{t,i}}[\Delta_{t,i}^{\mathrm{gate}}]$ equals the inclusion–exclusion contrast conditional on gate execution. This contrast can differ from $\Delta_{t,i}$ in Eq. 5 because the exclusion event $m_i \notin Y_t$ also includes cases where $i \notin \mathcal{V}_t$. When the gate is always executed, as in the controlled setting of Section 4.3, the two contrasts coincide.

D Benchmark Details

This appendix details the three benchmarks used in our experiments.

D.1 LifelongAgentBench

LifelongAgentBench (Zheng et al., 2025) evaluates the lifelong learning capabilities of LLM-based agents through sequential task completion in interactive environments. We use its two terminal-interaction domains.

Operating System (OS). The agent interacts with a Linux terminal to complete file management, system configuration, and process manipulation tasks. Each task provides a natural-language instruction and an initial system state, and the agent issues shell commands and receives terminal output until it solves the task or exhausts the maximum of 10 turns. Success is verified by an automated checker against predefined final-state conditions.

Database (DB). The agent operates on relational databases through an interactive SQL client, with tasks covering schema exploration, data retrieval, table manipulation, and multi-table joins of varying complexity. The maximum number of turns is 10, and success is determined by comparing the query result or database state against a reference answer.

Both environments organize tasks into epochs: within each epoch, the agent encounters a sequence of tasks drawn from a fixed pool, and the memory state persists within and across epochs. We report the last-epoch success rate (Last), measuring performance after learning from prior epochs, and the cumulative success rate (CSR), measuring the proportion of distinct tasks solved at least once over the ten epochs. The benchmark contains 500 OS tasks and 500 DB tasks.

D.2 ALFWORLD

ALFWorld (Shridhar et al., 2020b) is a text-based embodied environment that aligns TextWorld (Côté et al., 2018) interactive fiction games with ALFRED (Shridhar et al., 2020a) household tasks. The agent navigates rooms, interacts with objects, and completes household goals through text commands. ALFWorld provides six task types of increasing complexity:

- **Pick & Place (P&P):** pick up an object and place it at a target location.
- **Examine:** find an object and examine it under a light source.
- **Clean:** clean an object in a sink and place it at a target location.
- **Heat:** heat an object using a microwave and place it at a target location.
- **Cool:** cool an object in a fridge and place it at a target location.
- **Pick Two & Place (Pick-2):** pick up two instances of an object type and place them at a target location.

Each task type requires a different combination of navigation, object interaction, and appliance use. The environment returns text observations describing the current room, visible objects, and the result of each action, and success is binary once all goal conditions are satisfied. We use the 134 validation-unseen tasks distributed across the six types, report the per-type success rate and the macro-average, and cap each task at 30 agent action steps.

D.3 APPWORLD

AppWorld (Trivedi et al., 2024) is a compositional tool-use benchmark in which agents complete realistic multi-step workflows by invoking APIs of interconnected applications (calendar, email, file storage, messaging, and others). Each task specifies a user goal in natural language and provides documented API endpoints, and the agent must plan a sequence of API calls, handle cross-application dependencies, and verify that all goal conditions are satisfied. Tasks are organized into scenarios comprising three task variants each. We report Task Goal Completion (TGC), the fraction of tasks that pass all evaluation tests, and Scenario Goal Completion (SGC), the fraction of scenarios for which all task variants pass all evaluation tests. The benchmark contains 30 scenarios with 90 task goals, and each task allows at most 50 API calls.

E BASELINE DETAILS

This appendix describes the baseline implementations. All memory-centric baselines share the same memory writer, task order, black-box backbone, and token budget B_{mem} as CMPO, differing only in how memories are selected and how selection scores are updated. For Mem0 and MemP, we replace their native memory-construction modules with the shared writer and retain only their retrieval mechanisms, so these implementations are controlled shared-writer variants rather than full-system reproductions. Model-centric baselines share the white-box backbone and are tuned under matched compute budgets.

E.1 MEMORY-CENTRIC BASELINES

No Memory. The frozen agent completes each task using only the task instruction and interaction history, without external experience. This baseline isolates the intrinsic capability of the backbone and lower-bounds methods that use accumulated experience.

RAG. A standard retrieval-augmented baseline that ranks all memories by cosine similarity between the state embedding $\text{emb}(s_t)$ and each memory embedding $\text{emb}(m_i)$, and greedily includes memories in descending similarity order until the token budget B_{mem} is exhausted. Selection is deterministic given the state, and no feedback-based update is applied.

Mem0 (Chhikara et al., 2025) maintains a production-scale long-term memory through a lifecycle of extraction, consolidation, and update. New interactions are parsed into candidate facts, merged with semantically related entries, and stored in a queryable index, with retrieval combining semantic similarity and recency.

MemP (Fang et al., 2026) distills each completed trajectory into a step-by-step procedure that can be replayed for similar tasks, and retrieves the top-k procedures by task-description similarity.

MemRL (Zhang et al., 2026b) augments semantic retrieval with a per-memory utility u_i updated toward the observed trajectory return whenever the memory is retrieved, combining cosine similarity with the utility as the selection score.

MemQ (Liao et al., 2026) propagates return-based credit through a provenance directed acyclic graph (DAG) that records which memories contributed to each successful trajectory. Each node maintains a Q-value updated via backpropagation along the DAG, and the retrieval score combines semantic similarity with the current Q-value.

