SKILLGRAPH: SKILL-AUGMENTED REINFORCEMENT LEARNING FOR AGENTS VIA EVOLVING SKILL GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Skill libraries enable large language model agents to reuse experience from past interactions, but most existing libraries store skills as isolated entries and retrieve them only by semantic similarity. This leads to two key challenges for compositional tasks. Firstly, an agent must identify not only relevant skills but also how they depend on and build upon each other. Secondly, it also makes library maintenance difficult, since the system lacks structural cues for deciding when skills should be merged, split, or removed. We propose SKILLGRAPH, a framework that represents reusable skills as nodes in a directed graph, with typed edges encoding prerequisite, enhancement, and co-occurrence relations. Given a new task, SKILLGRAPH retrieves not just individual skills, but an ordered skill subgraph that can guide multi-step decision making. The graph is continuously updated from agent trajectories and reinforcement learning feedback, allowing both the skill library and the agent policy to improve together. Experiments on ALFWorld, WebShop, and seven search-augmented QA tasks show that SKILLGRAPH achieves state-of-the-art performance against memory-augmented RL methods, with especially large gains on complex tasks that require composing multiple skills.

1 INTRODUCTION

Large Language Model (LLM) agents have shown strong capabilities in complex interactive tasks, including web navigation (Yao et al., 2022a), embodied household manipulation (Shridhar et al., 2021), and tool-augmented question answering (Yao et al., 2022b). Yet most agents treat task as episode (Yao et al., 2022b; Shinn et al., 2023), struggling to learn from past successes or failures even when structurally similar problems have been encountered (Xia et al., 2026). Since many tasks share recurring subproblems and compositional action patterns, an agent that can *learn from experience*—extracting reusable knowledge from past interactions—would avoid redundant exploration, transfer strategies to similar tasks, and progressively build up the ability to solve more complex problems.

To reuse experience, a common approach is to maintain a *skill library*, which stores reusable units of knowledge for solving recurring subproblems (Wang et al., 2024; Zhao et al., 2024; Xia et al., 2026). A skill can be either manually designed by humans (Xu & Yan, 2026) or automatically acquired from agent experience—for instance, by distilling successful trajectories into natural language (Zhao et al., 2024; Xia et al., 2026) or executable programs (Wang et al., 2024). Compared with manually crafted skills, automatically acquired skills are more scalable and can continuously expand as the agent encounters new tasks and environments. Therefore, we focus on automatically acquiring skills from interaction trajectories.

Despite their promise, existing skill libraries are often organized as flat collections, where each skill is stored as an independent entry and retrieved mainly by semantic similarity (Xia et al., 2026; Zhao et al., 2024; Liu et al., 2026). This ignores the fact that skills are inherently related: some skills are prerequisites for others, some enhance others, and some frequently co-occur in successful trajectories. As a result, flat libraries suffer from two key limitations. First, **retrieval is not compositional**. Complex tasks often require an ordered sequence of skills; for example, a "heat and place" task in ALFWorld may require locating an object, picking it up, heating it with an appliance, and then placing it at the target destination. A flat Top-K retriever can return relevant skills, but it does not indicate their dependencies or execution order. Second, **skill updates are not structured**. When skills are maintained independently, the library lacks explicit evidence for merging redundant skills, splitting

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

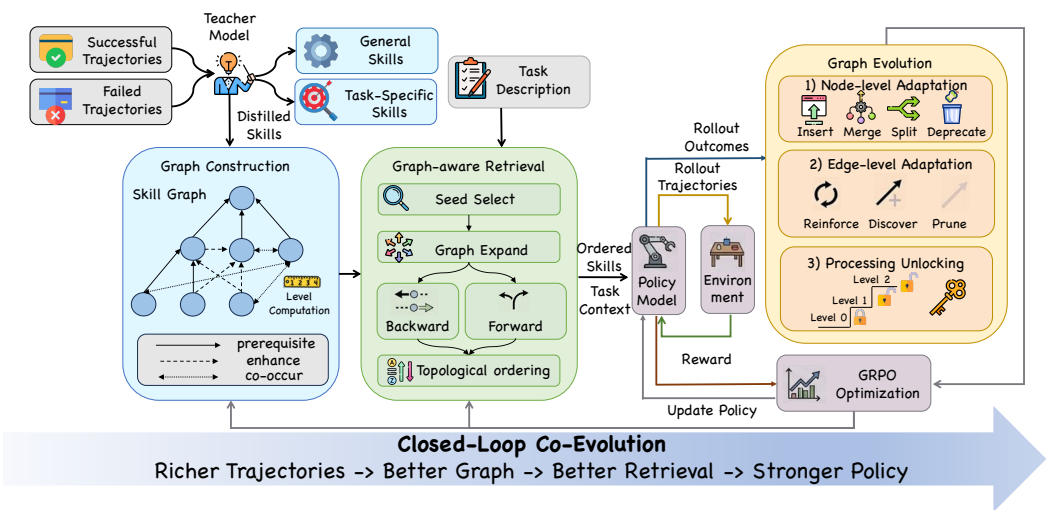

Figure 1: Overview of SKILLGRAPH. The skill graph and the agent's policy *co-evolve* through a closed loop: **(1)** graph construction distills skills and their typed relations (prerequisite, enhancement, co-occurrence) from trajectories; **(2)** graph-aware retrieval traverses these relations to produce dependency-ordered skill sequences that guide the policy; **(3)** graph evolution uses training feedback to refine skill nodes, adjust edge weights, and restructure the graph, which in turn improves future retrieval and policy learning.

overly broad skills, deprecating obsolete skills, or strengthening useful relations between skills (Xu & Yan, 2026). These limitations suggest that the core problem is not only how to acquire skills, but also how to *organize, retrieve, and update* them. If inter-skill relations are explicitly represented, retrieval can produce dependency-aware skill sequences rather than unordered hints, and both individual skills and their relations can be updated in a principled way.

Motivated by this, we propose SKILLGRAPH, a framework that organizes skills into a structured graph and co-evolves it with the agent's policy through reinforcement learning (RL). In SKILLGRAPH, nodes represent skills distilled from trajectories, while typed edges capture relations such as prerequisite, enhancement, and co-occurrence. SKILLGRAPH consists of three stages. First, **graph construction** builds an initial skill graph from interaction trajectories, making inter-skill relations explicit. Second, **graph-aware retrieval** starts from task-relevant seed skills, expands along graph edges, and orders retrieved skills according to their dependencies, producing a coherent skill sequence for decision-making. Third, **graph evolution** updates the graph during training by refining skill nodes and adjusting edge relations according to skill usage and success rate. Together, these stages form a closed loop: the skill graph provides structured guidance for policy learning, while the improving policy generates new trajectories that further refine the graph.

Empirically, we evaluate SKILLGRAPH on ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022a), and seven search-augmented question answering tasks (Kwiatkowski et al., 2019), covering embodied manipulation, web navigation, and information retrieval. Experimental results show that SKILLGRAPH achieves state-of-the-art performance across all benchmarks, with especially strong gains on complex multi-step tasks requiring skill composition. Further analysis shows that the graph structure improves skill reuse, reduces redundancy compared with flat libraries, and enables transfer of compositional knowledge from simpler tasks to more complex ones.

Our main contributions are summarized as follows:

- We propose a graph-structured formulation of skill library for LLM agents, where skills are connected by explicit prerequisite, enhancement, and co-occurrence relations.
- We introduce SKILLGRAPH, a closed-loop framework that supports dependency-aware skill retrieval and structured skill updates during RL.
- We conduct experiments on ALFWorld, WebShop, and seven search-augmented QA tasks, demonstrating state-of-the-art performance and substantial gains on complex multi-step tasks.

2 Related Work

Memory mechanisms in agents. External memory helps LLM agents reuse experience beyond the context window. Early methods store raw trajectories as examples (Zhao et al., 2024; Chhikara et al., 2025), while later work compresses experience into summaries or knowledge entries (Fang et al., 2025; Liu et al., 2026; Ouyang et al., 2025; Tang et al., 2025). Recent studies further apply RL directly to agent knowledge structures: MemRL (Zhang et al., 2026) performs runtime RL on episodic memory, MemEvolve (Zhang et al., 2025) meta-evolves memory systems, Mem-α (Wang et al., 2025) learns memory construction policies, and EvolveR (Wu et al., 2025) co-adapts the policy and memory bank. In contrast, SkillGraph represents experience as explicit skill abstractions with typed dependencies and evolves this structure jointly with the policy.

