WHICH MEMORY OPERATION DRIVES RECOVERY? A FACTORIAL STUDY OF RETRIEVE, WRITE, AND MANAGE ADAPTATION UNDER DOMAIN SHIFT

Zhaoxiang Feng* **Mingyang Yao*** **Charlie Sun*** **David Scott Lewis†**

ABSTRACT

Memory-augmented LLM agents use external stores to accumulate and reuse experience across lifelong deployment, yet existing work treats memory as a monolithic system, adapting retrieval while leaving write and manage policies fixed. We argue that memory adaptation should be studied as a factorial problem across three coupled operations: *retrieving* stored experience (PROVIDE), *writing* new experience (TAKE-IN), and *maintaining* store quality (MANAGE). We introduce OMAC, an online memory architecture controller that uses block-level Thompson sampling over the joint configuration space of all three operations, and a factorial ablation framework that isolates each operation's contribution by selectively enabling or disabling its adaptation. Through controlled factorial experiments on a streaming domain shift, we find that adaptive store management consistently outperforms retrieval-only adaptation for in-distribution learning, challenging the prevailing assumption that retrieval is the primary lever for memory-augmented agents. We further show that the value of memory adaptation is modulated by domain difficulty: when the base model is already competent, no adaptation strategy provides meaningful lift, whereas management adaptation yields substantial gains when the model struggles. These findings establish that the memory lifecycle deserves the same systematic, per-operation analysis that has been applied to retrieval, and that store management is an underexplored yet high-impact axis of improvement for deployed agents.

1 INTRODUCTION

Lifelong LLM agents operate in settings where tasks arrive as streams and the input distribution is non-stationary. While continual fine-tuning can adapt model weights, it is often impractical in deployment due to compute cost, catastrophic forgetting, and the need to serve multiple users from a single checkpoint. External memory offers a lightweight alternative: the agent accumulates experiences, retrieves relevant ones, and reuses them to improve future behavior without any weight update. A growing body of work demonstrates that agents can extract reusable abstractions and store them externally, from episodic memory banks (Zhong et al., 2024) and causal abstractions (Majumder et al., 2023) to distilled experience rules (Zhao et al., 2024; Shinn et al., 2023), workflow libraries (Wang et al., 2025b; 2024), and structured knowledge bases (Tang et al., 2025a).

However, memory is not a single retrieval knob. A deployed agent's memory involves three coupled operations: **PROVIDE** (what to retrieve, how much context to inject, how to gate against harmful retrieval), **TAKE-IN** (what to write, when to write, at what granularity), and **MANAGE** (how to prune stale items, consolidate redundant content, control store growth). These operations interact: if TAKE-IN writes noisy items, the store degrades; PROVIDE then retrieves distractors; MANAGE must compensate. LifelongAgentBench explicitly highlights that naïve memory replay can degrade performance when irrelevant content crowds out useful context (Zheng et al., 2025), and Agent KB similarly incorporates a disagreement gate to mitigate harmful knowledge interference (Tang et al., 2025a).

Prior adaptive-retrieval approaches (Wang et al., 2025a; Jeong et al., 2024; Tang et al., 2025c) learn online which retrieval strategy to use, but assume a static or externally-maintained corpus: they

*University of California San Diego †AI Executive Consulting (AIXC)

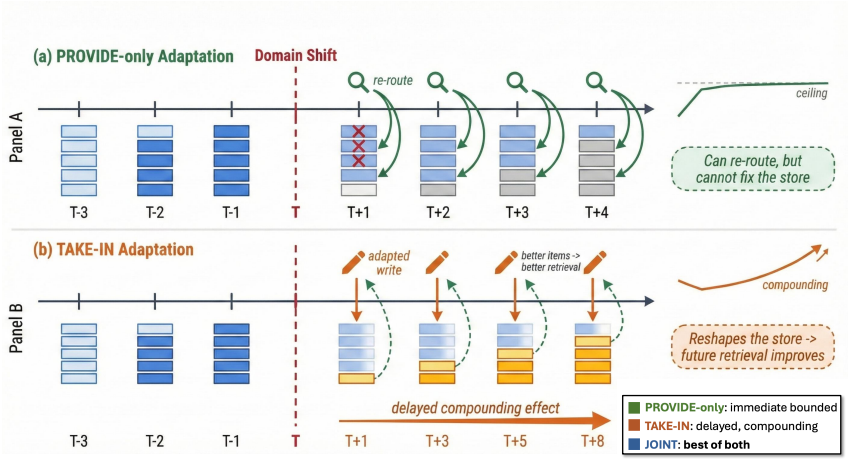

Figure 1: **Delayed compounding effect of write vs. retrieval adaptation.** **(a)** PROVIDE-only adaptation can re-route around stale items after a domain shift, but the store remains polluted with code-domain content (blue), yielding an immediate but bounded improvement (ceiling). **(b)** TAKE-IN adaptation reshapes the store itself: new math-domain items (orange) gradually replace stale ones, improving future retrieval quality with a delayed, compounding effect. Joint adaptation combines both mechanisms.

adapt PROVIDE in isolation. Meanwhile, agent memory systems like CLIN (Majumder et al., 2023), A-MEM (Xu et al., 2025b), and Agent KB (Tang et al., 2025a) update memory content online but with fixed write and manage policies. Offline evolution can discover strong memory configurations (Zhang et al., 2025b), but these are frozen at deployment and cannot respond to distribution shift. No prior work provides a **factorial decomposition** of which operation's adaptation drives recovery under domain shift. This gap is illustrated in Figure 1: retrieval-only adaptation can re-route around stale items but cannot fix a polluted store, while write adaptation reshapes the store itself with a delayed, compounding effect.

We address this gap with a controlled experiment. We introduce OMAC (Online Memory Architecture Controller), a block-level structured bandit that jointly selects configurations across all three operations using Thompson sampling with Bayesian linear regression. By independently enabling or disabling adaptation for each operation while freezing the others, we isolate each operation's contribution under a streaming code-to-math domain shift using LiveCodeBench (Jain et al., 2025) and MATH (Hendrycks et al., 2021).

Our study yields three findings: (1) memory provides substantial in-distribution benefit, raising code accuracy by 14–21 percentage points over a no-memory baseline; (2) online adaptation of MANAGE yields the largest pre-shift gains, outperforming retrieval-only adaptation by 6.8 points; and (3) post-shift recovery under an easy target domain is dominated by the base model's intrinsic competence rather than any memory adaptation strategy. These results suggest that practitioners should invest in adaptive store management alongside or even before sophisticated retrieval.

2 RELATED WORK

Agent memory systems. External memory for LLM agents has been explored through multiple paradigms. Episodic memory stores persist raw experiences across sessions (Zhong et al., 2024; Park et al., 2023); CLIN accumulates causal abstractions that transfer across tasks without parameter updates (Majumder et al., 2023); ExpeL distills experience into natural-language insights for future reuse (Zhao et al., 2024); and Reflexion stores verbal self-critiques as a feedback mechanism (Shinn et al., 2023). More recent systems emphasize richer memory structure: A-MEM organizes memories as linked atomic notes (Xu et al., 2025b), Agent Workflow Memory induces reusable procedural workflows (Wang et al., 2025b), and Voyager builds a growing skill library (Wang et al., 2024). Agent KB targets cross-framework experience sharing with a disagreement gate for interference con-

trol (Tang et al., 2025a). MemP frames memory management as a Markov decision process (Fang et al., 2025). These systems implement some or all of the PROVIDE/TAKE-IN/MANAGE lifecycle but with fixed heuristics for each operation. None provide factorial attribution of which operation matters under domain shift.

Adaptive retrieval. A separate line of work adapts retrieval strategy at deployment time. Adaptive-RAG routes queries to different retrieval depths based on complexity (Jeong et al., 2024). Astute RAG handles knowledge conflicts between retrieved passages and parametric knowledge (Wang et al., 2025a). Tang et al. use multi-armed bandits to route among heterogeneous retrievers under non-stationary query distributions (Tang et al., 2025c). MBA-RAG applies epsilon-greedy bandits to select retrieval strategies (Tang et al., 2025b). CAMAB formulates context segment selection as a combinatorial bandit (Pan et al., 2025). Amber introduces an adaptive memory-based optimizer with multi-granular content filtering (Qin et al., 2025). These methods represent increasingly sophisticated PROVIDE adaptation, but all assume a static or externally-maintained corpus. They do not address the key failure mode for lifelong agents: what gets written and how it is managed determines whether retrieval remains feasible over time. Our work extends the bandit adaptation paradigm from PROVIDE alone to the full PROVIDE $\times$ TAKE-IN $\times$ MANAGE lifecycle.