RoMeRL (Yang et al., 2026) addresses the memory-reward trap by compressing per-memory utilities into a fixed-dimensional reduced-order state updated from task returns, combining semantic similarity with the projection of each memory onto this state.

E.2 Model-Centric Baselines

Model-centric baselines require white-box access to the agent (parameters, gradients, or output logits) and are therefore run on white-box backbones in the corresponding block of Table 1.

ETO (Song et al., 2024) performs exploration-based trajectory optimization: successful and failed trajectories from iterative exploration are paired into preference tuples, and the policy is optimized through Direct Preference Optimization (Rafailov et al., 2023).

MEM1 (Zhou et al., 2026) trains the agent end-to-end with reinforcement learning to jointly reason and consolidate the interaction history into a compact, evolving internal memory state. We adopt the PPO configuration (Schulman et al., 2017) and reward shaping of the original paper.

JitRL (Li et al., 2026b) retrieves state-action-return experiences from a replay buffer to estimate action advantages and directly modulates the frozen agent output logits at inference time. Retrieval uses cosine similarity over state embeddings, and the modulation strength follows the annealing schedule of the original paper.

F Implementation Protocol

This appendix details the implementation of CMPO, while baseline-specific configurations follow the recommended settings of their respective papers. Table 4 lists the CMPO hyperparameters used in all experiments, and unless otherwise specified, the same values apply across benchmarks and backbones. The memory bank $\mathcal{M}_1$ and the pending-credit cache $\mathcal{Q}$ are empty at initialization, and each new memory is inserted with residual $\omega_i = 0$. The state embedding is computed once per task-level interaction, whereas each memory embedding is computed once upon insertion.

Table 4: Default hyperparameter values for CMPO.

Symbol	Meaning	Value	Notes
ϵ_0	Exploration rate for gate clipping	0.1	Section 3.1
α_0	Residual step size	0.5	Section 3.2
d	Retention factor per residual update	1.0	No decay by default
γ	Return discount factor	1.0	Shared across benchmarks
η_b	Baseline learning rate	0.05	Online linear value baseline tracking
H	Credit horizon (interactions)	3	Truncated at session end
B_{mem}	Memory-token budget in prompt	1024	Shared across all methods
$\text{sim}(\cdot,\cdot)$	Semantic similarity function	Cosine	Over embedded state and memories
Embedding model	Semantic encoder	text-embedding-3-large[1]	OpenAI text embedding

All memory-centric methods (including CMPO) share the same memory writer $\mathcal{W}$, which summarizes each completed interaction into a memory of at most 200 tokens using the prompt in Appendix L. Sharing this writer ensures that performance differences among memory-centric methods stem from selection and credit assignment rather than memory content.

[1] https://developers.openai.com/api/docs/models/text-embedding-3-large

Table 5: **Paired uncertainty of CMPO improvements (%).** Each entry reports the mean of three seed-wise paired improvements, $\Delta = \mathrm{CMPO} - \mathrm{BestBaseline}$, followed by its 95% paired t-confidence interval in brackets. For each backbone and metric, the strongest non-CMPO baseline is selected by its across-seed mean and then held fixed when forming the three paired differences. ALF. Macro denotes the macro-average over the six ALFWorld task types. All intervals use two degrees of freedom, and positive values favor CMPO.

Backbone	OS Last	OS CSR	DB Last	DB CSR	ALF. Macro	AppWorld TGC	AppWorld SGC
White-box model-centric comparison							
Llama-3.1-8B	$+1.5$ $[+0.0, +2.9]$	$+1.4$ $[+0.4, +2.4]$	$+1.7$ $[-0.1, +3.6]$	$+1.2$ $[+0.2, +2.2]$	$+0.3$ $[-0.7, +1.3]$	-0.4 $[-3.6, +2.8]$	$+1.1$ $[-8.5, +10.7]$
Qwen3-8B	$+3.1$ $[+0.8, +5.3]$	$+3.5$ $[+1.8, +5.3]$	$+0.7$ $[-1.1, +2.4]$	$+2.6$ $[+0.8, +4.4]$	$+1.0$ $[+0.1, +1.9]$	$+1.9$ $[-1.3, +5.0]$	$+2.2$ $[-10.4, +14.9]$
Black-box memory-centric comparison							
GPT-5.6-Luna	$+5.5$ $[+0.9, +10.1]$	$+6.0$ $[+4.0, +8.0]$	$+6.2$ $[+3.6, +8.8]$	$+3.8$ $[+1.5, +6.1]$	$+2.7$ $[-9.9, +15.3]$	$+3.3$ $[-8.7, +15.4]$	$+7.8$ $[+3.0, +12.6]$
Claude-Sonnet-5	$+4.7$ $[+1.2, +8.2]$	$+5.2$ $[+3.2, +7.2]$	$+4.7$ $[-2.1, +11.6]$	$+3.2$ $[+0.9, +5.5]$	$+1.1$ $[-7.7, +9.8]$	$+2.6$ $[-9.9, +15.0]$	$+5.6$ $[+0.8, +10.3]$

Backbones and inference. Black-box experiments use GPT-5.6-Luna and Claude-Sonnet-5 with decoding temperature 0.7 and top-p 0.95. Each interaction consists of one complete task-level rollout, with the retrieved memory set Y_t inserted as described in Section 3.1, and API failures are retried up to three times before the interaction is marked as failed. White-box experiments use Llama-3.1-8B and Qwen3-8B, on which CMPO uses only forward passes without accessing gradients or logits, whereas model-centric baselines follow their own training procedures.