Graph structures for LLMs. Graph structures have been used to enhance LLM reasoning (Besta et al., 2024), retrieval (Edge et al., 2024), and agent planning (Nonkes et al., 2024). For agent skills specifically, Graph of Skills (Li et al., 2026) proposes dependency-aware structural retrieval. SkillGraph differs in that it jointly evolves the graph topology and the policy through RL, enabling the graph to adapt continuously via training feedback rather than remaining static after construction.

Agent skill evolution. Agentic skills can compact reusable strategies for subtasks. Voyager (Wang et al., 2024) accumulates executable code skills, and ExpeL (Zhao et al., 2024) distills transferable strategic experience from trajectories. Most closely related, SkillRL (Xia et al., 2026) co-evolves a hierarchical skill bank with the agent's policy through recursive RL. SkillGraph builds on this line by elevating the flat skill bank into a structured dependency graph, enabling typed relational modeling and topology evolution throughout training.

3 SkillGraph

We present SkillGraph, a framework that organizes agent skills as a directed dependency graph and co-evolves the graph with the agent's policy through RL. The key insight is that explicitly modeling inter-skill relations enables two mutually reinforcing capabilities: *structured retrieval* that produces dependency-aware skill sequences for compositional planning, and *principled evolution* that uses training feedback to refine both individual skills and their relations. As illustrated in Figure 1, the framework consists of three stages—graph construction (Section 3.1), graph-aware retrieval (Section 3.2), and graph evolution (Section 3.3)—integrated into a closed-loop training procedure (Section 3.4).

3.1 Graph Construction

The first step is to build a skill graph that makes inter-skill relations explicit, providing the structural foundation for both retrieval and evolution.

Skill distillation. We collect trajectories by rolling out the base policy π_{base} in the environment. A teacher language model $\mathcal{M}$ then distills successful trajectories τ^+ and failed trajectories τ^- into two types of skills: *general skills*, which capture domain-independent reasoning strategies applicable across tasks (e.g., "verify each sub-goal before proceeding"), and *task-specific skills*, which encode strategies tied to particular task types (e.g., "check the microwave for heated objects"). Each skill is represented as a compact record containing a title, a core principle describing the strategy, an applicability condition, and a category label indicating its type.

Graph structure. The distilled skills form the node set $\mathcal{V}$ of a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E}$ denotes the edge set. To capture how skills relate to one another, we define three typed edges:

- **Prerequisite** ($A \xrightarrow{\texttt{prereq}} B$): skill A must be applied before skill B.
- **Enhances** ($A \xrightarrow{\texttt{enhance}} B$): general skill A improves the effectiveness of task-specific skill B.
- **Co-occurs** ($A \xleftrightarrow{\texttt{co_occur}} B$): skills A and B frequently appear together in successful episodes.

Each edge $e \in \mathcal{E}$ carries a weight $w(e) \in [0, 1]$ reflecting the strength of the relation, which is dynamically adjusted during training. Each node $v \in \mathcal{V}$ maintains running statistics—usage count

$n_{\text{use}}(v)$, success count $n_{\text{succ}}(v)$, and empirical success rate $\hat{p}(v) = n_{\text{succ}}(v)/n_{\text{use}}(v)$—that drive both evolution decisions and progressive unlocking in Section 3.3. Based on the directed prerequisite and enhancement edges, each node is assigned a topological level $\ell(v)$ indicating its position in the dependency hierarchy: level-0 skills have no prerequisites, while higher-level skills depend on lower-level ones. Details of edge initialization and level computation are provided in Appendix A.

3.2 Graph-Aware Retrieval

Flat skill retrieval returns a set of individually relevant skills but ignores their dependencies, making it inadequate for tasks that require ordered skill composition. To address this, we design a graph-aware retrieval procedure that traverses the skill graph to produce a dependency-respecting sequence of skills. Given a task description d with task type $t(d)$, retrieval proceeds in three steps.

Seed selection. We first identify task-relevant entry points from the currently active skill set $\mathcal{V}_{\text{active}} \subseteq \mathcal{V}$, which contains skills that have been progressively unlocked (see Section 3.3.3). From $\mathcal{V}_{\text{active}}$, we select all general skills and task-type-matched skills as seed nodes, where $\mathcal{R}$ denotes a retrieved subset of skill nodes:

$$\mathcal{R}_{\text{seed}} = \{v \in \mathcal{V}_{\text{active}} : \text{category}(v) = \texttt{general} \ \vee \ \text{category}(v) = t(d)\}. \tag{1}$$

Graph expansion. Starting from the seed set $\mathcal{R}_{\text{seed}}$, we expand in two complementary directions to recover the full dependency context:

- *Backward expansion* traverses incoming prerequisite edges via breadth-first search (BFS) up to a maximum depth D, producing the backward-expanded set $\mathcal{R}_{\text{BFS}}$ that recovers foundational skills the seeds depend on but that may belong to other task categories.
- *Forward expansion* explores outgoing edges via beam search with beam width B, producing the forward-expanded set $\mathcal{R}_{\text{beam}}$. Each candidate node v receives an expansion score $\sigma(v)$ propagated from its predecessors: $\sigma(v) = \max_{u \in \text{parents}(v)} \sigma(u) \cdot w(u, v)$, where seed nodes are initialized with $\sigma = 1$. This prioritizes skills connected by well-validated relations.

Topological ordering. The union of seeds, backward-expanded, and forward-expanded skills is topologically sorted according to the graph's dependency edges, producing an ordered skill sequence:

$$\mathcal{R}_{\text{ret}} = \text{TopoSort}_{\mathcal{G}}(\mathcal{R}_{\text{seed}} \cup \mathcal{R}_{\text{BFS}} \cup \mathcal{R}_{\text{beam}}). \tag{2}$$

This sequence, capped at $K_{\max}$ skills, is prepended to the task prompt as structured guidance for the policy. Because the ordering reflects dependency relations, the agent receives skills in a natural simple-to-complex order that mirrors how sub-tasks should be composed.

3.3 Graph Evolution

A static skill graph cannot keep pace with a continuously improving policy: new failure modes demand new skills, redundant skills accumulate, and the relative importance of inter-skill relations shifts over training. To address this, we evolve both the skill nodes and their edges at each validation step, driven by trajectory-level feedback.

3.3.1 Node-Level: Adaptive Granularity Control

We maintain appropriate skill granularity through four operations, each triggered by specific diagnostic signals from the training process.

Insert. When the agent fails on tasks that existing skills do not adequately cover, we generate targeted new skills. The teacher model $\mathcal{M}$ analyzes a batch of failed trajectories τ^- together with the current skill set $\mathcal{R}_{\text{existing}}$, and proposes up to m new skills addressing the identified failure causes:

$$\{s^1_{\text{new}}, \ldots, s^m_{\text{new}}\} = \mathcal{M}(\text{insert}, \tau^-, \mathcal{R}_{\text{existing}}). \tag{3}$$

Merge. Redundant skills inflate context length and dilute retrieval precision. We identify candidates for merging by measuring the overlap of their graph neighborhoods: let $\mathcal{N}(v)$ denote the set of neighbors of node v in $\mathcal{G}$; when two skills s_i and s_j share most of their neighbors (Jaccard similarity $J(\mathcal{N}(s_i), \mathcal{N}(s_j)) \geq \tau_{\text{merge}}$, where τ_{merge} is the merge threshold), they likely encode redundant strategies and are synthesized into a single unified skill by $\mathcal{M}$.

Split. Overly broad skills that conflate distinct sub-strategies exhibit moderate success rates despite high usage ($\hat{p}(v) \in [0.15, 0.4]$ and $n_{\text{use}}(v) \geq 10$). We decompose such skills into more focused sub-skills via $\mathcal{M}$, reconnecting them with prerequisite edges.

Deprecate. Skills that are frequently retrieved but consistently fail ($n_{\text{use}}(v) \geq 20$ and $\hat{p}(v) < 0.15$) are deprecated and excluded from future retrieval, preventing them from degrading policy performance.

