CoEvoSkill: Niche-Aware Co-Evolution of Skill Content and Graph Topology for Agents

Anonymous authors
Paper under double-blind review

Abstract

Skill libraries have emerged as a lightweight adaptation layer for large language model (LLM) agents, providing reusable procedures that can be deployed without modifying model weights. Existing methods improve such libraries along two isolated lines, refining the content of individual skills or evolving the graph that organizes them, and neither controls the execution layer, where refined skills can still be realized as invalid, repetitive, or premature steps. We argue that skill content, graph topology, and execution are mutually dependent, and introduce **CoEvoSkill** to close the loop. Skill texts are refined from failure attribution over execution traces; the dependency graph is updated with success-conditioned symbiosis scores; and a Monte Carlo tree search controller, guarded at every language realization before it changes the state, searches over graph-supported skills. The layers exchange evidence: guard rejections and search statistics drive content edits, edge updates, and the insertion of missing skills, while the evolved library reshapes the candidate space seen by search. We prove a fixed-point theorem showing that, under bounded edits and contractive topology changes, the alternating updates terminate at a stationary library that dominates single-layer alternatives. Across five benchmarks spanning embodied interaction, mathematical reasoning, and financial analysis, CoEvoSkill outperforms strong state-of-the-art methods and transfers across domains.

1 Introduction

Large language model (LLM) agents increasingly rely on reusable *skills* to preserve and reuse procedural knowledge acquired through interaction. A skill may encode a tool-use strategy, a reasoning procedure, a domain heuristic, or a recovery routine. Unlike parameter updates, such artifacts can be inspected, edited, transferred, and deployed while keeping the underlying model fixed, making skill libraries a flexible substrate for continual agent improvement (Schick et al., 2023; Wang et al., 2023a; Alzubi et al., 2026; Yang et al., 2026). As these libraries grow, however, a more difficult question arises: how should the library itself evolve, rather than simply accumulate more experience?

A skill library has two closely coupled aspects. The first is *content*: what knowledge or procedure each skill encodes. The second is *structure*: how skills depend on, specialize, and compose with one another. Content-oriented methods refine individual skills from execution feedback, for example through failure-driven updates or constrained textual optimization (Alzubi et al., 2026; Yang et al., 2026), while other approaches organize reusable behaviors into structured libraries or graphs to support retrieval, hierarchical planning, and composition (Bagaria et al., 2021; Yuan et al., 2024; Li et al., 2026). However, these two forms of adaptation are still largely treated as separate optimization problems.

This separation is restrictive because content and structure constrain one another. A skill that becomes a common prerequisite for several procedures should stay general and expose a stable interface; one that moves toward a higher abstraction should avoid duplicating its prerequisites. The dependency also runs in reverse: editing a skill can make an edge obsolete, expose redundancy, or enable a previously ineffective composition. Allowing both sides to evolve therefore creates a cou-

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.

