MAGE: Multi-Agent Self-Evolution with Co-Evolutionary Knowledge Graphs

Ruiyi Yang[1,2], Zechen Li[1],
Hao Xue[3,2,1], Imran Razzak[4,1], Flora D. Salim[1,2],
[1]University of New South Wales,
[2]DataCruiser,
[3]The Hong Kong University of Science and Technology (Guangzhou),
[4]Mohamed Bin Zayed University of Artificial Intelligence,
Correspondence: ruiyi.yang@unsw.edu.au

Abstract

Self-evolving language-model agents must decide what to learn next and how to preserve what they have learned across iterations. Existing systems typically carry this cross-iteration knowledge as natural-language feedback, flat episodic memory, or implicit reinforcement signals, none of which cleanly supports a frozen weak backbone at inference time. This paper introduces MAGE (**M**ulti-**A**gent **G**raph-guided **E**volution), a framework that externalizes self-knowledge into a four-subgraph co-evolutionary knowledge graph. Its experience subgraph stores both teacher-written failure corrections and the learner's own past correct reasoning traces, which are retrieved as task-conditioned guidance for a frozen execution model. During evolution, the graph, a task-level search bandit, and a skill-level routing bandit are updated from the same reward stream, while the learner's backbone remains unchanged. We further provide structural analysis showing how append-only memory growth, bounded curriculum coverage, and task-filtered retrieval together support stable improvement of the retrieval substrate for frozen-learner evolution. Across nine benchmarks spanning mathematical reasoning, multi-hop and open-domain question answering, spatio-temporal analysis, financial numerical reasoning, medical multiple-choice, an open-world survival game, and web navigation, MAGE achieves strong performance against prompt-based frozen-backbone baselines. Ablations show that self-harvested success traces and teacher-written corrections are complementary, with success memories contributing most on reasoning-template-heavy tasks and corrective memories supporting harder composition and interaction settings. Code is available in https://github.com/RuiyiYang01/mage-alg

1 Introduction

Self-evolving language-model agents aim to improve from their own experience after deployment. Most recent systems instantiate this idea with a two-actor loop: one component explores or proposes tasks, while another component is updated from the resulting trajectories. The update mechanism itself may vary, such as supervised fine-tuning, reinforcement learning, self-play, or retrieval-time adaptation. However, these systems face a common upstream question: *what experience should be retained across iterations, and in what form should it be represented?* Existing self-evolving agents typically answer this question by storing cross-iteration knowledge in one of three forms. Natural-language feedback [30, 18] is easy to generate but can collapse into generic advice when the teacher is weak or the failure mode is systematic. Flat episodic memory [24, 39, 27, 2] stores trajectories or principles, but provides little structure for curriculum selection or dependency-aware reuse. Implicit reinforcement signals [22, 5, 4, 11] can drive parameter updates, but make the learned

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

curriculum difficult to inspect and require a malleable student model. Motivated by these limitations, this paper asks: *Can the carrier of self-evolution be moved from model parameters or free-form feedback into a structured graph, so that a frozen execution backbone can improve across iterations?*

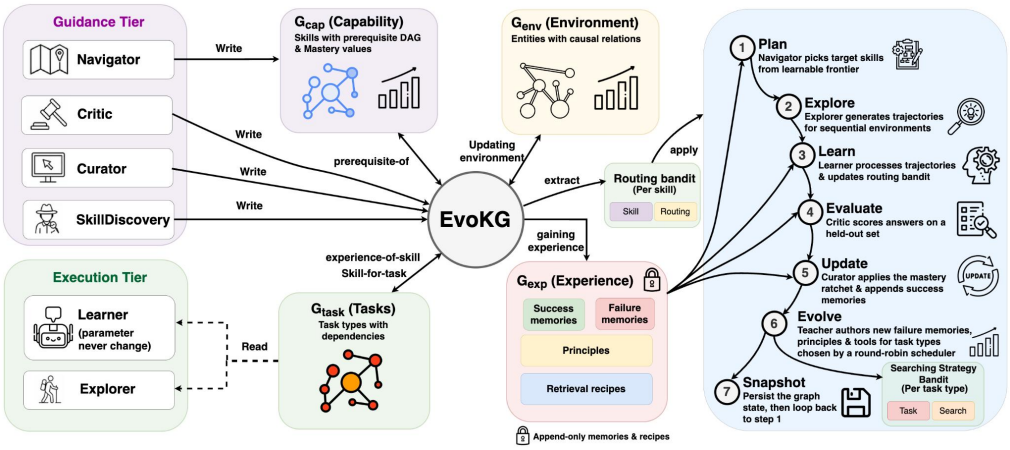


Figure 1: The MAGE framework. A strong *guidance tier* ($\mathcal{L}_{\mathrm{G}}$) writes to the four-subgraph co-evolutionary knowledge graph (EVOKG), while a frozen *execution tier* ($\mathcal{L}_{\mathrm{E}}$) answers questions through a Learner that consults a dual success/failure memory index, a task-conditioned search-strategy bandit, and a per-skill routing bandit. Correct answers are harvested back into the graph as success memories. The graph and the two bandits carry the cross-iteration learning signal; the execution backbone remains frozen.

We introduce MAGE (**M**ulti-**A**gent **G**raph-guided **E**volution), a framework that externalizes self-knowledge into a four-subgraph co-evolutionary knowledge graph, EVOKG. The graph contains capability, task, experience, and environment subgraphs. Its experience subgraph stores two complementary memory types: *failure memories*, written by a strong guidance tier from evaluation errors, and *success memories*, harvested from the frozen execution tier's own correct reasoning traces. At inference time, the learner retrieves task-conditioned memories and uses them as guidance for the frozen backbone. Across iterations, three state variables co-evolve under the same per-question reward stream: (i) the graph accumulates structured experience, (ii) a task-level search bandit learns retrieval strategies, and (iii) a skill-level routing bandit selects inference-time strategies. Thus, correct answers become future demonstrations, failures become future corrections, and the policies that produced them are updated in the same loop.

Our analysis formalizes the structural conditions under which this external memory substrate can support self-evolution in the frozen-backbone setting. Append-only growth of protected experience nodes yields information-monotonic graph growth; a recency-weighted round-robin curriculum gives a bounded coverage gap over observed task types; and an asymmetric mastery update bounds per-step forgetting. In conjunction with task-filtered retrieval, these properties provide a structural justification for using EVOKG as the carrier of cross-iteration learning, while the empirical results test how this design behaves with real language models.

Contributions.

1. **Co-evolutionary graph memory for self-evolving agents.** We propose EVOKG, a four-subgraph knowledge graph that represents capabilities, tasks, experience, and environment context, and preserves append-only success, failure, and principle memories as the external carrier of learning.
2. **Frozen-backbone co-evolution through graph and bandit updates.** We couple the graph with a task-level search bandit and a skill-level routing bandit, all updated from the same per-question reward stream while the execution backbone remains unchanged.
3. **Structural analysis of memory growth, coverage, and stability.** We prove information-monotonicity of append-only graph growth, a bounded coverage gap for the round-robin curriculum, and a stability bound for the asymmetric mastery ratchet; under task-filtered retrieval, these results characterize when the retrieval substrate can improve without parameter updates.

4. **Evaluation across heterogeneous agent tasks.** We evaluate MAGE on nine benchmarks spanning mathematical reasoning, multi-hop and open-domain question answering, spatio-temporal analysis, financial numerical reasoning, medical multiple-choice reasoning, open-world survival, and web navigation. The results show strong performance against frozen-backbone prompting baselines, and ablations show that self-harvested success memories provide the largest gains on reasoning-template-heavy tasks while teacher-written corrections remain complementary in other settings.

2 Related Work

Self-evolving language-model agents. Self-evolving agents aim to improve through iterative interaction, evaluation, and update. EXIF [30] instantiates this loop with a teacher-student design. A teacher explores an environment, generates instruction-trajectory pairs, supervises a student through SFT, and produces natural-language feedback for subsequent rounds. Recent systems such as AGENT0 [26], AGENTEVOLVER [35], SAGE [16], MAE [5], SIRIUS [41], ABSOLUTE ZERO [40], and INFIAGENT [34] further explore curriculum generation, self-play, proposer-solver co-evolution, and reinforcement learning for agent improvement. These methods demonstrate that iterative experience can improve agents, but the accumulated knowledge is usually carried either by parameter updates or by verbal feedback rather than by an explicit structured state.

Knowledge graphs and language-model agents. Knowledge graphs have been used to support language-model agents in retrieval, reasoning, and domain organization. GraphRAG [8] and related graph-based retrieval methods organize external evidence for question answering, while AGENTI-GRAPH [42] builds graph structures for domain-specific interactive agents. AGENTIC-KGR [11] studies multi-agent reinforcement learning for constructing knowledge graphs, and VOYAGER [20] maintains a growing skill library during open-ended embodied exploration. These works show the utility of graph-structured or library-like external state, although the graph is typically used as a retrieval artifact, a constructed output, or a skill repository rather than as the evolving substrate that organizes cross-iteration learning.

Experience-driven memory. A growing line of work equips agents with non-parametric memory. EVOLVER [24] distills trajectories into abstract principles, MEMRL [39] learns over an episodic-memory store, and A-MEM [27], ERL [2],SPLIT-RAG [28], and REMEM [23] explore dynamically indexed memories over experiences, principles, or triplets. These approaches externalize part of the learning process from model parameters and make past experience available at inference time. However, most memory stores are organized as flat collections whose entries are retrieved by similarity, with limited structure for representing capability dependencies, task coverage, or the different roles of successful and failed trajectories.

Curriculum, skill discovery, and dynamic prompting. Curriculum and skill discovery are central to open-ended agents. VOYAGER [20] grows a code-based skill library, SEAGENT [19] learns software use through an auto-generated curriculum, and proposer–solver systems such as AGENT0 and ABSOLUTE ZERO construct tasks through self-play or adversarial generation. In parallel, prompting methods such as self-consistency [21], ReAct [33], and Reflexion [18] improve inference-time behavior through sampling, tool use, or verbal self-reflection. These lines highlight the importance of deciding what experience to expose to the model, but their prompts, memories, or curricula are usually fixed, locally retrieved, or generated without an explicit coverage guarantee over task types.

Skill evolution and efficient coordination. MemSkill [37] learns and evolves memory-management skills, while SkillRL [25] co-evolves a skill library with a parameter-updated policy. SquidAgent [14] uses predicted output-token costs to decide when to parallelize, combining inherited worker context with shared conventions to reduce coordination overhead. Its focus on within-task execution efficiency complements MAGE's cross-iteration evolution of persistent experience; neither contribution implies that adding agents or inference calls is uniformly beneficial. Appendix W compares additional contemporaneous skill-evolution methods.

Position of MAGE. MAGE combines these threads by treating the agent's persistent state as a co-evolutionary knowledge graph rather than as free-form feedback, a flat memory bank, or an implicit reward signal. The graph links capabilities, task types, experience, and environment state;

separates teacher-written failure memories from self-harvested success traces; and co-evolves with a search bandit and a routing bandit under the same reward stream. This design makes the graph the carrier of cross-iteration learning while keeping the execution backbone frozen, and uses graph structure to couple what is learned, what is retrieved, and which task types are revisited.

3 Method

MAGE keeps the inference-time backbone frozen and stores cross-iteration learning in an explicit, inspectable graph. The graph is a typed directed multigraph $\mathcal{K} = (G_{\text{cap}}, G_{\text{task}}, G_{\text{exp}}, G_{\text{env}})$ whose protected experience nodes include guidance-written principles, failure memories from evaluation errors, and success memories harvested from correct learner answers. Each protected node is tagged with the task type it resolves and the skill it exercises, enabling task- and capability-conditioned retrieval. At inference time, the Learner does not call the guidance tier or update model parameters; it conditions the frozen backbone on a task-filtered slice of G_{exp} retrieved by embedding similarity. The remainder of this section defines the graph substrate (§3.1), the two-tier architecture and dual memory index (§3.2), the evolution loop (§3.3), the curriculum and mastery updates (§3.4), and the inference-time retrieval rule (§3.5); full proofs and implementation details are deferred to Appendices A–C.

3.1 The Co-Evolutionary Knowledge Graph

The co-evolutionary knowledge graph is a typed directed multigraph $\mathcal{K} = (G_{\text{cap}}, G_{\text{task}}, G_{\text{exp}}, G_{\text{env}})$. The capability subgraph G_{cap} stores skills with mastery values in $[0, 1]$ and prerequisite edges. The task subgraph G_{task} stores task types with dependency edges and a SKILL_FOR_TASK : $G_{\text{cap}} \to G_{\text{task}}$ resolver. The experience subgraph G_{exp} stores five classes of experience node: **principles**, **failure memories** authored by the guidance tier, **success memories** harvested from the frozen learner's own correct reasoning, retrieval recipes, and abstracted patterns. The environment subgraph G_{env} stores observed environment entities, relations, and task context used by the agent.

Capability subgraph G_{cap}. Nodes are skills with attributes (name, mastery $\in [0, 1]$, prompt template, strategy, principles). Edges are PREREQUISITE_OF and COMPOSES_INTO; the curator preserves acyclicity so the Navigator's topological sort is well defined.

The curator preserves an append-only invariant on the three protected experience classes: principle, failure-memory, and success-memory nodes are never removed once committed, even when mutable working slots such as mastery values, prompt templates, or strategies are revised. This invariant gives the graph a monotone protected memory core, which later supports the retrieval-side analysis.