3.3.2 EDGE-LEVEL: TOPOLOGY EVOLUTION

While node-level operations adjust *what* skills are available, edge-level operations adjust *how* skills relate to one another, directly shaping retrieval quality.

Path reinforcement. Successful trajectories provide evidence that the retrieved skill sequence was effective. We reinforce this signal by increasing the weight of every edge along the successful path:

$$w(e) \leftarrow \min\big(w(e) + \alpha,\ 1.0\big), \quad \forall e \in \text{path}(\tau^+), \tag{4}$$

where $\alpha \in (0, 1)$ is the reinforcement step size and $\text{path}(\tau^+)$ denotes the set of edges traversed by the skill sequence used in successful trajectory τ^+. This makes validated dependency paths more likely to be traversed in future retrieval.

Co-occurrence discovery. New inter-skill relations emerge as the policy improves. When two skills co-occur in a successful episode but are not yet connected in $\mathcal{G}$, we add a `co_occur` edge to capture this discovered association.

Decay and pruning. To prevent stale relations from persisting indefinitely, all edge weights undergo multiplicative decay with decay factor $\gamma \in (0, 1)$: $w(e) \leftarrow \gamma \cdot w(e)$ at each checkpoint. Edges whose weights fall below a pruning threshold $w_{\min}$ are removed from $\mathcal{E}$. After all updates, node levels $\ell(v)$ are recomputed to reflect the new topology.

3.3.3 PROGRESSIVE UNLOCKING

Exposing the agent to complex, high-level skills before it has mastered their prerequisites can hinder learning. To implement a curriculum over skill complexity, SKILLGRAPH progressively unlocks skills based on their topological level. Initially, only level-0 foundational skills are active. Let L denote the current highest active level. When the average success rate of level-L skills exceeds an unlocking threshold θ_{unlock}, level-$(L{+}1)$ skills are activated:

$$\bar{p}(L) = \frac{1}{|\{v : \ell(v) = L\}|} \sum_{v:\, \ell(v)=L} \hat{p}(v) \ \geq\ \theta_{\text{unlock}} \implies \mathcal{V}_{\text{active}} \leftarrow \mathcal{V}_{\text{active}} \cup \{v : \ell(v) = L + 1\}. \tag{5}$$

This ensures that the agent builds competence from the ground up, with advanced compositional skills becoming available only when their foundations are reliable.

3.4 POLICY OPTIMIZATION AND CLOSED-LOOP TRAINING

We optimize the skill-augmented policy π_θ, parameterized by θ, using GRPO (Shao et al., 2024). For each task, we sample a group of G rollouts from π_θ conditioned on the task description d and the retrieved skill sequence $\mathcal{R}_{\text{ret}}$. Each rollout i receives a binary reward $R_i \in \{0, 1\}$ indicating task success, and the estimated advantage $\hat{A}_i$ is computed by within-group normalization:

$$\hat{A}_i = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^G)}{\text{std}(\{R_j\}_{j=1}^G) + \epsilon}, \tag{6}$$

where ϵ is a small constant for numerical stability. The policy is updated via the clipped surrogate objective with a KL penalty anchored to the reference policy π_{ref} (initialized from the SFT model):

$$\mathcal{L}(\theta) = \mathbb{E}\Big[\min\Big(r(\theta)\,\hat{A}_i,\ \text{clip}(r(\theta), 1{-}\epsilon_c, 1{+}\epsilon_c)\,\hat{A}_i\Big) - \beta\, D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\Big], \tag{7}$$

Algorithm 1 SKILLGRAPH: Skill-Augmented RL for Agents via Evolving Skill Graphs

Require: Base policy π_{base}, teacher model $\mathcal{M}$, environment Env, unlocking threshold θ_{unlock}
Ensure: Trained policy π_{θ^*}, evolved skill graph $\mathcal{G}^*$

— Graph Construction —
$\mathcal{T}^+, \mathcal{T}^- \leftarrow \text{Rollout}(\pi_{\text{base}}, \text{Env})$
$\mathcal{V} \leftarrow \mathcal{M}(\mathcal{T}^+, \mathcal{T}^-)$ ▷ Distill general & task-specific skills
$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \leftarrow \text{InitGraph}(\mathcal{V})$ ▷ Add prereq, enhance, co-occur edges
Compute topological levels $\ell(v)$ for all $v \in \mathcal{V}$
— Cold-Start SFT —
$\theta \leftarrow \text{SFT}(\pi_{\text{base}}, \mathcal{M}(\text{Env}, \mathcal{G}))$; $\pi_{\text{ref}} \leftarrow \pi_\theta$
$\mathcal{V}_{\text{active}} \leftarrow \{v : \ell(v) = 0\}$; $L \leftarrow 0$ ▷ Unlock level-0 skills
— Closed-Loop RL Training —
for epoch $= 1$ to N **do**
  for each task d with type $t(d)$ **do**
    Graph-Aware Retrieval:
      $\mathcal{R}_{\text{seed}} \leftarrow \{v \in \mathcal{V}_{\text{active}} : \text{category}(v) = \texttt{general} \vee \text{category}(v) = t(d)\}$
      $\mathcal{R}_{\text{BFS}} \leftarrow \text{BackwardBFS}(\mathcal{R}_{\text{seed}}, \mathcal{G}, D)$; $\mathcal{R}_{\text{beam}} \leftarrow \text{ForwardBeam}(\mathcal{R}_{\text{seed}}, \mathcal{G}, B)$
      $\mathcal{R}_{\text{ret}} \leftarrow \text{TopoSort}_{\mathcal{G}}(\mathcal{R}_{\text{seed}} \cup \mathcal{R}_{\text{BFS}} \cup \mathcal{R}_{\text{beam}})$ ▷ Cap at $K_{\max}$ skills
    Sample G rollouts $\{\tau^{(i)}\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid d, \mathcal{R}_{\text{ret}})$; Update θ via GRPO
  end for
  if validation step **then**
    Graph Evolution:
    Node-level: Insert / Merge / Split / Deprecate skills via $\mathcal{M}$
    Edge-level: Reinforce paths in τ^+; Discover new co-occur edges; Decay & prune weak edges
    Recompute topological levels $\ell(v)$
    Progressive Unlocking: if $\bar{p}(L) \geq \theta_{\text{unlock}}$ **then** $\mathcal{V}_{\text{active}} \leftarrow \mathcal{V}_{\text{active}} \cup \{v : \ell(v) = L{+}1\}$; $L \leftarrow L{+}1$
  end if
end for
return $\pi_\theta, \mathcal{G}$

where $r(\theta) = \pi_\theta / \pi_{\text{old}}$ is the importance sampling ratio between the current and previous policies, ϵ_c is the clipping parameter, β is the KL penalty coefficient, and D_{KL} denotes the Kullback–Leibler divergence.

At each validation step, the full graph evolution pipeline is executed, creating a closed training loop: the improving policy generates richer trajectories that refine the skill graph through node- and edge-level updates, while the refined graph provides higher-quality structured retrieval that accelerates subsequent policy learning. The complete procedure is summarized in Algorithm 1.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Environments. ALFWorld (Shridhar et al., 2021) is a text-based household interaction environment that covers six task categories (Pick, Look, Clean, Heat, Cool, Pick2), each requiring multi-step goal-directed manipulation. WebShop (Yao et al., 2022a) presents a web navigation challenge in which agents must search, browse, and purchase products meeting specific user requirements. For search-augmented question answering, we evaluate on three single-hop benchmarks—NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023)—and four multi-hop benchmarks—HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023).

Baselines. We compare SKILLGRAPH against four groups of methods. **(1) Closed-source LLMs**: GPT-4o (Hurst et al., 2024) and Gemini-2.5-Pro (Comanici et al., 2025), serving as strong references. **(2) Prompt-based and memory-augmented methods**: ReAct (Yao et al., 2022b), Reflexion (Shinn et al., 2023), Mem0 (Chhikara et al., 2025), MemP (Fang et al., 2025), ExpeL (Zhao et al., 2024), and SimpleMem (Liu et al., 2026), which use in-context experience without parameter updates. **(3) RL-based methods**: RLOO (Ahmadian et al., 2024) and GRPO (Shao et al., 2024). **(4) Memory-augmented RL methods**: MemRL (Zhang et al., 2026), EvolveR (Wu et al., 2025), Mem0+GRPO,

Table 1: Main results on ALFWorld and WebShop. ALFWorld reports per-subtask and overall success rates (%); WebShop reports task score and success rate (%). **Bold** and underline denote the best and second-best results, respectively.