Online configuration tuning. Thompson sampling provides a principled exploration-exploitation framework (Russo et al., 2018), and non-stationary bandit algorithms handle changing reward distributions via sliding windows and discounted updates (Garivier & Moulines, 2011). StreamBench benchmarks continuous improvement of language agents in a streaming setting aligned with our protocol (Wu et al., 2024). MemEvolve (Zhang et al., 2025b) shows that memory architectures can be evolved offline over the PROVIDE/TAKE-IN/MANAGE configuration space, but the resulting configurations are frozen at deployment. Our contribution is extending this to online adaptation: treating the same configuration space as a compositional action in a non-stationary contextual bandit that adapts continuously during deployment. Additional related work on memory management and self-evolution is discussed in Appendix B.

3 METHOD

3.1 PROBLEM FORMULATION

We consider a deployment stream of episodes $t = 1, \ldots, T$. Each episode provides a task q_t and a verifier V returning a score $s_t \in [0, 1]$. The agent uses a frozen base model f_θ augmented with an external memory store $\mathcal{S}_t$ that evolves as the agent writes experiences. No model weights are updated; adaptation occurs only in the memory configuration and store contents. At each episode, the memory system executes three operations parameterized by a configuration triple $c_t = (c_t^{\text{prov}}, c_t^{\text{take}}, c_t^{\text{man}})$:

$$R_t = \textsc{Provide}(q_t, \mathcal{S}_t;\ c_t^{\text{prov}}), \tag{1}$$

$$y_t = f_\theta(q_t, R_t), \quad s_t = V(y_t), \tag{2}$$

$$W_t = \textsc{Take-In}(q_t, y_t, s_t;\ c_t^{\text{take}}), \tag{3}$$

$$\mathcal{S}_{t+1} = \textsc{Manage}(\mathcal{S}_t \cup W_t;\ c_t^{\text{man}}). \tag{4}$$

This lifecycle is common across agent memory systems (Majumder et al., 2023; Xu et al., 2025b; Fang et al., 2025), but typically with fixed heuristics for each operation. Our contribution is making each operation independently configurable and adaptable.

Each operation has discrete knobs. PROVIDE controls the retriever type (lexical, recency, hybrid), top-k (3, 5, 8), context budget (200, 400, 600 tokens), and a retrieval gate (on/off). TAKE-IN controls the write trigger (always, success-only, failure-only), granularity (episode trace, distilled insight), write budget (100, 200 tokens), and deduplication (on/off). MANAGE controls pruning strategy (FIFO, LRU, score-based), pruning threshold (0.7, 0.9, 1.0), and consolidation (on/off). The full configuration space $\mathcal{C} = \mathcal{C}_{\text{prov}} \times \mathcal{C}_{\text{take}} \times \mathcal{C}_{\text{man}}$ contains $|\mathcal{C}| = 23{,}328$ configurations. Because TAKE-IN and MANAGE execute *after* the episode reward is observed, per-episode updates cannot learn their value from same-episode feedback. We therefore operate the controller at block level: the configuration is selected once per block of B episodes and held fixed within the block. Block reward aggregates episode-level feedback, enabling delayed-credit assignment for write and manage deci-

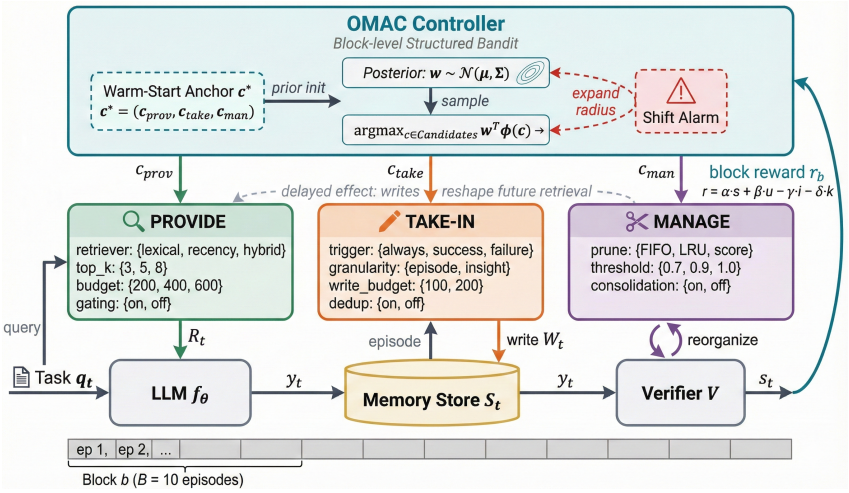

Figure 2: **OMAC system architecture.** The controller (top) selects a memory configuration triple $(c_{\text{prov}}, c_{\text{take}}, c_{\text{man}})$ via Thompson sampling over a structured candidate set. Three operation modules (middle) execute retrieval, writing, and management with the selected knobs. The agent (bottom) solves tasks using the LLM augmented with retrieved context, and block-level reward feeds back to update the controller's posterior. The dashed arrow between TAKE-IN and PROVIDE highlights the delayed effect: write decisions reshape the store, changing future retrieval quality.

sions. This design choice directly addresses the delayed effect shown in Figure 1: write decisions at block b affect retrieval quality at block $b{+}1$ and beyond.

3.2 OMAC CONTROLLER

OMAC selects memory configurations using non-stationary Thompson sampling with Bayesian linear regression (Figure 2). The controller operates above any memory provider implementation as a meta-controller, following the algorithmic foundations of Thompson sampling (Russo et al., 2018) adapted for structured, non-stationary environments (Garivier & Moulines, 2011).

Feature encoding. Each configuration c is encoded as a feature vector $\phi(c) \in \mathbb{R}^d$ comprising three groups of features: (1) **main effects**: one-hot indicators for each knob value across all three operations ($\sim$28 raw features, padded with within-operation interactions to $\sim$51); (2) **cross-operation interactions**: pairwise outer products of key knobs between operation pairs, capturing dependencies such as the interaction between retriever type (PROVIDE) and write granularity (TAKE-IN), yielding $\sim$48 features; and (3) **memory-state features**: store size bucket, fullness fraction, rolling accuracy, domain entropy, and episodes since last alarm (5 features). The total dimensionality is $d \approx 104$.

The cross-operation interaction features are critical for capturing the coupling between operations. For example, the combination of aggressive pruning (MANAGE) with high-volume writing (TAKE-IN) behaves differently than aggressive pruning with conservative writing. Without interaction features, the controller would treat each operation as independent, missing these compositional effects.

Reward model and posterior inference. We model block reward as linear in the features:

$$r_b = \phi(c_b)^\top \mathbf{w} + \epsilon_b, \quad \epsilon_b \sim \mathcal{N}(0, \sigma^2), \tag{5}$$

where $r_b = \alpha \bar{s}_b + \beta \bar{u}_b - \gamma \bar{i}_b - \delta \bar{k}_b$ aggregates four block-level signals: mean task score $\bar{s}_b$ (primary), mean retrieval utility $\bar{u}_b$ (how often retrieved items appeared in correct solutions), mean interference rate $\bar{i}_b$ (how often retrieved items appeared in incorrect solutions), and mean cost $\bar{k}_b$ (normalized token usage). We use weights $(\alpha, \beta, \gamma, \delta) = (1.0, 0.2, 0.3, 0.1)$, chosen to emphasize task performance while penalizing interference more than cost. The multi-objective formulation follows the spirit of latency-aware bandit routing (Tang et al., 2025c). We maintain a Bayesian posterior $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ updated via standard Bayesian linear regression. To handle non-stationarity, we use

a sliding window of the most recent $W = 50$ block observations: old observations are discarded as new ones arrive, allowing the posterior to track regime changes without explicit change-point detection. Given the observation history $\{(\boldsymbol{\phi}_i, r_i)\}_{i \in \text{window}}$, the posterior parameters are:

$$\boldsymbol{\Sigma}^{-1} = \sigma_0^{-2}\mathbf{I} + \sigma^{-2} \sum_i \boldsymbol{\phi}_i \boldsymbol{\phi}_i^\top, \tag{6}$$

$$\boldsymbol{\mu} = \boldsymbol{\Sigma} \left(\sigma^{-2} \sum_i \boldsymbol{\phi}_i r_i \right), \tag{7}$$

with prior variance $\sigma_0^2 = 1.0$ and noise variance $\sigma^2 = 0.5$. At each block, OMAC samples a weight vector $\tilde{\mathbf{w}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and selects the configuration that maximizes the predicted reward over a candidate set:

$$c_b = \arg\max_{c \in \mathcal{C}_{\text{cand}}} \boldsymbol{\phi}(c)^\top \tilde{\mathbf{w}}. \tag{8}$$

The stochasticity of the sampled weights provides natural exploration, with the degree of exploration determined by posterior uncertainty (Russo et al., 2018).