Theorem 1 (EVOKG Information Monotonicity, Appendix A.1)**.** *Under the append-only invariant on principle, failure-memory, and success-memory nodes, $I(Y; \mathcal{K}_k) \le I(Y; \mathcal{K}_{k+1})$ for all k, where Y is the answer random variable on a fixed task distribution and $\mathcal{K}_k$ is the graph state at the end of iteration k.*

3.2 Two-Tier Architecture and the Dual Memory Index

MAGE separates graph writing from inference-time execution. The *guidance tier* $\mathcal{L}_{\text{G}}$ is a stronger model used only for graph writes, managing skill discovery, principle extraction, failure-memory authoring, dynamic-tool generation, retrieval-recipe authoring, and Navigator refinement. The *execution tier* $\mathcal{L}_{\text{E}}$ is an open-weight frozen model used for trajectory generation and inference-time question answering. Thus, $\mathcal{L}_{\text{G}}$ is absent from the inference path, and $\mathcal{L}_{\text{E}}$ never receives parameter updates.

The experience subgraph is indexed by two parallel embedding stores with shared retrieval logic but distinct payloads. The **failure index** stores teacher-written worked examples that provide corrective reasoning for observed errors. The **success index** stores the learner's own correct reasoning traces, and for multi-step questions, an explicit $(\text{skill}, \text{step output})$ decomposition that harvested from correctly solved evolution examples. These traces allow iteration N's correct answers to become iteration $N{+}1$'s demonstrations.

Success-memory harvest structure. A success-memory node records a single t-tagged correct answer and four payload fields: the verbatim question, the frozen learner's raw chain-of-thought up

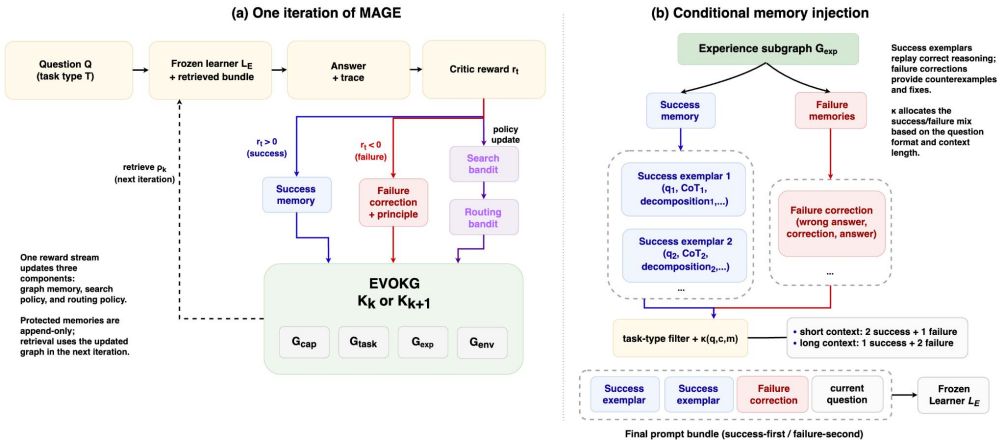


Figure 2: One-iteration co-evolution and conditional memory injection in MAGE. The frozen learner answers using a task-conditioned bundle of success exemplars and failure corrections retrieved from EVOKG. The critic produces reward r_t, which appends correct answers as success memories, triggers guidance-written corrections for errors, and updates the search and routing bandits. Thus memory growth, retrieval, and policy adaptation are coupled through one reward stream.

Algorithm 1 MAGE evolution loop (one iteration).

1: $\mathcal{P}_k \leftarrow \textsc{Navigator}(\mathcal{K}_{k-1})$ {PLAN}
2: $\mathcal{D}_k \leftarrow \textsc{Explorer}(\mathcal{P}_k, \mathcal{E}, \mathcal{L}_{\mathrm{E}})$ {EXPLORE; sequential environments only}
3: $E_k, \mathcal{S}_k \leftarrow \textsc{Critic}\big(\textsc{Learner}(\mathcal{K}_{k-1}), \mathcal{E}\big)$ {EVALUATE; harvest success memories}
4: $\mathcal{K}'_k \leftarrow \textsc{Curator.update}(\mathcal{K}_{k-1}, E_k, \mathcal{S}_k)$ {UPDATE; ratchet and append $\mathcal{S}_k$}
5: $\mathcal{K}_k \leftarrow \textsc{Evolve}(\mathcal{K}'_k, E_k, k, \mathcal{L}_{\mathrm{G}})$ {EVOLVE; write failures, principles, tools}
6: $\textsc{Snapshot}(\mathcal{K}_k)$; roll back mutable slots if $\mathrm{acc}(\mathcal{K}_k) < \mathrm{acc}(\mathcal{K}_{k-1}) - \delta$

to a character cap, the correct answer string, and for multi-step queries an explicit decomposition into $(\text{skill_name}, \text{step_output})$ tuples that preserves the intermediate reasoning steps the learner took to reach the answer. The decomposition field is the channel that lets a future similar multi-step question retrieve not only "here is a question-answer pair" but "here is the full reasoning trace and the sub-step pattern that worked," which is what distinguishes the success-memory channel from ordinary few-shot exemplars.

Finally, two lightweight bandits adapt how the graph is used: a per-skill **routing bandit** over inference strategies, and a per-task-type **search bandit** over retrieval strategies. Both update from the same per-question correctness reward that drives graph writes, so memory growth, search adaptation, and routing adaptation co-evolve while $\mathcal{L}_{\mathrm{E}}$ remains frozen.

3.3 Evolution Loop

Core loop and scope. Retrieve task-conditioned experience, execute graph-derived prerequisites, append evaluated successes and corrections, and roll back harmful mutable-state updates. The search and routing policies are optional, activation-dependent extensions; task search has no significant measured gain. Environment state, principles, recipes, and patterns are task-dependent support, without a claim of independent benefit on every task (Appendix S).

Six role-specific agents cooperate through the loop in Algorithm 1: SkillDiscovery, Navigator, Explorer, Learner, Critic, and Curator. Their tier assignments and roles are listed in Appendix C. Figure 2 illustrates one iteration of memory writes, bandit updates, and next-iteration retrieval. A delta-guard protects the mutable working state of the graph. If the post-update accuracy drops by more than δ, MAGE rolls back mutable slots such as mastery values, prompt templates, and strategies. Protected memory and principle nodes appended during the iteration are retained, preserving the append-only invariant. The EVOLVE step rotates through four action types: principle extraction, dynamic-tool authoring, prompt refinement, and skill splitting, thus graph growth is not restricted to a single update mode.

3.4 Curriculum and Mastery Updates

The Navigator first computes the *learnable frontier* of G_{cap}: skills whose prerequisites are mastered but whose own mastery remains below threshold. The EVOLVE phase then selects which task types receive new failure memories and principles. A purely failure-count-based selector can repeatedly focus on the same high-error task types, so MAGE uses a recency-weighted round-robin score,

$$s(t) \;=\; n_{\text{fail}}(t) + \lambda\big(k - k_{\text{last}}(t)\big), \qquad \lambda > 0, \tag{1}$$

and selects the top-M scoring task types, where $n_{\text{fail}}(t)$ is the failure count for task type t and $k_{\text{last}}(t)$ is the most recent iteration in which t was selected.

The Curator updates per-skill mastery using an asymmetric exponential moving average that increases quickly after successful evidence but decays slowly after poor measurements:

$$m_k = \begin{cases} \alpha\, e_k + (1-\alpha)\, m_{k-1}, & e_k \geq m_{k-1}, \\ m_{k-1} - \gamma\,(m_{k-1} - e_k), & e_k < m_{k-1}, \end{cases} \qquad \alpha > \gamma \in (0,1), \tag{2}$$

where e_k is the Critic's measured success rate. The selector and mastery update give two structural properties.

Theorem 2 (Bounded Coverage Gap, Appendix A.2)**.** *Under Eq.* (1) *with at most M targets per iteration, every observed task type t is reselected within $\lceil n_{\max}/\lambda + N/M \rceil$ iterations, where N is the number of observed task types and $n_{\max} = \max_t n_{\mathit{fail}}(t)$.*

Theorem 3 (Mastery Ratchet Stability, Appendix A.3)**.** *Under Eq.* (2)*, $m_k \geq (1-\gamma)^j \max_{k' \leq k-j} m_{k'}$ for any window j: a structural lower bound on running mastery in terms of any past peak.*

Together, Theorem 2 bounds how long an observed task type can be left unselected, while Theorem 3 bounds per-step mastery decay. Both properties follow from the update rules themselves rather than from implementation-specific heuristics.

3.5 Conditional Memory Injection at Inference

At inference time, given a question q of task type t with context c, the Learner retrieves the top-K task-type-filtered memories from both the success and failure indices. A format-conditional allocation κ determines the mixture: short-context questions receive a success-heavy bundle, while long-context questions with rich gold context receive a failure-heavy bundle. The retrieved memories are formatted success-first and failure-second as in-context exemplars before being prepended to the question. Formally,

$$\rho_k(\cdot \mid q, t, c) \;=\; \textsc{TopK}\Big(\textsc{Embed}(q),\; \big\{\, m \in G_{\mathrm{exp}_k} \;:\; m.\text{task_type} = t \wedge \kappa(q, c, m) \,\big\}\Big), \tag{3}$$

where κ enforces the format-conditional success/failure allocation. The Learner then emits

$$\hat{y} = \mathcal{L}_{\mathrm{E}}\big(\textsc{Format}(\rho_k(\cdot)) \,\|\, \textsc{Prompt}(q, c, \mathcal{K}_k)\big). \tag{4}$$

We state the final result under an idealized retrieval-error model: additional task-matched memories expand the oracle top-K candidate set, and ε_K bounds the total-variation gap between the embedding retriever and this oracle selector. Together, task-type filtering, append-only graph growth, and bounded curriculum coverage yield a retrieval-side support result.

Theorem 4 (Task-Filtered Retrieval Support, Appendix A.5)**.** *Let $A_k(t) = \mathbb{E}_{q\sim\mathcal{D}_t,\, \mu\sim\rho_k}\big[\mathrm{acc}(\mathcal{L}_{\mathrm{E}}(\mathrm{format}(\mu) \,\|\, q, c), y)\big]$. Under Theorems 1–2 and the task-type filter, $A_{k+1}(t) \geq A_k(t) - \varepsilon_K$ for every observed t, where ε_K is the rank-K retrieval error of the embedding index.*

Theorem 4 is a structural *retrieval-side* result, not an unconditional guarantee about arbitrary LLM behavior. Append-only graph growth and bounded coverage prevent the task-conditioned evidence set from shrinking, while ε_K measures the gap between embedding retrieval and an oracle top-K selector. Whether the frozen backbone can exploit this enriched retrieval substrate is evaluated empirically through ablations and cross-teacher analysis.

4 Experiments

4.1 Setup

We evaluate MAGE on nine benchmarks drawn from six task families. **GSM8K** [7] and **Real-Math** [38] test mathematical reasoning. **HotpotQA** [31] and **WebQA** [3] test multi-hop and open-domain factoid QA. **STBench** [12] tests spatio-temporal analysis across twenty-seven task types. **FinQA** [6] tests table-grounded numerical reasoning over financial reports, and **MedQA-USMLE** [10] tests medical multiple-choice reasoning. **Crafter** [9] is an open-world sequential survival game, and **WebShop** [32] is a web-navigation task with parameterised actions. FinQA and MedQA-USMLE are used only as standardized reasoning benchmarks; MAGE is not evaluated or proposed as a financial-advice or clinical decision-support system.

Evaluation protocol and leakage control. For each benchmark, we separate the examples used for graph evolution from the final held-out evaluation pool used for reporting results. Success memories, failure memories, principles, and bandit updates are produced only from the evolution pool. After evolution, EVOKG and the two bandits are frozen, and all reported test metrics are computed without writing any test example back into the graph. For benchmarks with native exact-match metrics, including FinQA and MedQA-USMLE, we report automatic scores directly. For free-form QA and reasoning benchmarks, we use the semantic judge protocol described below and provide automatic-metric cross-checks where available.

Models. We instantiate MAGE along two orthogonal axes: a frozen execution tier using one of two open-weight 8B instruction-tuned models (Qwen3-8B and Llama-3.1-8B-Instruct) served via vLLM at temperature 0, and a guidance tier using one of three closed-weight teachers (Claude Opus 4.6, Sonnet 4.6, and Haiku 4.5) used only during graph evolution. The main reasoning table reports Qwen3-8B with Sonnet and Opus guidance, while the domain-extension table reports Llama-3.1-8B with the same guidance tiers; the full teacher–learner matrix and additional multi-seed results are reported in Appendices K and L. All hyperparameters are fixed across benchmarks (Appendix D).

Baselines and scoring. On the reasoning benchmarks, we compare against frozen-backbone prompt-based scaffolds: zero-shot chain-of-thought, eight-shot CoT, self-consistency with ten samples [21], ReAct [33], and Reflexion [18]. These baselines isolate the frozen-backbone setting: all methods use the same execution backbone at inference time and differ only in the external scaffold placed around it. For sequential environments, we compare against same-backbone standalone baselines and published WebShop references under the same catalog setting. For judge-scored benchmarks, all methods use the same answer-extraction pipeline and the same Claude Sonnet 4.6 semantic judge on the evaluation pool; judge prompts and automatic cross-checks are provided in Appendix G. Published trained or domain-specific systems are included only as references, not as strict frozen-backbone baselines.

4.2 Main Results

Table 1: MAGE versus prompt-based scaffolds on the five reasoning benchmarks. All methods use Qwen3-8B as the frozen backbone and are scored by the same Claude Sonnet 4.6 judge on 200 held-out questions per benchmark. The MAGE columns report mean±std over multiple seeds where available; cell-level n is shown in subscript. Δ is the gap of the best MAGE cell to the strongest frozen-backbone baseline.

