PAGEDMEM: LEARNING TO REMEMBER AT THE RIGHT GRANULARITY FOR SELF-EVOLVING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite the increasing reliance of LLM agents on persistent memory, whether past experience improves or degrades their performance depends on the granularity at which it is remembered. Memory that is over-broad provides no actionable guidance; memory that is over-strict constrains or even misleads the agent when the current task is inconsistent with it. Although injected text is formally soft guidance, an over-strict procedure acts as a hard behavioral prior: it suppresses exploration and triggers negative transfer when the current environment diverges from the source trajectory. Memory injection can therefore degrade rather than improve performance when the stored experience is more specific than the current execution supports, even if the memory content itself is correct. We introduce PAGEDMEM, a self-evolving memory system that treats representation granularity and exposure granularity as separable design variables. Unlike prior methods that fix the granularity of memory at construction and inject retrieved content wholesale, PAGEDMEM keeps multiple levels of abstraction available and progressively exposes the level that execution currently requires. It consolidates free-form reflections into semantic pages with three levels: a short header, an open-ended summary, and provenance-linked supporting records. We conduct extensive experiments on ALFWorld with three model backbones. On every backbone, PAGEDMEM improves success and efficiency over ReAct, AWM, and MemP and substantially reduces invalid actions. These results show that representing and using memory at the right granularity can further improve agents' ability to adapt and generalize.

1 INTRODUCTION

Long-horizon agents repeatedly encounter tasks that share partial structure. A prior trajectory may reveal how to recover from a failed action (Shinn et al., 2023), which lessons transfer across tasks (Zhao et al., 2024), or how to avoid a recurring trap (Wang et al., 2025; Fang et al., 2026b). Reusing such experience can improve both task success and execution efficiency, motivating a growing line of work on persistent memory for language-model agents. Existing systems store and retrieve episodic observations (Park et al., 2023), manage context through paging (Packer et al., 2023), maintain long-term user memory (Zhong et al., 2024), and distill workflows, procedures, or reasoning-oriented items from past trajectories (Wang et al., 2025; Fang et al., 2026b; Ouyang et al., 2026).

Reusing experience, however, raises a design question that is often hidden behind the retrieval interface: *at what granularity should experience be represented and exposed?* A memory that is too broad provides only vague reminders, while a memory that is too specific can bind the agent to the objects, states, and action order of an earlier trajectory. Although injected text is formally soft guidance, a detailed procedure can act as a hard behavioral prior, suppressing exploration and causing negative transfer when the current environment differs from the source trajectory.

Existing memory systems often store raw trajectories, successful routines, or procedural workflows, then retrieve and inject them as complete items. This flat use couples representation granularity, how experience is organized in storage, with exposure granularity, how much is shown at execution time; detailed evidence cannot remain available without being injected. It also encourages premature anchoring to a historical action sequence before the environment confirms that it applies, and it lacks a usage-driven lifecycle in which downstream outcomes reshape the memory store.

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.

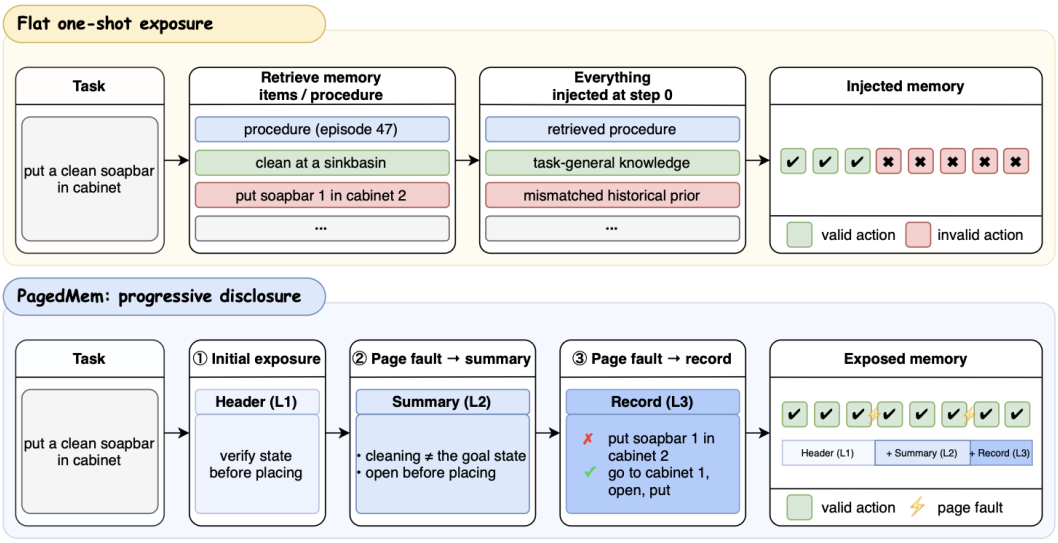

Figure 1: Flat one-shot exposure injects the full retrieved procedure at step 0, including a historical prior that does not match the current task, and surfaces invalid actions; PAGEDMEM progressively exposes an L1 header, L2 summary, and L3 record as execution triggers page faults.

PagedMem is built on a simple insight: representation granularity and exposure granularity should be treated as separate design choices. It consolidates free-form reflections into semantic pages with a short header, an open-ended summary, and provenance-linked records. At episode start, only a page header is exposed. Execution-triggered page faults then expand the page or replace it when the current guidance proves insufficient, and downstream utility updates exposed pages across episodes. The policy remains frozen, so adaptation is confined to the external memory store and the memory system evolves during test-time operation.

We evaluate PAGEDMEM on ALFWorld with three backbones: Qwen3.5-35B-A3B, DeepSeek-V4-Flash, and GPT-5.6-Luna. It attains 84.3% to 85.0% success on Seen and 90.3% to 92.5% on Unseen, outperforming ReAct, AWM, and MemP on both splits for every backbone, and it substantially reduces invalid actions relative to MemP. Page faults occur frequently enough to show that progressive disclosure is actively used, and granularity ablations show that removing either multi-granularity pages or progressive disclosure lowers success and increases invalid actions.