3.3 CANDIDATE SET AND EXPLORATION

The full configuration space ($|\mathcal{C}| = 23{,}328$) is too large to enumerate at each block. We construct a tractable candidate set $\mathcal{C}_{\text{cand}}$ around an anchor configuration c^*:

Hamming neighborhood. We include all configurations within Hamming distance r of c^* in the discrete knob space. At radius r=2, this produces a manageable set that captures local variations while remaining computationally tractable.

Sentinel configurations. We add $n_s = 3$ diverse sentinel configurations sampled to be far from the anchor (Hamming distance ≥ 3), providing long-range exploration to escape local optima.

Warm-start mechanism. During the first $B_{\text{warm}} = 3$ blocks, we restrict the candidate radius to r=1, constraining early exploration to small perturbations of c^*. This warm-start phase reduces the risk of catastrophic configurations during initial deployment while still gathering informative observations. After B_{warm} blocks, the radius expands to r=2.

Shift alarm. We track the fraction of recent blocks in which the controller selects a non-anchor configuration (exploration rate). When this rate exceeds a threshold of $\tau = 2.0$ standard deviations above the historical mean over a sliding window, we trigger a shift alarm that: (1) resets the observation window, discarding potentially outdated reward signals; and (2) expands the candidate radius, encouraging broader exploration. This provides an implicit shift signal through exploration dynamics rather than requiring an explicit domain detector.

3.4 FACTORIAL ABLATION DESIGN

To isolate each operation's contribution, we define a mask $\mathbf{m} = (m_{\text{prov}}, m_{\text{take}}, m_{\text{man}}) \in \{0,1\}^3$ specifying which operations the controller may adapt. For masked operations ($m = 0$), the configuration is frozen at the warm-start value c^*; only unmasked operations ($m = 1$) participate in bandit selection. Table 1 lists the six resulting conditions and their masks. When the mask restricts adaptation to a single operation, the candidate set is also restricted to vary only that operation's knobs, reducing the effective search space and focusing exploration.

Algorithm 1 summarizes the full controller loop (hyperparameters and feature details in Appendix A). The key design choices that distinguish OMAC from prior bandit-based retrieval routers (Tang et al., 2025c;b) are: (1) the action space spans the joint configuration of all three memory operations, not just retrieval routing; (2) block-level aggregation enables delayed credit assignment for write and manage decisions; and (3) the factorial mask enables controlled ablation studies that isolate each operation's contribution.

4 EXPERIMENTAL DESIGN

Evaluation stream and conditions. We concatenate 479 code episodes from LiveCodeBench (Jain et al., 2025), a streaming benchmark of programming problems with executable

Algorithm 1 OMAC Controller Loop

Require: Task stream $\{q_t\}_{t=1}^{T}$, block size B, mask $\mathbf{m}$, anchor c^*
Initialize posterior $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with prior; history $\mathcal{H} \leftarrow \emptyset$
for block $b = 1, 2, \ldots, \lceil T/B \rceil$ **do**
  $\mathcal{C}_{\text{cand}} \leftarrow$ BUILDCANDIDATES(c^*, $\mathbf{m}$, radius r)
  Sample $\tilde{\mathbf{w}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ ▷ Thompson sampling
  $c_b \leftarrow \arg\max_{c \in \mathcal{C}_{\text{cand}}} \boldsymbol{\phi}(c)^\top \tilde{\mathbf{w}}$
  $c_b \leftarrow$ APPLYMASK(c_b, c^*, $\mathbf{m}$) ▷ freeze masked ops
  for episode t in block b **do**
    $R_t \leftarrow$ PROVIDE$(q_t, \mathcal{S}_t;\ c_b^{\text{prov}})$
    $y_t \leftarrow f_\theta(q_t, R_t)$; $s_t \leftarrow V(y_t)$
    $W_t \leftarrow$ TAKE-IN$(q_t, y_t, s_t;\ c_b^{\text{take}})$
    $\mathcal{S}_{t+1} \leftarrow$ MANAGE$(\mathcal{S}_t \cup W_t;\ c_b^{\text{man}})$
  end for
  $r_b \leftarrow \alpha \bar{s}_b + \beta \bar{u}_b - \gamma \bar{i}_b - \delta \bar{k}_b$ ▷ block reward
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{\phi}(c_b), r_b)\}$; trim to window W
  Update posterior $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ from $\mathcal{H}$
  if SHIFTALARM($\mathcal{H}$) **then** reset window; expand r
end for

Table 1: **Factorial conditions.** Each row is one experimental system. Colored ADAPT cells indicate that the controller tunes that operation's knobs online; gray FROZEN cells indicate the operation stays at the warm-start anchor c^*; dashes indicate no memory. The evaluation stream concatenates 479 code episodes (LiveCodeBench) followed by 500 math episodes (MATH), creating a single domain shift at episode 479.

System	PROVIDE	TAKE-IN	MANAGE	Mask $\mathbf{m}$
NO-MEMORY	—	—	—	n/a
FROZEN-EVOLVED	frozen	frozen	frozen	$(0, 0, 0)$
ONLINE-JOINT	**adapt**	**adapt**	**adapt**	$(1, 1, 1)$
PROVIDE-ONLY	**adapt**	frozen	frozen	$(1, 0, 0)$
TAKE-IN-ONLY	frozen	**adapt**	frozen	$(0, 1, 0)$
MANAGE-ONLY	frozen	frozen	**adapt**	$(0, 0, 1)$

test suites, and 500 math episodes from MATH (Hendrycks et al., 2021), competition-level mathematics with exact-match verification. This creates a single domain shift at episode 479, yielding 979 total episodes per run. The memory store accumulates code-domain experiences before the shift, creating a realistic scenario where the agent must handle a store populated with potentially stale content. This streaming evaluation protocol follows the philosophy of StreamBench (Wu et al., 2024) and Evo-Memory (Wei et al., 2025), which advocate for sequential task streams to evaluate memory evolution. We evaluate the six conditions defined in Section 3.4, all sharing the same base LLM, the same task stream, and the same warm-start anchor configuration c^*. The conditions differ only in the adaptation mask $\mathbf{m}$ (Table 1).

Setup. The anchor configuration c^* uses mid-range settings across all operations: lexical retrieval with k=5 and a 400-token budget (no gating) for PROVIDE; write-always at episode granularity with a 200-token budget (no deduplication) for TAKE-IN; and FIFO pruning at threshold 0.9 (no consolidation) for MANAGE. This configuration represents a reasonable default that avoids extreme choices in any operation; in a full deployment pipeline, c^* could be the output of an offline evolution procedure such as MemEvolve (Zhang et al., 2025b), but here we use a curated mid-range anchor to avoid confounding adaptation effects with the quality of the initial search. Our experimental infrastructure, including the memory provider abstraction, the PROVIDE/TAKE-IN/MANAGE lifecycle, and the discrete knob space, builds on the MemEvolve codebase; the primary addition is the online OMAC controller and the factorial ablation framework described in Section 3.4. All condi-

Table 2: Main results across six factorial conditions. Pre-shift = code (episodes 1–479), post-shift = math (episodes 480–979). Mean ± 95% bootstrap CI over 10 seeds. Best values **bolded**.