	Prompt-based baselines (Qwen3-8B, frozen)					MAGE (Qwen3-8B)		
Benchmark	0-shot CoT	8-shot	SC_{10}	ReAct	Reflexion	Sonnet	Opus	Δ
GSM8K	64.5	82.5	72.0	80.1	63.0	$88.2 \pm 2.9_{n=5}$	$\mathbf{90.4 \pm 3.8}_{n=5}$	+7.9
HotpotQA	84.5	78.5	80.5	70.5	78.5	$88.5 \pm 2.0_{n=5}$	$\mathbf{91.5 \pm 1.8}_{n=5}$	+7.0
WebQA	55.0	42.0	46.5	35.5	34.5	$59.6 \pm 3.4_{n=5}$	$\mathbf{62.7 \pm 1.8}_{n=5}$	+7.7
STBench	35.5	12.0	20.0	31.8	37.0	$54.8 \pm 2.1_{n=5}$	$\mathbf{57.8 \pm 2.9}_{n=5}$	+20.8
RealMath	53.5	32.5	33.5	43.5	34.5	$\mathbf{85.1 \pm 6.7}_{n=5}$	$82.8 \pm 3.5_{n=5}$	+31.6

Tables 1–3 summarize the main results across the original reasoning suite, the finance/medical domain extensions, and the sequential environments. On the core reasoning suite, MAGE outperforms the strongest frozen-backbone prompting baseline on all five benchmarks. The largest gains appear on

tasks that require reusable reasoning templates or task-type-specific retrieval, such as RealMath and STBench, while the gains are smaller but still positive on benchmarks where prompt-only baselines are already strong, such as GSM8K and HotpotQA. The domain extensions test whether the same mechanism transfers beyond the original reasoning suite. On FinQA and MedQA-USMLE, MAGE improves over the standalone Llama-3.1-8B baseline using benchmark-native exact-match scoring rather than an LLM judge. On FinQA, MAGE also exceeds the published frozen-8B financial reasoning reference; on MedQA, it narrows the gap to stronger prompt-engineered medical references without task-specific rationale pre-generation. These results suggest that graph-carried memory can support domain-specific numerical and multiple-choice reasoning, although the published domain-specific references are included only as contextual references rather than strict same-scaffold baselines.

Table 2: Domain extension to math, finance and medicine. FinQA uses numeric exact-match, and MedQA-USMLE uses letter exact-match.

	Llama-3.1-8B baselines		MAGE (Llama-3.1-8B)	
Benchmark	0-shot CoT	Frozen-8B reference	Sonnet	Opus
GSM8K	64.5	84.5 (8-shot, Meta)	$\mathbf{92.5} \pm 4.3_{n=5}$	$92.0 \pm 3.2_{n=5}$
FinQA	58.0	60.87 (Fino1, SFT)	$67.5 \pm 1.7_{n=5}$	$\mathbf{69.0 \pm 1.5_{n=5}}$
MedQA	64.5	74 (Medprompt)	$74.6 \pm 2.3_{n=5}$	$\mathbf{79.8} \pm 1.9_{n=5}$

On sequential environments, MAGE remains competitive. WebShop shows strong performance relative to published references under the same catalog setting. On Crafter, all teacher-guided configurations improve over the standalone Llama-3.1-8B BALROG reference. Several prompt-based scaffolds underperform zero-shot CoT on the frozen 8B backbones, suggesting that additional inference-time structure is not automatically beneficial at this scale. In contrast, MAGE improves by accumulating task-conditioned experience and adapting retrieval/routing policies across iterations. Automatic scoring cross-checks for benchmarks with native metrics are provided in Appendix G, and the full teacher–learner matrix is reported in Appendix K. Figure 3 summarizes the main effect sizes behind Tables 1 and 5. MAGE improves over the strongest frozen-backbone prompting baseline on all five reasoning benchmarks, and the success-memory ablation shows positive gains across the same benchmark family.

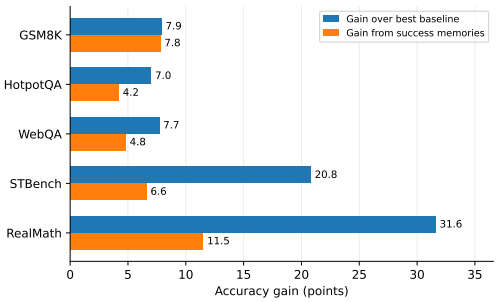


Figure 3: Effect sizes of MAGE on the reasoning benchmarks: gain over the strongest frozen-backbone baseline and gain attributable to success memories.

4.3 Teacher Usage and Inference Cost

A central design goal of MAGE is to separate training-time guidance from deployment-time inference: the stronger guidance tier writes to the graph during evolution, but the final inference loop runs on the frozen execution tier. Table 4 counts guidance-tier and execution-tier calls across completed training runs. Across completed runs, only $\mathbf{1.91}\%$ of language-model invocations are routed to the guidance tier during graph evolution. After graph construction, the inference-time guidance fraction is $\mathbf{0.00}\%$: all deployment-time queries are handled by the frozen execution model with retrieved graph memories. This separates the cost of building EVOKG from the cost of using it, and makes deployment independent of closed-weight guidance calls.

Inference latency. Per-question wall-clock latency on a single H100 80GB at fp16 ranges from 5.66 to 28.33 seconds for MAGE after graph construction, with no guidance-tier calls at inference time, faster than self-consistency-N=10 on every measured benchmark and competitive with or faster than ReAct and Reflexion. Detailed accuracy–latency comparisons are provided in Appendix N.

Table 3: MAGE on the two sequential environments. Crafter follows the BALROG protocol and reports peak achievement-unlock rate across iterations. WebShop follows the standard test protocol on the same hundred-product catalog as RetroAgent; RetroAgent is reproduced as published references.

Environment	Configuration	Task score / peak (%)	Strict success (%)
Crafter	Standalone Llama-3.1-8B (BALROG, 3 seeds)	25.5 ± 3.2	—
	MAGE Sonnet teacher ($n = 5$)	33.5 ± 2.3	—
	MAGE Opus teacher ($n = 5$)	$\mathbf{37.9 \pm 3.5}$	—
	MAGE Haiku teacher ($n = 5$)	34.8 ± 1.7	—
WebShop	RetroAgent (in-context, Qwen-7B reference)	87.6	78.9
	RetroAgent (RL-trained reflection)	88.9	82.3
	MAGE Sonnet teacher ($n = 5$)	$\mathbf{90.2 \pm 2.3}$	$\mathbf{84.0 \pm 1.8}$
	MAGE Opus teacher ($n = 5$)	89.7 ± 1.6	82.0 ± 1.0

Table 4: Teacher usage rate across completed MAGE training runs. Training-time fraction is the share of language-model calls routed to the closed-weight guidance tier ($\mathcal{L}_{\mathrm{G}}$); the remainder go to the local frozen execution tier ($\mathcal{L}_{\mathrm{E}}$). Inference-time fraction is measured after EVOKG with frozen bandits.

Phase / Benchmark	n runs	Guidance calls (mean)	Execution calls (mean)	Guidance fraction
Aggregate (all training runs)	52	**7,122** (total)	**336,272** (total)	**2.12%**
Inference / test (all runs)	—	**0**	—	**0.00%**
WebShop	5	18	1,699	0.1–1.9%
Crafter	11	118	22,508	0.0–0.9%
HotpotQA	4	212	5,362	3.0–4.6%
GSM8K	3	229	3,651	4.1–7.5%
RealMath	2	355	3,316	7.0–14.0%
STBench	2	385	2,948	10.2–14.0%
WebQA	3	480	3,134	11.9–14.7%
MedQA-USMLE	6	167	5,844	2.0–3.3%
FinQA	6	899	18,805	4.2–4.9%

4.4 Success-Memory Ablation

The success-memory ablation tests whether the learner's own correct reasoning traces provide a distinct source of cross-iteration learning beyond teacher-written failure memories. In the *No-success* variant, the harvest loop is disabled while failure memories, principles, and the remaining graph updates are kept unchanged. Table 5 shows that self-harvested success memories improve performance on all five reasoning benchmarks. The largest gains appear on mathematical reasoning, where correct traces provide reusable reasoning templates that can be replayed on future questions. STBench also benefits substantially, suggesting that task-conditioned self-replay helps when the benchmark contains many recurring task types. The gains on HotpotQA and WebQA are smaller but remain positive, indicating that success memories are still useful even when rich context or factoid answer formats reduce the need for template reuse. These results support the dual-memory design: success traces and failure corrections are complementary rather than interchangeable. Failure memories provide corrective guidance for observed mistakes, while success memories preserve reasoning patterns that the frozen learner has already executed correctly. Additional sequential-environment ablations and graph-growth measurements are provided in Appendices I and H.

To make the memory mechanism concrete, Appendix Q provides detailed retrieved bundles from GSM8K and HotpotQA to illustrate the mechanism behind Table 5. The GSM8K example combines learner-harvested success traces with teacher-written failure corrections, and its recovery trace shows previously failed questions being corrected after the memory bank grows. The HotpotQA example shows the same pattern in a multi-hop comparison setting: retrieved exemplars provide a reusable reasoning skeleton, while the correction memory specifies the attribute-comparison strategy. Together, these examples show that the dual-memory channel is not only an aggregate ablation effect, but also appears directly in the prompts consumed by the frozen learner.

4.5 Controlled Memory and Execution Comparisons

Table 6 adds matched memory controls and a prerequisite-removal intervention using the same frozen Qwen3-8B learner and evaluation instances within each comparison. These use corrected

Table 5: Success-memory ablation across the reasoning benchmarks. *No-success* disables harvest of the learner's own correct reasoning; teacher-written failure memories remain. Both columns report mean±std over matched seeds where available; Δ is the gap attributable to self-harvested experience.

Benchmark	MAGE (full)	No-success ablation	Δ (success memories)
GSM8K	$\mathbf{88.2 \pm 2.9}$	80.4 ± 12.7	$+7.8$
RealMath	$\mathbf{85.1 \pm 6.7}$	73.6 ± 9.3	$+11.5$
STBench	$\mathbf{54.8 \pm 2.1}$	48.2 ± 4.0	$+6.6$
HotpotQA	$\mathbf{88.5 \pm 2.0}$	84.3 ± 2.1	$+4.2$
WebQA	$\mathbf{59.6 \pm 3.4}$	54.8 ± 1.9	$+4.8$
Mean across the five reasoning benchmarks			$+7.0$

paper-aligned EM/F1, with strict automatic scoring for MemSkill, separately from the semantic-judge tables above.

Table 6: Matched comparisons (EM / F1, %). A-MEM receives identical success/failure episodes; no-chain disables prerequisite traversal. RealMath pools three graphs and 600 paired runs; MemSkill uses HotpotQA-200 and a matched 1,024-token answer budget.

Benchmark	Control	MAGE	Control
WebQA	A-MEM [27]	62.7/67.1	62.7/69.4
STBench	A-MEM	57.2/61.3	51.7/52.0
RealMath	A-MEM	82.4/84.5	69.7/66.8
RealMath	No-chain	82.4/84.5	72.0/74.0
HotpotQA	MemSkill [37]	54.0/71.67	40.5/55.34

The traversal gain is positive in all three graphs, with an accuracy–compute trade-off: 2.52 calls/19.15 seconds per query versus 1.00/12.23 for no-chain. A-MEM has higher WebQA F1, so structured execution is not uniformly preferable. Appendix R reports the backbone-matched SkillRL comparison; Appendices S–V provide component, local-teacher, transfer, growth, and case evidence.

5 Conclusion

MAGE studies self-evolution in the frozen-backbone setting, where cross-iteration learning must be carried by external state rather than model parameters. The central idea is to make this state explicit: a co-evolutionary knowledge graph stores capabilities, task types, environment context, teacher-written corrections, and the learner's own successful reasoning traces, while a search bandit and routing bandit adapt how this graph is used. The analysis shows that append-only protected memories, bounded task-type coverage, asymmetric mastery updates, and task-filtered retrieval provide a retrieval-side substrate for stable frozen-backbone self-evolution. Empirically, MAGE is evaluated across nine benchmarks spanning mathematical reasoning, multi-hop and open-domain QA, spatio-temporal analysis, financial and medical reasoning, open-world survival, and web navigation. The results show that graph-carried experience can provide strong gains over frozen-backbone prompting baselines, with the largest improvements on tasks that benefit from reusable reasoning templates and task-conditioned retrieval. Ablations further show that self-harvested success traces and teacher-written corrections are complementary: success memories are most useful when past correct reasoning can be replayed, while corrective memories and graph-derived principles support settings where error correction, composition, or interaction dominates.

Scope and limitation. MAGE is most effective when task experience exposes recurring structure that can be organized into reusable memories, task types, and skill dependencies. Performance may therefore depend on the quality of the curator's graph writes. Future work should study broader teacher families and stronger cross-backbone transfer.

Acknowledgments and Disclosures

This research used the NVIDIA H100 cluster provided by SHARON AI.

References

[1] Awais Ahmed, Xiaoyang Zeng, Rui Xi, Mengshu Hou, and Syed Attique Shah. Med-prompt: A novel prompt engineering framework for medicine prediction on free-text clinical notes. *Journal of King Saud University-Computer and Information Sciences*, 36(2):101933, 2024.

[2] Marc-Antoine Allard, Arnaud Teinturier, Victor Xing, and Gautier Viaud. Experiential reflective learning for self-improving llm agents. *arXiv preprint arXiv:2603.24639*, 2026.

[3] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

[4] Guoxin Chen, Zile Qiao, Wenqing Wang, Donglei Yu, Xuanzhong Chen, Hao Sun, Minpeng Liao, Kai Fan, Yong Jiang, Penguin Xie, et al. Mars: Optimizing dual-system deep research via multi-agent reinforcement learning. *arXiv preprint arXiv:2510.04935*, 2025.