Our contributions are threefold: (1) We identify a granularity mismatch in agent memory: overly broad memory provides no actionable guidance, while overly specific memory can act as a hard behavioral prior; we frame memory use as the joint problem of representation granularity and exposure granularity. (2) We propose PAGEDMEM, a self-evolving memory system that consolidates reflections into multi-level semantic pages, exposes them coarse-to-fine through execution-triggered page faults, and updates them from downstream utility. (3) On ALFWorld with three model backbones, PAGEDMEM improves success over ReAct, AWM, and MemP and substantially reduces invalid actions; granularity ablations show that removing either multi-granularity pages or progressive disclosure degrades performance.

2 PRELIMINARY

LLM Agents with Memory. We consider an agent interacting with an environment over an episode. Given a task description q, at each step t the agent observes history $h_t = (a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ and selects an action a_t with a frozen language-model policy π_θ; the environment returns observation o_t. The episode ends upon task completion or after a budget of B steps, yielding outcome $y \in \{0, 1\}$. A memory system augments the policy with an external store $\mathcal{M}$ and an *exposure function* E that selects and formats memory content:

$$a_t \sim \pi_\theta\big(\cdot \mid q, h_t, m_t\big), \qquad m_t = E(\mathcal{M}, q, h_t). \tag{1}$$

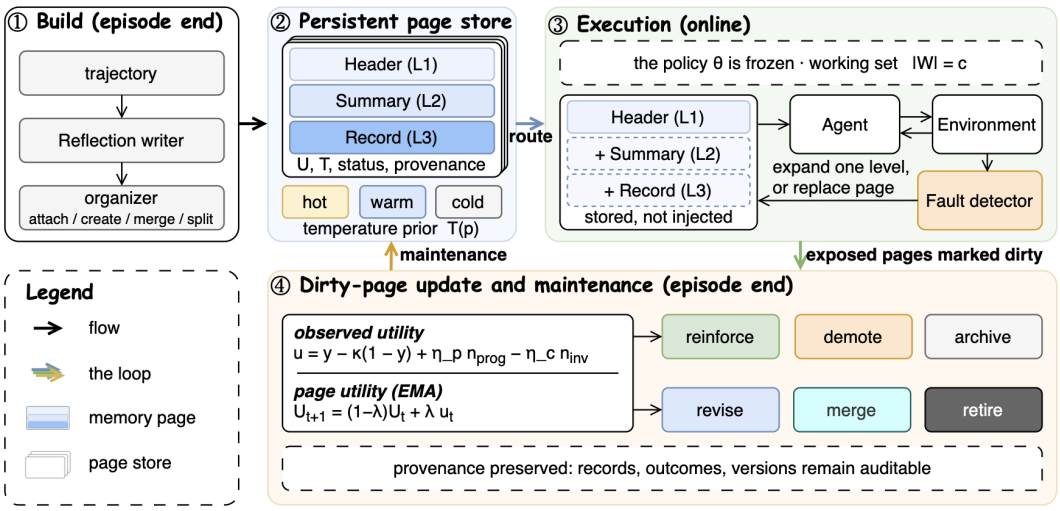

Figure 2: Overview of PAGEDMEM. The reflection writer and organizer build an evolving page store; execution routes one page into a bounded working set, exposes it through the agent, and triggers expansion or replacement on faults; dirty pages are then updated and maintained from downstream utility.

The store $\mathcal{M}$ may itself change between and within episodes. In this formulation, the policy parameters θ never change; all adaptation occurs through $\mathcal{M}$ and E.

The flat exposure paradigm. Most memory systems instantiate E as one-shot retrieval,

$$m_q \;=\; \mathrm{TopK}_{\mathrm{sim}}(\mathcal{M}, q), \qquad m_t = m_q \quad \forall\, t \in [1, B], \tag{2}$$

decided once at the query and fixed for the episode. Under this paradigm the retrieval unit is the exposure unit, which causes three problems: representation and exposure granularity are conflated; procedure-like content can prematurely anchor the agent, and lengthy injections can dilute relevant context (Liu et al., 2024); and downstream utility does not reshape memory organization, so stale and useful items compete equally. This raises our research question: *at what granularity should experience be represented and exposed?*

Two granularity axes. We distinguish two design axes of a memory system. The *representation granularity* is the level of abstraction at which an item is stored in $\mathcal{M}$. The *exposure granularity* is the level of detail of the injected content m_t at step t. Under the flat paradigm of Eq. 2 the two coincide, because the retrieved item is injected in full; PagedMem decouples them, so that detailed evidence can remain available without being injected up front.

3 PAGEDMEM

PagedMem instantiates the exposure function E as a paging lifecycle over fine-grained REFLECTIONRECORDs and aggregated MEMORYPAGEs. It separates representation granularity from exposure granularity through four components: multi-level pages, a bounded working set, execution-triggered page faults, and utility-driven maintenance. The policy parameters remain frozen; all adaptation occurs through the memory store $\mathcal{M}$ and the exposure function E (Eq. 1).

The core algorithmic idea of PagedMem is to decouple memory representation from memory exposure. Representation granularity determines how concretely experience is retained, while exposure granularity determines how much historical detail enters the prompt at step t. PagedMem keeps both coarse and fine evidence in the same page, but exposes only the level required by the current execution state, so specificity is available on demand rather than imposed by default.

3.1 MEMORY REPRESENTATION

After episode e, a reflection writer produces a free-form reflection describing useful strategies, failure modes, recovery behavior, and relevant environmental checks. The reflection is stored as a REFLECTIONRECORD with provenance rather than as a fixed action script:

$$r = (\texttt{reflection_text}, \texttt{task_description}, \texttt{trajectory_id}, \texttt{outcome}, \texttt{created_at}, \texttt{page_id}, \texttt{exposure_count}).$$

We write r_e for the record produced by episode e; records retain their source trajectory and outcome.