Condition	Pre-shift (Code)	Post-shift (Math)	Overall	AURC (post)
NO-MEMORY	0.394 ± 0.016	0.935 ± 0.029	0.665	0.946 ± 0.021
FROZEN-EVOLVED	0.538 ± 0.105	0.955 ± 0.008	0.747	0.948 ± 0.007
ONLINE-JOINT	0.582 ± 0.094	0.954 ± 0.008	0.768	0.952 ± 0.010
PROVIDE-ONLY	0.540 ± 0.073	0.947 ± 0.024	0.744	0.942 ± 0.015
TAKE-IN-ONLY	0.570 ± 0.082	0.956 ± 0.006	0.763	**0.953** ± 0.008
MANAGE-ONLY	**0.608** ± 0.091	**0.957** ± 0.005	**0.782**	0.947 ± 0.008

tions, including FROZEN-EVOLVED (which deploys c^* without any online adaptation), start from this same anchor, ensuring that observed differences reflect online adaptation rather than the initial configuration. Code episodes are graded by comparing predictions against executable test suites; a prediction is correct if it passes all test cases. Math episodes are graded by extracting the numerical or symbolic answer and comparing against the gold answer under normalization (simplifying fractions, standardizing notation). Both domains produce binary scores ($s_t \in \{0, 1\}$).

Metrics and implementation. We report: (1) rolling accuracy (window of 10 episodes) over the stream; (2) mean accuracy per domain phase (pre-shift code, post-shift math); (3) area under the recovery curve (AURC) in the post-shift region, computed as the normalized integral of rolling accuracy from the shift point to the end of the stream; and (4) 95% bootstrap confidence intervals computed over 10 seeds with 1,000 bootstrap resamples. The base model is Qwen3-8B (`qwen/qwen3-8b`) accessed via the OpenRouter API with a token budget of 400. Each condition is run with 10 random seeds, yielding 60 independent runs (6 conditions $\times$ 10 seeds). Block size $B = 10$ episodes. Each run processes 979 episodes, totaling 58,740 episodes and API calls across all runs. We chose 10 seeds to balance statistical reliability against computational cost; the resulting bootstrap CIs are sufficiently narrow to distinguish the main effects ($\pm$0.01–0.10).

5 RESULTS

5.1 MEMORY PROVIDES SUBSTANTIAL IN-DISTRIBUTION BENEFIT

Table 2 presents the main results. The most striking finding is the large gap between NO-MEMORY and all memory-equipped conditions during the pre-shift code phase. Without memory, the agent achieves 39.4% accuracy on code tasks. With memory (any configuration, frozen or adapted), accuracy rises to 53.8–60.8%, a gain of 14–21 percentage points. This confirms that external memory provides substantial value for in-distribution deployment even with a relatively small model, consistent with findings across the agent memory literature (Zhao et al., 2024; Majumder et al., 2023; Liu et al., 2025).

5.2 ONLINE ADAPTATION IMPROVES IN-DISTRIBUTION LEARNING

Among memory-equipped conditions, online adaptation consistently outperforms the frozen baseline during the code phase, confirming that dynamic configuration tuning provides value beyond a fixed deployment strategy. FROZEN-EVOLVED achieves 53.8% on code, while the best single-operation ablation (MANAGE-ONLY) reaches 60.8%, a 7-point improvement. ONLINE-JOINT achieves 58.2%, and TAKE-IN-ONLY 57.0%. This ordering is notable: MANAGE adaptation alone provides the largest in-distribution benefit.

Why does MANAGE dominate? The mechanism is intuitive and follows directly from the memory lifecycle structure. During the code phase, the store grows continuously as the agent writes new experiences. Without adaptive management, the store accumulates both useful and noisy items, eventually degrading retrieval precision. Adaptive pruning keeps the store lean and relevant, improving the signal-to-noise ratio of future retrievals. This is a form of indirect retrieval improvement: rather than optimizing how to query a noisy store (PROVIDE), MANAGE improves the store itself.

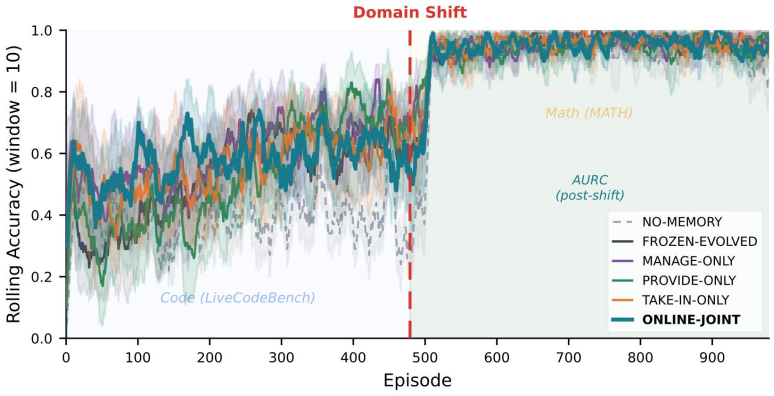

Figure 3: Rolling accuracy (window=10) over the episode stream for all six conditions, averaged over 10 seeds with 95% bootstrap CI bands. The vertical dashed line marks the code-to-math domain shift at episode 479. All conditions recover rapidly after the shift because Qwen3-8B is intrinsically stronger at math than code.

This finding aligns with the growing emphasis on memory management in recent work (Zheng et al., 2025; Xu et al., 2025a; Fang et al., 2025), which argues that controlling what stays in memory is at least as important as controlling what is retrieved.

The wide confidence intervals (standard deviation ≈ 0.10) reflect the stochastic nature of bandit exploration during early blocks. Some seeds find good configurations quickly; others explore longer. This variance decreases as the controller accumulates observations.

5.3 Post-shift recovery is dominated by domain difficulty

Figure 3 shows the recovery curves. After the domain shift from code to math, all conditions jump to high accuracy within the first post-shift block. This rapid convergence occurs because Qwen3-8B achieves $\sim$94% accuracy on MATH tasks even without memory, compared to $\sim$39% on LiveCodeBench without memory. The shift is from a hard domain to an easy one.

Post-shift AURC values are tightly clustered: 0.942 (PROVIDE-ONLY) to 0.953 (TAKE-IN-ONLY), a spread of only 0.011 with all confidence intervals overlapping. Time-to-recovery is uniformly minimal: all conditions reach target performance within the first post-shift block. This ceiling effect compresses the differences between adaptation strategies. When the base model is already near-perfect on the target domain, there is little room for memory to help or hurt.

5.4 Factorial decomposition: pre-shift vs. post-shift

The factorial structure reveals a clear asymmetry between the two phases:

Pre-shift (code, harder domain): Adaptation differences are meaningful. The ordering MANAGE-ONLY $>$ ONLINE-JOINT $>$ TAKE-IN-ONLY $>$ PROVIDE-ONLY $\approx$ FROZEN-EVOLVED suggests that store quality is the bottleneck. MANAGE adaptation directly improves store quality; TAKE-IN adaptation reshapes what enters the store; PROVIDE adaptation can only route around existing content. This ordering is consistent with the delayed-effect hypothesis (Figure 1): operations that modify the store have compounding effects, while retrieval-only adaptation provides immediate but bounded improvement.

Post-shift (math, easier domain): Adaptation differences are negligible. All memory conditions perform within 1 percentage point of each other (94.7–95.7%), and even NO-MEMORY reaches 93.5%. The ceiling eliminates the signal needed to distinguish operation contributions.

This asymmetry suggests that the value of factorial memory adaptation is *domain-difficulty-dependent*. When the agent struggles, adapting how memory is managed provides meaningful lift.

When the agent is already competent, the marginal value of any memory optimization is small. This echoes the transfer asymmetry reported in prior work: Agent KB (Tang et al., 2025a) finds that reasoning experience benefits software-engineering tasks but not vice versa, and SEDM (Xu et al., 2025a) observes asymmetric knowledge diffusion across domains. Our finding extends this pattern to the *adaptation* level—not just what is stored, but how the memory lifecycle is controlled.

5.5 Exploration dynamics

OMAC explores most aggressively under the ONLINE-JOINT condition (exploration rate 1.0), consistent with having the largest configuration space. The single-operation conditions show exploration rates of 0.94–0.97. The shift alarm mechanism triggered inconsistently across seeds, as the jump in performance at the domain boundary confounded the spike-detection heuristic. In settings where the shift causes a performance *drop* rather than a jump, we expect the alarm to be more informative.