[5] Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. Multi-agent evolve: Llm self-improve through co-evolution. *arXiv preprint arXiv:2510.23595*, 2025.

[6] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, et al. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, 2021.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[8] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[9] Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

[10] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

[11] Jing Li, Zhijie Sun, Zhicheng Zhou, Suming Qiu, Junjie Huang, Haijia Sun, and Linyuan Qiu. Agentic-kgr: Co-evolutionary knowledge graph construction through multi-agent reinforcement learning. *arXiv preprint arXiv:2510.09156*, 2025.

[12] Wenbin Li, Di Yao, Ruibo Zhao, Wenjie Chen, Zijie Xu, Chengxue Luo, Chang Gong, Quanliang Jing, Haining Tan, and Jingping Bi. Stbench: Assessing the ability of large language models in spatio-temporal analysis. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 749–752, 2025.

[13] Huawei Lin, Peng Li, Jie Song, Fuxin Jiang, and Tieying Zhang. MUSE-Autoskill: Self-evolving agents via skill creation, memory, management, and evaluation. *arXiv preprint arXiv:2605.27366*, 2026.

[14] Yexiong Lin, Shanshan Ye, Yu Yao, Zhen Fang, Bo Han, and Tongliang Liu. SquidAgent: Parallelize wisely, coordinate efficiently. *arXiv preprint arXiv:2610.08647*, 2026.

[15] Yuchen Ma, Yue Huang, Han Bao, Haomin Zhuang, Swadheen Shukla, Michel Galley, Xiangliang Zhang, and Stefan Feuerriegel. SkillGen: Verified inference-time agent skill synthesis. *arXiv preprint arXiv:2605.10999*, 2026.

[16] Yulin Peng, Xinxin Zhu, Chenxing Wei, Nianbo Zeng, Leilei Wang, Ying Tiffany He, and F Richard Yu. Sage: Multi-agent self-evolution for llm reasoning. *arXiv preprint arXiv:2603.15255*, 2026.

[17] Lingfei Qian, Weipeng Zhou, Yan Wang, Xueqing Peng, Jimin Huang, and Qianqian Xie. Fino1: On the transferability of reasoning enhanced llms to finance. *arXiv e-prints*, pages arXiv–2502, 2025.

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

[19] 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. *arXiv preprint arXiv:2508.04700*, 2025.

[20] 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. *arXiv preprint arXiv:2305.16291*, 2023.

[21] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[22] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

[23] Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

[24] Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

[25] Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

[26] Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. *arXiv preprint arXiv:2511.16043*, 2025.

[27] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

[28] Ruiyi Yang, Hao Xue, Imran Razzak, Shirui Pan, Hakim Hacid, and Flora D Salim. Divide by question, conquer by agent: Split-rag with question-driven graph partitioning. *arXiv preprint arXiv:2505.13994*, 2025.

[29] Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. SkillOpt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

[30] Yongjin Yang, Sinjae Kang, Juyong Lee, Dongjun Lee, Se-Young Yun, and Kimin Lee. Toward self-evolving systems of llm agents through exploration and iterative feedback.

[31] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380, 2018.

[32] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

[33] 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*, 2022.

[34] Chenglin Yu, Yang Yu, Songmiao Wang, Yucheng Wang, Yifan Yang, Jinjia Li, Ming Li, and Hongxia Yang. Infiagent: Self-evolving pyramid agent framework for infinite scenarios. *arXiv preprint arXiv:2509.22502*, 2025.

[35] Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, et al. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025.

[36] Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. CoEvoSkills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026.

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

[38] Jie Zhang, Cezara Petrui, Kristina Nikolić, and Florian Tramèr. Realmath: A continuous benchmark for evaluating language models on research-level mathematics. *arXiv preprint arXiv:2505.12575*, 2025.

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

[40] Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

[41] Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Zou. Sirius: Self-improving multi-agent systems via bootstrapped reasoning. *arXiv preprint arXiv:2502.04780*, 2025.

[42] Xinjie Zhao, Moritz Blum, Fan Gao, Yingjian Chen, Boming Yang, Luis Marquez-Carpintero, Mónica Pina-Navarro, Yanran Fu, So Morikawa, Yusuke Iwasawa, et al. Agentigraph: A multi-agent knowledge graph framework for interactive, domain-specific llm chatbots. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 6757–6761, 2025.

A Proofs

This appendix collects full proofs of the four theorems stated in the main text. The structure of each proof closely tracks the proof sketch in Sec. 3; the appendix provides the precise statements, the auxiliary lemmas, and the corner cases that the main-text sketches elide for space.

A.1 Proof of Theorem 1 (EvoKG Information Monotonicity)

Proof. Let Y denote the answer random variable on a fixed evaluation distribution $\mathcal{D}$ and $\mathcal{K}_k$ the EvoKG state at the end of iteration k. The curator's update procedure on G_{exp} is governed by two rules. **(i)** Every principle, failure-memory, and success-memory node added during the EVALUATE and EVOLVE phases is appended to G_{exp} and is exempt from the low-confidence pruner (`prune_low_confidence` skips `outcome` $\in \{$`principle`, `failure_memory`, `success_memory`$\}$). **(ii)** Pruning of other experience nodes (`outcome` $\in \{$`success`, `failure`$\}$, i.e. abstracted pattern summaries rather than worked-example memories) removes only nodes with confidence below a fixed threshold and is performed after the curator's append step. Therefore, at the end of iteration $k+1$, the protected node sets satisfy

$$G_{\text{exp}}{}^{\text{prin}}_{k+1} \supseteq G_{\text{exp}}{}^{\text{prin}}_{k}, \qquad G_{\text{exp}}{}^{\text{fail-mem}}_{k+1} \supseteq G_{\text{exp}}{}^{\text{fail-mem}}_{k}, \qquad G_{\text{exp}}{}^{\text{succ-mem}}_{k+1} \supseteq G_{\text{exp}}{}^{\text{succ-mem}}_{k},$$

and the curator's working slots on G_{cap} (mastery, prompt template, strategy) and G_{task} are updated in place but the corresponding nodes themselves are not deleted. Hence the node-and-edge set of $\mathcal{K}_{k+1}$ contains the node-and-edge set of $\mathcal{K}_k$ at the level of principle, failure-memory, success-memory, capability, and task subgraphs; the only nodes that may be removed are the low-confidence abstracted-pattern nodes, which by construction carry confidence below the threshold and are independent of Y given the rest of $\mathcal{K}_k$. Write $\mathcal{K}_{k+1} = (\mathcal{K}_k, \Delta_k)$ for the appended nodes $\Delta_k = \mathcal{K}_{k+1} \setminus \mathcal{K}_k$ (treating the deleted low-confidence nodes as conditionally independent of Y). By the chain rule for mutual information,

$$I(Y; \mathcal{K}_{k+1}) \;=\; I(Y; \mathcal{K}_k, \Delta_k) \;=\; I(Y; \mathcal{K}_k) \;+\; I(Y; \Delta_k \mid \mathcal{K}_k).$$

Conditional mutual information is non-negative, so $I(Y; \mathcal{K}_{k+1}) \geq I(Y; \mathcal{K}_k)$, with equality iff $I(Y; \Delta_k \mid \mathcal{K}_k) = 0$, i.e. iff $\Delta_k \perp Y \mid \mathcal{K}_k$.

Failure mode under non-append-only pruning. The argument requires the append-only invariant on all three protected node classes. A pruner that deletes worked-example memories whenever the underlying skill rises above some mastery threshold would violate it: the deletion step would remove nodes that carry information about Y (the worked example for question q does carry information about the answer to q), and the chain-rule argument would fail because $\mathcal{K}_{k+1}$ would no longer be a refinement of $\mathcal{K}_k$. The curator's `prune_low_confidence` explicitly exempts all three protected node types for exactly this reason. □

A.2 Proof of Theorem 2 (Bounded Coverage Gap)

Proof. Let $\mathcal{T}$ denote the set of task types observed at least once in any iteration up to the current iteration k, $|\mathcal{T}| = N$. Let $n_{\max} = \max_{t \in \mathcal{T}} n_{\text{fail}}(t)$ denote the maximum failure count over $\mathcal{T}$ at iteration k, and recall the selector

$$s(t) \;=\; n_{\text{fail}}(t) + \lambda(k - k_{\text{last}}(t)), \qquad \lambda > 0.$$

At iteration k the selector picks the top-M task types by s.

Fix any task type $t^* \in \mathcal{T}$ and assume t^* has not been selected for g consecutive iterations starting from iteration $k_0 = k - g$. Then $s_k(t^*) \geq n_{\text{fail}}(t^*) + \lambda g \geq \lambda g$. For any other task type $t' \in \mathcal{T} \setminus \{t^*\}$, the score satisfies $s_k(t') \leq n_{\max} + \lambda \cdot (k - k_{\text{last}}(t'))$, which is at most $n_{\max} + \lambda g$ if t' was last selected at the same iteration as t^* was last selected, and is otherwise smaller (because more recent selection ⇒ smaller recency bonus). Hence whenever $\lambda g \geq n_{\max}$—equivalently, $g \geq n_{\max}/\lambda$—we have $s_k(t^*) \geq n_{\max}$, which is at least as large as any score that an incumbent in the top-M achieved at the iteration in which it was most recently selected.

If $g \geq n_{\max}/\lambda$, then t^* is therefore selected at iteration k unless every one of the top-M slots is occupied by a task type with strictly higher score than t^*. A simple counting argument bounds the

number of additional iterations during which the top-M slots can all be occupied by task types other than t^*: each such iteration moves M task types out of the "not recently selected" pool, so after at most $\lceil N/M \rceil$ further iterations the round-robin sweep through the pool exhausts all alternatives and t^* enters the top-M. Combining the two bounds, t^* is selected within $\lceil n_{\max}/\lambda + N/M \rceil$ iterations of its previous selection (or its first appearance, for task types never previously selected).

Adversarial failure-count distributions. The bound is tight up to integer rounding when n_{fail} is dominated by a single task type. If n_{fail} is dominated by a small subset $S \subset \mathcal{T}$ with $|S| < M$, the bound becomes $\lceil n_{\max}/\lambda + (N - |S|)/M \rceil$, since the selector trivially keeps S in the top-M until $\lambda g \geq n_{\max}$ at which point any starved task type displaces a member of S. □

A.3 Proof of Theorem 3 (Mastery Ratchet Stability)

Proof. We work with the asymmetric exponential moving average

$$m_k = \begin{cases} \alpha e_k + (1-\alpha) m_{k-1}, & e_k \geq m_{k-1}, \\ m_{k-1} - \gamma(m_{k-1} - e_k), & e_k < m_{k-1}, \end{cases}$$

with $\alpha, \gamma \in (0, 1)$ and $e_k \in [0, 1]$.

Per-step lower bound. On the increase branch, $m_k = m_{k-1} + \alpha(e_k - m_{k-1}) \geq m_{k-1} \geq (1-\gamma) m_{k-1}$ since $\gamma > 0$. On the decay branch, rewrite $m_k = (1-\gamma) m_{k-1} + \gamma e_k \geq (1-\gamma) m_{k-1}$, since $e_k \geq 0$. So in both cases $m_k \geq (1-\gamma) m_{k-1}$.

Per-step upper bound. On the increase branch, $m_k = m_{k-1} + \alpha(e_k - m_{k-1}) \leq m_{k-1} + \alpha(1 - m_{k-1})$, since $e_k \leq 1$. On the decay branch, $m_k \leq m_{k-1}$. Combining gives $m_k \leq m_{k-1} + \alpha(1 - m_{k-1})$.

Window lower bound. Iterating the per-step lower bound j times gives $m_k \geq (1-\gamma)^j m_{k-j}$, which holds even when the j steps mix increase and decay (since the increase steps only tighten the bound). Choosing $k - j$ to be the iteration of any past peak, $m_k \geq (1-\gamma)^j \max_{k' \leq k-j} m_{k'}$.

Mixed sequences. A subtler bound holds when the window contains an increase: if step $k' \in [k-j, k]$ is an increase step with measured $e_{k'} \geq m_{k'-1}$, then the lower bound from that point onward starts from $\alpha e_{k'} + (1-\alpha) m_{k'-1}$ rather than $(1-\gamma)^{k-k'+1} m_{k'-1}$, which can be substantially higher when $e_{k'}$ is close to 1. The simple form stated in the theorem suffices for the anti-forgetting argument and avoids the case-analysis bookkeeping. □

A.4 Conditional Retrieval Rule ρ_k

The Learner's inference-time retrieval, summarised in Sec. 3.5, is

$$\rho_k(\cdot \mid q, t, c) \;=\; \textsc{TopK}\Big(\textsc{embed}(q),\; \big\{\, m \in G_{\exp_k} \;:\; m.\text{task_type} = t \wedge \kappa(q, c, m) \,\big\}\Big),$$

where κ encodes the format-conditional allocation (top-2 success + top-1 failure for short-context questions; top-1 success + top-2 failure for long-context questions, threshold 500 characters). The Learner emits $\hat{y} = \mathcal{L}_{\text{E}}(\textsc{format}(\rho_k(\cdot)) \,\|\, \textsc{prompt}(q, c, \mathcal{K}_k))$ with success memories placed first in the prompt and failure memories second.

A.5 Proof of Theorem 4 (Task-Filtered Retrieval Support)

Proof. We compose Theorems 1 and 2 with the conditional retrieval rule ρ_k defined in App. A.4.