A MEMORYPAGE aggregates semantically related records. It couples multi-level content with bookkeeping attributes,

$$p \;=\; (\text{hdr}_p,\ \text{sum}_p,\ \mathcal{R}_p), \qquad \text{with attributes } (U(p),\ T(p),\ \sigma(p)), \tag{3}$$

where hdr_p is a short header, sum_p an open-ended summary, $\mathcal{R}_p$ a set of provenance-linked supporting records, $U(p)$ the observed utility, $T(p)$ a temperature prior, and $\sigma(p)$ the lifecycle status. A page is a current interpretation of its evidence rather than an immutable truth, so page formation is incremental: a new record can be attached, a new page created, redundant pages merged, and a page with stable internal conflicts split. The implementation uses lightweight similarity to propose candidates and an LLM organizer to choose the operation and refresh page text.

Each page exposes three levels: a short header, an open-ended summary, and a provenance-linked supporting record. The exposed content at level i is

$$\ell_i(p) \;=\; \begin{cases} \text{hdr}_p, & i = 1 \ \ (\text{L1}), \\ \text{hdr}_p \oplus \text{sum}_p, & i = 2 \ \ (\text{L2}), \\ \ell_2(p) \oplus r_p, & i = 3 \ \ (\text{L3}), \end{cases} \tag{4}$$

where $r_p \in \mathcal{R}_p$ is the currently exposed supporting record and $\oplus$ denotes text concatenation. The levels form an ordered set $\text{L1} \prec \text{L2} \prec \text{L3}$ of increasing specificity, keeping detail one fault away rather than either fully injected or absent.

The levels make the abstraction-specificity trade-off explicit. L1 provides a low-commitment reminder, L2 explains the context and checks behind it, and L3 grounds the advice in a concrete success or failure. This ordered traversal lets the agent stop at an abstract level when it can adapt on its own, and descend to evidence only when a concrete case is needed. For example, a page may advise verifying state-changing operations before placement; its summary explains why operation issuance does not imply state change, while supporting records preserve the episodes behind that lesson.

3.2 WORKING SET AND PROGRESSIVE EXPOSURE

Let $\mathcal{P} \subseteq \mathcal{M}$ be the page set and q the current task. PagedMem maintains a bounded working set $W \subseteq \mathcal{P}$ with $|W| \le c$; this paper uses $c = 1$, so W contains a single resident page. Pages are scored by

$$S(p, q) = \text{sim}(p, q) + \alpha\, U(p) + \beta\, T(p) - \gamma\ \text{red}(p, W), \tag{5}$$

where sim is lightweight similarity, $U(p)$ observed utility, $T(p)$ a temperature prior, and $\text{red}(p, W)$ a redundancy penalty. Starting from $W = \emptyset$, pages are added greedily until $|W| = c$:

$$W \;\leftarrow\; W \cup \left\{ \arg\max_{p \in \mathcal{P} \setminus W}\ S(p, q) \right\}. \tag{6}$$

Routing is therefore candidate generation rather than the final exposure decision. It narrows the store to a small working set, while execution decides which level of that page becomes visible. With $c = 1$, the working set acts as an interference-control mechanism rather than a claim about the optimal memory capacity; the score favors task-relevant, useful, and warm pages, and can be extended to larger c by the same greedy rule.

At step t, the resident page p_t begins at L1 and exposes

$$m_t \;=\; \ell_{k_t}(p_t), \tag{7}$$

where $k_t \in \{1, 2, 3\}$ is the current level. A fault fires when an online execution-deficit score exceeds a threshold:

$$F(h_t) \triangleq (g(h_t) \geq \tau), \tag{8}$$

where $g(h_t)$ aggregates observable signals such as repeated actions, invalid actions, stalled progress, and observation-plan mismatch. These signals instantiate the detector rather than define an exhaustive trigger set.

These triggers are online proxies for a granularity deficit: execution indicates that the current level of guidance is too coarse, too specific, or no longer applicable. The detector never reads benchmark answers, hidden evaluator labels, or future observations.

When a fault fires, the resident state follows

$$(p_{t+1}, k_{t+1}) = \mathrm{Transit}(p_t, k_t, h_{t+1}), \tag{9}$$

where Transit advances the exposure level via $\varphi(k) = \min(k + 1, 3)$ when possible; if the current page is exhausted or mismatched, it pages in a recovery page and resets exposure to L1. With no fault, the state is unchanged. Algorithm 1 expands this transition. A cooldown window prevents churn. Memory content is advisory throughout, so the agent may ignore or abandon it.

3.3 Dirty-Page Updates and Maintenance

Every page exposed during an episode is marked dirty. The system records the highest level reached, subsequent actions, observed progress, invalid actions, and final outcome. These signals define the observed utility of page p after episode e:

$$u_e(p) = u_{\mathrm{outcome}} + u_{\mathrm{progress}} - u_{\mathrm{cost}}, \tag{10}$$

with

$$u_{\mathrm{outcome}} = y - \kappa\,(1 - y), \qquad u_{\mathrm{progress}} = \min(\bar{u},\, \eta_p\, n_{\mathrm{progress}}), \qquad u_{\mathrm{cost}} = \eta_c\, n_{\mathrm{invalid}}, \tag{11}$$

where y indicates task success and $n_{\mathrm{progress}}, n_{\mathrm{invalid}}$ count progress and invalid actions. Utility is aggregated by an exponential moving average with coefficient λ:

$$U_{e+1}(p) = (1 - \lambda)\, U_e(p) + \lambda\, u_e(p). \tag{12}$$

All coefficients are fixed protocol parameters (Table 2 in Appendix A); each term is bounded for the fixed interaction budget.