Training and evaluation. We generate a fixed task order per random seed and apply it to all methods to avoid confounding by task ordering. All memory-centric methods run for 10 epochs with the memory bank persisting across epochs, each epoch attempts every task instance exactly once (one rollout per task), and all methods use one rollout per interaction under the black-box protocol, ensuring matched interaction budgets. We report the mean over three independent seeds ($\{0, 1, 2\}$), which control task ordering, gate sampling in CMPO, and any stochastic components of the baselines. Shaded regions in learning curves denote one standard deviation.

Compute and reproducibility. Black-box experiments run on a single server with an Intel® Xeon® Gold 6430 processor (60 MB cache, 2.10 GHz) and 768 GB of DDR5 memory, where all agent inference is served through API calls and requires no local accelerator. White-box experiments run on the same server, additionally using a single node with 8 NVIDIA A100 GPUs (80 GB, SXM4) connected via NVLink, on which inference for the frozen backbones and training for the model-centric baselines are both performed. Every reported number is averaged over the three seeds $\{0, 1, 2\}$ described above, and within each seed all compared methods reuse the same task order, prompt templates, memory writer, and decoding settings.

G Paired Uncertainty Analysis

To quantify the uncertainty of the improvements in Table 1, we conduct a paired analysis over three random seeds. Within each seed, CMPO and all baselines share the same task order and experimental configuration, so seed-level differences compare methods under matched conditions. For each backbone and metric, we select the strongest non-CMPO baseline by its across-seed mean, hold it fixed across the three paired comparisons, and compute the seed-level paired improvement $\Delta_s = P_s^{\mathrm{CMPO}} - P_s^{\mathrm{baseline}}$, where P_s^{CMPO} and P_s^{baseline} denote the metric values of CMPO and the selected baseline under seed s. We report the mean improvement with the 95% paired t-confidence interval $\bar{\Delta} \pm t_{0.975,2} s_\Delta / \sqrt{3}$, where $t_{0.975,2} = 4.303$ and s_Δ is the sample standard deviation of Δ_s. For ALFWorld, the six task-type scores are macro-averaged within each seed before computing Δ_s. As shown in Table 5, CMPO obtains a positive mean improvement in 27 of the 28 comparisons, with 15 confidence intervals lying strictly above zero and one additional interval touching zero at its lower bound. The improvements are most consistent in the black-box setting, where 9 of 14 intervals exclude zero and the remaining five retain positive means. In the white-box setting, the effect sizes are smaller relative to the seed-level spread, so fewer intervals exclude zero, yet the improvement remains positive in 13 of 14 cases. The residual uncertainty is most pronounced on AppWorld, whose

Table 6: **Component-wise ablation of CMPO (%).** Each variant removes or replaces exactly one component while retaining the memory writer, token budget, stochastic-gate protocol, prompts, and decoding settings. All variants are evaluated for ten online epochs with one rollout per task and averaged over three paired random seeds. LifelongAgentBench reports last-epoch success rate and cumulative success rate (Last/CSR), ALFWorld reports per-type success rate, and AppWorld reports TGC and SGC. Comparisons should be made only within the same backbone.