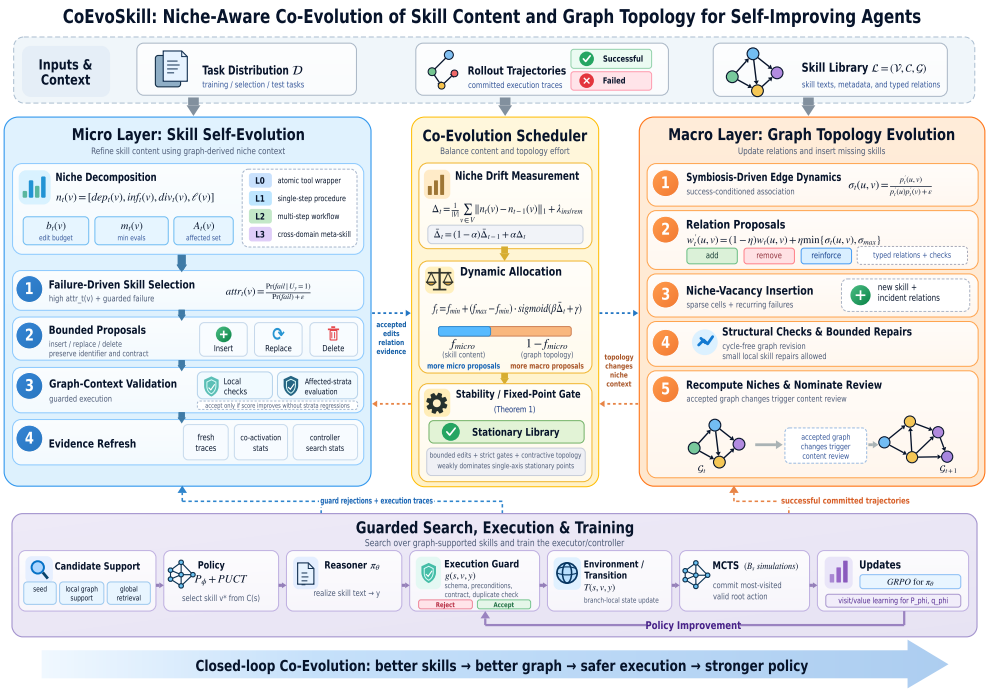

Figure 1: **Overview of CoEvoSkill.** The framework jointly evolves skill content and graph topology through two interacting layers. The micro layer refines skill content using its current structural role and validates proposed edits in graph context. The macro layer updates relations among skills from trajectory feedback and reorganizes the skill graph. A co-evolution scheduler coordinates the two processes. Topology changes provide structural context for subsequent content refinement, while accepted content updates feed back into graph evolution. A guarded Monte Carlo tree search controller connects both layers to execution.

pled, moving-target process, in which each layer continuously reshapes the objective of the other, rather than the simple union of two independent loops. A useful framework must expose enough structural context to guide local edits, preserve a feedback path through which content changes reorganize the library, and keep the joint process stable.

We propose **CoEvoSkill**, a niche-aware framework that jointly adapts skill content and graph topology. We use a skill's *niche* to denote its current structural role, characterized by dependency depth, downstream influence, interaction diversity, and abstraction level. This representation guides content refinement: skills in different roles are revised differently, and candidate edits are evaluated for local utility and for compatibility with dependent skills, so the graph acts as part of the context that determines how procedural knowledge is revised, not merely as a retrieval index.

CoEvoSkill couples this with topology evolution: evidence from successful trajectories updates relations and identifies roles the library lacks. The processes form a closed loop, as topology changes alter the niche context for later content edits and accepted edits update the evidence for graph reorganization. A scheduler allocates effort by the rate of structural change, and we study stability under explicit assumptions on bounded updates, validation, and topology dynamics.

Figure 1 illustrates the framework. We evaluate on five benchmarks spanning embodied interaction, mathematical reasoning, and financial reasoning, which differ in format but all involve reusable procedures and multi-step problem solving. CoEvoSkill consistently improves over content-only and structure-only alternatives, with larger gains on tasks that depend on procedural composition, and ablations show the benefit cannot be attributed to either layer alone. Our main contributions are:

- We formulate skill-library improvement as a *coupled content–structure adaptation problem*, emphasizing the bidirectional dependence between what a skill encodes and the role it plays in a compositional library.
- We introduce **CoEvoSkill**, which uses graph-derived niche context to guide skill refinement and validation while feeding accepted edits back into topology evolution through a coordinated co-evolution process.
- We study the stability of the coupled updates and evaluate across diverse agent and reasoning tasks, with controlled ablations and analyses of the evolving skill structure.