After episode e, the new record r_e is added and the dirty pages $\mathcal{D}_e \subseteq \mathcal{P}$ are maintained:

$$\mathcal{M}_{e+1} = \mathrm{Maintain}(\mathcal{M}_e,\, \mathcal{D}_e,\, r_e). \tag{13}$$

Maintenance makes granularity adaptive across episodes. A page that yields progress and success is reinforced; one whose summary is outdated is revised; one that repeatedly fails to help is demoted, archived, or retired. This keeps the active store compact while preserving provenance, so the system can change both what it remembers and at what level it exposes it. Maintenance reinforces, revises, demotes, merges, archives, or retires pages according to observed utility; archived and retired pages leave normal routing but retain their records and provenance. Algorithm 1 summarizes the complete loop, and the policy parameters remain frozen throughout.

Overall, PagedMem treats memory exposure as a granularity-control problem rather than a one-shot retrieval problem. The same evidence can remain stored at multiple levels, but only the level that execution currently requires is injected; when that level proves insufficient or inappropriate, the system escalates, replaces, or revises it.

4 Experiments

4.1 Setup

Protocol. The evaluation uses ALFWorld (Shridhar et al., 2021) with frozen Seen140 and Unseen134 task manifests. All methods use the same task streams, policy backbone, action format, and 30-step interaction budget. Under this aligned protocol, we reproduce and compare PagedMem with ReAct, AWM, and MemP.

Table 1: ALFWorld results across three backbones under the shared 30-step protocol, with success rates, steps, and invalid actions per split; bold marks the best result in each backbone block.

Method	Success			Efficiency (Seen / Unseen)		
	Seen	Unseen	Combined	Mean steps	Med. steps	Invalid
Qwen3.5-35B-A3B						
ReAct	98/140 (70.00)	104/134 (77.61)	202/274 (73.72)	17.96 / 16.67	17 / 15.5	143 / 113
AWM	101/140 (72.14)	101/134 (75.37)	202/274 (73.72)	18.67 / 18.28	17.5 / 18	147 / 155
MemP	109/140 (77.86)	116/134 (86.57)	225/274 (82.12)	15.20 / 14.43	11 / 11.5	67 / 46
PagedMem	**118/140 (84.29)**	**124/134 (92.54)**	**242/274 (88.32)**	**14.68 / 13.11**	**11 / 10**	**11 / 4**
DeepSeek-v4-Flash						
ReAct	94/140 (67.14)	102/134 (76.12)	196/274 (71.53)	18.63 / 18.04	19 / 17.5	113 / 133
AWM	106/140 (75.71)	111/134 (82.84)	217/274 (79.20)	17.83 / 17.42	17.5 / 17	97 / 112
MemP	110/140 (78.57)	119/134 (88.81)	229/274 (83.58)	14.64 / 14.02	11.5 / 10.5	82 / 90
PagedMem	**119/140 (85.00)**	**121/134 (90.30)**	**240/274 (87.59)**	**13.00 / 13.75**	**9 / 10.5**	**9 / 6**
GPT-5.6-Luna						
ReAct	80/140 (57.14)	87/134 (64.93)	167/274 (60.95)	20.18 / 19.81	23.5 / 22.5	193 / 165
AWM	90/140 (64.29)	111/134 (82.84)	201/274 (73.36)	18.87 / 16.34	19.5 / 15	53 / 43
MemP	82/140 (58.57)	93/134 (69.40)	175/274 (63.87)	20.35 / 17.59	25.5 / 14	422 / 282
PagedMem	**119/140 (85.00)**	**122/134 (91.04)**	**241/274 (87.96)**	**16.49 / 15.76**	**15.5 / 14**	**12 / 5**

Baselines. We compare against ReAct (Yao et al., 2023), AWM (Wang et al., 2025), and MemP (Fang et al., 2026b). We evaluate all methods on three backbones: Qwen3.5-35B-A3B, DeepSeek-V4-Flash, and GPT-5.6-Luna.

Metrics and reporting. We report results across these three backbones, including success counts and rates, mean and median steps, and total invalid actions per split, plus an operational accounting of memory exposure, updates, and errors.

4.2 Main Results

Table 1 reports the completed comparison. On Qwen3.5-35B-A3B, PagedMem obtains the highest success on both splits: it improves over ReAct by 14.29 points on Seen and 14.93 points on Unseen, and over MemP by 6.43 and 5.97 points. Its combined success is 242/274 (88.32%), versus 225/274 (82.12%) for MemP and 202/274 (73.72%) for both ReAct and AWM. Across the three backbones, PagedMem attains the highest success on both splits. AWM improves over ReAct on Seen (+2.14) but underperforms on Unseen (-2.24), indicating that its gains are not uniform across splits. PagedMem also uses the fewest mean and median steps on both splits, reducing mean steps relative to MemP by 0.52 and 1.32. Its advantage is therefore not accuracy-only.

4.3 Ablation Study

The central claim of this work is that representation granularity and exposure granularity must be decoupled. To isolate the contribution of each axis, we conduct two ablations on Qwen3.5-35B-A3B over the same ALFWorld protocol: one removes multi-level representation, and the other removes progressive exposure. All other components are held fixed (Figure 3).

Representation granularity. To ablate representation granularity, we replace each multi-level semantic page with a flat bank of raw reflection records. All records remain available, but their hierarchical organization is gone: every record is injected directly, so the system has only a single level of memory representation to expose. Success drops from 84.3% to 80.0% on Seen and from 92.5% to 87.3% on Unseen, and invalid actions rise from 11/4 to 57/40. The degradation concentrates in execution behavior rather than task success: without the multi-granularity page, procedure-like detail arrives as an undifferentiated block, and this over-specific guidance surfaces as invalid actions. This pattern matches the signature predicted by the granularity-mismatch account.