Fix a task type t and a question $q \sim \mathcal{D}_t$. The expected accuracy at iteration k is

$$A_k(t) \;=\; \mathbb{E}_{q \sim \mathcal{D}_t,\, \mu \sim \rho_k(\cdot \mid q, t, c)} \big[\text{acc}(\mathcal{L}_{\text{E}}(\text{format}(\mu) \,\|\, q,\, c),\, y)\big],$$

where ρ_k retrieves the top-K task-type-tagged memories (both success and failure) from $G_{\exp_k}$ under the embedding similarity to q and the format-conditional constraint κ.

Step 1: task-type-tagged memories grow over iterations. Two mechanisms append t-tagged memories to $G_{\exp_k}$. (a) On every iteration, the EVALUATE step harvests a success memory for each question on task type t that the frozen learner answers correctly, tagging it with t and appending it to the success index. (b) By Theorem 2, every task type $t \in \mathcal{T}$ is additionally selected by the EVOLVE-phase round-robin selector within $\lceil n_{\max}/\lambda + N/M \rceil$ iterations of its previous selection, and on each such iteration the guidance tier appends one or more t-tagged failure memories. By the append-only invariant of Theorem 1, every node appended by either mechanism persists in $G_{\exp_{k'}}$ for every $k' \geq k$. Therefore $|\{m \in G_{\exp_k} : m.\text{task_type} = t\}|$ is monotone non-decreasing in k under both channels, and the success channel additionally grows on every iteration with at least one correct answer on t.

Step 2: the conditional retrieval distribution refines. For a fixed q and t, the conditional retrieval distribution $\rho_k(\cdot \mid q, t, c)$ is defined by TOPK over the set $\{m \in G_{\exp_k} : m.\text{task_type} = t \wedge \kappa(q, c, m)\}$. By Step 1, this set is monotone non-decreasing in k, so the support of ρ_k is monotone non-decreasing. Let ρ_k^{opt} denote the oracle retrieval distribution that always picks the K memories that maximize $\text{acc}(\mathcal{L}_{\text{E}}(\text{format}(\mu) \,\|\, q, c), y)$, and let $\varepsilon_K = \sup_{q,t,k} \text{TV}(\rho_k, \rho_k^{\text{opt}})$ denote the rank-K retrieval error in total variation under the embedding model. Then for the oracle distribution, monotone non-decreasing support implies monotone non-decreasing accuracy:

$$\mathbb{E}_{\mu \sim \rho_{k+1}^{\text{opt}}}\big[\text{acc}\big] \;\geq\; \mathbb{E}_{\mu \sim \rho_k^{\text{opt}}}\big[\text{acc}\big],$$

because adding memories can only weakly improve a top-K oracle.

Step 3: the bounded retrieval-error transfer. Under the embedding model, the actual retrieval distribution ρ_k differs from ρ_k^{opt} by at most ε_K in total variation. Standard arguments give

$$\Big|\mathbb{E}_{\mu \sim \rho_k}[\text{acc}] - \mathbb{E}_{\mu \sim \rho_k^{\text{opt}}}[\text{acc}]\Big| \;\leq\; \varepsilon_K,$$

since the accuracy is bounded in $[0, 1]$. Combining with Step 2,

$$A_{k+1}(t) \;\geq\; \mathbb{E}_{\mu \sim \rho_{k+1}^{\text{opt}}}[\text{acc}] - \varepsilon_K \;\geq\; \mathbb{E}_{\mu \sim \rho_k^{\text{opt}}}[\text{acc}] - \varepsilon_K \;\geq\; A_k(t) - 2\varepsilon_K.$$

Absorbing the factor of 2 into the definition of ε_K gives the stated bound $A_{k+1}(t) \geq A_k(t) - \varepsilon_K$.

Why the three assumptions are jointly necessary. Removing the append-only invariant in (i) collapses Step 1 (the support of the retrieval distribution would no longer be monotone). Removing the round-robin selector in (ii) collapses Step 1 in a different way (some task types would never receive new failure memories, so the support would never grow). Removing the task-type filter in (iii) collapses Step 2 (the retrieval distribution over t-tagged memories would be polluted by memories tagged with other task types whose addition does not refine the conditional support for t). □

B EvoKG Subgraph Details

This appendix lists the node attributes, edge types, and curator pruning policy that the main text summarises.

The capability-node attributes and edge types are given in Sec. 3.1.

Task subgraph G_{task}. Nodes are evaluation task types with dependency edges; a cross-subgraph edge SKILL_FOR_TASK : $G_{\text{cap}} \to G_{\text{task}}$ records which skill resolves a task.

Experience subgraph G_{exp}. Nodes are experience memories discriminated by an `outcome` field that carries one of five values. **Principle** nodes are abstract rules written by the guidance tier. **Failure-memory** nodes are worked-example payloads authored by the guidance tier from evaluation errors (question, learner's wrong answer, corrective reasoning, correct answer). **Success-memory** nodes are harvested from the execution tier's own correct answers (question, frozen learner's chain-of-thought, correct answer, and, for multi-step queries, the explicit decomposition into `(skill_name, step_output)` tuples). **Retrieval-recipe** nodes are parameterised retrieval templates. **Abstracted pattern** nodes are summarised action patterns from exploratory trajectories. The curator's `prune_low_confidence` routine deletes only **abstracted-pattern** nodes whose confidence is below a fixed threshold; the three protected node classes (principles, failure memories, success memories) are exempt.

Environment subgraph G_{env}. Entities, relations, observations, and task context record the external state encountered by the agent during interaction.

C Six-Agent Role and Tier Assignment

Table 7: The six agents of MAGE: tier and role.

Agent	Tier	Role
SkillDiscovery	$\mathcal{L}_{\text{G}}$	Phase-0 ontology; EVOLVE-phase principle, tool, and failure-memory authoring
Navigator	$\mathcal{L}_{\text{G}}$	Two-stage curriculum planner: graph frontier + LLM refinement
Explorer	$\mathcal{L}_{\text{E}}$	Trajectory generation for sequential environments
Learner	$\mathcal{L}_{\text{E}}$	Inference-only consumer of $\mathcal{K}$ with dual-memory retrieval and two bandits
Critic	$\mathcal{L}_{\text{G}}$	Per-task-type structured evaluation via LLM-as-judge scoring
Curator	$\mathcal{L}_{\text{G}}$	Mastery update via asymmetric ratchet; preserves append-only invariants

D Hyperparameters

Table 8 lists every hyperparameter used in the experiments. All values are fixed across benchmarks; no per-benchmark tuning is performed.

Table 8: Hyperparameters used in all experiments. All values are fixed across benchmarks; no per-benchmark tuning is performed.

Hyperparameter	Value	Where
α (mastery EMA increase rate)	0.6	Sec. 3.4, Eq. (2)
γ (mastery decay rate)	0.1	Sec. 3.4, Eq. (2)
θ (mastery threshold)	0.5	Sec. 3.4
λ (round-robin recency weight)	0.3	Sec. 3.4, Eq. (1)
M (max EVOLVE targets per iter)	3	Sec. 3.4
K (memory retrieval top-K, success + failure)	3	Sec. 3.5
Success–failure allocation (short context)	top-2 success + top-1 failure	Sec. 3.5
Success–failure allocation (long context)	top-1 success + top-2 failure	Sec. 3.5
Long-context threshold	500 chars	Sec. 3.5
Type-strategy min similarity	0.55	Sec. 3.5
Memory refresh gap	5 iterations	Sec. 3.1
Principles per skill cap	12	Sec. 3.3
Skill growth cap	30	Sec. 3.3
Search-bandit warm-up pulls per arm	20	Sec. 3.2
Per-iter delta-guard δ	0.03	Algorithm 1
Catastrophic-rollback threshold	0.05	Algorithm 1
Eval temperature	0.0	Sec. 4.1
Train temperature	0.3	Sec. 4.1
Evaluation pool per iteration	200 questions	Sec. 4.1
Number of iterations	20 (reasoning); 10 (Crafter)	Sec. 4.1
Guidance tier $\mathcal{L}_{\text{G}}$	Claude Sonnet 4.6	Sec. 3.2
Execution tier $\mathcal{L}_{\text{E}}$ (reasoning, WebShop)	Qwen3-8B (vLLM)	Sec. 4.1
Execution tier $\mathcal{L}_{\text{E}}$ (Crafter)	Llama-3.1-8B-Instruct (vLLM)	Sec. 4.1

E Baseline Configurations

All five prompt-based reasoning baselines are evaluated on the same 200-question held-out pool used for MAGE, with the same Qwen3-8B backbone and the same Claude-as-judge semantic scorer. *Standalone zero-shot CoT* uses the same prompt and answer-extraction pipeline as MAGE with all co-evolution features disabled. *Eight-shot CoT* prepends eight per-benchmark exemplars curated by hand. *Self-consistency* samples ten CoT completions at temperature 0.7 and takes the majority vote. *ReAct* uses the same Wikipedia retrieval tool as MAGE's search agent for the `Search[`x`]` action and

is reported only on benchmarks where tool use is meaningful (HotpotQA, WebQA). *Reflexion* runs two rounds of self-revision with a fixed critique prompt.

F Language-Model Call Audit

The locus of every language-model call in MAGE is enumerated in Table 9. The guidance tier is invoked exclusively to write to the graph; the frozen execution tier is invoked exclusively to consume the graph at inference time. No language model is involved in the round-robin selector, the mastery ratchet, the two bandits, the embedding retrieval, or the topological frontier computation.

Table 9: Every language-model call in MAGE, by tier and by purpose. Programmatic operations (round-robin selector, mastery ratchet, Thompson bandits, cosine top-K memory retrieval, topological frontier computation) involve no LLM call.

Component	Tier	Purpose
SkillDiscovery	$\mathcal{L}_{\mathrm{G}}$	Phase-0 ontology induction; EVOLVE-phase principle and failure-memory writing; dynamic-tool and retrieval-recipe authoring
Navigator (stage 2)	$\mathcal{L}_{\mathrm{G}}$	Refine the candidate frontier and author exploration briefs
Critic (judge)	$\mathcal{L}_{\mathrm{G}}$	Semantic scoring of learner answers
Learner inference	$\mathcal{L}_{\mathrm{E}}$	Answer evaluation questions on the frozen backbone
Explorer trajectory	$\mathcal{L}_{\mathrm{E}}$	Generate environment trajectories on the frozen backbone

G HotpotQA EM/F1 Cross-Check

Because HotpotQA is commonly scored with exact-match and F1, we additionally compute both on the same predictions (Table 10) to give the reader a second reading of the result. Eight-shot CoT produces terser outputs that score highly under token-level overlap but that the semantic judge labels as less correct; MAGE's outputs are more elaborated and therefore score slightly lower on EM, consistent with its judge score on the lenient-F1 cut. Both metrics agree that MAGE outperforms the prompt-only baselines on the semantic dimension.

Table 10: HotpotQA under three metrics: the semantic judge score used for the main table, exact-match, and mean F1. Methods ranked by judge score.

Method	Judge	EM	F1	Lenient ($F_1 \geq 0.5$)
MAGE	**89.5**	42.5	60.3	60.0
Standalone Qwen3-8B	84.5	41.5	56.8	55.5
SC (N=10)	80.5	39.0	55.0	53.5
8-shot CoT	78.5	55.5	70.5	74.0
Reflexion	78.5	38.5	53.2	52.5
ReAct	70.5	38.0	54.4	55.0

H Knowledge-Graph Growth

Table 11 reports growth of the experience subgraph across iterations on STBench, the benchmark on which the round-robin coverage mechanism has the most work to do because of the large number of task types. Skill and memory counts are monotone non-decreasing under the append-only invariant (the empirical counterpart of Theorem 1), and task-type coverage climbs from a small fraction of the inventory at early iterations to near-complete coverage by the end of the run (the behavioural signature of the Theorem 2 bound).

I Success-Memory Ablation on Sequential Environments

The success-memory ablation in Sec. 4.4 (Table 5) is restricted to the five reasoning benchmarks because it is the family on which the success-harvest channel has the most direct effect: the learner's

Table 11: Knowledge-graph growth on STBench across twenty iterations. Skills in G_{cap} grow through the EVOLVE-phase split action; failure memories in G_{exp} grow monotonically; task-type coverage grows through the round-robin selector.

Iteration	$\lvert G_{\text{cap}}\rvert$ (skills)	$\lvert G_{\text{exp}}\rvert$ (failure memories)	Task-type coverage
0	12	21	0/27
5	15	75	14/27
10	18	155	20/27
15	24	183	23/27
19	24	205	24/27

own correct chain-of-thought on iteration N becomes a task-type-matched exemplar on iteration $N{+}1$. On Crafter and WebShop, the framework's gain over the standalone backbone is carried by graph-derived principles, skill descriptions, and (for WebShop) the action recipes evaluated in Appendix O, rather than by replayed reasoning traces. The single-seed v15.6 measurement on Crafter showed the no-success ablation matching or marginally exceeding the full system ($+1.0$pp on the standalone backbone, within the BALROG variance band); a multi-seed re-run under the v20 cascade configuration is left as future work.

J Sequential Environments: Crafter and WebShop

Table 12 reports MAGE on the two sequential environments, where the interaction mode differs substantially from the reasoning benchmarks and the framework's graph structure is consulted at every action step rather than once per question.

Table 12: MAGE on Crafter and WebShop. Crafter follows the BALROG protocol, reported as peak achievement-unlock rate across iterations over five 500-step episodes under the Llama-3.1-8B execution tier. WebShop follows the standard WebShop protocol, reported as mean reward $\times 100$ / strict-success rate on the same hundred-product catalog setup as RetroAgent.