2 Related Work

Skill content optimization. A growing line of work refines individual skill documents without modifying weights. EvoSkill (Alzubi et al., 2026) applies evolutionary operators (insert, delete, replace) to skill text and selects survivors by task performance. SKILLOPT (Yang et al., 2026) formulates bounded text optimization as a constrained search over edit operations, keeping refinements within a prescribed edit budget. These methods treat each skill as an independent target and do not model inter-skill dependencies. CoEvoSkill makes every content edit ecologically aware: the niche constrains the edit budget, abstraction level, and validation scope from the skill's graph position.

Skill graph and hierarchical organization. Graph-based skill organization has been studied in both reinforcement learning (RL) and LLM-agent settings. Deep Skill Graphs (Bagaria et al., 2021) learn hierarchical option graphs for long-horizon RL, and Plan4MC (Yuan et al., 2024) constructs skill graphs for Minecraft by learning basic skills and planning over their relations. SKILLGRAPH (Li et al., 2026) organizes LLM-agent skills into a typed dependency graph and co-evolves topology with the policy via group-relative policy optimization (GRPO). Its graph evolution is phenotypically frozen, however: node text stays static while topology changes. CoEvoSkill closes this loop by letting topology changes trigger content re-evaluation and vice versa.

Memory and experience accumulation. Agent memory systems accumulate experiences for reuse. Mem0 (Mem0, 2025) maintains a structured memory layer with automatic deduplication, Reflexion (Shinn et al., 2023) stores verbal self-reflections across attempts, and Voyager (Wang et al., 2023a) builds an open-ended skill library through iterative prompting. These systems store skills but do not systematically refine content or reorganize structure over time; CoEvoSkill can serve as an evolutionary layer on top of any memory system.

Mathematical and financial reasoning. For mathematical reasoning, ToRA (Gou et al., 2024) integrates tool use with reasoning and PAL (Gao et al., 2023) offloads computation to a program interpreter. For financial reasoning, FINDER (Khatuya et al., 2025) offers a retrieval- and program-augmented question-answering framework and Fin-o1 (Qian et al., 2025) applies reinforcement learning to financial reasoning. CoEvoSkill is orthogonal to these domain-specific methods: it provides a general skill-evolution substrate that can incorporate domain skills (a program calculator for math, a table parser for finance) and optimize them jointly.

3 Method

We formalize the skill library as an evolving ecosystem and present CoEvoSkill's three mechanisms: niche-conditioned content refinement, symbiosis-driven graph updates, and adaptive co-evolution scheduling. A guarded skill-search controller connects them to execution: the library defines candidate procedures, search selects and instantiates them, and execution traces guide subsequent updates.

3.1 Preliminaries

A skill library is $\mathcal{L} = (V, C, \mathcal{G})$, where V contains persistent skill identifiers, C contains their texts and metadata, and $\mathcal{G} = (V, E, w)$ is a weighted, typed multigraph. Each skill has an applicability condition and an input/output contract. A prerequisite edge $u \to v$ specifies a required preceding skill; a co-activation edge records empirical co-use; a specialization edge links a general procedure

to a narrower variant. The prerequisite subgraph is acyclic, and terminal skills that produce final answers are distinguished from continuation skills.

At execution step i, the state s_i contains the task, accepted steps, observations, executed skill identifiers, and remaining budget. A controller selects a skill v_i, the language agent π_θ produces its realization y_i, and an execution guard decides whether it may update the state. Search requires branch-local execution: reasoning histories are copied and interactive environments support snapshots or deterministic replay.

We separate training tasks D_{tr}, an adaptation-selection set D_{sel}, and a final test set D_{test}. For a fixed executor $\Xi = (\pi_\theta, P_\phi, q_\psi, \mathrm{guard})$ and search budget B_s, we evaluate

$$J(\mathcal{L}; \Xi) = \mathbb{E}_{x,\tau}[r(\tau; x)], \quad r \in [0,1], \quad x \sim \mathcal{D},\ \tau \sim \mathrm{Exec}(x; \mathcal{L}, \Xi, B_s). \tag{1}$$

The executor is frozen while comparing revisions; GRPO updates the language agent between library-update phases, and search-policy/value learning provide auxiliary controller updates. Test tasks never update the library or controller.