Method	ALFWorld							WebShop	
	Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Closed-source LLMs									
GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0	31.8	23.7
Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3	42.5	35.9
Prompt-based Agentic or Memory-based Methods									
ReAct	48.5	35.4	34.3	13.2	18.2	17.6	31.2	46.2	19.5
Reflexion	62.0	41.6	44.9	30.9	36.3	23.8	42.7	58.1	28.8
Mem0	54.0	55.0	26.9	36.4	20.8	7.69	33.6	23.9	2.00
MemP	54.3	38.5	48.1	56.2	32.0	16.7	41.4	25.3	6.40
ExpeL	21.0	67.0	55.0	52.0	11.0	6.00	46.3	30.9	11.2
SimpleMem	64.5	33.3	20.0	12.5	33.3	3.84	29.7	33.2	8.59
RL-based Methods									
RLOO	87.6	<u>78.2</u>	87.3	81.3	71.9	48.9	75.5	80.3	65.7
GRPO	90.8	66.1	89.3	74.7	72.5	64.7	77.6	79.3	66.1
Memory-Augmented RL-based Methods									
MemRL	62.8	38.5	22.2	12.5	8.00	0.00	21.4	29.5	9.20
EvolveR	64.9	33.3	46.4	13.3	33.3	33.3	43.8	42.5	17.6
Mem0+GRPO	78.1	54.8	56.1	31.0	65.0	26.9	54.7	58.1	37.5
SimpleMem+GRPO	89.5	63.6	60.0	50.0	64.9	26.3	62.5	67.8	46.9
SkillRL	<u>97.9</u>	71.4	<u>90.0</u>	<u>90.0</u>	**95.5**	**87.5**	<u>89.9</u>	<u>85.2</u>	<u>72.7</u>
SKILLGRAPH (Ours)	**100.0**	**80.0**	**100.0**	**100.0**	<u>95.3</u>	<u>83.3</u>	**93.1**	**91.5**	**84.4**

SimpleMem+GRPO, and SkillRL (Xia et al., 2026). For search-augmented QA, we additionally compare against CoT (Wei et al., 2022), RAG (Arslan et al., 2024), Search-o1 (Li et al., 2025), Search-R1 (Jin et al., 2025) and ZeroSearch (Sun et al., 2025).

Implementation details. We adopt Qwen2.5-7B-Instruct (Yang et al., 2025) as the base policy π_{base}, initialized via cold-start SFT, and OpenAI o3 (Jaech et al., 2024) as the teacher model $\mathcal{M}$ for skill distillation, SFT data generation, and graph evolution operations. RL training uses GRPO with learning rate 1×10^{-6}, KL coefficient $\beta = 0.01$, clipping parameter $\epsilon_c = 0.2$, train batch size 16, and group size $G = 8$. For graph-aware retrieval, we cap the retrieved skill sequence at $K_{\max} = 8$, set backward-BFS depth $D = 2$, and forward beam width $B = 3$. For graph evolution, edges are initialized with weights $w = 0.3$ (co-occur) and $w = 0.2$ (enhance); at each validation checkpoint, successful paths receive additive reinforcement $\alpha = 0.05$, all weights decay by factor $\gamma = 0.99$, and edges below $w_{\min} = 0.05$ are pruned. Node-level evolution uses merge threshold $\tau_{\text{merge}} = 0.85$, and at most $m = 3$ newly inserted skills per update. Progressive unlocking activates level-$(L{+}1)$ skills when the average success rate of level-L skills exceeds $\theta_{\text{unlock}} = 0.6$.

4.2 MAIN RESULTS

Comparison with baselines. Table 1 reports results on ALFWorld and WebShop. SKILLGRAPH achieves the best overall performance on both benchmarks. *(i)* Notably, SKILLGRAPH with a 7B open-source model substantially outperforms closed-source LLMs: it surpasses GPT-4o by 45.1 points and Gemini-2.5-Pro by 32.8 points on ALFWorld, and exceeds both by over 48 points on WebShop, demonstrating that structured skill reasoning can compensate for the scale gap. *(ii)* Compared with prompt-based and memory methods, SKILLGRAPH outperforms the best method (ExpeL) by 46.8 points on ALFWorld, with the largest gains on Clean (100.0 vs. 55.0) and Heat (100.0 vs. 52.0). These subtasks require executing prerequisite actions in a strict order, which flat retrieval cannot enforce but graph-aware retrieval handles naturally. *(iii)* Over the vanilla GRPO baseline with the same optimizer, SKILLGRAPH improves by 15.5 and 18.3 points on ALFWorld and WebShop respectively, directly quantifying the benefit of graph-structured skill guidance in reducing exploration burden. *(iv)* Against the strongest prior method SkillRL, SKILLGRAPH achieves slightly

Table 2: Results on search-augmented QA. SKILLGRAPH is trained on NQ$^{\heartsuit}$ and HotpotQA$^{\heartsuit}$ (in-domain) and evaluated zero-shot on the remaining five benchmarks$^{\spadesuit}$ (out-of-domain).

Method	Single-Hop QA			Multi-Hop QA				Avg.
	NQ$^{\heartsuit}$	TriviaQA$^{\spadesuit}$	PopQA$^{\spadesuit}$	HotpotQA$^{\heartsuit}$	2Wiki$^{\spadesuit}$	MuSiQue$^{\spadesuit}$	Bamboogle$^{\spadesuit}$	
Qwen2.5-7B-Instruct	11.6	35.6	1.20	16.4	22.2	4.80	14.4	15.2
CoT	12.8	35.6	3.80	16.2	22.6	6.60	24.0	17.4
RAG	27.4	58.2	17.8	25.8	23.2	9.40	16.8	25.5
Search-o1	19.4	40.6	11.4	17.0	27.0	8.60	30.4	22.1
Search-R1	39.3	61.0	39.7	37.0	40.1	14.6	36.8	38.5
ZeroSearch	43.6	61.8	**51.5**	34.6	35.2	18.4	27.8	39.1
EvolveR	43.5	63.4	44.6	38.2	42.0	15.6	54.4	43.1
SkillRL	45.9	63.3	45.9	43.2	40.3	**20.2**	**73.8**	47.1
SKILLGRAPH (Ours)	**48.0**	**63.8**	48.5	**44.7**	**43.4**	19.5	72.6	**48.9**

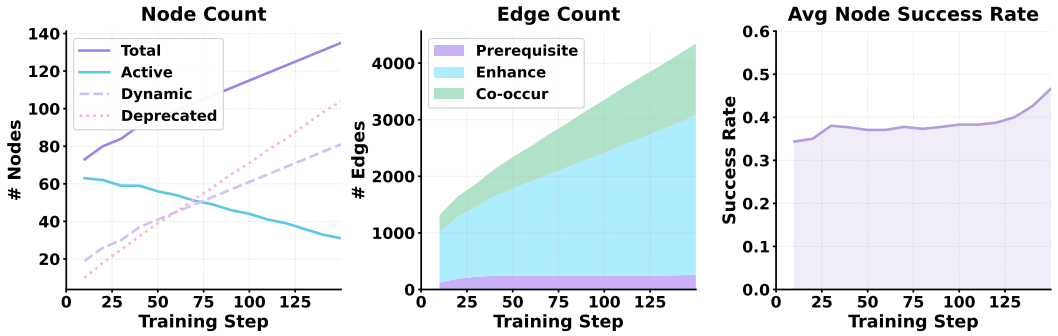

Figure 2: Skill graph evolution over training on WebShop. Left: node counts (total, active, inserted, deprecated). Middle: edge counts by type. Right: average node success rate.

higher ALFWorld performance while gaining 11.7 points on WebShop. The gap stems from the evolving graph structure: graph evolution continuously refines the skill set and discovers inter-skill relations (e.g., query refinement $\rightarrow$ attribute matching $\rightarrow$ price comparison), providing higher-quality compositional guidance than a static flat skill bank.