Variant	LifelongAgentBench		ALFWorld						AppWorld		LifelongAgentBench		ALFWorld						AppWorld	
	OS Last/CSR	DB Last/CSR	P&P	Examine	Clean	Heat	Cool	Pick-2	TGC	SGC	OS Last/CSR	DB Last/CSR	P&P	Examine	Clean	Heat	Cool	Pick-2	TGC	SGC
White-box	Llama-3.1-8B										Qwen3-8B									
Credit Construction																				
w/o Value Baseline	64.4/75.8	54.7/84.2	79.2	74.1	77.4	76.8	74.6	66.7	13.0	7.8	68.3/80.2	59.2/88.6	80.6	75.9	79.6	78.3	76.2	68.6	15.6	10.0
Naive Running Difference	65.5/76.7	55.0/84.5	80.6	72.2	76.3	75.4	73.0	64.7	13.3	7.8	67.8/79.8	58.5/87.9	77.8	74.1	77.4	76.8	74.6	66.7	14.8	7.8
w/o Propensity Correction	66.8/77.8	56.5/85.6	81.9	75.9	77.4	78.3	76.2	68.6	14.4	8.9	69.9/81.6	60.9/90.1	81.9	77.8	80.6	79.7	77.8	72.5	17.8	12.2
w/o Exclusion Credit	65.5/76.7	54.7/84.0	79.2	70.4	75.3	75.4	73.0	64.7	12.6	7.8	68.3/80.5	59.2/88.6	80.6	74.1	78.5	76.8	74.6	66.7	15.9	10.0
Policy Parameterization																				
w/o Semantic Prior	66.0/77.5	56.4/85.5	83.3	74.1	76.3	76.8	74.6	66.7	13.0	6.7	69.8/81.5	61.7/90.4	83.3	77.8	80.6	78.3	79.4	70.6	15.6	11.1
w/o Learned Residual	64.0/74.7	52.7/82.4	77.8	68.5	74.2	73.9	71.4	60.8	10.7	5.6	67.1/78.6	57.3/86.3	80.6	72.2	77.4	75.4	76.2	64.7	15.2	8.9
Memory-Bank Scaling																				
w/o Log-Bank Correction	63.7/74.9	52.8/83.1	80.6	72.2	77.4	75.4	73.0	62.7	11.9	4.4	66.9/79.1	57.1/87.2	81.9	75.9	79.6	76.8	76.2	68.6	14.1	6.7
w/o Update Normalization	62.3/74.3	55.0/84.3	77.8	74.1	78.5	76.8	77.8	66.7	10.4	3.3	68.0/80.2	56.6/86.7	80.6	77.8	80.6	78.3	77.8	70.6	14.4	5.6
CMPO (Full)	67.0/78.2	57.0/86.0	83.3	75.9	78.5	78.3	77.8	68.6	14.8	10.0	70.5/82.3	62.1/90.8	83.3	79.6	81.7	79.7	79.4	72.5	18.5	12.2
Black-box	GPT-5.6-Luna										Claude-Sonnet-5									
Credit Construction																				
w/o Value Baseline	82.2/86.3	68.9/95.6	93.1	90.7	91.4	85.5	90.5	84.3	32.2	27.8	83.3/86.7	67.1/94.5	93.1	90.7	88.2	85.5	88.9	82.4	32.6	27.8
Naive Running Difference	83.3/87.0	68.1/95.2	91.7	88.9	90.3	85.5	88.9	82.4	31.9	25.6	83.3/87.0	67.7/94.9	93.1	88.9	87.1	84.1	87.3	80.4	32.6	26.7
w/o Propensity Correction	85.0/88.1	70.5/97.0	95.8	94.4	92.5	87.0	92.1	86.3	35.2	32.2	85.6/88.2	69.4/96.2	88.9	92.6	90.3	87.0	88.9	84.3	36.3	32.2
w/o Exclusion Credit	81.9/85.6	67.7/94.9	88.9	90.7	90.3	84.1	88.9	80.4	30.7	26.7	82.6/85.6	66.4/94.5	91.7	88.9	87.1	85.5	87.3	78.4	31.9	25.6
Policy Parameterization																				
w/o Semantic Prior	85.3/88.3	71.8/97.3	95.8	92.6	91.4	85.5	90.5	84.3	32.6	30.0	85.0/88.6	69.8/96.1	95.8	90.7	89.2	85.5	90.5	82.4	33.3	28.9
w/o Learned Residual	68.1/73.2	58.5/83.2	94.4	83.3	88.2	82.6	85.7	78.4	20.4	13.3	70.0/75.4	57.6/82.8	90.3	83.3	86.0	84.1	85.7	76.5	23.0	15.6
Memory-Bank Scaling																				
w/o Log-Bank Correction	79.9/84.0	65.3/93.5	95.8	90.7	90.3	87.0	88.9	82.4	29.6	25.6	80.9/85.1	64.6/93.1	93.1	88.9	88.2	88.4	88.9	80.4	30.4	24.4
w/o Update Normalization	78.6/83.3	69.1/95.9	94.4	94.4	93.5	87.0	93.7	84.3	27.8	23.3	82.7/86.7	63.4/92.0	94.4	94.4	89.2	87.0	92.1	84.3	31.5	23.3
CMPO (Full)	86.3/89.1	72.4/97.8	95.8	96.3	93.5	88.4	93.7	88.2	36.3	34.4	86.1/89.0	70.5/96.7	95.8	94.4	90.3	88.4	92.1	86.3	36.7	33.3

compositional multi-step structure induces greater seed-level variance. These results indicate that most advantages of CMPO in Table 1 persist across paired random seeds rather than being driven by a single favorable run.

H Ablation Analysis (*RQ4*)

To examine the role of each CMPO component, we conduct one-at-a-time ablations across two white-box and two black-box backbones under the protocol stated in the caption of Table 6: each variant removes or replaces exactly one component and runs online for ten epochs with one rollout per task, and Table 6 reports epoch-10 performance and, for LAB, cumulative success over the ten epochs, averaged over three paired random seeds. Because the components interact within the unified memory policy, the reported differences measure the structural importance of each component rather than additive standalone contributions.

w/o Value Baseline. This variant sets $b(s_t) = 0$ in Eq. 11, weighting raw returns directly by inverse propensities. Although asymptotically unbiased, omitting baseline subtraction amplifies single-rollout gradient variance on complex tasks: SGC decreases by 6.7 points on GPT-5.6-Luna and 5.6 points on Claude-Sonnet-5, and OS Last drops by 4.1 and 2.8 points, respectively, confirming that state baseline subtraction is necessary for variance reduction in single-rollout credit estimation.