3.2 Niche Decomposition

The niche of a skill summarizes its structural role:

$$\mathbf{n}_t(v) = [\mathrm{dep}_t(v), \mathrm{inf}_t(v), \mathrm{div}_t(v), \ell(v)]. \tag{2}$$

Here dep is the longest incoming prerequisite path, inf counts distinct prerequisite descendants within two hops, and div is the entropy of domain labels among co-activation neighbors, zero for an empty neighborhood. The metadata label $\ell \in \{0,1,2,3\}$ denotes an atomic tool wrapper, single-step procedure, multi-step workflow, or cross-domain procedure; it describes abstraction rather than topological depth. Domain and abstraction labels are assigned at skill creation and remain fixed during local edits.

We normalize the first two features with fixed clipping scales, the entropy by $\log \max(2, |\mathcal{D}_{\mathrm{dom}}|)$, and ℓ by three, giving $\bar{\mathbf{n}}_t(v) \in [0,1]^4$. These features control refinement:

$$\begin{aligned} b_t(v) &= \min\{B_{\max}, \lceil b_0 + b_1 \overline{\mathrm{dep}}_t(v) \rceil\}, \\ m_t(v) &= m_0 + \lceil m_1 \overline{\mathrm{inf}}_t(v) + m_2 \overline{\mathrm{div}}_t(v) \rceil, \\ \mathcal{A}_t(v) &= \{v\} \cup \mathrm{Desc}_{\mathrm{pre}}(v) \cup \mathrm{Nbr}_{\mathrm{co}}(v). \end{aligned} \tag{3}$$

$b_t(v)$ bounds token-level edit distance and $m_t(v)$ sets the minimum number of affected-task evaluations: greater dependency depth allows more text for coordination, while influence and diversity allocate extra validation. These are bounded allocation heuristics with nonnegative coefficients fixed before evaluation. The editor preserves the abstraction ℓ, and validation is stratified across the domains in $\mathcal{A}_t(v)$, so broad-use skills receive cross-domain checks rather than only checks on the triggering failure. Appendix C works through these quantities on a concrete three-node subgraph.

3.3 Micro Layer: Skill Self-Evolution

The micro layer changes skill text while holding identifiers, contracts, and graph structure fixed, following a propose-and-test procedure that builds on failure-driven refinement and bounded text optimization (Alzubi et al., 2026; Yang et al., 2026), with graph context setting the edit constraints and evaluation scope.

Stage 1: Failure-driven selection. Let $U_v(\tau)$ indicate that v was committed on an executed trajectory; retrieval or an uncommitted search visit does not count as use. On current trajectories from the frozen executor, compute

$$a_t(v) = \frac{\widehat{\Pr}(\mathrm{fail} \mid U_v = 1)}{\widehat{\Pr}(\mathrm{fail}) + \epsilon}. \tag{4}$$

Skills above a threshold and minimum usage become candidates; repeated guard failures also nominate skills, including ones never committed, with rejections counted once per state–skill pair. These nominate for diagnosis rather than causal responsibility, since task difficulty and controller choices matter and untried skills get no estimate; the editor receives the failed realization and violated check to separate an unclear instruction from a wrong invocation.

Stage 2: Bounded proposals. For each candidate, the editor receives its text, representative failures and successes, and the contracts of prerequisites and affected neighbors. It proposes insertions, replacements, or deletions with token-level edit distance at most $b_t(v)$, leaving the identifier, applicability condition, and contract unchanged; a change in the skill's role instead requires a separately validated library revision.