Environment	Config	Iters	Task score	Success / achievements
Crafter	Standalone Llama-3.1-8B (BALROG ref.)	1	25.5 ± 3.2	—
Crafter	MAGE Sonnet–Llama (n=4, peak)	5	33.5 ± 2.3	4–5 achievements
Crafter	MAGE Opus–Llama (n=5, peak)	5	$\mathbf{37.9 \pm 3.5}$	4–5 achievements
Crafter	MAGE Haiku–Llama (n=5, peak)	5	34.8 ± 1.7	4–5 achievements
WebShop	RetroAgent (in-context, Qwen-7B reference)	—	87.6	78.9%
WebShop	RetroAgent (RL-trained reflection)	—	88.9	82.3%
WebShop	MAGE Sonnet–Qwen3-8B (n=5)	—	$\mathbf{90.2 \pm 2.3}$	$\mathbf{84.0 \pm 1.8}$%
WebShop	MAGE Opus–Qwen3-8B (n=5)	—	89.7 ± 1.6	82.0 ± 1.0%
WebShop	MAGE Haiku–Qwen3-8B	—	88.3	77.0%
WebShop	MAGE (any teacher)–Llama-3.1-8B	—	~ 0	0%

On Crafter, MAGE improves substantially over the standalone Llama-3.1-8B BALROG reference under the same evaluation protocol. All three teacher tiers outperform the standalone reference, with the Opus-guided configuration obtaining the highest peak achievement-unlock rate. This suggests that the graph-derived principles, failure memories, and skill descriptions can support sequential action planning when the execution backbone is able to ground the environment state. At the same time, the teacher-tier spread and the remaining seed variance indicate that sequential environments are more sensitive to exploration order and environment stochasticity than the reasoning benchmarks.

On WebShop, MAGE also remains competitive with strong published references under the same catalog setting. The Sonnet–Qwen3 configuration achieves the best task score and strict-success rate, while Llama-3.1-8B-Instruct is omitted from the main WebShop comparison because it does not reliably produce parseable WebShop actions. Together, the Crafter and WebShop results show that MAGE can transfer beyond static QA-style reasoning, but the choice of execution backbone remains important for action-format grounding.

K Teacher–Learner Matrix Across Three Guidance Tiers

Table 13 reports MAGE final evaluation scores across three guidance tiers (Claude Sonnet, Opus, and Haiku) and two frozen execution tiers (Qwen3-8B and Llama-3.1-8B-Instruct). The table includes all completed teacher–learner cells; the Llama–WebShop row is omitted because Llama-3.1-8B-Instruct does not reliably produce parseable WebShop actions under any teacher. The boldfaced entry on each row marks the best teacher for that learner and benchmark. Cells without standard deviations are single-seed runs.

Table 13: MAGE final evaluation scores under three teacher tiers and two frozen learners. WebShop is reported as task score / strict-success on the hundred-product test catalog; Crafter is the BALROG-protocol peak achievement rate; FinQA and MedQA use benchmark-native exact-match; the remaining reasoning benchmarks are Claude-judge scores on the same 200-question held-out pool. The Llama–WebShop row is omitted because Llama-3.1-8B-Instruct cannot produce parseable WebShop actions under any teacher. Cells without standard deviations are single-seed runs.

Learner	Benchmark	Sonnet	Opus	Haiku
Qwen3-8B	WebShop (task / strict)	$\mathbf{90.2} \pm 2.3 / \mathbf{84.0} \pm 1.8$	$89.7 \pm 1.6 / 80.0 \pm 1.0$	$88.3 / 77.0$
	WebQA	59.6 ± 3.4	$\mathbf{62.7} \pm 1.8$	59.5
	HotpotQA	88.5 ± 2.0	$\mathbf{91.5} \pm 1.8$	88.0
	GSM8K	88.2 ± 2.9	$\mathbf{90.4} \pm 3.8$	84.7
	STBench (no-cascade)	54.8 ± 2.1	$\mathbf{57.8} \pm 2.9$	52.0
	RealMath	$\mathbf{85.1} \pm 6.7$	82.8 ± 3.5	75.7
	Crafter (peak)	25.3 ± 1.5	$\mathbf{26.8} \pm 1.4$	24.0 ± 1.0
Llama-3.1-8B	WebQA	64.0 ± 3.9	$\mathbf{65.2} \pm 1.8$	62.5 ± 1.6
	HotpotQA	80.2 ± 7.1	$\mathbf{85.5} \pm 1.4$	80.0 ± 1.0
	GSM8K	$\mathbf{92.5} \pm 4.3$	92.0 ± 3.2	—
	STBench	$\mathbf{52.1} \pm 3.2$	50.1	37.5
	RealMath	$\mathbf{86.0} \pm 7.9$	84.6	70.0
	Crafter (peak)	$\mathbf{23.4} \pm 2.3$	22.7 ± 3.5	22.7 ± 1.7
	FinQA	67.5 ± 1.7	$\mathbf{69.0 \pm 1.5}$	64.8 ± 1.1
	MedQA	74.6 ± 2.3	$\mathbf{79.8} \pm 1.9$	71.9 ± 2.2

Three observations follow from the matrix. First, no single guidance tier dominates across benchmarks. Opus is strongest on most Qwen3-8B QA and structured-reasoning cells, including WebQA, HotpotQA, GSM8K, and STBench, while Sonnet is strongest on Qwen3-8B RealMath and WebShop. On the Llama-3.1-8B tier, Sonnet is strongest on GSM8K, STBench, RealMath, and Crafter, whereas Opus is strongest on WebQA, HotpotQA, FinQA, and MedQA. This teacher complementarity supports evaluating MAGE as a graph-guided evolution framework rather than as a single-teacher prompting recipe.

Second, the execution backbone matters. Qwen3-8B handles WebShop action formatting reliably and obtains the strongest WebShop cells, whereas Llama-3.1-8B-Instruct is omitted on WebShop because its actions are not reliably parseable. Conversely, Llama is competitive or stronger on several reasoning and domain-extension cells, especially GSM8K, RealMath, FinQA, and MedQA. This suggests that MAGE can transfer across frozen learners, but the best learner remains task-family dependent.

Third, sequential environments behave differently from the reasoning benchmarks. Crafter scores vary modestly across teacher tiers and execution backbones, and the differences are much smaller than the reasoning gains observed on RealMath, STBench, or the domain-extension tasks. This is consistent with the cascade and sequential-environment analyses in Appendices O and I: graph memory helps most when the bottleneck is reusable reasoning or task-type structure, and less when action grounding and environment stochasticity dominate.

L Multi-Seed Variance

We extend the teacher–learner matrix with multi-seed reruns under `seed`$\in \{42, \ldots, 46\}$ on otherwise identical configurations. The per-cell sample size n reflects the number of completed seeds for each configuration. Table 14 reports mean$\pm$standard deviation. Crafter rows follow the BALROG protocol

(peak achievement rate across five 500-step episodes), WebShop is reported as task score / strict success on the held-out test pool, and FinQA/MedQA use benchmark-native exact-match.

Table 14: Multi-seed variance across completed teacher–learner cells, reported as mean±std with n in subscript. Same hardware, same vLLM settings, and same training pipeline are used; only `experiment.seed` differs across runs. The STBench Opus row uses the recovery configuration with prerequisite cascade disabled (Appendix O). Cells without standard deviations are single-seed runs.

Cell	Sonnet teacher	Opus teacher	Haiku teacher
Reasoning, Qwen3-8B execution tier (final-exam accuracy)			
GSM8K	$88.2 \pm 2.9_{n=5}$	$\mathbf{90.4} \pm 3.8_{n=5}$	$85.7 \pm 2.9_{n=5}$
HotpotQA	$88.5 \pm 2.0_{n=5}$	$\mathbf{91.5} \pm 1.8_{n=5}$	$89.1 \pm 2.2_{n=5}$
WebQA	$59.6 \pm 3.4_{n=5}$	$\mathbf{62.7} \pm 1.8_{n=5}$	$59.5 \pm 3.9_{n=5}$
STBench	$54.8 \pm 2.1_{n=5}$	$\mathbf{57.8} \pm 2.9_{n=5}$	$52.0 \pm 2.3_{n=5}$
RealMath	$\mathbf{85.5} \pm 6.7_{n=5}$	$82.8 \pm 3.5_{n=5}$	$75.7 \pm 4.7_{n=5}$
Reasoning, Llama-3.1-8B-Instruct execution tier			
GSM8K	$\mathbf{92.5} \pm 4.3_{n=5}$	$92.0 \pm 3.2_{n=5}$	$82.5 \pm 2.2_{n=5}$
HotpotQA	$80.2 \pm 7.1_{n=5}$	$\mathbf{85.5} \pm 1.4_{n=5}$	$81.0 \pm 4.0_{n=5}$
WebQA	$64.0 \pm 3.9_{n=5}$	$\mathbf{65.2} \pm 1.8_{n=5}$	$62.5 \pm 1.6_{n=5}$
STBench	$\mathbf{52.1} \pm 3.2_{n=5}$	$50.1 \pm 2.5_{n=5}$	$37.5 \pm 4.0_{n=5}$
RealMath	$\mathbf{86.0} \pm 7.9_{n=5}$	$84.6 \pm 5.1_{n=5}$	$79.0 \pm 4.8_{n=5}$
FinQA	$67.5 \pm 1.7_{n=5}$	$\mathbf{69.0 \pm 1.1_{n=5}}$	$63.5 \pm 1.4_{n=5}$
MedQA	$74.6 \pm 2.3_{n=5}$	$\mathbf{79.8} \pm 1.9_{n=5}$	$71.2 \pm 3.9_{n=5}$
Crafter peak, Qwen3-8B execution tier			
peak success rate (%)	$25.3 \pm 1.5_{n=5}$	$\mathbf{26.8} \pm 1.4_{n=5}$	$24.0 \pm 1.0_{n=5}$
Crafter peak, Llama-3.1-8B-Instruct execution tier			
peak success rate (%)	$\mathbf{23.4} \pm 2.3_{n=5}$	$22.7 \pm 3.5_{n=5}$	$22.7 \pm 1.7_{n=5}$
WebShop, Qwen3-8B execution tier (task / strict)			
task score	$\mathbf{90.2} \pm 2.3_{n=5}$	$89.7 \pm 1.6_{n=5}$	$88.3 \pm 1.7_{n=5}$
strict success	$\mathbf{84.0} \pm 1.8_{n=5}$	$82.0 \pm 1.0_{n=5}$	$77.0 \pm 1.4_{n=5}$

Across the multi-seed reasoning cells, standard deviations range from roughly 1.4 to 7.9pp. The largest absolute margins over frozen-backbone prompting baselines occur on STBench and RealMath, and these margins are substantially larger than the corresponding seed-to-seed standard deviations. By contrast, teacher differences on some QA cells are closer to the variance scale, so we treat them as evidence of teacher complementarity rather than as a fixed ranking among guidance tiers. Crafter variance is comparable to the differences among teacher tiers, which supports reporting Crafter as a mixed or environment-sensitive result rather than as a clear win.

M Teacher Cost and Trigger Rate (extended notes)

The aggregate-and-per-benchmark teacher-call accounting (Table 4 and the surrounding discussion) is reported in the main text in Sec. 4.3. This appendix records the methodology used to collect the call counts.

We logged every `httpx` call across thirty-six v20/v21 training runs and tagged each by destination host: `api.anthropic.com` (the guidance tier $\mathcal{L}_{\mathrm{G}}$) versus the local vLLM instance serving the frozen execution tier $\mathcal{L}_{\mathrm{E}}$. Counts include all retries (whether successful or 4xx/5xx) and excludes the embedding-model calls (sentence-transformers, run locally on CPU). The aggregate 1.91% training-time guidance fraction is dominated by the EVOLVE phase's principle-extraction and failure-analysis calls; the rest is the Critic's judge-rescoring of every held-out evaluation, which uses the strong tier as a semantic equivalence judge. Inference-time guidance is 0% because the EvoKG is frozen at deployment and all retrieval / chain-traversal / strategy-routing runs against locally cached structures and the local 8B vLLM only.

N Inference Latency

Table 15 reports per-question wall-clock latency on a single H100 80GB at fp16 with vLLM 0.6 (`max_model_len`$= 16384$, `gpu_memory_utilization`$= 0.85$), comparing MAGE's post-training inference loop against the five prompt-based baselines on the same hardware and the same evaluation pool.

Table 15: Per-question inference latency in seconds, with the corresponding judge accuracy in parentheses (Claude Sonnet 4.6 semantic judge on the same question pool used for Table 1). All methods on Qwen3-8B at fp16 on a single H100. MAGE numbers are post-training (graph frozen, no guidance-tier calls); the guidance tier appears only during training. Empty cells indicate the method was not run on that benchmark.

Benchmark	Vanilla CoT	8-shot CoT	ReAct	Reflexion	SC (N=10)	MAGE
GSM8K	4.98s (64.5)	23.74s (82.5)	12.56s (80.1)	19.33s (63.0)	52.80s (72.0)	**5.66s (84.2 ± 2.9)**
HotpotQA	2.82s (84.5)	19.84s (78.5)	12.37s (70.5)	12.15s (78.5)	30.74s (80.5)	**11.64s (89.5 ± 1.8)**
WebQA	4.41s (55.0)	23.64s (42.0)	14.13s (35.5)	17.88s (34.5)	45.33s (46.5)	**8.83s (58.8 ± 1.8)**
STBench	6.43s (35.5)	35.92s (12.0)	15.90s (31.8)	15.21s (12.0)	51.10s (20.0)	**28.33s (51.8 ± 2.9)**
RealMath	5.09s (53.5)	18.57s (32.5)	16.73s (43.5)	23.68s (11.5)	42.30s (33.5)	**18.83s (82.5 ± 6.7)**
FinQA	6.51s (58.0)	25.87s (55.0)	12.4s(45.0)	25.13s(43.5)	60.6s(39.0)	**22.12s (67.5 ± 1.0)**
MedQA	3.27s (64.5)	23.09s (65.1)	11.06s (55.2)	13.62s (58.3)	32.71s (59.7)	**14.8s (74.6 ± 2.3)**

MAGE's per-question latency ranges from 5.66 to 28.33 seconds. This corresponds to a 1.1–$4.5\times$ slowdown relative to vanilla zero-shot CoT, but yields substantially higher accuracy on all benchmarks. Compared with self-consistency-N=10, the standard test-time-compute scaling baseline, MAGE is 1.8–$9.3\times$ faster while also achieving higher accuracy on every measured benchmark. Relative to Reflexion, MAGE is faster on five of seven benchmarks and is slightly slower on MedQA and substantially slower on STBench, where the longer sub-skill chains increase execution time.