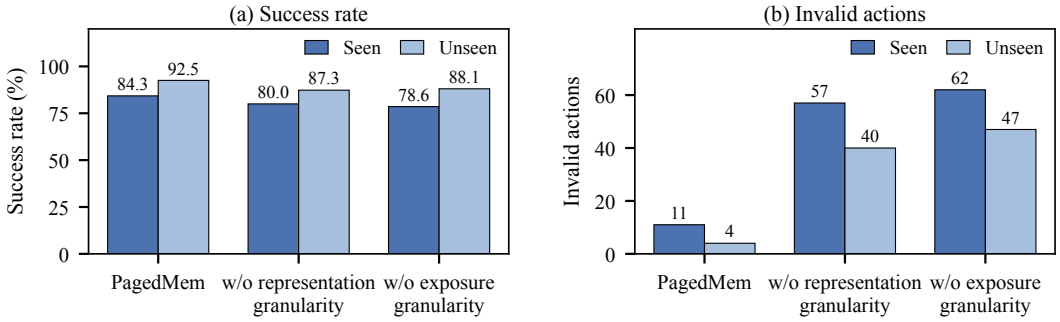

Figure 3: Granularity ablations on Qwen3.5-35B-A3B: removing multi-level representation or progressive exposure reduces success and increases invalid actions.

Exposure granularity. To ablate exposure granularity, we keep the multi-level page representation unchanged but disable progressive disclosure: the full page is injected at episode start. Exposure therefore becomes one-shot while representation stays multi-granular. Success falls to 78.6% on Seen and 88.1% on Unseen, and invalid actions rise further, to 62/47, the highest among the three PagedMem variants in Figure 3. Only the timing of exposure changes. A purely exposure-level ablation both lowers success and multiplies invalid actions several-fold; on Seen it even degrades success below the flat representation. This confirms that the behavioral gain comes from the execution-conditioned, coarse-to-fine page-in rather than from the page structure alone, and that coupling the two axes costs more than dropping the page altogether. Together, the two ablations show that both granularity axes carry independent, load-bearing components of the improvement.

4.4 Case Studies

To examine how the granularity of agent memory affects behavior within an episode, we compare two representative cases from the DeepSeek-V4-Flash runs. In both cases, PagedMem solves the episode while ReAct, AWM, and MemP fail; the first case isolates execution-triggered exposure adjustment, and the second shows how task-level guidance avoids an over-specific search prior.

Case 1: Fault-triggered progressive disclosure

Task: *put a hot cup in cabinet.*

PagedMem

Starts at L1. A `task_entity_mismatch` fault at step 4 expands the page to L2. The agent takes `cup 1` instead of the mug, heats it, and places it. Success in 20 steps, 0 invalid actions. ✓

Baselines

× ReAct: 30-step failure, 2 invalid actions.
× AWM: 30-step failure, 7 invalid actions.
× MemP: 30-step failure, 6 invalid actions.

Case 2: Task-level guidance avoids over-specific search

Task: *put a clean ladle in diningtable.*

PagedMem

The exposed L1 guidance keeps the agent on the visible diningtable instead of an exhaustive container search. The agent takes the ladle, cleans it at the sinkbasin, returns, and places it. Success in 6 steps, 0 invalid actions. ✓

Baselines

× ReAct: 30-step failure.
× AWM: 30-step failure.
× MemP: 30-step failure.

All three open cabinets and drawers without locating the ladle.

These two cases expose complementary granularity failures. In the hot-cup case, the baselines do not fail for lack of memory content; they fail because their guidance is not revised at the level required when execution detects an entity mismatch. PagedMem's L1-to-L2 transition supplies that missing specificity at the moment of failure, and the episode completes without invalid actions.

In the ladle case, the mismatch is in the opposite direction: the baselines spend the budget on a low-level container search, whereas PagedMem's abstract L1 guidance is sufficient to start from the visible tabletop and complete the task in six steps. The contrast therefore concerns representation and exposure granularity rather than memory quantity alone.

The two cases also mirror the aggregate pattern through different behavioral signatures. The hot-cup case shows the invalid-action reduction: PagedMem records zero invalid actions, whereas ReAct, AWM, and MemP accumulate 2, 7, and 6 before exhausting the budget. The ladle case shows the efficiency gain: PagedMem needs 6 steps, while all three baselines need 30. Together, the traces connect the measured success and efficiency improvements to the same mechanism: abstract guidance is exposed first, and more specific or different guidance is brought in only when execution shows that the current level is insufficient. These traces are qualitative and do not establish causality on their own, but they exhibit the behavioral signature predicted by the granularity account.

5 Analysis

To understand how the granularity of agent memory shapes downstream behavior, we analyze the results along three dimensions: negative transfer caused by granularity mismatch, progressive multi-granularity exposure, and the contribution of the two granularity axes.

5.1 Granularity mismatch induces negative transfer.

Across the three backbones, adding memory does not uniformly improve over the memory-free ReAct baseline. On Qwen3.5-35B-A3B, AWM reduces Unseen success from 104/134 (77.6%) to 101/134 (75.4%). The error behavior is more revealing. On GPT-5.6-Luna, MemP produces 422/282 invalid actions, compared with 193/165 for ReAct and 12/5 for PagedMem. On DeepSeek-V4-Flash, MemP produces 82/90 invalid actions versus 9/6 for PagedMem. This pattern is consistent with granularity mismatch: when injected memory is more specific than the current execution supports, it can act as an overly strong prior and drive the policy toward erroneous actions. PagedMem avoids this failure mode and attains the lowest invalid-action count on every backbone.

5.2 Progressive multi-granularity exposure adapts memory to execution.

PagedMem does not merely retain multiple levels of memory; it controls when each level becomes visible. Across the Seen and Unseen runs, it triggers 160 and 131 page faults, showing that execution-triggered exposure is actively exercised. The hot-cup case isolates this mechanism: at step 4, a task-entity mismatch fault expands the resident page from L1 to L2. The agent abandons the wrong object, finds the cup, and completes the task in 20 steps with zero invalid actions, whereas ReAct, AWM, and MemP exhaust the 30-step budget with 2, 7, and 6 invalid actions. The one-shot exposure ablation in Section 4.3 provides complementary evidence: with the same multi-level pages but no progressive exposure, success falls from 118/140 and 124/134 to 110/140 and 118/134, and invalid actions rise from 11/4 to 62/47. The gain therefore comes from execution-conditioned exposure rather than from page structure alone. Appendix E further shows that this efficiency advantage persists across the evaluation stream, rather than being driven by a small subset of episodes.