Stage 3: Graph-context validation. A local check rejects malformed edits and contract changes, and a low-cost screen on affected training tasks filters clearly harmful proposals. Surviving proposals are evaluated with the complete guarded executor under old and proposed libraries, using the same selection tasks, seed panel, and search budget. Besides the aggregate selection score, we measure affected strata: tasks previously using v, its prerequisite descendants, or co-activation neighbors, partitioned by domain, with strata fixed before evaluation. A revision is accepted only if

$$\begin{aligned} \widehat{J}(\mathcal{L}';\Xi) - \widehat{J}(\mathcal{L};\Xi) &\geq \delta, \\ \widehat{J}_a(\mathcal{L}';\Xi) - \widehat{J}_a(\mathcal{L};\Xi) &\geq -\xi \quad \forall a, \end{aligned} \tag{5}$$

where $\delta > 0$ is the required gain and $\xi \geq 0$ a regression tolerance; the panel must cover the strata and $m_t(v)$ evaluations or the edit is deferred. Edits commit sequentially against the latest library, and only the untouched test set measures generalization.

Stage 4: Evidence refresh. After an accepted revision, stale caches are dropped and fresh traces collected. Only committed trajectories determine co-activation frequencies, while search returns are reserved for controller learning, so heavily explored branches are not mistaken for frequently useful combinations.

3.4 Symbiosis-Driven Edge Dynamics

The macro layer revises relations and introduces missing skills. Let N_t^+ be the number of successful committed trajectories, $n_t^+(u)$ the number using u, and $n_t^+(u,v)$ the number using both. For sufficiently supported pairs, define

$$\begin{aligned} p_t^+(u) = n_t^+(u)/N_t^+, \qquad p_t^+(u,v) = n_t^+(u,v)/N_t^+, \\ \sigma_t(u,v) = \frac{p_t^+(u,v)}{p_t^+(u)p_t^+(v) + \epsilon}. \end{aligned} \tag{6}$$

We call σ a symbiosis score but use it as a success-conditioned association statistic that proposes relationships for evaluation; it does not identify causal synergy or interference. Pairs below a minimum joint count, or windows without successful trajectories, do not trigger association updates.

Relation proposals. For co-activation edges we propose the smoothed weight

$$w_t'(u,v) = (1-\eta)w_t(u,v) + \eta \min\{\sigma_t(u,v), \sigma_{\max}\}. \tag{7}$$

Separate addition and removal thresholds propose new or obsolete edges. Observed order can nominate a prerequisite, but admission also requires a contract-supported dependency where u supplies a precondition of v; specialization requires the same function with narrower applicability, so relations are not inferred from order or vocabulary alone. Prerequisite proposals that introduce cycles are rejected, and σ updates co-activation weights only.

Niche-vacancy insertion. Sparse abstraction–domain cells only nominate gaps; insertion also requires recurring failures that existing skills cannot handle. The editor proposes a new skill with contract and incident relations, evaluated as a complete revision, and a graph revision may atomically repair a bounded number of affected skills so a relation whose benefit depends on adapting its procedures is not rejected; repair count and budget are fixed in advance.

All graph changes, with any associated repairs, pass structural checks and Eq. equation 5; affected strata cover changed endpoints and their dependency and co-activation neighborhoods in either graph, and new skills additionally require coverage of the targeted task family. Accepted changes recompute niches and nominate affected skills for content review, while fresh traces after content edits determine later relation proposals.

3.5 Co-Evolution Scheduling and Stability

Each round freezes Ξ and alternates content and graph proposals under a fixed evaluation budget. To allocate proposal opportunities, measure niche drift between consecutive accepted versions. With $I_t = V_t \cap V_{t-1}$ and $U_t = V_t \cup V_{t-1}$,

$$\Delta_t = \frac{\sum_{v \in I_t} \|\bar{\mathbf{n}}_t(v) - \bar{\mathbf{n}}_{t-1}(v)\|_1}{4 \max(1, |I_t|)} + \frac{|V_t \triangle V_{t-1}|}{\max(1, |U_t|)}. \tag{8}$$

The second term accounts for insertion or removal. An exponential average $\bar{\Delta}_t = (1-\kappa)\bar{\Delta}_{t-1} + \kappa\Delta_t$ controls the probability of a micro proposal,

$$f_t = f_{\min} + (f_{\max} - f_{\min}) \operatorname{sigmoid}(\beta \bar{\Delta}_t + \gamma), \tag{9}$$

with $0 < f_{\min} < f_{\max} < 1$ and $\beta > 0$, so structural changes increase attention to content while both proposal types remain available. Rejections do not reset the estimate, initialized to zero each phase. This is a resource-allocation heuristic; each phase has finite proposal and execution budgets, and proposal generation, screening, validation, and evidence refresh all count toward reported computation.