6 Discussion

Our most important caveat is that the code-to-math shift with Qwen3-8B is not the kind of challenging shift our framework was designed to study. Because the model's math accuracy far exceeds its code accuracy (94% vs. 39% without memory), post-shift recovery is trivial and all conditions converge within the first block. The title asks which operation drives recovery, but the honest answer from our data is that *domain difficulty* drives recovery: when the base model is already near-perfect on the target domain, no memory adaptation strategy can meaningfully differentiate itself. A more informative experiment would use a shift where the model is weak on both sides (e.g., code to a low-resource language, or standard math to adversarial math), or the reverse direction (math to code), where stale math memories would actively interfere with code retrieval. We view the current post-shift results as establishing that our experimental framework works end-to-end, rather than as a definitive answer to which operation matters most for recovery.

However, the hard-to-easy shift direction, while limiting the post-shift story, actually strengthens the pre-shift finding. During the 479-episode code phase, the base model struggles (39% without memory), creating exactly the conditions where memory adaptation should matter. The clear ordering MANAGE-ONLY $>$ ONLINE-JOINT $>$ TAKE-IN-ONLY $>$ PROVIDE-ONLY $\approx$ FROZEN-EVOLVED emerges in this regime without any confound from an impending easy recovery. The MANAGE-ONLY advantage of 6.8 points over PROVIDE-ONLY is an in-distribution learning result, and we present it as such rather than as a domain-shift result.

That said, the pairwise differences between adaptive conditions do not reach statistical significance at $\alpha = 0.05$ (Appendix C): only the FROZEN-EVOLVED vs. NO-MEMORY gap ($\Delta = +0.144$, $p = 0.003$) is significant, while MANAGE-ONLY vs. TAKE-IN-ONLY merely approaches it ($p = 0.059$). The ordering is consistent across seeds but the wide confidence intervals ($\pm$0.08–0.10) prevent definitive ranking. Larger-scale experiments with more seeds or longer streams would be needed to confirm the MANAGE advantage with statistical rigor.

The pre-shift results challenge the common assumption that retrieval adaptation is the primary lever for memory-augmented agents. MANAGE-ONLY outperforms PROVIDE-ONLY by 6.8 percentage points during the code phase, suggesting that keeping the store clean is more valuable than optimizing how to query a noisy store, at least in the early deployment regime where the store is growing rapidly. This finding echoes the observations of MemP (Fang et al., 2025), which models memory management as an MDP, and SEDM (Xu et al., 2025a), which validates candidate memories before admission. If confirmed under harder shifts, these results suggest that practitioners building lifelong agents should invest in adaptive store management (pruning, consolidation, quality gating) alongside or even before sophisticated retrieval strategies. The standard approach of optimizing retrieval while leaving write and manage policies fixed may leave significant performance on the table, consistent with the broader trend toward active memory management (Xu et al., 2025b; Zhang et al., 2026; Chhikara et al., 2025). More broadly, the factorial decomposition offers a tool for auditing how a lifelong agent's memory evolves: by toggling each operation's adaptation independently, developers can identify which component is responsible for behavioral changes and constrain the others when predictability or safety is required.

Shift alarm behavior. The shift alarm triggered in only 3 of 40 adaptive runs, which may appear to indicate a design flaw. However, this is a predictable consequence of the hard-to-easy shift direction: the alarm monitors for exploration rate spikes driven by reward drops, but our domain shift causes a reward *jump*, not a drop. The alarm is designed for the common case where a shift degrades performance and the controller must explore to find a new good configuration. When performance improves at the shift, the controller has no reason to increase exploration, and the alarm correctly does not fire. In a hard-to-hard shift where stale memories cause a performance drop, we expect the alarm to behave as intended. Incorporating feature-space statistics (e.g., domain entropy of retrieved items) alongside reward-based detection would make the alarm responsive to shift direction regardless of difficulty, and is a natural extension.

Limitations. Several limitations apply beyond the domain-difficulty asymmetry discussed above. We evaluate a single LLM (Qwen3-8B); memory dynamics may differ with larger models that have stronger parametric knowledge or longer context windows. Our 979-episode stream with a single shift is a controlled first step toward truly lifelong evaluation; sequential multi-domain shifts (e.g., code $\rightarrow$ math $\rightarrow$ reasoning) would test whether the MANAGE advantage compounds or saturates over successive transitions. The warm-start mechanism (Section 3.3) is proposed but not validated against a cold-start baseline. We also omit controls that would strengthen causal claims: a random-configuration baseline, an oracle tuned on the target domain, and token-matched placebo retrieval. The memory search space is discretized and restricted: MANAGE is limited to pruning and consolidation, TAKE-IN cannot introduce tool-use or API-based memory types, and the write budget is capped at 200 tokens. Richer operation spaces that include memory systems with fundamentally different architectures (e.g., graph-structured stores, generative memory) would test whether the MANAGE advantage holds beyond our current design space. These are important directions for a full-length study.

7 CONCLUSION

We presented a factorial study of memory operation adaptation under domain shift. Using OMAC, a block-level Thompson sampling controller over the joint PROVIDE $\times$ TAKE-IN $\times$ MANAGE configuration space, we isolated each operation's contribution across six conditions and 60 experimental runs. Our results show that memory provides 14–21 points of in-distribution benefit, that online adaptation of MANAGE yields the largest pre-shift gains, and that post-shift recovery under an easy target domain is dominated by the base model's competence rather than memory adaptation strategy. These findings motivate future work on harder domain shifts, richer management policies, and end-to-end evaluation of the memory lifecycle.

REFERENCES

Ryan Prescott Adams and David J. C. MacKay. Bayesian online changepoint detection. *CoRR*, abs/0710.3742, 2007. URL https://arxiv.org/abs/0710.3742.

Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. In *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*, pp. 443–448. SIAM, 2007. doi: 10.1137/1.9781611972771.42. URL https://doi.org/10.1137/1.9781611972771.42.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading. *CoRR*, abs/2310.05029, 2023. doi: 10.48550/ARXIV.2310.05029. URL https://doi.org/10.48550/arXiv.2310.05029.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In Inês Lynce, Nello Murano, Mauro Vallati, Serena Villata, Federico Chesani, Michela Milano, Andrea Omicini, and Mehdi Dastani (eds.), *ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025)*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pp. 2993–

3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *CoRR*, abs/2508.06433, 2025. doi: 10.48550/ARXIV.2508.06433. URL https://doi.org/10.48550/arXiv.2508.06433.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann (eds.), *Algorithmic Learning Theory - 22nd International Conference, ALT 2011, Espoo, Finland, October 5-7, 2011. Proceedings*, volume 6925 of *Lecture Notes in Computer Science*, pp. 174–188. Springer, 2011. doi: 10.1007/978-3-642-24412-4_16. URL https://doi.org/10.1007/978-3-642-24412-4_16.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/6ddc001d07ca4f319af96a3024f6dbd1-Abstract-Conference.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 7036–7050. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.NAACL-LONG.389. URL https://doi.org/10.18653/v1/2024.naacl-long.389.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John F. Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26396–26415. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/lee24c.html.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Yitao Liu, Chenglei Si, Karthik R. Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 14179–14198. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.694/.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter A. Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. CLIN: A continually learning language agent for rapid task adaptation and generalization. *CoRR*, abs/2310.10134, 2023. doi: 10.48550/ARXIV.2310.10134. URL https://doi.org/10.48550/arXiv.2310.10134.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. doi: 10.48550/ARXIV.2310.08560. URL https://doi.org/10.48550/arXiv.2310.08560.

E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954. doi: 10.2307/2333009.

Deng Pan, Keerthiram Murugesan, Nuno Moniz, and Nitesh V. Chawla. Context attribution with multi-armed bandit optimization. *CoRR*, abs/2506.19977, 2025. doi: 10.48550/ARXIV.2506.19977. URL https://doi.org/10.48550/arXiv.2506.19977.

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

Qitao Qin, Yucong Luo, Yihang Lu, Zhibo Chu, Xiaoman Liu, and Xianwei Meng. Towards adaptive memory-based optimization for enhanced retrieval-augmented generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, volume ACL 2025 of *Findings of ACL*, pp. 7991–8004. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.418/.

Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling. *Found. Trends Mach. Learn.*, 11(1):1–96, 2018. doi: 10.1561/2200000070. URL https://doi.org/10.1561/2200000070.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, Ge Zhang, Jiaheng Liu, Xingyao Wang, Sirui Hong, Chenglin Wu, Hao Cheng, Chi Wang, and Wangchunshu Zhou. Agent KB: leveraging cross-domain experience for agentic problem solving. *CoRR*, abs/2507.06229, 2025a. doi: 10.48550/ARXIV.2507.06229. URL https://doi.org/10.48550/arXiv.2507.06229.