Generalization to search-augmented QA. Table 2 reports results on seven QA benchmarks. Trained only on NQ and HotpotQA, SKILLGRAPH achieves the highest average performance (48.6) and generalizes zero-shot to five unseen datasets. On single-hop tasks, SKILLGRAPH surpasses all baselines on NQ (48.0) and TriviaQA (63.8), improving over SkillRL by 2.1 and 0.5 respectively. This advantage stems from graph evolution, which keeps the skill set aligned with the evolving policy rather than relying on a fixed skill library. On multi-hop tasks, SKILLGRAPH leads on HotpotQA (44.7) and 2Wiki (43.4), where prerequisite-ordered retrieval helps decompose chained queries into sub-questions. These results confirm that the structured skill representation learned from two training domains transfers effectively to unseen tasks, demonstrating strong generalization without task-specific adaptation.

4.3 ANALYSIS

Table 3: Ablation study on ALFWorld and WebShop success rate(%).

Method	ALFWorld	WebShop
SKILLGRAPH	**93.1**	**84.4**
w/o Graph Structure	89.9	72.7
w/o Graph-aware Ret.	59.4	79.7
w/o Graph Evolution	78.2	70.3
w/o Cold-Start SFT	73.4	67.2

Ablation study. Table 3 isolates each component's contribution. The components exhibit complementary strengths across environments. On ALFWorld, removing graph-aware retrieval causes the largest single drop (-33.7), confirming that the rigid multi-step subtasks (e.g., Clean, Heat) critically depend on prerequisite-ordered skill sequences, consistent with the large gains reported in the main results. On WebShop,

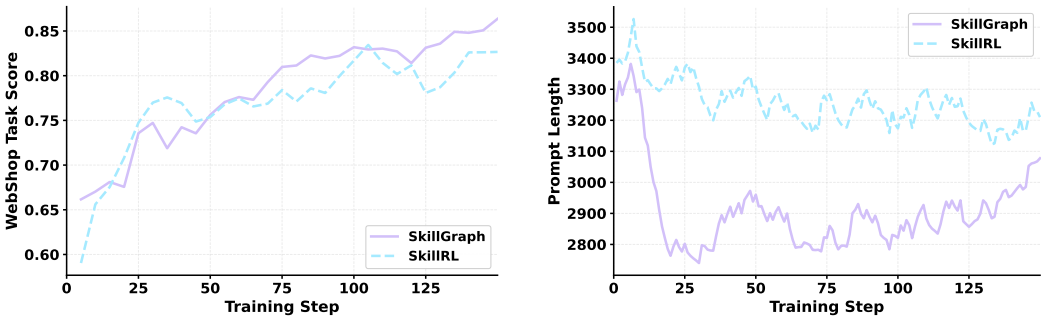

Figure 3: Training dynamics and context efficiency. Left: WebShop task score over training epochs. Right: average prompt length during training.

graph evolution (-14.1) and graph structure (-11.7) contribute the most, indicating that WebShop benefits primarily from maintaining a high-quality, evolving skill set—the correct skills matter more than their ordering in this flexible navigation setting, which explains why the graph structure gap over SkillRL ($+11.7$) is larger than the retrieval ordering gap. Cold-start SFT yields a large combined drop, confirming that a good initialization is essential for RL convergence in complex agent environments.

Skill graph evolution dynamics. Figure 2 tracks graph statistics over training. Node count grows from $\sim$20 to $\sim$140 via failure-driven insertion, but the active count plateaus earlier as deprecation prunes failing skills—a self-regulating loop that prevents unbounded growth. Co-occur edges grow fastest through automatic discovery, while prerequisite and enhance edges increase steadily via path reinforcement, showing that the graph discovers relational structure beyond initial construction. The average node success rate rises confirming that evolution progressively filters low-quality skills while reinforcing useful ones.

Convergence and context efficiency. Figure 3 (left) shows that SKILLGRAPH surpasses SkillRL after roughly 50 training steps and maintains a consistently higher score thereafter, converging to a superior final performance. The faster convergence is driven by dependency-ordered retrieval reducing early-stage exploration and progressive unlocking acting as an automatic curriculum. Figure 3 (right) shows that graph-guided retrieval maintains shorter prompts than flat retrieval throughout training, because graph traversal limits the retrieved set to topologically relevant skills rather than all semantically similar entries, improving both inference cost and signal-to-noise ratio.

5 CONCLUSION

We presented SKILLGRAPH, a framework that organizes agent skills into a structured dependency graph with typed relational edges and co-evolves the graph with the agent's policy through RL. By unifying graph construction, graph-aware retrieval, and graph evolution into a closed training loop, SKILLGRAPH addresses three key limitations of flat skill libraries: weak compositional planning, poor granularity control, and the inability to accumulate inter-skill relational signals. Experiments on ALFWorld, WebShop, and seven search-augmented QA benchmarks demonstrate state-of-the-art performance, with the largest gains on complex multi-step tasks that require ordered skill composition.

Limitations and future work. Our current framework relies on a strong teacher model for skill distillation and graph-adaptive operations, which introduces additional inference cost during graph evolution. Exploring lightweight alternatives such as self-distillation or critic-based skill generation could reduce this dependency. Additionally, the skill graph is currently constructed and evolved within a single environment; investigating cross-environment skill transfer—where a graph trained on one domain bootstraps learning in another—is a promising direction. Finally, scaling SKILLGRAPH to larger base models and more diverse task distributions remains an open question worth exploring.

AI USE STATEMENT

Large language models were used in this work in two capacities. **(1) As part of the research methodology:** LLMs serve as the teacher model for skill distillation, SFT data generation, and graph evolution operations (insertion, merge, split), and as the base policy fine-tuned via RL, as described in Section 3. **(2) For writing assistance:** LLMs were used to polish the language and improve the presentation of this manuscript. All LLM-assisted content has been manually reviewed, verified, and edited by the authors. The authors take full responsibility for the accuracy and integrity of all claims, results, and statements presented in this paper.

ETHICS STATEMENT

This work evaluates agents in simulated household tasks, web navigation, and search-augmented question answering. Skills distilled from trajectories may inherit biases or errors from the underlying language models and teacher-generated guidance. Applications beyond the evaluated environments require careful validation and appropriate safeguards. Appendix G discusses potential benefits, risks, and limitations, and Appendix H describes the assets used in this work.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 describe the proposed framework. Appendices A and B provide graph construction, evolution, and implementation details. Appendix C lists training and graph hyperparameters, and Appendix D provides representative prompt templates.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Muhammad Arslan, Hussam Ghanem, Saba Munawar, and Christophe Cruz. A survey on rag with llms. *Procedia computer science*, 246:3781–3790, 2024.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

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

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

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.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Dawei Li, Zongxia Li, Hongyang Du, Xiyang Wu, Shihang Gui, Yongbei Kuang, and Lichao Sun. Graph of skills: Dependency-aware structural retrieval for massive agent skills. *arXiv preprint arXiv:2604.05333*, 2026.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*, 2026.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 9802–9822, 2023.

Noa Nonkes, Sergei Agaronian, Evangelos Kanoulas, and Roxana Petcu. Leveraging graph structures to detect hallucinations in large language models. In *Proceedings of TextGraphs-17: Graph-based Methods for Natural Language Processing*, pp. 93–104, 2024.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*, 2025.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-{\alpha}: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

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.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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*, pp. 2369–2380, 2018.

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, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022b. URL https://openreview.net/forum?id=tvI4u1ylcqs.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*, 2025.

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.

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

A SUPPLEMENTARY DETAILS FOR SKILLGRAPH

This appendix provides formal definitions, derivations, and implementation specifics that supplement the method description in Section 3.