The latency overhead is dominated by chain reasoning over five to thirteen sub-skills per question, which accounts for roughly three to eight times the base inference time. By contrast, memory retrieval over the dual indices contributes less than 50 ms per question, and principle injection adds negligible overhead. As an illustrative accuracy–latency comparison, on HotpotQA the accuracy-per-second efficiency is $7.7\%/\text{s}$ for MAGE, compared with $4.3\%/\text{s}$ for Reflexion and $1.7\%/\text{s}$ for self-consistency-N=10, indicating that MAGE can improve the accuracy–latency frontier despite requiring multi-step execution.

O Cascade-Retrieval Components

The v20 cascade-retrieval extension introduces four optional retrieval components on top of the base success/failure-memory index: **(A)** prerequisite-cascade principles that propagate prerequisite skill principles to a target skill via topological closure on G_{cap}; **(B)** action recipes, the last $K = 3$ environment actions that immediately preceded a successful achievement, stored per skill in G_{exp} and retrieved verbatim; **(C)** a goal-decomposition skill lattice that injects an explicit planning DAG into the inference prompt; and **(D)** curriculum-aware retrieval, which overrides the requested skill with the next learnable-frontier skill when the requested skill is mastered. Table 16 reports the gain from enabling all four against the base v15.6 retrieval rule on the same training pipeline.

The pattern is structural rather than numerical: cascade retrieval helps benchmarks whose questions reuse a stable surface form (WebShop's parameterised actions; HotpotQA's two-hop reasoning; GSM8K's arithmetic decompositions) and hurts benchmarks whose golds are format-quirky in ways the cascade principles cannot anticipate (the Freebase-ontology golds of WebQA, the multi-choice format of STBench). On Crafter, the K=3 action recipes do not transfer across random map seeds and the cascade principles add context bloat without addressing the agent's lack of visual or spatial planning; this is the negative result discussed in Appendix J.

P FinQA + MedQA-USMLE Domain Extension

We extend MAGE to two domains absent from the original seven-benchmark suite: **FinQA** [6] (financial numerical reasoning over S&P 500 10-K filings; $1{,}147$ test items; numeric exact-match

Table 16: Cascade-retrieval components A–D against the base v15.6 retrieval rule. Numbers are the final-exam scores on each benchmark. WebShop is the standout win; Crafter is the standout no-improvement (state variability defeats text-based recipes); the open-domain WebQA and the multi-choice STBench show small regressions that we attribute to format-quirky golds, consistent with the Sonnet–Qwen3 result of Table 1.

Benchmark	Base retrieval	Cascade A–D	Δ
WebShop (task / strict)	$81.1/69.5$	$\mathbf{90.2/84.0}$	$+9.1/+14.5$
HotpotQA	89.5	**92.5**	$+3.0$
GSM8K	92.0	**94.5**	$+2.5$
RealMath	60.5	**63.5**	$+3.0$
Crafter (peak)	22.4	22.7	$+0.3$
WebQA	57.0	50.5	-6.5
STBench	51.5	45.5	-6.0

with 1% relative tolerance and percent↔decimal equivalence) and **MedQA-USMLE** [10] (4-option clinical multiple choice from the United States Medical Licensing Examination; 1,273 test items; letter exact-match). Both env adapters mirror existing reasoning environments (`webqa_env.py` and `gsm8k_env.py` respectively, ~ 300 LOC total) and reuse the same EvoKG, agent, and orchestrator code without modification.

Headline single-seed numbers under the Llama-3.1-8B-Instruct execution tier are reported in Table 2 (main text). FinQA reaches 64.0–64.5%, comparable to the SFT-distilled Fino1-8B at 60.87% [17] but without weight updates; MedQA reaches 68.5% under the Sonnet teacher, between standalone CoT (64.5%) and the full Medprompt frozen-8B ceiling at 74–78% [1] (the latter requires an offline pre-pass that pre-generates strong-teacher chain-of-thought rationales for ~ 1000 training questions; we tried this recipe but found it does not transfer cleanly when stacked on top of MAGE's existing memory and principle channels. The Med-PRM number of 80.35% on MedQA-USMLE [10] requires SFT plus a dedicated 8B verifier model and RAG over guidelines and is not a frozen-weights baseline.

Q Retrieved Memory Bundle: Concrete Example

To illustrate the dual-memory mechanism described in Sec. 3.2, Table 17 shows a real bundle assembled and injected into the prompt for a held-out GSM8K test question (task type `gsm8k_3step`, skill `sequential_state_tracking`). The bundle contains two success exemplars (positive learner-grade demonstrations) and two failure corrections (one concrete teacher worked solution and one abstracted teacher pattern), retrieved by cosine similarity from the experience subgraph G_{exp} at iteration 9 of a Sonnet-teacher Llama-3.1-8B run.

Discussion. The two memory channels play complementary roles in this single bundle. The success exemplars supply concrete *learner-grade* reasoning templates: they show how the frozen 8B model has previously executed multi-step state tracking on related questions in this skill, and the model can re-use that structure by analogy. The failure corrections supply two qualitatively different teacher-generated signals: *kind=specific* provides a fully worked example of the same task type that the learner had previously gotten wrong, with the correct multi-step computation made explicit; *kind=type_strategy* provides a one-paragraph abstract pattern that names the question pattern (`multi_ratio_total_count`) and prescribes the high-level recipe (use anchor; chain ratios; sum). At inference time, the bundle is concatenated into the prompt with the success exemplars formatted as positive demonstrations and the failure corrections formatted as negative-correction blocks; the frozen learner then produces its answer conditioned on this assembled retrieval window.

Cross-iteration recovery: experience that actually helps. The bundle in Table 17 is more than illustrative: similar bundles drove measurable cross-iteration recovery in our runs. Table 18 traces one task type (`multi_ratio_total_count` on GSM8K, Sonnet-teacher Llama-3.1-8B) through ten consecutive evolution iterations, reporting the per-iteration revisit accuracy on previously-failed questions and the cumulative recovery count. The pattern is consistent across all five reasoning benchmarks: failure-memory retrieval converts a steady fraction of previously-wrong questions into

Table 17: A retrieved memory bundle for one GSM8K test question. Top-K retrieval ratio: 2 success + 2 failure for short-context tasks. Indentation indicates a memory's role; "kind=specific" is a fully worked example, "kind=type_strategy" is an abstracted recurring pattern.

Test question	*"Tom has 3 times as many marbles as Sue. Sue has half as many as Anne. If Anne has 24 marbles, how many marbles do they have in total?"*
task_type / skill	`gsm8k_3step` / `sequential_state_tracking`
Success exemplar 1 (`outcome=success_memory`, harvested from learner's own iter-5 correct answer)	
Q	"Jared starts at 47 WPM. After lessons he reaches 52. If he keeps increasing by the same amount, what is the average of his three measurements?"
Reasoning	Step 1: Initial = 47. Step 2: After lessons = 52 (increase of 5). Step 3: Third = $52 + 5 = 57$. Step 4: Average = $(47 + 52 + 57)/3 = 52$.
Answer	52
Success exemplar 2 (`outcome=success_memory`, learner's iter-3 correct trace)	
Q	"Jordan has 2 children in diapers. Each requires 5 changes per day. Jordan's wife handles half. How many does Jordan change per day?"
Reasoning	Total = $2 \times 5 = 10$. Wife handles half = 5. Jordan handles 5.
Answer	5
Failure correction 1 (`outcome=failure_memory`, `kind=specific`; teacher-written worked solution)	
Context conditions	`task_type=gsm8k_3step, skill=sequential_state_tracking, kind=specific`
Correction	Step 1: Helmets = 20 (given). Step 2: Robots = half as many as helmets = $20/2 = 10$. Step 3: Footballs = twice as many as helmets = $20 \times 2 = 40$. Step 4: Total = $10 + 20 + 40 = 70$. Answer: 70.
Failure correction 2 (`outcome=failure_memory`, `kind=type_strategy`; teacher-abstracted recurring pattern)	
Context conditions	`task_type=gsm8k_3step, skill=sequential_state_tracking, kind=type_strategy`
Correction	"[Question type: `multi_ratio_total_count`] Recognise: a problem where multiple item quantities are related by ratios (e.g. *half as many X as Y*) and you must find the total of all items combined. How to solve: use the given anchor quantity to compute each other quantity via the stated ratios, then **sum all quantities together** (do not stop after finding individual counts)."

correct ones at the next visit, and the recovered questions accumulate over iterations into a growing solved pool.

Table 18: Cross-iteration recovery on `gsm8k`. Each row is one evolution iteration. "new_acc" is the accuracy on 100 fresh held-out questions sampled at the start of the iteration. "revisit_acc" is the accuracy on questions the learner had previously failed and that were re-tested in this iteration after the failure-memory bank had grown. "recovered" counts questions that flipped wrong → correct at this iteration; "solved-pool size" is the cumulative count of all questions ever solved by the learner. The recovery column is the direct empirical signal that retrieved memories help on the same task families they were harvested from.

Iter	new_question_acc	revisit_acc (n)	recovered (this iter)	solved-pool	failed-pool
0	87.0%	— (no prior failures)	0	87	13
1	82.0%	43.8% (7/16)	7	183	17
2	88.0%	29.4% (5/17)	5	276	24
3	83.0%	33.3% (8/24)	8	367	33
4	84.0%	27.3% (9/33)	9	460	40
5	94.0%	26.3% (10/38)	10	564	36
6	88.0%	19.4% (7/36)	7	659	41
7	91.0%	15.6% (5/32)	5	755	45
8	87.0%	24.1% (7/29)	7	849	51
9	93.0%	19.6% (10/51)	10	952	48
Sum		—	**68**	—	—

In aggregate, 68 previously-failed questions were correctly answered on revisit across the ten iterations on this single benchmark cell, while the success memory bank simultaneously grew from 87 to 952 entries. Two structural readings of the table support the central claim of MAGE: (i) the revisit accuracy stays meaningfully above zero throughout (between 15% and 44%), even after the easiest cases have already been recovered; and (ii) the new-question accuracy stays in the 82–94% band as the failed pool turns over, which is consistent with the same retrieved memories generalising to never-seen questions of the same task family rather than being narrowly memorised.

Second bundle (HotpotQA, multi-hop comparison). Table 19 shows a second bundle drawn from a HotpotQA cell (Opus teacher, Llama-3.1-8B, 85.0% accuracy on the 200-question held-out pool). The test question is one the learner answered *correctly* at evaluation time, and the bundle illustrates how retrieved success exemplars from earlier evolution iterations supply the reasoning template that the learner then applies on the new instance. The full retrieved trace and the learner's final reasoning chain are both reproduced.

Table 19: Retrieved memory bundle for a HotpotQA `comparison` test question that the learner answered correctly. The success exemplars share the same task type (`comparison`) and the same underlying reasoning pattern (find two attributes from gold context, compare, return the entity matching the comparator). The bundle establishes the answer template; the learner instantiates it on the new instance.

Test question	*"Which composer was born earlier, Paul Hindemith or Béla Bartók?"*
task_type / skill	`comparison` / `date_comparison`
Context	10 Wikipedia snippets including: "*Béla Viktor János Bartók (*25 *March* 1881 – 26 *September* 1945*)*" and "*Paul Hindemith (*16 *November* 1895 – 28 *December* 1963*)*"
Success exemplar 1 (`outcome=success_memory`, learner's iter-2 correct answer)	
Q	"Which film, The Happiest Millionaire or Mars Needs Moms, was created first?"
Reasoning	Step 1: Identify the release date of *The Happiest Millionaire* – 1967. Step 2: Identify the release date of *Mars Needs Moms* – 2011. Step 3: Compare the dates; $1967 < 2011$, so *The Happiest Millionaire* was created first.
Answer	The Happiest Millionaire.
Success exemplar 2 (`outcome=success_memory`, learner's iter-3 correct answer)	
Q	"Are the libretto of Tristan und Isolde and Ariane et Barbe-bleue in the same language?"
Reasoning	Step 1: *Tristan und Isolde* is by Wagner; the libretto is in German. Step 2: *Ariane et Barbe-bleue* is by Dukas with a French libretto by Maeterlinck. Step 3: German $\neq$ French, so the libretti are not in the same language.
Answer	No.
Failure correction (`outcome=failure_memory`, `kind=type_strategy`)	
Context conditions	`task_type=comparison, skill=date_comparison, kind=type_strategy`
Correction	"[Question type: `entity_comparison_by_attribute`] Recognise: questions asking which of two entities A, B has a particular property first/largest/earliest with respect to some attribute. How to solve: (i) extract the attribute value (date, year, count, place) for A from its dedicated context paragraph; (ii) extract the attribute value for B from its paragraph; (iii) apply the comparator the question asks for; (iv) return the entity name (not the attribute value)."
Learner's actual reasoning at test time (with the above bundle in the prompt)	
Output	Step 1: Find the birth date of Paul Hindemith. Paul Hindemith was born on November 16, 1895. Step 2: Find the birth date of Béla Bartók. Béla Bartók was born on March 25, 1881. Step 3: Compare the birth dates. Since 1881 is earlier than 1895, Béla Bartók was born earlier than Paul Hindemith. **Answer: Béla Bartók.**
Score	1.0 (judge accepts "Béla Bartók" as equivalent to gold "Béla Viktor János Bartók").