Xiaqiang Tang, Qiang Gao, Jian Li, Nan Du, Qi Li, and Sihong Xie. MBA-RAG: a bandit approach for adaptive retrieval-augmented generation through question complexity. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pp. 3248–3254. Association for Computational Linguistics, 2025b. URL https://aclanthology.org/2025.coling-main.218/.

Xiaqiang Tang, Jian Li, Nan Du, and Sihong Xie. Adapting to non-stationary environments: Multi-armed bandit enhanced retrieval-augmented generation on knowledge graphs. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 12658–12666. AAAI Press, 2025c. doi: 10.1609/AAAI.V39I12.33380. URL https://doi.org/10.1609/aaai.v39i12.33380.

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö. Arik. Astute RAG: overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 30553–30571. Association for Computational Linguistics, 2025a. URL https://aclanthology.org/2025.acl-long.1476/.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. Evo-memory: Benchmarking LLM agent test-time learning with self-evolving memory. *CoRR*, abs/2511.20857, 2025. doi: 10.48550/ARXIV.2511.20857. URL https://doi.org/10.48550/arXiv.2511.20857.

Cheng-Kuang Wu, Zhi Rui Tam, Chieh-Yen Lin, Yun-Nung Chen, and Hung-yi Lee. Streambench: Towards benchmarking continuous improvement of language agents. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/c189915371c4474fe9789be3728113fc-Abstract-Datasets_and_Benchmarks_Track.html.

Haoran Xu, Jiacong Hu, Ke Zhang, Lei Yu, Yuxin Tang, Xinyuan Song, Yiqun Duan, Lynn Ai, and Bill Shi. SEDM: scalable self-evolving distributed memory for agents. *CoRR*, abs/2509.09498, 2025a. doi: 10.48550/ARXIV.2509.09498. URL https://doi.org/10.48550/arXiv.2509.09498.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: agentic memory for LLM agents. *CoRR*, abs/2502.12110, 2025b. doi: 10.48550/ARXIV.2502.12110. URL https://doi.org/10.48550/arXiv.2502.12110.

Wei Yang, Jinwei Xiao, Hongming Zhang, Qingyang Zhang, Yanna Wang, and Bo Xu. Coarse-to-fine grounded memory for LLM agent planning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 13029–13056. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.659. URL https://doi.org/10.18653/v1/2025.emnlp-main.659.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *CoRR*, abs/2509.24704, 2025a. doi: 10.48550/ARXIV.2509.24704. URL https://doi.org/10.48550/arXiv.2509.24704.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems. *CoRR*, abs/2512.18746, 2025b. doi: 10.48550/ARXIV.2512.18746. URL https://doi.org/10.48550/arXiv.2512.18746.

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, 2026. URL https://arxiv.org/abs/2602.02474.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, Zhong-Zhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating LLM agents as lifelong learners. *CoRR*, abs/2505.11942, 2025. doi: 10.48550/ARXIV.2505.11942. URL https://doi.org/10.48550/arXiv.2505.11942.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 19724–19731. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

A OMAC CONTROLLER DETAILS

Knob space. Table 3 lists the complete configuration space.

Table 3: Memory operation knobs and their discrete values.

Operation	Knob	Values	$\lvert\cdot\rvert$
PROVIDE	Retriever	lexical, recency, hybrid	3
	Top-k	3, 5, 8	3
	Budget (tokens)	200, 400, 600	3
	Gating	on, off	2
TAKE-IN	Write trigger	always, success, failure	3
	Granularity	episode, insight	2
	Write budget	100, 200	2
	Dedup	on, off	2
MANAGE	Prune strategy	FIFO, LRU, score	3
	Threshold	0.7, 0.9, 1.0	3
	Consolidation	on, off	2

Feature vector. The feature vector $\phi(c) \in \mathbb{R}^{104}$ comprises: (1) one-hot encoding of each knob value ($3+3+3+2+3+2+2+2+3+3+2 = 28$ binary features, padded to 51 with cross-knob within-operation interactions); (2) pairwise cross-operation interactions (outer products of 2 key knobs per operation pair, $\sim$48 features); (3) memory-state features (store size bucket, fullness fraction, rolling accuracy, domain entropy, episodes since alarm; 5 features).

Hyperparameters. Block size $B = 10$. Sliding window $W = 50$ blocks. Prior variance $\sigma_0^2 = 1.0$. Noise variance $\sigma^2 = 0.5$. Warm-start blocks = 3. Shift alarm threshold $\tau = 2.0$ standard deviations. Sentinel configs $n_s = 3$. Reward weights: $\alpha = 1.0$, $\beta = 0.2$, $\gamma = 0.3$, $\delta = 0.1$.

B Additional Related Work

B.1 Memory management and self-evolution

The importance of active memory management has received growing attention. LifelongAgentBench demonstrates that naïve experience replay can degrade performance due to irrelevant content and context limits, proposing group self-consistency as a mitigation (Zheng et al., 2025). SEDM introduces a verifiable memory lifecycle with paired A/B replay to estimate marginal utility before memory admission, plus cross-domain knowledge diffusion that abstracts items into more general forms (Xu et al., 2025a). MemSkill treats memory operations as learnable skills that can be evolved and transferred under distribution shift (Zhang et al., 2026). Evo-Memory provides a streaming benchmark framework that evaluates memory evolution across ten diverse datasets, comparing multiple memory modules under a unified harness (Wei et al., 2025). Contextual Experience Replay accumulates and synthesizes past experiences into a dynamic buffer for retrieval-based self-improvement (Liu et al., 2025). MemGPT introduces hierarchical memory with explicit working-memory paging (Packer et al., 2023), and Mem0 emphasizes production-scale memory consolidation and retrieval (Chhikara et al., 2025). These works motivate the importance of MANAGE but study it as an integrated system component rather than as an independently controllable and adaptable operation. Our factorial design isolates MANAGE adaptation from PROVIDE and TAKE-IN to measure its independent contribution.

B.2 Other related work

Several additional lines of work are relevant. HippoRAG (Gutierrez et al., 2024) proposes neurobiologically inspired graph-structured memory for relational retrieval, representing a richer storage paradigm than our flat memory store. Coarse-to-Fine Grounded Memory (Yang et al., 2025) uses multi-granularity memory representations for agent planning, highlighting that different granularities serve different retrieval needs. MemWalker (Chen et al., 2023) and ReadAgent (Lee et al., 2024) explore structured memory navigation strategies that could complement our retrieval knobs. RAG (Lewis et al., 2020) provides the canonical retrieve-then-generate baseline underlying most agent memory systems, while Generative Agents (Park et al., 2023) introduced the influential reflection-based memory consolidation pattern that many subsequent systems build on. MemGen (Zhang et al., 2025a) proposes a fundamentally different approach through generative latent memory tokens, representing an alternative to our explicit memory store paradigm.

C Per-Run Results and Statistical Tests

This section provides the complete per-seed data underlying the aggregate results in Table 2, together with pairwise significance tests.

Variance and outliers. Figure 4 shows individual seed traces for all six conditions. Pre-shift variance differs sharply: NO-MEMORY has negligible seed-to-seed variation (std = 0.021) since performance depends only on the base model, while FROZEN-EVOLVED has the highest variation (std = 0.139, range 17.4%–68.8%) despite using a fixed configuration. This high variance arises because the frozen write policy accumulates whatever experiences the early task stream provides; if early tasks yield useful patterns the store helps, otherwise it adds noise. Online adaptation does not reduce this variance (MANAGE-ONLY: std = 0.120), suggesting that adaptation shifts the mean upward without tightening it. Two severe outliers — FROZEN-EVOLVED seed 9 (pre-shift 0.174) and ONLINE-JOINT seed 7 (pre-shift 0.226) — inflate confidence intervals substantially. We retain all seeds to reflect realistic deployment variance.

Phase decorrelation. Pre-shift and post-shift accuracy are moderately correlated for FROZEN-EVOLVED ($r = +0.60$): seeds that build a useful code store also tend to perform well on math, because store quality transfers. Under online adaptation, this correlation vanishes ($r \approx 0$), as Thompson sampling's stochastic policy decouples outcomes across phases.