A.1 LEVEL COMPUTATION

Each node $v \in \mathcal{V}$ is assigned a topological level $\ell(v)$ used for progressive unlocking and dependency-respecting retrieval ordering. Levels are computed via BFS over the directional dependency edges:

$$\ell(v) = \begin{cases} 0 & \text{if } v \text{ has no prerequisite/enhancement parents,} \\ \max_{u:(u,v)\in\mathcal{E}_{\text{dep}}} \ell(u) + 1 & \text{otherwise,} \end{cases} \tag{8}$$

where $\mathcal{E}_{\text{dep}} = \{e \in \mathcal{E} : \text{type}(e) \in \{\texttt{prereq}, \texttt{enhance}\}\}$. We exclude `co_occur` edges from this computation because co-occurrence captures symmetric association rather than directional dependency; including them would introduce cycles and blur the prerequisite hierarchy. Levels are recomputed after every graph evolution step to reflect topology changes.

A.2 EDGE INITIALIZATION

Before training begins, edges are initialized with structural priors rather than learned from data: `co_occur` edges ($w = 0.3$) connect task-specific skills within the same category, and `enhance` edges ($w = 0.2$) connect each general skill to all task-specific skills. No `prereq` edges are created at initialization; they emerge through graph evolution as the agent discovers ordering dependencies.

A.3 STATISTICS UPDATE

At each validation checkpoint, node statistics are updated incrementally before evolution decisions are made:

$$\hat{p}(v) \leftarrow \frac{n_{\text{succ}}(v) + n_{\text{succ}}^{\text{new}}(v)}{n_{\text{use}}(v) + n_{\text{use}}^{\text{new}}(v)}, \tag{9}$$

where $n_{\text{succ}}^{\text{new}}(v)$ and $n_{\text{use}}^{\text{new}}(v)$ are the success and usage counts observed in the latest trajectory batch. A skill receives one usage count when it is retrieved into the prompt and one success count when the corresponding rollout succeeds.

A.4 NODE EVOLUTION: ADDITIONAL DETAILS

Insertion and edge bootstrapping. Newly inserted skills start as isolated nodes with no edges. Connections are established in subsequent checkpoints via the co-occurrence discovery mechanism: if a new skill and an existing skill co-appear in at least $c_{\min} = 2$ successful episodes, a `co_occur` edge is added automatically.

Merge: edge inheritance. When two skills v_i and v_j are merged into v_{merged}, the merged node inherits the union of edges from both originals. Duplicate edges to the same neighbor are resolved by keeping the higher weight.

Split: edge reconnection. When a skill v is split into sub-skills $\{v'_1, v'_2, \ldots\}$, the sub-skills are connected by `prereq` edges in the order produced by the teacher model. Existing edges of v are redistributed to the sub-skill whose description is most relevant.

A.5 PROGRESSIVE UNLOCKING: ADDITIONAL DETAILS

During the initial warmup phase (the first 5 training steps), only non-deprecated level-0 skills are active: $\mathcal{V}_{\text{active}}^{(0)} = \{v : \ell(v) = 0\}$. After warmup, unlocking is checked at each validation checkpoint. Success rates are smoothed with a Beta$(1, 1)$ prior to avoid premature unlocking from small sample sizes. If the newly unlocked level already satisfies the threshold, multiple levels can be unlocked within a single checkpoint, enabling rapid progression when the policy has strong foundational competence.

B Additional Implementation Details

Skill schema. Each natural-language skill is stored as a compact record containing a unique skill identifier, a short title, a principle, an applicability condition, and a category (`general` or an environment-specific task type). The same record format is used by flat skill-library baselines; SkillGraph augments it with graph metadata including level, exposure count, successful-exposure count, success rate, creation step, and deprecation status.

Co-occurrence edge threshold. New `co_occur` edges require at least $c_{\min} = 2$ co-appearances in successful validation episodes before being added, preventing spurious edges from single lucky episodes. Deprecated nodes are retained in the saved graph for auditability but excluded from $\mathcal{V}_{\text{active}}$.

C Experimental Details

Metric definitions. For ALFWorld, we report success rate. For WebShop, Table 1 reports both normalized task score and binary success rate, while Table 3 reports task score to match the training-curve analysis. For QA benchmarks, we report exact-match accuracy under the search-augmented QA evaluation protocol. The search experiments use a tool-augmented QA environment where the retriever returns top-3 passages from a Wikipedia index built with an E5 retriever. SkillGraph is trained on NQ and HotpotQA and evaluated on the seven datasets reported in Table 2.

Hyperparameters. Table 4 lists the training hyperparameters for each environment. Table 5 lists the SkillGraph-specific hyperparameters, which are shared across all three environments.

Table 4: Training hyperparameters per environment.

Hyperparameter	ALFWorld	WebShop	Search QA
Optimization			
Learning rate	1×10^{-6}	1×10^{-6}	1×10^{-6}
LR warmup ratio	–	–	0.1
KL coefficient β	0.01	0.01	0.001
Entropy coefficient	–	–	0
Invalid-action penalty	0.1	0.1	0.01
Batch sizes			
Train batch size	16	16	256
Validation batch size	64	64	512
Group size G	8	8	4
PPO mini-batch size	128	64	256
PPO micro-batch / GPU	4	4	8
Sequence lengths			
Max prompt length	4096	6000	5000
Max response length	512	768	700
Max episode steps	50	15	4
Infrastructure			
Tensor-parallel size	4	4	1
GPU memory utilization	0.5	0.7	0.5
Param offload	✓	✓	–
Optimizer offload	✓	✓	–
Schedule			
Total epochs	200	200	200
Save frequency	10	10	5
Validation frequency	5	5	5

Table 5: SKILLGRAPH hyperparameters (shared across all environments).

Hyperparameter	Symbol	Value
Graph-aware retrieval		
Retrieved skill cap	$K_{\max}$	8
Backward BFS depth	D	2
Forward beam width	B	3
Node-level evolution		
Max new skills per update	m	3
Merge threshold (Jaccard)	τ_{merge}	0.85
Deprecation threshold	–	0.15
Min usage for deprecation	–	20
Edge-level evolution		
Path reinforcement step	α	0.05
Edge decay factor	γ	0.99
Edge pruning threshold	$w_{\min}$	0.05
Progressive unlocking		
Curriculum warmup epochs	–	5
Level unlock threshold	θ_{unlock}	0.6

D PROMPT TEMPLATES

This section gives representative prompt templates used by SKILLGRAPH. Environment-specific prompts differ mainly in the action space and observation format, while the retrieved skill block is shared across ALFWorld, WebShop, and search-augmented QA.

Agent prompt with retrieved skills. The following template shows the common structure used when skill memory is enabled. The ALFWorld and WebShop variants replace the action-space description with admissible environment actions, while the search variant replaces it with the choice between issuing a `<search>` query and returning a `<answer>`.

Agent Prompt Template

```
You are an expert agent operating in the target environment.
Your task is to: {task_description}

## Retrieved Relevant Experience
{retrieved_skills}

## Current Progress
Prior to this step, you have already taken {step_count} step(s).
Below are the most recent observations and actions:
{action_history}

Current observation:
{current_observation}

Admissible actions:
[{admissible_actions}]

Now it is your turn to take an action.
First reason step-by-step within <think> </think> tags.
Then choose one admissible action within <action> </action> tags.
```

Graph-ordered skill injection. In graph retrieval mode, retrieved skills are rendered in dependency order before being inserted into the agent prompt. This makes the graph structure visible to the policy without requiring special model architecture changes.

Graph-Ordered Skill Block

```
### Skills (ordered by dependency)
- **[category] Skill Title** [skill_id]: Skill principle.
_Apply when: Applicability condition._
- **Next Skill Title** [skill_id]: Skill principle.
_Apply when: Applicability condition._

### Mistakes to Avoid
- **Don't**: Failure pattern or bad action.
**Instead**: Corrective strategy.
```

Failure-driven skill insertion prompt. During dynamic updates, failed validation trajectories are summarized and passed to the teacher model. The teacher is asked to produce a small number of new skills and to avoid duplicating existing skill titles. Returned identifiers are reassigned by the implementation to prevent collisions.