Theorem 1 (Co-evolution fixed point)**.** *Fix the executor Ξ and the panel, inducing a deterministic score $\widehat{J} \in [0, 1]$. Assume **(A1)** bounded edits, namely micro edit distance at most $b_t(v)$ and a bounded number of macro repairs; **(A2)** strict gates (Eq. equation 5), under which each accepted revision raises $\widehat{J}$ by at least $\delta > 0$ and bounds every stratum regression by ξ; and **(A3)** contractive topology, where the macro map contracts normalized niche displacement by some $\rho < 1$. Then (i) the number of accepted revisions per phase is at most $\lfloor (1 - \widehat{J}(\mathcal{L}_0))/\delta \rfloor$, after which the process reaches a stationary library with no passing micro or macro proposal; and (ii) this library weakly dominates the content-only and graph-only stationary points, strictly whenever a coupled revision that changes both layers is accepted.*

Proof sketch. By A2 every acceptance raises the same bounded score by at least δ, giving (i) through $\widehat{J}(\mathcal{L}_0) + K\delta \leq 1$, while A3 contracts niche motion so the process settles rather than oscillates. The joint proposal set contains every single-layer proposal and A2 makes acceptance monotone, so the joint process replays either single-layer path, giving (ii). The full assumptions, the role of ρ, the complete proof, and the stochastic-evaluation caveat are in Appendix E.

Scope. The claim is stated per phase for a fixed executor, bounds accepted rather than attempted changes, and does not imply a global optimum across GRPO rounds; under stochastic evaluation the gate is an empirical safeguard.

Within each round, micro and macro proposals are generated according to f_t, screened structurally and on affected training tasks, and committed only when Eq. equation 5 holds; after a round the library and controller are frozen while θ is updated by GRPO, after which fresh searches update ϕ, ψ and the joint checkpoint is scored on D_{sel}.

3.6 Graph-Aware Search and Policy Training

Monte Carlo tree search (MCTS) (Silver et al., 2016) gives both evolution layers a common execution interface; following language-agent tree search (Zhou et al., 2024; Qi et al., 2025), we treat a procedure-level action as distinct from its language realization, where actions are reusable, evolving skills.

Candidate support and selection. From the last accepted skill in the current branch, or semantic seed skills at the root, we traverse at most two hops over outgoing directed and undirected co-activation edges. Local candidates are ranked by semantic similarity plus λ_g times their strongest path product (edge weights normalized over each node's outgoing neighbors; unit weights for prerequisite and specialization edges), keeping the top k_{local}. Global semantic retrieval (Lewis et al., 2020) adds at most k_{explore} further skills:

$$C(s) = \text{Eligible}_s(C_{\text{local}}(s; \mathcal{G}) \cup C_{\text{explore}}(s)). \tag{10}$$

Eligibility enforces applicability and satisfied prerequisites, replacing blocked skills with eligible unmet ancestors: prerequisite edges restrict eligibility, co-activation edges supply complements,

and specialization edges supply alternatives. The policy $P_\phi(v \mid s)$ normalizes over $C(s)$ and PUCT selects

$$v^* = \arg\max_{v \in C(s)} \bar{Q}(s,v) + c_{\text{puct}} P_\phi(v \mid s) \frac{\sqrt{1 + \sum_u N(s,u)}}{1 + N(s,v)}. \tag{11}$$