Post-shift stability and AURC variance. Post-shift accuracy is remarkably stable: 55 of 60 runs exceed 93.0% on math, confirming that the ceiling is a property of the base model. AURC variance

is highest for NO-MEMORY (std = 0.039) and PROVIDE-ONLY (std = 0.031), and lowest for MANAGE-ONLY (std = 0.007) and TAKE-IN-ONLY (std = 0.008). Conditions that adapt store content produce the most consistent post-shift trajectories, suggesting that stabilizing the store also stabilizes recovery.

Pairwise significance. Table 5 reports bootstrap tests (10,000 resamples) for differences in pre-shift accuracy. The only significant comparison at $\alpha = 0.05$ is FROZEN-EVOLVED vs. NO-MEMORY ($\Delta = +0.144$, $p = 0.003$), confirming that memory provides a reliable benefit. No pairwise comparison between adaptive conditions reaches significance, though MANAGE-ONLY vs. TAKE-IN-ONLY approaches it ($p = 0.059$). Larger-scale experiments would be needed to definitively rank the adaptive conditions.

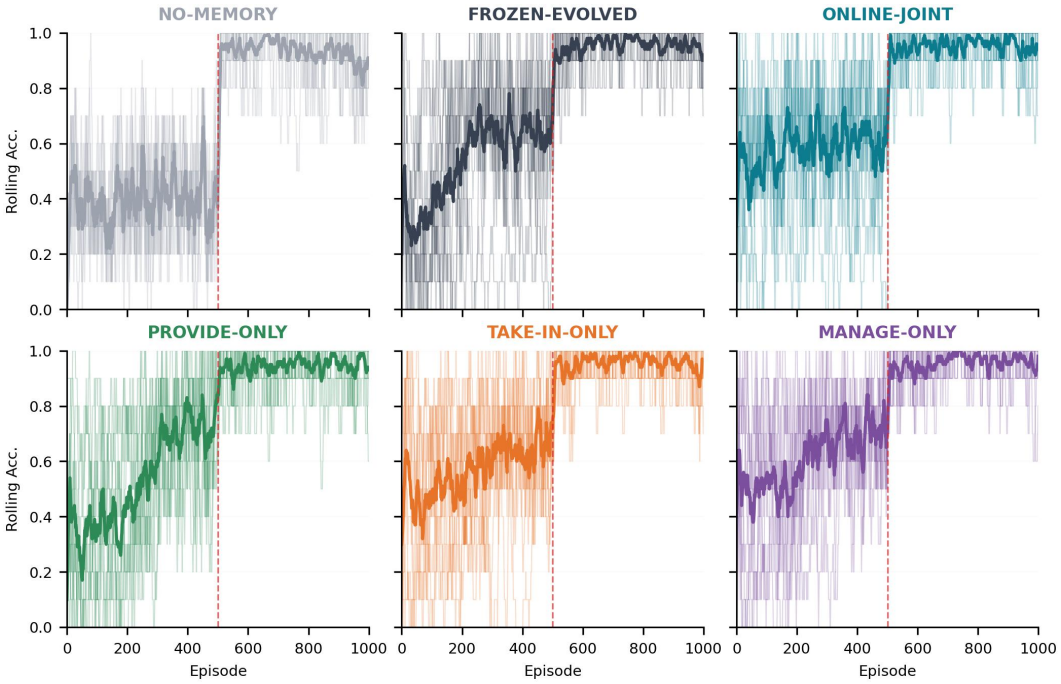

Figure 4: Per-seed recovery curves (rolling accuracy, window=10). Light lines: individual seeds; bold lines: seed-averaged mean. Dashed red line: domain shift.

D Store Size and Memory Turnover

Figure 5 shows memory store size over blocks for all memory-equipped conditions. All conditions reach the 200-item capacity well before the domain shift at block 50. Conditions with a frozen write-always policy (FROZEN-EVOLVED, PROVIDE-ONLY, MANAGE-ONLY) reach the cap deterministically at block 19. TAKE-IN-ONLY is slower (mean block 28.3, range 25–31), as its adaptive write trigger learns to be selective. This 9-block delay does not hurt performance (57.0% pre-shift vs. 54.0% for PROVIDE-ONLY), suggesting that writing fewer, higher-quality items is as effective as writing everything.

The domain shift has no visible effect on store size because stores are already saturated. Any post-shift improvement must therefore come from *replacing* existing items rather than adding new ones. The interference spike in Figure 6 confirms that this replacement takes approximately 10 blocks ($\sim$100 episodes): with 200 items and $\sim$10 new items written per block, even a partial infusion of math-domain items reduces stale-content interference before full turnover completes. This turnover dynamic operates identically across all conditions, explaining why post-shift convergence is universal.

Table 4: Full per-seed results across all 60 runs.

Condition	Seed	Pre-shift	Post-shift	Overall	AURC
NO-MEMORY	0	0.378	0.886	0.632	0.8788
	1	0.384	0.950	0.667	0.9452
	2	0.394	0.950	0.672	0.9478
	3	0.394	0.966	0.680	0.9648
	4	0.404	0.956	0.680	0.9494
	5	0.388	0.946	0.667	0.9402
	6	0.364	0.912	0.638	0.9050
	7	0.446	0.848	0.647	0.8438
	8	0.386	0.962	0.674	0.9562
	9	0.404	0.972	0.688	0.9664
FROZEN-EVOLVED	0	0.658	0.946	0.802	0.9456
	1	0.584	0.956	0.770	0.9554
	2	0.528	0.960	0.744	0.9564
	3	0.484	0.972	0.728	0.9688
	4	0.652	0.964	0.808	0.9584
	5	0.550	0.954	0.752	0.9558
	6	0.688	0.960	0.824	0.9586
	7	0.586	0.962	0.774	0.9618
	8	0.478	0.948	0.713	0.9484
	9	0.174	0.932	0.553	0.9284
ONLINE-JOINT	0	0.682	0.966	0.824	0.9626
	1	0.628	0.946	0.787	0.9446
	2	0.610	0.968	0.789	0.9642
	3	0.620	0.954	0.787	0.9490
	4	0.674	0.942	0.808	0.9402
	5	0.592	0.932	0.762	0.9304
	6	0.590	0.956	0.773	0.9546
	7	0.226	0.952	0.589	0.9446
	8	0.650	0.960	0.805	0.9606
	9	0.546	0.964	0.755	0.9624
PROVIDE-ONLY	0	0.660	0.950	0.805	0.9500
	1	0.492	0.952	0.722	0.9474
	2	0.668	0.962	0.815	0.9604
	3	0.530	0.852	0.691	0.8560
	4	0.678	0.960	0.819	0.9590
	5	0.582	0.960	0.771	0.9582
	6	0.498	0.966	0.732	0.9644
	7	0.456	0.956	0.706	0.9548
	8	0.418	0.948	0.683	0.9464
	9	0.416	0.966	0.691	0.9638
TAKE-IN-ONLY	0	0.606	0.956	0.781	0.9502
	1	0.592	0.938	0.765	0.9392
	2	0.494	0.948	0.721	0.9468
	3	0.338	0.950	0.644	0.9458
	4	0.430	0.968	0.699	0.9662
	5	0.640	0.962	0.801	0.9604
	6	0.688	0.958	0.823	0.9542
	7	0.640	0.954	0.797	0.9528
	8	0.682	0.962	0.822	0.9594
	9	0.586	0.960	0.773	0.9600
MANAGE-ONLY	0	0.712	0.966	0.839	0.9620
	1	0.468	0.962	0.715	0.9596
	2	0.648	0.950	0.799	0.9462
	3	0.362	0.954	0.658	0.9522
	4	0.474	0.962	0.718	0.9604
	5	0.676	0.942	0.809	0.9370
	6	0.728	0.960	0.844	0.9588
	7	0.712	0.956	0.834	0.9528
	8	0.638	0.958	0.798	0.9578
	9	0.660	0.956	0.808	0.9524

E CONTROLLER DYNAMICS AND REWARD DECOMPOSITION

Exploration behavior. ONLINE-JOINT maintains a near-constant exploration rate of 1.0 throughout the stream: the joint configuration space is large enough that the posterior never concentrates on a single configuration with only 10 episodes of feedback per block. Single-operation conditions show slightly lower rates (0.85–1.0). The domain shift produces no visible change in

Table 5: Pairwise bootstrap significance tests on pre-shift (code) accuracy. Δ = mean difference (A−B), p = one-sided p-value for A > B. ✓ = significant at $\alpha = 0.05$.