5.3 Representation and exposure granularity are both necessary.

The granularity ablations in Section 4.3 remove one granularity axis at a time. Flattening the representation while keeping all records lowers success from 84.3% to 80.0% on Seen and from 92.5% to 87.3% on Unseen, and increases invalid actions from 11/4 to 57/40. Disabling progressive exposure while retaining multi-level pages lowers success to 78.6% on Seen and 88.1% on Unseen, and raises invalid actions to 62/47. Neither axis alone preserves the full behavior; on Seen, one-shot exposure even falls below the flat-representation setting. The case studies mirror these two failure directions: the hot-cup case requires finer detail after an execution mismatch, whereas the ladle case shows that abstract guidance avoids an over-specific search procedure. Across all three backbones, the complete system achieves the best success and the lowest invalid-action count on both splits, and it also uses the fewest mean steps. We therefore conclude that the benefit of PagedMem comes from controlling both representation and exposure granularity, not from storing more memory.

6 Related Work

Agent Memory and retrieval architectures. Memory has become a standard component of language-model agents, allowing them to persist information and reuse past interactions (Zhang et al., 2024; Du et al., 2025; Hu et al., 2025b; Huang et al., 2026). Existing systems organize memory in various forms, including episodic observations for simulation (Park et al., 2023) and human-inspired episodic memory for long-context models (Fountas et al., 2025), operating-system-style context paging (Packer et al., 2023), hierarchical working memory (Hu et al., 2025a), long-term user memory with forgetting (Zhong et al., 2024), evolving note networks (Xu et al., 2025), production-scale conversational memory (Chhikara et al., 2025; Kang et al., 2025), neurobiologically inspired graph memory (Jiménez Gutiérrez et al., 2024), and staged sensory-to-long-term memory (Fang et al., 2026a). Retrieval-augmented generation conditions generation on retrieved context (Lewis et al., 2020), while long-context studies show that injected content can dilute or bury relevant evidence (Liu et al., 2024). These works primarily advance how memory is stored, indexed, and retrieved. PagedMem asks a complementary question: at what granularity should stored experience be exposed during execution, and how should exposure outcomes reshape the store? Across these designs, the exposed unit is typically the retrieved item itself, even when memory is organized at multiple granularity levels. PagedMem instead makes exposure granularity an explicit decision variable: a page can remain stored at multiple granularities while execution controls which granularity enters the agent context.

Agent Memory for self-evolving agents. A related line of work distills reusable experience from past trajectories. Reflexion converts episode feedback into verbal self-reflections in an episodic buffer (Shinn et al., 2023); ExpeL extracts and retrieves cross-task lessons (Zhao et al., 2024); AutoGuide generates context-aware guidelines from prior trajectories (Fu et al., 2024); Voyager grows a skill library in an open-ended world (Wang et al., 2023); Synapse retrieves trajectory exemplars for computer control (Zheng et al., 2024); ReasoningBank induces reasoning-oriented memory from successes and failures (Ouyang et al., 2026); and Titans updates a neural memory at test time (Behrouz et al., 2025). Contextual experience replay and cross-domain memory reuse extend this idea across tasks (Liu et al., 2025; Tang et al., 2025; Qian et al., 2024), while reflective memory management studies when stored experience should be revised or forgotten (Tan et al., 2025; Suzgun et al., 2025). Recent work further explores self-evolving agents that couple reflection with memory for autonomous improvement (Liang et al., 2024; Sun et al., 2025). Workflow- and procedure-based methods, including AWM (Wang et al., 2025) and MemP (Fang et al., 2026b), form a more structured branch of this line. These methods show that memory can drive self-evolution, but they typically fix the abstraction at construction time and expose retrieved items at a single level. PagedMem instead begins with an abstract page header, expands to summaries or supporting records only after execution-triggered faults, and updates pages according to downstream utility. Its self-evolution therefore emerges from the granularity of exposure and maintenance, rather than from changing policy parameters.

7 Conclusion

We introduce PagedMem, a self-evolving memory system that treats agent memory as a problem of representation and exposure granularity. It consolidates free-form reflections into multi-level semantic pages, keeps a compact task-conditioned working set, and progressively exposes each page when execution signals that the current level of guidance is insufficient. Downstream utility then revises and maintains the exposed pages, allowing memory to evolve during test-time operation while the policy parameters remain fixed. Extensive experiments on ALFWorld across three model backbones show that PagedMem improves success and efficiency while substantially reducing invalid actions relative to ReAct, AWM, and MemP. Further analyses show that both granularity axes are necessary: removing multi-level representation or progressive exposure degrades success and increases invalid actions, and the case studies illustrate how fault-triggered exposure corrects an entity mismatch while abstract guidance avoids over-specific search. Our findings suggest that controlling both representation and exposure granularity offers a practical route toward adaptive, self-evolving agents in long-horizon settings.

AI USE STATEMENT

Generative AI tools were used solely to improve the grammar, clarity, and readability of text written by the authors. All AI-suggested edits were reviewed and approved by the authors to ensure that they did not alter the technical meaning of the paper. The authors take full responsibility for the final content of this work.

REFERENCES

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems 38*, pp. 125925–125962, 2025.

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

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z. Pan. Rethinking memory in llm based agents: Representations, operations, and emerging topics. 2025.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. Lightmem: Lightweight and efficient memory-augmented generation. In *The Fourteenth International Conference on Learning Representations*, 2026a.

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, 2026b.

Zafeirios Fountas, Martin Benfeghoul, Adnan Oomerjee, Fenia Christopoulou, Gerasimos Lampouras, Haitham Bou Ammar, and Jun Wang. Human-inspired episodic memory for infinite context llms. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems 37*, pp. 119919–119948, 2024.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, 2025a.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents. 2025b.