Failure-Driven Skill Insertion Prompt

```
Analyze these failed agent trajectories and suggest NEW skills
to add to the skill bank.

FAILED TRAJECTORIES:
Example 1:
Task: {task}
Task Type: {task_type}
Trajectory (last 5 steps):
Action: {action}
Observation: {observation}
...

EXISTING SKILL TITLES (avoid duplicating these):
{existing_titles}

Generate 1-{max_new_skills} NEW actionable skills that would
help avoid these failures.
Each skill must have: skill_id, title (3-5 words), principle
(1-2 sentences), when_to_apply.

Use skill_ids: {dyn_id_list}

Return ONLY a JSON array of skills, no other text.
```

Skill merge and split prompts. For graph evolution, the teacher is also used as a skill-bank curator. Merge prompts ask it to combine two semantically overlapping skills into one concise skill. Split prompts ask it to decompose a high-usage but low-success skill into two or three simpler sub-skills, optionally conditioned on failure contexts where the original skill did not help. Both prompts require the same JSON schema as insertion: `skill_id`, `title`, `principle`, and `when_to_apply`.

E Additional Search Training Results

Table 6 reports intermediate validation checkpoints for the search-augmented QA SkillGraph run. The final checkpoint at step 200 gives the best average score, while NQ and HotpotQA peak slightly earlier at step 180. We report the unified step-200 checkpoint in Table 2 for a single consistent model selection rule across datasets.

F Compute Resources

All training experiments are conducted on a single node equipped with 8× NVIDIA A100 80GB GPUs, 224 CPU cores, and 2048 GB of system memory. The total compute budget across all training runs amounts to approximately 280 GPU-hours.

Table 6: Search-augmented QA validation accuracy (%) for SKILLGRAPH over training. NQ and HotpotQA are in-domain training datasets; the remaining datasets are held-out transfer evaluations.

Step	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg.
30	39.2	58.9	40.6	39.7	35.4	16.3	69.0	42.4
60	44.9	61.8	44.1	42.4	39.1	17.6	69.8	45.3
90	46.0	62.1	45.6	42.8	40.0	18.1	68.1	46.4
120	46.2	62.5	45.1	43.6	41.9	18.4	70.6	46.9
150	46.4	63.6	45.6	44.3	43.3	19.4	70.6	47.9
180	**48.3**	63.4	47.0	**44.8**	41.9	19.4	72.2	48.1
200	48.0	**63.8**	**48.5**	44.7	**43.4**	**19.5**	**72.6**	**48.9**

Table 7: Licenses of datasets, environments, and models used in this work.

Asset	Type	License	Reference
Environments			
ALFWorld	Environment	MIT	Shridhar et al. (2021)
WebShop	Environment	MIT	Yao et al. (2022a)
Datasets — Single-hop QA			
Natural Questions (NQ)	Dataset	Apache 2.0	Kwiatkowski et al. (2019)
TriviaQA	Dataset	Apache 2.0	Joshi et al. (2017)
PopQA	Dataset	MIT	Mallen et al. (2023)
Datasets — Multi-hop QA			
HotpotQA	Dataset	CC BY-SA 4.0	Yang et al. (2018)
2WikiMultiHopQA	Dataset	Apache 2.0	Ho et al. (2020)
MuSiQue	Dataset	CC BY 4.0	Trivedi et al. (2022)
Bamboogle	Dataset	MIT	Press et al. (2023)
Models			
Qwen2.5-7B-Instruct	Model	Apache 2.0	Yang et al. (2025)
OpenAI o3	API Service	Proprietary	Jaech et al. (2024)

G BROADER IMPACT

This work proposes a general framework for organizing and evolving reusable skills in LLM-based agents. We discuss potential broader impacts below.

Positive impacts. By enabling agents to accumulate structured knowledge from experience and reuse it across tasks, SKILLGRAPH can improve the sample efficiency and reliability of autonomous agents in domains such as household assistance, web navigation, and information retrieval. The graph-structured skill memory also enhances interpretability: users can inspect which skills were retrieved, how they are related, and why certain decisions were made, facilitating human oversight of agent behavior. Furthermore, the progressive unlocking mechanism provides a built-in safety property—agents are restricted to well-mastered foundational skills before being exposed to more complex behaviors, reducing the risk of premature deployment of unreliable capabilities.

Potential risks and limitations. As with other LLM agent systems, SKILLGRAPH inherits the biases and failure modes of the underlying language model. Skills distilled from trajectories may encode undesirable patterns if the training data contains biased behaviors. The teacher model used for graph evolution (e.g., skill insertion, merge, split) may introduce errors or hallucinated skills, which could propagate through the graph. We mitigate this through the deprecation mechanism that removes consistently failing skills, but additional safeguards (e.g., human-in-the-loop skill review) may be necessary for safety-critical applications. Our current evaluation focuses on simulated environments; deployment in real-world settings would require careful validation of skill quality and additional safety constraints.

H ASSET LICENSES

Table 7 summarizes the licenses of all datasets, environments, and models used in this work. All assets are publicly available and permit academic research use.

I SUPPLEMENTARY EXPERIMENTS AND ANALYSIS

This appendix examines the contributions of graph structure and maintenance, sensitivity to maintenance thresholds, and the computational cost of SKILLGRAPH. We also discuss task-specific error patterns and the interaction between graph evolution and policy optimization. Unless otherwise stated, ALFWorld results are success rates (%), WebShop results are task scores on a 0–100 scale, and performance differences are absolute points.

I.1 CONTROLLED GRAPH INTERVENTIONS

Intervention design. We examine four interventions that change skill organization or graph operations within the same base-model, cold-start SFT, and GRPO framework. The flat-library variant retains the skill content but removes explicit graph topology. The prerequisite-removal variant deletes prerequisite relations while retaining the other edge types. Retrieval without topology selects relevant skills without dependency expansion or topological ordering. The static-graph variant retains graph-based retrieval but disables graph evolution during training. These comparisons probe the roles of relational structure, dependency recovery, retrieval, and adaptation, respectively.

Table 8: Graph interventions on ALFWorld success rate and WebShop task score. Drops are absolute differences from the full method in each column.

	ALFWorld success		WebShop score	
Variant	Score	Drop	Score	Drop
Full method	93.1	–	91.5	–
Flat skill library	89.9	3.2	85.2	6.3
Remove prerequisite edges	87.5	5.6	82.5	9.0
Retrieval without topology	59.4	33.7	83.3	8.2
Static graph	78.2	14.9	70.3	21.2

Results and interpretation. Table 8 shows that every intervention reduces performance in both environments. Replacing the graph with a flat skill library lowers ALFWorld success from 93.1 to 89.9 and WebShop score from 91.5 to 85.2, corresponding to drops of 3.2 and 6.3 points. Removing prerequisite edges incurs drops of 5.6 and 9.0 points, supporting the value of explicit dependencies alongside associative connections. On ALFWorld, retrieval without topology produces the largest decrease, 33.7 points, consistent with the need to recover and order intermediate actions in household tasks. On WebShop, the static graph produces the largest decrease, 21.2 points, suggesting that adapting skill content and relations is particularly useful when search and product-selection strategies must improve with the policy.

Together, these results highlight complementary roles for graph-aware retrieval and evolution. Dependency recovery and ordering provide structured guidance for ALFWorld's sequential tasks, while continued refinement of skills and relations supports WebShop's search and product-selection decisions.

I.2 JOINT SENSITIVITY TO MAINTENANCE THRESHOLDS

Settings. We use shared default thresholds across environments and examine two joint perturbations, denoted Loose and Strict in Table 9. Each setting changes the update, merge, deprecation, and level-unlock thresholds together while keeping the remaining settings fixed. A lower merge threshold allows more pairs of skills to qualify for merging, while a lower deprecation threshold retains low-success skills longer. Lowering the unlock threshold makes higher-level skills available earlier; raising it requires stronger evidence of competence at the current level. These joint settings examine how the balance between skill maintenance and progressive exposure affects performance.

Table 9: Joint sensitivity to update, merge, deprecation, and unlock thresholds on ALFWorld success rate and WebShop task score.

Setting	Update	Merge	Deprecate	Unlock	ALFWorld	WebShop
Default	0.40	0.85	0.15	0.60	93.1	91.5
Loose	0.30	0.75	0.10	0.50	87.5	89.1
Strict	0.60	0.95	0.20	0.70	89.8	85.7

Results. The default configuration performs best among the three settings in both environments. Relative to the default, the loose setting reduces ALFWorld success by 5.6 points and WebShop score by 2.4 points; the strict setting reduces them by 3.3 and 5.8 points, respectively. Both perturbations remain above the skill-free GRPO reference in Table 1: the loose and strict settings exceed GRPO by 9.9 and 12.2 points on ALFWorld, and by 9.8 and 6.4 points on WebShop task score. Thus, the advantage over GRPO persists under both tested joint perturbations. ALFWorld is more sensitive to the loose setting, whereas WebShop is more sensitive to the strict setting, suggesting that the shared default provides an effective balance between skill refinement and progressive exposure across the two environments.