Condition A	Condition B	Δ	95% CI	p	Sig.
MANAGE-ONLY	ONLINE-JOINT	+0.026	[-0.092, +0.160]	0.350	—
MANAGE-ONLY	PROVIDE-ONLY	+0.068	[-0.035, +0.165]	0.098	—
MANAGE-ONLY	TAKE-IN-ONLY	+0.038	[-0.011, +0.081]	0.059	—
MANAGE-ONLY	FROZEN-EVOLVED	+0.070	[-0.036, +0.188]	0.109	—
ONLINE-JOINT	FROZEN-EVOLVED	+0.044	[-0.074, +0.151]	0.224	—
ONLINE-JOINT	PROVIDE-ONLY	+0.042	[-0.038, +0.113]	0.140	—
TAKE-IN-ONLY	PROVIDE-ONLY	+0.030	[-0.081, +0.137]	0.295	—
TAKE-IN-ONLY	FROZEN-EVOLVED	+0.031	[-0.066, +0.141]	0.292	—
PROVIDE-ONLY	FROZEN-EVOLVED	+0.002	[-0.069, +0.079]	0.480	—
FROZEN-EVOLVED	NO-MEMORY	+0.144	[+0.046, +0.224]	0.003	✓

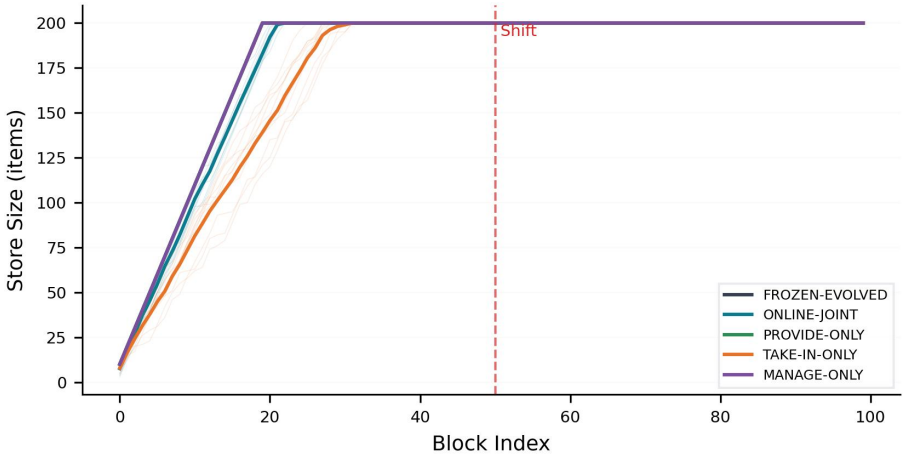

Figure 5: Store size evolution over blocks. All conditions reach the 200-item capacity before the shift. TAKE-IN-ONLY grows slowest due to selective write adaptation.

exploration, and the shift alarm triggered in only 3 of 40 adaptive runs (0 ONLINE-JOINT, 1 TAKE-IN-ONLY, 2 MANAGE-ONLY). The hard-to-easy shift causes a reward *jump* rather than a drop, so reward-based shift detection is ineffective. Detecting easy-target shifts would require monitoring feature-space statistics (e.g., domain entropy) rather than reward alone. We note that established change-point detection methods—CUSUM (Page, 1954), ADWIN (Bifet & Gavaldà, 2007), and Bayesian online change-point detection (Adams & MacKay, 2007)—could replace our heuristic; we opted for the lightweight exploration-rate check to avoid additional distributional assumptions, but incorporating principled CPD is a natural extension.

Reward component breakdown. Figure 6 decomposes the four reward components over blocks. The **task score** (top-left) mirrors the recovery curves; pre-shift, conditions are partially separable, but post-shift convergence compresses all to near-identical trajectories. **Retrieval utility** (top-right) remains high (0.6–1.0) throughout, though the consistently high values may indicate the utility proxy is too permissive. **Cost** (bottom-right) decreases modestly at the shift, reflecting shorter math responses.

The most interpretable pattern is in the **interference rate** (bottom-left): a universal spike to 1.0 immediately after the domain shift, decaying over ∼10 blocks to 0.0. This spike is identical across all five memory conditions, confirming it is a property of the domain shift rather than any adaptation strategy. When the domain changes, every retrieved item is a stale code-domain memory, producing an interference rate of 1.0 by construction. As the agent writes math experiences and old items are pruned, the store turns over and interference drops. This provides direct evidence for the delayed-effect hypothesis (Figure 1) and explains why all conditions converge post-shift: the same natural

turnover mechanism operates regardless of whether adaptation is enabled. The interference penalty ($\gamma = 0.3$) in the reward formulation ensures the controller receives a penalty signal during this transient; without it, the reward would not reflect the contamination of stale memories.

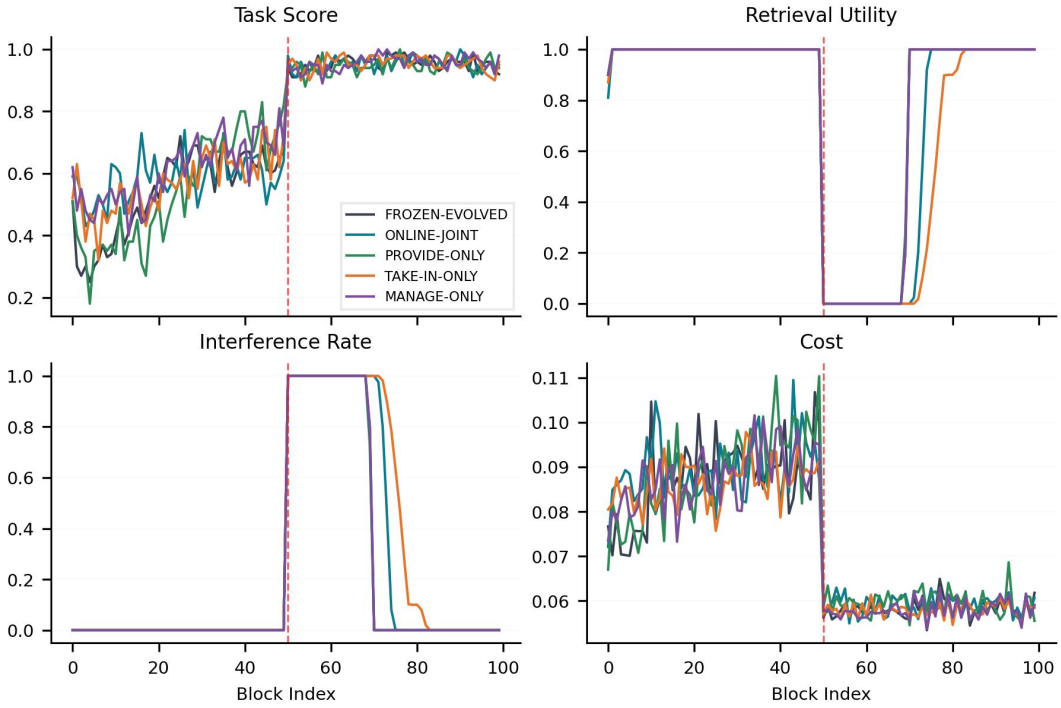

Figure 6: Reward component breakdown over blocks. Top-left: task score. Top-right: retrieval utility. Bottom-left: interference rate (note the spike at the shift). Bottom-right: normalized cost. Block-level means over 10 seeds.

F Sensitivity Considerations

Our experiments use a single block size ($B = 10$), sliding window ($W = 50$), and reward weight vector $(\alpha, \beta, \gamma, \delta) = (1.0, 0.2, 0.3, 0.1)$. We did not perform a full hyperparameter sweep due to computational constraints (60,000 API calls per full run). We note three considerations. **Block size**: smaller blocks provide faster feedback but noisier rewards; $B = 10$ is comparable to bandit-based RAG routing (Tang et al., 2025c). **Sliding window**: $W = 50$ retains $\sim$500 episodes of history; shorter windows track non-stationarity more aggressively at the cost of higher posterior variance. **Reward weights**: the interference penalty ($\gamma = 0.3$) exceeds the utility bonus ($\beta = 0.2$), penalizing harmful retrieval more than rewarding helpful retrieval; the interference spike at the shift (Figure 6) validates this choice. **Action selection**: we use Thompson sampling throughout; comparing against UCB-based selection (Garivier & Moulines, 2011) would test robustness to the exploration strategy but was not performed in this study.