Naive Running Difference. This variant replaces propensity-weighted credit with the running average difference $\bar{G}^{+}_{t,i} - \bar{G}^{-}_{t,i}$ between inclusion and exclusion returns for each memory, omitting inverse-propensity weighting. Naive contrasts mitigate leakage under initial uniform exploration, but their accuracy degrades as the policy evolves: on AppWorld, SGC drops by 8.9 points on GPT-5.6-Luna and 6.7 points on Claude-Sonnet-5, and Pick-2 decreases by 5.9 points across black-box models. Because selective exposure renders memory sampling non-uniform, unweighted sample averages accumulate the selection bias of all past epochs, whereas propensity correction removes it within each single rollout.

w/o Propensity Correction. This variant replaces the credit in Eq. 11 with the Bernoulli logit-gradient estimator $(G^{(H)}_t - b(s_t))(z_{t,i} - p_{t,i})$ for gates in the unclipped region, and assigns zero gradient where

probability clipping is active. Here $G_t^{(H)}$ denotes the H-interaction return, $b(s_t)$ the value baseline for state s_t, and $z_{t,i}$ and $p_{t,i}$ the sampled outcome and the inclusion probability of memory gate i at interaction t. Relative to CMPO, This variant lowers performance by up to 1.9 percentage points in the white-box setting, with larger drops of 2.2 points in SGC on GPT-5.6-Luna and 3.2 points in Cool success rate on Claude-Sonnet-5. These results favor updating memory residuals directly from the inclusion–exclusion contrast. For an unclipped gate, the expected signal is $p_{t,i}(1-p_{t,i})\Delta_{t,i}^{\mathrm{gate}}$, where $\Delta_{t,i}^{\mathrm{gate}}$ is the gate-local return contrast, so the learning signal vanishes as $p_{t,i}$ approaches zero or one. CMPO removes this attenuation through inverse-propensity weighting, which preserves the expected contrast for included memories as well as for retrieved, rarely excluded ones. The observed gains are consistent with this mechanism, which helps CMPO promote overlooked useful memories and correct overly favored harmful ones.

w/o Exclusion Credit. This variant retains the propensity-weighted inclusion term $(G_t^{(H)} - b(s_t))z_{t,i}/p_{t,i}$ but assigns zero credit to excluded gates ($z_{t,i}=0$). The degradation exceeds that of removing propensity correction alone: averaged over the four backbones, Examine and Pick-2 drop by 5.6 and 6.4 points, and SGC by 5.0 points. The pronounced losses on multi-memory retrieval tasks show that observing only inclusion returns cannot disentangle helpful memories from co-retrieved irrelevant or harmful entries.

w/o Semantic Prior. This variant sets $e_{t,i}=0$ while preserving learned residuals and log-bank corrections. Simple navigation tasks such as P&P remain unaffected, but tasks requiring fine-grained state–memory alignment decline moderately, averaging 4.0 points across Examine, Pick-2, and SGC under black-box LLMs. Although $e_{t,i}\in[-1,1]$ has a narrower range than $-\log|\mathcal{M}_t|$, it provides critical state-conditioned initialization in early epochs, before the residuals $\omega_{t,i}$ accumulate enough feedback to dominate the selection margin.

w/o Learned Residual. This variant fixes $\omega_{t,i}=0$, preventing environmental feedback from refining future memory exposure, and causes the most severe degradation of all ablations: average reductions of 17.1 points across OS Last, TGC, and SGC under black-box LLMs, and of 7.4 and 7.8 points on Examine and Pick-2 in the white-box setting. Semantic retrieval alone therefore cannot support effective self-evolution, and persistent residuals are indispensable for converting credit feedback into improved selection policies.

w/o Log-Bank Correction. This variant removes $-\log|\mathcal{M}_t|$ from the selection logit in Eq. 7, allowing the prior exposure probability of neutral memories to grow with the bank size. In the white-box setting, SGC decreases by 5.6 points and OS Last by 3.3–3.6 points, and the drop is larger under black-box LLMs, where SGC declines by 8.9 points and DB Last by 6.5 points on average. Regulating aggregate exposure is therefore critical for preventing prompt saturation as memories accumulate over extended interactions.

w/o Update Normalization. This variant omits the normalization factor $1/|\mathcal{M}_t|$ in Eq. 12, causing single-step updates to scale with the memory bank size. The degradation concentrates on long-horizon and compositional metrics, with SGC dropping by 6.7 points on average in the white-box setting and by 10.6 points under black-box LLMs. Together with the log-bank ablation, this confirms that both exposure-level and update-level scaling are essential for stable online learning as the memory bank expands.

I Hyperparameter Sensitivity Analysis (*RQ5*)

Sensitivity to α_0 and ϵ_0. To examine whether CMPO depends on careful tuning, we jointly vary the residual step size $\alpha_0\in\{0.1,0.3,0.5,0.7,1.0\}$ and exploration rate $\epsilon_0\in\{0.01,0.05,0.1,0.2,0.3\}$ on GPT-5.6-Luna and Claude-Sonnet-5, keeping all other configurations fixed, and evaluate each configuration for ten online epochs with one rollout per task over three random seeds. As shown in Fig. 4, the default configuration $(\alpha_0,\epsilon_0)=(0.5,0.1)$ lies within 0.4 percentage points of the best result in all eight settings, and the region $\alpha_0\in[0.5,0.7]$, $\epsilon_0\in[0.1,0.2]$ forms a stable near-optimal plateau in which performance varies by at most 1.5 point. Performance degrades toward the extremes: at $\epsilon_0=0.1$, moving α_0 from 0.5 to 0.1 or 1.0 causes average losses of 3.6 and 2.9 points across the eight settings, indicating that both insufficient adaptation and overly aggressive residual updates are detrimental, and at $\alpha_0=0.5$, extreme exploration rates ($\epsilon_0\in\{0.01,0.3\}$) yield average losses of 2.3 points, reflecting the trade-off between inadequate exploration and excessive random exposure. Sensitivity also varies with the benchmark: the full-grid variation is only 3.4–4.4 points on ALFWorld