Wei-Chieh Huang, Weizhi Zhang, Yueqing Liang, Yuanchen Bei, Yankai Chen, Tao Feng, Xinyu Pan, Zhen Tan, Yu Wang, Tianxin Wei, et al. A survey of agent memory in the second half: Towards self-evolving and long-horizon agents. *Transactions on Machine Learning Research*, 2026. Survey Certification.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, 2024.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent. 2025.

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 *Advances in Neural Information Processing Systems*, 2020.

Xuechen Liang, Yangfan He, Yinghui Xia, Xinyuan Song, Jianhui Wang, Meiling Tao, Li Sun, Xinhang Yuan, Jiayi Su, Keqin Li, et al. Self-evolving agents with reflective and memory-augmented abilities. 2024.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 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.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. 2023.

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.

Cheng Qian, Shihao Liang, Yujia Qin, Yining Ye, Xin Cong, Yankai Lin, Yesai Wu, Zhiyuan Liu, and Maosong Sun. Investigate-consolidate-exploit: A general strategy for inter-task agent self-evolution. 2024.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *The Ninth International Conference on Learning Representations*, 2021.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. Seagent: Self-evolving computer use agent with autonomous learning from experience. 2025.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. 2025.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8416–8439, 2025.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. 2025.

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. *Transactions on Machine Learning Research*, 2023.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems 38*, pp. 20004–20031, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. 2024.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, 2024.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*, 2024.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, 2024.

A Algorithm and Protocol Parameters

To better illustrate the online execution of PagedMem, Algorithm 1 instantiates the loop described in Section 3: it builds a task-conditioned working set (Eq. 5), exposes the resident page at increasing levels (Eq. 4), detects execution-triggered page faults (Eq. 8), and updates every exposed page from its observed utility (Eq. 10). Table 2 lists the protocol parameters.

Algorithm 1 PagedMem algorithm

Require: external memory store $\mathcal{M}$ with page set $\mathcal{P}$; task description q; working-set capacity c; exposure levels L1 $\prec$ L2 $\prec$ L3; interaction budget B
Require: current episode index e; frozen policy π_θ; routing weights α, β, γ; utility coefficients $\kappa, \bar{u}, \eta_p, \eta_c, \lambda$
$p_1 \leftarrow \arg\max_{p \in \mathcal{P}} (\mathrm{sim}(p, q) + \alpha\, U(p) + \beta\, T(p))$; $W \leftarrow \{p_1\}$; $k_1 \leftarrow 1$ {first working-set page and resident page (Eqs. 5 and 6)}
for $i = 2$ **to** c **do**
  $p^\star \leftarrow \arg\max_{p \in \mathcal{P} \setminus W} (\mathrm{sim}(p, q) + \alpha\, U(p) + \beta\, T(p) - \gamma\ \mathrm{red}(p, W))$ {Eq. 5}
  $W \leftarrow W \cup \{p^\star\}$
end for
for $t = 1$ **to** B **do**
  $m_t \leftarrow \ell_{k_t}(p_t)$ {injected memory at level k_t (Eq. 4)}
  $a_t \sim \pi_\theta(\cdot \mid q, h_t, m_t)$; $o_t \leftarrow \textsc{EnvStep}(a_t)$
  if the episode terminates **then**
    break
  end if
  $f \leftarrow F(h_{t+1})$ {Eq. 8}
  if f **then**
    if $k_t = 3 \ \vee\ \mathrm{mismatch}(h_{t+1})$ **then**
      $p_{t+1} \leftarrow \mathrm{Recovery}(h_{t+1}, \mathcal{P})$; $W \leftarrow (W \setminus \{p_t\}) \cup \{p_{t+1}\}$; $k_{t+1} \leftarrow 1$ {exhausted or mismatched: replace the page (Eq. 9)}
    else
      $p_{t+1} \leftarrow p_t$; $k_{t+1} \leftarrow \varphi(k_t) = \min(k_t + 1, 3)$ {insufficient but recoverable: escalate one level}
    end if
    apply a cooldown window to prevent uncontrolled page churn
  else
    $p_{t+1} \leftarrow p_t$; $k_{t+1} \leftarrow k_t$
  end if
end for
$\mathcal{D}_e \leftarrow$ {pages exposed during episode e} {mark exposed pages dirty}
for all $p \in \mathcal{D}_e$ **do**
  $u_{\mathrm{outcome}}(p) \leftarrow y - \kappa\,(1 - y)$; $u_{\mathrm{progress}}(p) \leftarrow \min(\bar{u},\ \eta_p\, n_{\mathrm{progress}}(p))$
  $u_{\mathrm{cost}}(p) \leftarrow \eta_c\, n_{\mathrm{invalid}}(p)$ {Eq. 11}
  $u_e(p) \leftarrow u_{\mathrm{outcome}}(p) + u_{\mathrm{progress}}(p) - u_{\mathrm{cost}}(p)$ {Eq. 10}
  $U_{e+1}(p) \leftarrow (1 - \lambda)\, U_e(p) + \lambda\, u_e(p)$ {Eq. 12}
end for
write reflection record r_e; $\mathcal{M}_{e+1} \leftarrow \mathrm{Maintain}(\mathcal{M}_e, \mathcal{D}_e, r_e)$ {Eq. 13: attach / create / merge / split; reinforce / revise / demote / merge / archive / retire}
return updated store $\mathcal{M}_{e+1}$

B Page schema and implementation details

A page stores the following fields: page identifier, header, summary, member record identifiers, conflicting record identifiers, optional neighboring page identifiers, creation and update timestamps, version, access count, page-fault count, observed utility, temperature, and lifecycle status. A record stores its reflection text, task description, source trajectory, outcome, timestamps, page assignment,

Table 2: Fixed protocol parameters used across tasks and benchmarks.