I.3 Teacher Usage, Training Compute, and Efficiency

Teacher-model usage. The supplementary comparisons reuse the public SkillRL cold-start SFT data. During training, o3 supports failure analysis, skill insertion, and merge/split verification through calls triggered by graph-maintenance conditions. The policy and graph algorithms handle action generation, traversal, topological sorting, edge-weight updates, pruning, and final evaluation. Table 10 reports average additional usage of 102.5 calls on ALFWorld and 74.3 calls on WebShop, costing US$1.91 and US$1.39, respectively. The combined incremental API cost for graph maintenance is US$3.30.

Table 10: Average additional o3 calls and API cost for graph maintenance on ALFWorld and WebShop.

Environment	Average o3 calls	Additional cost (USD)
ALFWorld	102.5	1.91
WebShop	74.3	1.39
Combined cost	–	3.30

Training time and historical cost estimate. The supplementary comparison runs use four A100 GPUs each. Table 11 shows that SkillGraph reduces training time from 32.26 to 31.59 hours on ALFWorld and from 12.34 to 11.39 hours on WebShop, corresponding to reductions of 2.1% and 7.7%. Across the two environments, training time decreases by 1.62 hours, or 3.6%. To express compute and teacher usage in the same monetary units, we use the historical estimate of US$21.9576 per hour for an eight-A100 instance, equivalent to US$2.7447 per A100-hour. Estimated GPU cost is the training duration multiplied by four GPUs and this hourly rate.

Table 11: Training wall-clock hours on four A100 GPUs and estimated GPU costs for the supplementary comparison runs.

Environment	Method	Hours	Estimated GPU cost (USD)
ALFWorld	SkillRL / flat skills	32.26	354.18
ALFWorld	SkillGraph	31.59	346.82
WebShop	SkillRL / flat skills	12.34	135.48
WebShop	SkillGraph	11.39	125.05
Combined	SkillRL / flat skills	44.60	489.65
Combined	SkillGraph	42.98	471.87

Relative to the estimated GPU costs of SKILLGRAPH, additional teacher usage amounts to 0.55% on ALFWorld, 1.11% on WebShop, and 0.70% across both environments. Local RL training therefore accounts for most of the estimated expense in these comparison runs, with graph-maintenance calls adding a small API cost.

Context and step efficiency. Graph-aware retrieval expands from task-relevant seeds and caps the resulting skill sequence, supplying a compact set of skills with their dependency context. Table 12 reports an average WebShop prompt length of $1{,}419$ tokens, compared with $1{,}781$ for flat skills, a reduction of 362 tokens (20.3%). The final step-token count decreases from 1.81M to 1.37M (24.3%), while step time decreases from 302.9 to 292.4 seconds (3.5%).

Table 12: WebShop prompt length, step-token count, and step time for flat skills and SKILLGRAPH.

Measurement	SkillRL / flat skills	SKILLGRAPH
Average prompt length (tokens)	1,781	1,419
Final step-token count	1.81M	1.37M
Step time (seconds)	302.9	292.4

The reductions in prompt length and token count are consistent with the shorter-context trend in Figure 3. Together with the lower step time, they show that graph-aware retrieval improves context efficiency while reducing training-step runtime in the WebShop comparison.

Time measurements. Table 13 summarizes the recorded time measurements for closed-source models and the two local RL methods. For the local RL comparison, the entries correspond to training hours: SKILLGRAPH takes 42.98 hours across ALFWorld and WebShop, compared with 44.60 hours for SkillRL. The reduction in both environments complements the context-efficiency results in Table 12.

Table 13: Recorded time measurements on ALFWorld and WebShop. The SkillRL and SKILLGRAPH entries report training hours.

Method	ALFWorld	WebShop	Total
GPT-4o	13.88	2.24	16.12
Gemini-2.5-Pro	28.82	22.45	51.27
SkillRL	32.26	12.34	44.60
SKILLGRAPH	31.59	11.39	42.98

I.4 QA CLARIFICATIONS AND QUALITATIVE ERROR ANALYSIS

QA results and comparisons. Table 2 reports NQ and PopQA scores of 48.0 and 48.5 for SKILLGRAPH, exceeding SkillRL by 2.1 and 2.6 points. ZeroSearch achieves the highest PopQA score of 51.5, while SKILLGRAPH achieves the highest arithmetic mean over the seven benchmarks at 48.6, compared with 47.5 for SkillRL. On MuSiQue, SKILLGRAPH scores 19.5, exceeding EvolveR's 15.6 by 3.9 points and falling below SkillRL's 20.2 by 0.7 points. On 2Wiki, SKILLGRAPH achieves 43.4, compared with EvolveR's 42.0. These results show broad gains over SkillRL alongside variation across individual QA tasks.

The following qualitative examples illustrate task-specific failure mechanisms and the guidance that graph-organized skills can provide.

ALFWorld errors. ALFWorld requires the agent to track object locations, inventory, and completed subgoals across a sequence of constrained actions. State-tracking failures can lead an agent to continue searching after observing the target or to overlook an object inside a closed container. Execution failures include attempting to clean, heat, or cool an object before acquiring it, choosing the wrong device, or omitting final placement. A skill chain such as *find* $\rightarrow$ *pick* $\rightarrow$ *transform* $\rightarrow$ *place* makes the dependency structure explicit and can help preserve the overall plan across local decisions. This procedural guidance complements the policy's interpretation of observations and tracking of environment state.

WebShop errors. WebShop requires satisfying several product constraints simultaneously. A product may match the requested category while violating a size, color, or price requirement, and an apparently suitable listing may still require selecting the correct option before purchase. A chain such as *query refinement* → *filtering* → *option verification* → *final check* encourages the agent to retain all constraints through the interaction and verify the final selection. Graph evolution can refine these strategies as failures expose missing checks, which is consistent with the static-graph result in Table 8.

MuSiQue difficulties. MuSiQue requires recovering implicit bridge entities and carrying them into subsequent queries to assemble a complete evidence chain. An incorrect intermediate entity can direct the next search toward irrelevant evidence, while incomplete retrieved evidence can leave the final answer unsupported even when the reasoning steps are ordered correctly. Under a limited search budget, successful execution depends on accurate entity selection, focused follow-up queries, and evidence integration. These demands provide a plausible explanation for the task's lower absolute scores and motivate skills that explicitly preserve intermediate entities and verify evidence coverage.

I.5 Additional Methodological Details

Three relation types. The three relation types serve distinct operations in graph-aware retrieval. Prerequisite edges recover dependencies and determine execution order; enhancement edges connect general advice to task-specific procedures; and co-occurrence edges expand retrieval toward skills that have been useful together. Co-occurrence expresses associative relevance, while prerequisite and enhancement edges define the dependency levels described in Appendix A.1. Together, these relations support dependency recovery, general-skill propagation, and associative expansion. The prerequisite-removal results in Table 8 further support the role of explicit ordering relations in both environments.

Meaning of co-evolution. Co-evolution refers to the feedback loop between policy learning and graph maintenance. GRPO updates the policy from rollouts conditioned on retrieved skills. Subsequent validation trajectories update skill-usage and success statistics, which guide changes to skill nodes, relation weights, and the active skill set. The revised graph then changes the guidance supplied to later policy updates. Teacher calls support selected skill-generation and verification operations within this loop. Policy optimization and discrete graph updates therefore interact through trajectory feedback and the skill guidance used in subsequent rollouts.

Experimental setting. The supplementary comparisons use Qwen2.5-7B-Instruct, keeping the base policy fixed when examining skill organization and maintenance. ALFWorld and WebShop evaluate the framework in sequential manipulation and multi-constraint navigation, respectively. The QA experiments train on NQ and HotpotQA and evaluate zero-shot on five additional datasets, assessing transfer of the search-augmented policy to unseen QA tasks.