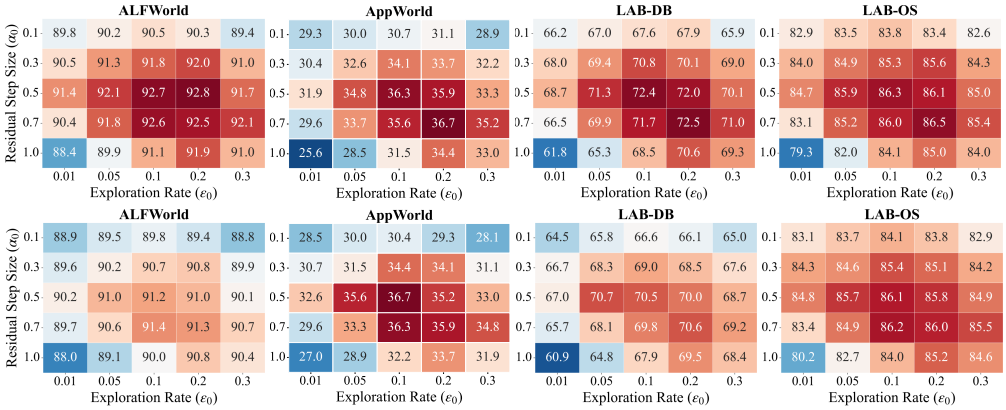


Figure 4: **Sensitivity to the residual step size α_0 and exploration rate ϵ_0.** Top and bottom rows correspond to GPT-5.6-Luna and Claude-Sonnet-5, respectively, and Columns report ALFWorld macro SR, AppWorld TGC, LAB-DB last-epoch SR, and LAB-OS last-epoch SR. Each cell reports the mean epoch-10 performance (%) over three seeds.

but reaches 9.7–11.1 points on LAB-DB and AppWorld, whose sparse feedback and compositional execution make erroneous memory updates more costly. Overall, CMPO remains stable over a broad operating region while avoiding extreme step-size and exploration combinations.

Sensitivity to B_{mem}, d, H, and γ. We further vary one hyperparameter at a time while fixing the others to their main-text values, evaluating $B_{\text{mem}} \in \{256, 512, 1024, 2048, 4096\}$, $d \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, $H \in \{1, 2, 3, 4, 5\}$, and $\gamma \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ on both black-box backbones and all four benchmarks over ten epochs and three seeds. As shown in Fig. 5, the default $H = 3$ lies within 0.6 points of the best result in every setting, while $H = 1$ and $H = 5$ underperform with average losses of 2.8 and 1.2 points. The inferiority of $H = 1$ highlights that task-level rollouts are causally coupled across interactions via the memory writer $\mathcal{W}$: a successful selection at interaction t yields a completed trajectory from which $\mathcal{W}$ distills high-leverage procedural strategies into $\mathcal{M}_{t+1}$, directly assisting the subsequent $1 \sim 2$ interactions. A myopic single-step horizon discards this delayed positive externality, whereas overly extended horizons ($H \geq 5$) dilute localized credit with downstream task stochasticity. The retention factor d has a stronger asymmetric effect: reducing d to 0.2 causes an average loss of 7.1 points, whereas $d = 0.8$ stays within 0.4 points of $d = 1.0$. Persistent credit is thus essential in our ten-epoch setting where reusable skills recur across tasks, although a smaller d would be preferable in non-stationary environments where old memories rapidly become obsolete, so the observed optimum near $d = 1.0$ reflects the relatively stable task distribution rather than a universal recommendation. The discount factor γ exhibits a similar but milder trend: $\gamma \in [0.8, 1.0]$ stays within 0.5 points of the default, whereas $\gamma = 0.2$ lowers performance by up to 3.3 points on LAB-DB and AppWorld. Because aggressive discounting suppresses the delayed rewards that propagate through $\mathcal{W}$ within the short horizon $H = 3$, a near-undiscounted return preserves the writer-mediated coupling that motivates $H > 1$. The token budget B_{mem} shows diminishing returns: reducing it to 256 lowers performance by 5.1 points on average (particularly on LAB-DB and AppWorld), whereas increasing it from 1024 to 2048 changes performance by only -0.2 to $+0.9$ points, and 4096 yields no consistent gain. These effects are smallest on ALFWorld and largest on the more compositional LAB-DB and AppWorld tasks. Overall, the selected configuration is consistently near-optimal across both backbones and avoids the additional context cost of a larger memory budget.