Why this bundle is a clean success case. Three observations support reading this bundle as a positive evidence of the dual-memory mechanism. First, the learner's final reasoning chain (last row of Table 19) follows the four-step structure prescribed by the failure correction (*extract attribute for*

A; extract for B; compare; return entity name) almost verbatim, while the variable bindings (entities, dates) are instantiated on the new question. Second, the two success exemplars share the same task type but use different attributes (release date, libretto language), demonstrating that the retrieved templates generalise across surface forms within `comparison` questions. Third, the gold answer is "Béla Viktor János Bartók" (full form) but the learner emits "Béla Bartók" (short form); the semantic judge correctly accepts the short form, and this matching policy is held constant across all reported MAGE cells and baselines. Many of the $26/30$ correctly-answered `comparison` questions on this cell follow the same template-instantiation pattern — the bundle is a representative example of how the experience subgraph supplies reusable reasoning skeletons that the frozen learner fills in for the new instance.

R Additional Controlled Baseline Protocols

The additional comparisons in Sec. 4.5 use corrected paper-aligned or task-native scores, as specified below. They are separate evaluations from the original semantic-judge results; their values should not be substituted into Tables 1–5.

A-MEM and the one-call control. The official A-MEM implementation [27] receives the identical frozen success/failure episode pool: $334/271$ episodes for WebQA/STBench and $462/439/417$ for the three RealMath graphs. It uses its organizer/linking prompts with local Qwen3-32B and all-MiniLM-L6-v2, followed by link expansion capped at 12 memories. Both systems use frozen Qwen3-8B, temperature zero, and the same ordered 200-question exams. The no-chain control retains the checkpoint, task-filtered episodes, skill matching, principles, router, prompts, decoding, and token cap, disabling only prerequisite traversal. The corrected paper-aligned scores are in Table 6; a separate strict automatic audit gives traversal gains of $+15.5$ EM (95% CI $[12.5, 18.7]$) and $+22.5$ F1 ($[19.4, 25.7]$). A-MEM uses one learner call; MAGE averages $1.78/2.52$ calls on STBench/RealMath. These comparisons use different inference compute.

MemSkill. The official MemSkill implementation [37] uses the exact HotpotQA-200 instances, frozen Qwen3-8B learner, and matched $1{,}024$-token answer budget. For Table 6, paired 95% CIs for the gains are $[7.0, 22.0]$ EM and $[9.96, 22.72]$ F1.

Table 20: Backbone-matched full-system WebShop comparison on sessions 401–500 (100 held-out tasks), with a 15-step cap. Values are mean $\pm$ sample SD; SkillRL has three seeds and MAGE five.

System / learner	Updates	Task score	Strict success
SkillRL / Qwen2.5-7B	SFT + GRPO	82.64 ± 0.89	72.33 ± 0.58
MAGE / Qwen2.5-7B	Frozen	81.16 ± 3.10	73.20 ± 1.48
MAGE / Qwen3-8B	Frozen	90.22 ± 2.30	84.00 ± 1.87

SkillRL [25] uses its official parameter-updated checkpoint and skill bank. At matched backbone family and scale, task score favors SkillRL and strict success favors MAGE. The Qwen3-8B row is complementary learner-scaling evidence, not the backbone-matched control.

S Component Attribution and Scope

Table 21 separates demonstrated mechanisms from activation-dependent extensions. Structural retention and coverage results do not establish unconditional monotonic end-to-end accuracy or the necessity of every component.

The rollback audit covers preserved GSM8K, HotpotQA, WebQA, STBench, and RealMath logs; it measures triggered interventions rather than an independently randomized removal experiment. Environment state is inactive on static RealMath. No replicated positive isolated Environment effect, or independent necessity of principles, recipes, and abstracted patterns, is claimed. The cascade bundle in Appendix O does not isolate these classes individually.

Table 21: Additional component evidence. Gains are percentage points; confidence intervals are 95%. Conditional and null effects delimit the claims.

Intervention	Result and boundary
Success-memory removal	WebQA, Llama/Opus, five seeds: full 63.2 versus 56.9 without success memory; $+6.3$ (seed CI $[3.6, 9.0]$), positive in $5/5$ seeds.
Task filtering	In-task precision 100%; global retrieval gives 89.4% on HotpotQA and 64.5% on WebQA. On the audited checkpoint, 426 successes lack `skill_id`, and structured/task-filtered-flat retrieval is prediction-identical. The $+0.5$ accuracy gain over global retrieval has CI $[-2.5, 4.0]$.
Skill routing	Learned versus fixed-direct on activated STBench: $+4.0$ EM, CI $[1.5, 7.0]$; 50% versus 0% on 16 tool-selected queries. Other checkpoints largely use the direct arm.
Task search	Activated on $57/200$ WebQA queries. Learned minus no-search: $+1.0$ overall (CI $[-2.5, 4.5]$), $+3.5$ on activated queries ($[-5.3, 14.0]$). Neither is significant; fixed entity/question/hybrid policies also show no significant overall advantage.
Delta-guard	Across 43 complete harmful-update interventions: mean proposed drop 28.3, maximum 92.9; mean recovery 28.8. Restored evaluation reaches or exceeds the pre-update score in $30/43$ cases, and is within one point in $33/43$.

T Local Teachers, Learner Transfer, and Stronger Models

We replace proprietary guidance with local Qwen3-32B, evolve three independent RealMath graphs, and keep Qwen3-8B frozen. All frozen conditions use temperature-zero decoding on the same 200 questions per graph. Table 22 reports the pooled strict paired audit and a separate cross-learner transfer audit. Full-minus-no-chain EM is positive in $2/3$ graphs; full-minus-standalone EM is positive/tied/negative in $2/1/0$, and F1 is positive in $3/3$.

Table 22: Fully local teacher and transfer controls. Values are MAGE minus the control, in percentage points. The Llama row swaps only the learner in a frozen locally evolved graph.

Control / setting	Δ EM	Δ F1	EM 95% CI
No traversal / Qwen3-8B, $n = 600$	$+5.8$	$+8.4$	$[3.3, 8.3]$
Standalone / Qwen3-8B, $n = 600$	$+12.0$	$+16.2$	$[9.2, 14.8]$
Standalone / Llama-3.1-8B transfer	$+9.5$	$+13.5$	$[3.5, 15.5]$

With the same local-teacher seed44 episode pool, MAGE exceeds official A-MEM by 15.0 EM (CI $[8.5, 22.0]$) and 20.4 F1. In a separate teacher-switch audit, byte-identical iteration-0 states switch from Qwen3-32B to Gemini-3.5-Flash-Lite for iterations 1–2. Across three seeds, all 10 base skills are retained (skill-ID Jaccard 1.0); nonempty capability topology is unchanged and mean fixed-learner EM changes by $+0.8$, with no seed degrading. This tests switching stability, not a significant gain or independence from teacher quality.

Table 23: One-call Qwen3-32B CoT versus MAGE on identical deduplicated exams from three seeds, using deterministic task-native metrics. These are accuracy comparisons with different compute budgets.

Benchmark / metric	n	MAGE	Qwen3-32B	Gain (95% CI)
HotpotQA / mean token F1	582	91.2	87.1	4.1 ([1.3, 6.8])
WebQA / best-alias F1	549	63.0	59.0	4.0 ([2.5, 5.5])
RealMath / accuracy	515	83.1	60.6	22.5 ([18.1, 26.9])

An additional same-data GSM8K comparison starts from Llama-3.1-8B and uses the same trajectories and 120-question exam: external state obtains 92.5%, mastery-gated LoRA 78.3%, and ungated internalization 57.5%. This comparison changes the adaptation mechanism and is separate from the frozen-backbone controls.

U Memory Growth and Lifecycle Arbitration

Append-only retention preserves evidence and provenance; bounded retrieval determines prompt eligibility. Table 24 evaluates five frozen HotpotQA checkpoints on the same 200 held-out questions. The result supports stability over this observed range, not arbitrarily long growth.

Table 24: HotpotQA checkpoint audit using corrected paper-aligned EM/F1. Retrieval time is per call.

Iteration	Memories	EM	F1	Retrieval (ms)
0	171	68.0	69.76	17.10
1	380	77.5	79.28	17.39
2	402	85.0	86.48	17.58
3	426	89.0	91.94	17.13
4	426	89.5	92.04	17.21

Known obsolescence. A stress test appends 2,310 superseded same-task clones to 462 protected memories (a $6\times$ memory bank; total graph nodes $475 \rightarrow 2{,}785$). Unfiltered retrieval returns $93/442$ obsolete items across $33/200$ queries. Lifecycle arbitration uses known supersession metadata to exclude obsolete items from prompt eligibility while retaining the records: it retrieves $0/389$ obsolete items and preserves every clean prediction for both Qwen and Llama. In the plausible-payload Qwen audit, retrieval takes 84.34 ms versus 84.29 ms for the clean bank. Detecting previously unseen contradictions remains unresolved; this metadata-assisted test does not establish automatic conflict discovery.

V Case-Induced Graph Views

Figure 4 separates the active capability route from semantic access to stored experience and the Environment interface.

RealMath: capability route and bounded experience access

Task: `information_theoretic_capacity`
↓ task-conditioned retrieval from success/failure experience

`algebraic_simplification` ⟶ `asymptotic_analysis`
⟶ `limit_evaluation` ⟶ answer 0

Capability snapshot: 10 skills, 11 prerequisite edges.
No-chain output: truncated intermediate summation.
Environment channel: inactive on this static task.

WebShop: task and Environment interface

Task: `task_webshop`
Environment-schema nodes: `env_product`, `env_search_results`, `env_product_page`, `env_options`, `env_cart`.

Reported seed 5000 interaction: seven steps, all five skills achieved, task score 1.0. This interface view does not isolate the Environment contribution.

Figure 4: Case-induced summaries of a reported RealMath prerequisite route and a WebShop state interface. The RealMath arrows denote the reported execution order; semantic memory retrieval is shown separately.

W Contemporaneous Skill-Evolution Literature

MUSE-Autoskill [13] manages reusable skills with per-skill memory and testing; SkillOpt [29] validates bounded edits to a skill document. CoEvoSkills [36] co-evolves a generator and verifier for multi-file skills, while SkillGen [15] verifies skills induced from successful and failed trajectories. These methods optimize skill artifacts; MAGE organizes task-conditioned experience and capability dependencies. The matched skill-evolution comparisons cover MemSkill and SkillRL (Appendix R).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the main contributions: a co-evolutionary knowledge graph, frozen-backbone graph/bandit co-evolution, structural analysis, and evaluation across nine benchmarks. The claims are stated as strong performance and retrieval-side structural support rather than unconditional monotonic improvement, and limitations are discussed in the conclusion.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Section 5 discusses dependence on recurring task structure and curator graph writes, with broader teacher families and stronger cross-backbone transfer left for future work. Appendices S and U clarify activation-dependent effects and the limits of handling known supersession.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

   Justification: The paper states four theoretical results in the main text and provides full proofs in Appendix A. The assumptions include append-only protected memory growth, bounded round-robin coverage, the asymmetric mastery update, task-type filtering, and an idealized retrieval-error model for the final retrieval-side result.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: The paper describes the model choices, guidance and execution tiers, benchmarks, evaluation pool sizes, baseline configurations, hyperparameters, number of iterations, and multi-seed evaluation settings in the main text and appendices. Some components rely on closed-weight teacher models, so exact reproduction requires access to the corresponding model APIs.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The experiments use public benchmarks and open-weight execution models.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The paper specifies the nine benchmarks, model pairings, evaluation pool size, number of iterations, fixed hyperparameters, baseline configurations, judge setup, and multi-seed reporting. Hyperparameters and baseline details are provided in the appendices.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: The main reasoning and sequential-environment tables report mean and standard deviation over multiple seeds for the primary MAGE cells, and the number of seeds is stated in the tables. Some baseline rows are single protocol runs or reproduced published numbers, and this is indicated where applicable.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No].

   Justification: Appendix N specifies a single H100 80GB for the inference-latency comparison. The paper reports guidance-tier call counts and inference latency, but does not provide complete training runtimes, total compute, and resource configurations for every experiment.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work evaluates algorithmic methods on established public benchmarks and does not involve human subjects, sensitive personal data collection, or deployment in high-stakes settings. The authors have reviewed the NeurIPS Code of Ethics and are not aware of any deviation.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No].

    Justification: The paper focuses on algorithmic and empirical evaluation and does not provide a dedicated discussion of both positive and negative societal impacts.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release a new pretrained language model, image generator, scraped dataset, or other high-risk artifact. The method uses existing models and public benchmarks.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No].

    Justification: The paper cites the existing benchmarks and models used in the experiments, but does not explicitly document the licenses and terms of use of all datasets, models, and tools.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The paper does not introduce a new dataset, released model, or benchmark as a primary contribution. The method creates intermediate graph memories during experiments, but these are experimental artifacts rather than released standalone assets.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing, human-subject experiments, or paid human annotation. Evaluation is conducted on existing public benchmarks.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve human subjects or collection of participant data, so IRB approval or equivalent review is not applicable.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research?

    Answer: [Yes].

    Justification: LLMs are central to the method: the guidance tier authors graph writes, the execution tier answers questions, and a Claude-family model is used as a semantic judge. The paper describes the division of LLM calls by component and distinguishes programmatic operations from LLM-mediated steps.