Symbol	Meaning	Value
c	working-set capacity (Eq. 6)	1
κ	failure penalty in u_{outcome} (Eq. 11)	0.25
$\bar{u}$	cap on u_{progress} (Eq. 11)	0.5
α	utility weight in routing (Eq. 5)	0.12
β	temperature weight in routing (Eq. 5)	1.00
γ	redundancy weight in routing (Eq. 5)	0.10
η_p	progress weight (Eq. 11)	0.04
η_c	invalid-action weight (Eq. 11)	0.02
λ	EMA mixing coefficient (Eq. 12)	0.2

exposure statistics, downstream progress and success counts, harmful or no-progress counts, and archive status.

Retrieval uses lightweight lexical similarity or embeddings only to generate page candidates; it does not determine the final exposure. The organizer then decides whether a reflection should attach to an existing page or create a new page, and maintenance can merge or archive pages. The policy backbone remains unchanged, so all memory adaptation occurs through page construction, exposure, and test-time updates.

C PROMPTS

We list the main prompt templates used by PAGEDMEM. Braces denote runtime placeholders that are instantiated by the implementation.

C.1 POLICY PROMPT

System and user prompts for action selection

```
System:
You are an expert agent operating in a text-based interactive
    environment.
Reason step-by-step inside <think>...</think>, then choose exactly one
     action and put it inside <action>...</action>.

User:
Task: {task}
{memory}
{history}
Observation: {observation}
Actions:
{actions}
{extra_instruction}
```

C.2 REFLECTION WRITER PROMPT

System and user prompts for reflection extraction

```
System:
You extract cautious reusable lessons from agent experience.

User:
Review this interaction episode and write one compact reflection for
    future agents.
```

```
Extract flexible strategy and recovery advice rather than a fixed
    action script. For failures,
identify what stalled and what to try differently. Generalize only as
    far as the evidence supports.
Return JSON only: {"reflection":"..."}
Task: {task}
Success: {success}
Trajectory: {trajectory}
```

C.3 PAGE ORGANIZER PROMPTS

System and user prompts for page organization

```
System:
You organize free-form agent reflections into semantic memory pages.
Pages are soft, reusable advice, not rigid procedures. Preserve
    exceptions and do not invent
facts. Return valid JSON only.

Page formation:
Create one semantic page from the reflections below. The header must
    be one short
high-level reminder (at most 18 words). The summary must be compact
    but complete,
reflection-style advice (at most 150 words), mention important
    recovery lessons, and
remain adaptable to the current observation. Return
{"header":"...","summary":"..."}.
Task family: {task_family}
Evidence: {evidence}

Attach or create:
Decide whether the new reflection belongs in one candidate page or
    needs a new page.
Attach only when it expresses compatible advice; a different object is
     not by itself
a new topic. If attaching, revise the page header and summary to
    incorporate useful
new or conflicting evidence without making a rigid action script.
    Return
{"decision":"attach|create","page_id":"candidate id or empty",
"header":"...","summary":"...","reason":"..."}.
New reflection: {new_reflection}
Candidates: {candidates}

Page revision:
Revise this exposed semantic page after a real execution. If the
    evidence does not
justify a content change, preserve the current meaning. Keep the
    header short and the
summary soft/adaptable; never turn it into a fixed action script.
    Return
{"header":"...","summary":"..."}.
Current page: {page}
New episode reflection: {new_reflection}
Episode feedback: {episode_feedback}
Page evidence: {evidence}
```

D Implementation Details

All reported runs use deterministic decoding with temperature 0.0. Policy action calls use a maximum of 4096 generated tokens. The reflection writer and page organizer use a maximum of 2048 generated tokens and request JSON objects. For each reported backbone, the policy and memory models are instantiated with the same backbone model.

E Step Efficiency over the Evaluation Stream

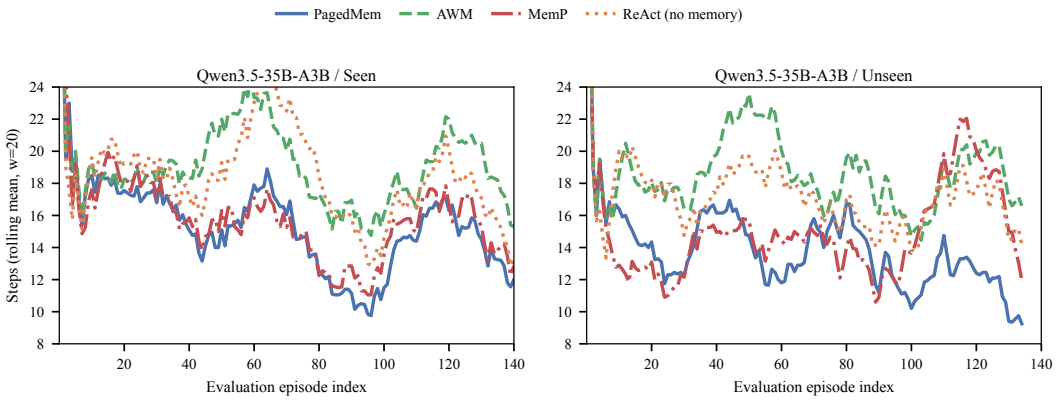

Figure 4: Required steps over the evaluation stream on Qwen3.5-35B-A3B.

Figure 4 reports the steps for all four methods on Qwen3.5-35B-A3B. PagedMem attains the lowest overall mean on both splits: 14.68 and 13.11 steps, compared with 15.20 and 14.43 for MemP, 17.96 and 16.67 for ReAct, and 18.67 and 18.28 for AWM. The advantage is also visible over the stream: on Seen, PagedMem's mean over the first 20 episodes is 17.55 and falls to 12.00 over the final 20; on Unseen, it falls from 14.35 to 9.25. The memory baselines do not reach the same final level: in the last 20 episodes, MemP requires 12.85 and 11.95 steps, ReAct 13.15 and 14.15, and AWM 15.30 and 16.60. This pattern shows that the efficiency gain is sustained across the evaluation stream rather than being driven by a few early episodes, and it is consistent with utility-driven page updates gradually aligning the exposed memory with the evaluation tasks.