J Generalization on Held-Out Tasks (RQ6)

To evaluate whether CMPO acquires transferable procedural strategies rather than acting as a task-specific answer cache, we measure zero-shot adaptation on disjoint held-out task splits. All benchmarks are partitioned into non-overlapping training and test sets: LifelongAgentBench allocates 350 training and 150 test tasks for OS and DB each, introducing novel shell utilities, system environments, and database schemas; ALFWorld provides 66 training tasks and 68 out-of-distribution (OOD) test tasks with unseen room layouts; AppWorld reserves 10 unseen scenarios (30 sub-goals) out of 30 for testing. Methods undergo online self-evolution on the training set for ten epochs under the single-rollout protocol with token budget $B_{\text{mem}} = 1024$, after which the evolved memory bank $\mathcal{M}$

Table 7: **Generalization on held-out tasks and unseen environments (%).** Agents undergo online self-evolution on the disjoint training tasks for ten epochs. The resulting memory bank $\mathcal{M}$ and learned residuals $\{\omega_i\}$ are then frozen and evaluated on held-out tasks with one rollout per task, averaged over three random seeds. LAB evaluates 150 unseen OS and 150 unseen DB tasks; ALFWorld evaluates 68 out-of-distribution (OOD) tasks with novel room layouts; AppWorld evaluates 10 unseen multi-application scenarios (30 sub-goals).

Method	LAB (Held-out)		ALFWorld (OOD)						AppWorld (Held-out)		LAB (Held-out)		ALFWorld (OOD)						AppWorld (Held-out)	
	OS Test	DB Test	P&P	Examine	Clean	Heat	Cool	Pick-2	TGC	SGC	OS Test	DB Test	P&P	Examine	Clean	Heat	Cool	Pick-2	TGC	SGC
White-box	Llama-3.1-8B										Qwen3-8B									
ETO	50.4	44.7	83.3	70.4	75.0	**80.6**	69.7	62.5	10.0	6.7	57.6	52.0	**86.1**	66.7	75.0	**80.6**	72.7	62.5	12.2	6.7
MEM1	57.1	46.4	77.8	74.1	77.1	72.2	72.7	**70.8**	**13.3**	6.7	60.0	51.3	77.8	74.1	77.1	75.0	75.8	66.7	14.4	6.7
JitRL	55.6	46.9	**86.1**	70.4	**79.2**	75.0	69.7	66.7	11.1	6.7	58.2	**54.7**	83.3	70.4	79.2	72.2	75.8	70.8	13.3	**10.0**
CMPO	**62.2**	**51.6**	80.6	**77.8**	75.0	77.8	**75.8**	66.7	12.2	**10.0**	**64.7**	54.2	80.6	**77.8**	**81.3**	77.8	**78.8**	**75.0**	**15.6**	**10.0**
Black-box	GPT-5.6-Luna										Claude-Sonnet-5									
No Memory	55.6	48.7	86.1	77.8	81.3	77.8	81.8	70.8	21.1	16.7	59.1	46.4	88.9	77.8	83.3	77.8	75.8	66.7	22.2	16.7
RAG	59.8	46.4	83.3	81.5	83.3	80.6	81.8	66.7	15.6	10.0	67.1	47.3	86.1	81.5	81.3	80.6	78.8	70.8	17.8	13.3
Mem0	62.0	52.9	88.9	77.8	85.4	83.3	84.8	75.0	17.8	13.3	61.3	53.1	88.9	81.5	83.3	83.3	81.8	75.0	20.0	13.3
MemP	66.0	50.7	91.7	85.2	85.4	80.6	84.8	70.8	20.0	16.7	73.6	56.7	91.7	77.8	85.4	86.1	84.8	79.2	21.1	16.7
MemRL	73.8	53.3	94.4	85.2	87.5	**88.9**	84.8	83.3	24.4	16.7	72.0	51.1	94.4	81.5	85.4	83.3	81.8	83.3	25.6	20.0
MemQ	66.0	61.1	88.9	88.9	85.4	83.3	90.9	79.2	27.8	20.0	72.9	**60.9**	86.1	85.2	85.4	80.6	87.9	75.0	28.9	20.0
RoMeRL	72.0	60.9	88.9	92.6	89.6	86.1	87.9	79.2	25.6	20.0	74.4	57.1	94.4	88.9	**89.6**	86.1	87.9	79.2	27.8	23.3
CMPO	**78.9**	**66.4**	**97.2**	**96.3**	**91.7**	**88.9**	**93.9**	**87.5**	**32.2**	**26.7**	**81.6**	60.7	**97.2**	**92.6**	87.5	**88.9**	**90.9**	**87.5**	**32.2**	**26.7**

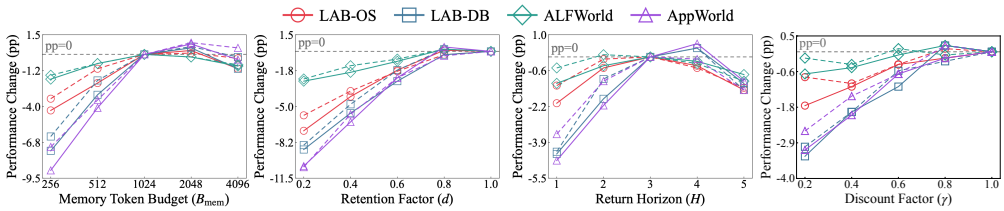


Figure 5: **Sensitivity to the return horizon H, retention factor d, memory-token budget B_{mem} and discount factor γ.** Solid and dashed curves denote GPT-5.6-Luna and Claude-Sonnet-5, respectively, and colors and markers distinguish the four benchmarks. Each panel varies one hyperparameter while fixing the others to the main-text configuration $H = 3$, $d = 1.0$, $B_{\text{mem}} = 1024$, $\gamma = 1.0$. The vertical axis reports the mean epoch-10 performance change (in percentage points) over three seeds relative to this configuration, and the gray dashed line marks $\text{pp} = 0$.

and learned residuals $\{\omega_i\}$ (or baseline policy parameters) are frozen and evaluated on the held-out tasks with one rollout per task across three random seeds.

As summarized in Table 7, CMPO maintains superior transfer performance across all backbones. In the white-box setting, CMPO achieves average success rates of 56.9% and 59.4% across held-out OS and DB tasks, and ALFWorld macro-average success rates of 75.6% and 78.6%, under Llama-3.1-8B and Qwen3-8B, respectively. On AppWorld, the corresponding TGC scores are 12.2% and 15.6%, with SGC of 10.0% under both backbones. In the black-box setting, CMPO achieves the highest generalization scores under both GPT-5.6-Luna and Claude-Sonnet-5, 92.6% and 90.8% macro-average on ALFWorld, and 32.2% and 32.2% TGC with 26.7% SGC on AppWorld. These results highlight the structural advantage of gate-local credit assignment for zero-shot transfer. Baselines relying on shared-return utility updates (such as MemRL and MemQ) degrade severely on held-out tasks because undivided trajectory returns over-credit task-specific scripts, which become irrelevant distractors in novel contexts. In contrast, CMPO isolates the gate-local incremental contribution of each entry through signed propensity-corrected credit, penalizing context-dependent shortcuts while consolidating invariant procedural heuristics, such as proactive schema verification in DB or systematic search routines in ALFWorld. CMPO thereby mitigates negative experience transfer and remains robust in unseen deployment environments.

K Limitations and Future Directions

While CMPO effectively mitigates memory credit leakage in long-horizon interactions, it presents limitations that warrant future investigation.

Joint optimization with memory writing. CMPO focuses on which memories to expose and treats the memory writer $\mathcal{W}$ as a fixed component shared across all compared methods, which isolates the effect of credit assignment from that of memory generation. A natural extension is to jointly optimize

writing and selection under a unified propensity-corrected objective, in which the writer proposes candidate memories and CMPO provides per-candidate credit signals for content-level refinement, potentially improving the quality of the memory bank while retaining single-rollout efficiency.

Cross-session and multi-user memory sharing. Our experiments consider a single agent interacting within one session or across successive sessions of the same user. Extending CMPO to cross-session and multi-user memory sharing raises further questions about cross-agent credit attribution and privacy-aware exposure control. The propensity-corrected estimator generalizes naturally to these settings, although the corresponding selection policies and update rules warrant further study.

L Prompt Templates

This appendix presents the prompt templates used for agent interaction and memory operations across the three benchmarks. All prompts follow a unified structure: a system instruction defining the role and constraints, followed by the task description, retrieved memories (if any), and interaction history.

L.1 Agent Interaction Prompts

System Prompt for OS Tasks

You are a Linux system administrator. You will be given a task to complete using shell commands. At each step, you observe the terminal output and issue the next command.
Rules:
- Issue exactly one command per turn.
- Do not use interactive editors (e.g., vim, nano). Use command-line tools instead.
- When the task is complete, output: DONE.

Relevant Experience:
{retrieved_memories}
Task:
{task_instruction}
Interaction History:
{history}

System Prompt for DB Tasks

You are a database administrator. You will be given a task involving relational databases. At each step, you issue a SQL query or command and observe the result.
Rules:
- Issue exactly one SQL statement per turn.
- Always examine the schema before writing complex queries.
- When the task is complete, output: DONE.

Relevant Experience:
{retrieved_memories}
Task:
{task_instruction}
Interaction History:
{history}

System Prompt for ALFWorld Tasks

You are a household robot operating in a text-based environment. You will be given a task to complete by navigating rooms and interacting with objects.
Rules:
- Issue exactly one action per turn from the set of admissible actions.
- Explore systematically: check likely locations before exhaustive search.
- When the task is complete, the environment will signal success.

Relevant Experience:
{retrieved_memories}
Task:
{task_instruction}
Current Observation:
{observation}

System Prompt for AppWorld Tasks

You are a digital assistant that completes user requests by calling APIs of interconnected applications. You have access to the following API endpoints:
{api_documentation}
Rules:
- Call exactly one API per turn.
- Verify prerequisites before calling dependent APIs.
- Handle errors gracefully and retry with corrected parameters.
- When all goals are complete, output: DONE.

Relevant Experience:
{retrieved_memories}
User Request:
{task_instruction}
Execution History:
{history}

L.2 Memory Writing Prompt

The memory writer distills completed interactions into concise, reusable experience entries. It receives the full trajectory and produces a structured memory.

Memory Writing Prompt

You are a memory curator for an AI agent. Given a completed interaction trajectory, extract a concise and reusable experience entry that could help the agent solve similar tasks in the future.
Trajectory:
Task: {task_instruction}
Actions and Observations: {trajectory}
Outcome: {success_or_failure}
Instructions:
- Summarize the key strategy or insight, not the full trajectory.
- If the task failed, describe what went wrong and what should be done differently.
- If the task succeeded, describe the critical steps that led to success.
- Keep the memory under 200 tokens.

Output format:
[Task Type]: Brief description of the task category.
[Strategy]: The key insight or approach.
[Outcome]: Success/Failure and brief reason.