Guarded expansion. On first expanding (s, v), the reasoner samples $y \sim \pi_\theta(\cdot \mid s, \text{text}(v))$ in a branch-local state. A fixed guard specified by the task adapter checks schema and argument types, executable preconditions, admissible state changes, and mechanically checkable contract constraints,

$$g(s,v,y) = \prod_{j \in \mathcal{H}} g_j(s,v,y), \qquad g_j \in \{0,1\}, \tag{12}$$

also rejecting duplicate actions under unchanged state. A rejection backs up -1 without a child and disables that realization for the search; an acceptance creates $s' = T(s,v,y)$, with continuation/terminal value heads supplying leaf returns (Appendix D). After B_s simulations the most-visited guard-valid root action is committed and search restarts; absent a valid action the episode fails, and budget never overrides the guard.

Policy and controller updates. The language agent is trained with GRPO (Shao et al., 2024), with the library, guard, and controller frozen. For each task we sample G direct trajectories (skills from P_ϕ, realizations from $\pi_{\theta_{\text{old}}}$; a guard rejection scores zero), so search-selected paths are not reused as on-policy samples. With group-relative advantage $\widehat{A}_i = (r_i - \bar{r})/(\text{std}(r) + \epsilon)$ we optimize the clipped surrogate with a KL penalty (Eq. equation 15, Appendix D); constant groups contribute zero advantage. Fresh searches then supply visit-based policy and backed-up value targets for ϕ, ψ (Eqs. equation 16–equation 14), without updating θ. Ground truth may supervise these targets but is never seen by the guard, selector, or value model during selection or testing.

4 Experiments

We organize the experiments around three questions: (**Q1**) Does jointly evolving skill content and graph topology outperform methods that optimize only one layer or neither? (**Q2**) Which components and interactions account for the gains? (**Q3**) How does the ecosystem change, does the joint process behave as the fixed-point analysis predicts, and what library does it produce?

4.1 Experimental Setup

Benchmarks. We use five benchmarks from three domains that differ in action space, trajectory length, and required composition:

- **Embodied AI:** ALFWorld (Shridhar et al., 2021), a text household simulator with six task types chaining navigation and manipulation; success rate over the 134 unseen tasks.
- **Mathematical reasoning:** GSM8K (Cobbe et al., 2021) (1,319 grade-school problems) and MATH (Hendrycks et al., 2021) (5,000 competition problems); accuracy on the final number or `\boxed` answer.
- **Financial analysis:** FinQA (Chen et al., 2021) (1,147) and ConvFinQA (Chen et al., 2022) (1,151 multi-turn); execution accuracy over grounded table operations.

Test splits are disjoint from all training data. Pairing one long-horizon embodied task with two symbolic-reasoning families is deliberate: if co-evolution only helped within one task family, the cross-domain averages would expose it.

Training pipeline. All learnable methods use Qwen2.5-7B-Instruct, trained in two stages. **Cold-start supervised fine-tuning (SFT):** since the instruct checkpoint has not learned the skill-conditioned format and direct RL gives many malformed rollouts, for each domain we sample 300 tasks, retrieve the seed subgraph, and generate four rollouts per task with a Qwen2.5-14B-Instruct editor; trajectories that are correct and invoke ≥ 1 skill are kept ($\approx$ 1.8k). We fine-tune one epoch at 5×10^{-6} (cosine, 0.1 warm-up), stopping before the distribution collapses; this teaches the format

Table 1: **Main results across five benchmarks in three domains (%).** Open models use Qwen2.5-7B-Instruct, share the cold-start SFT checkpoint and GRPO budget, and are evaluated with one episode per task; closed models are zero-shot. ToRA and FINDER are evaluated only on their target domains and excluded from the average. Best per column in **bold**.

Method	ALFWorld	GSM8K	MATH	FinQA	ConvFinQA	Avg
GPT-4o (zero-shot)	48.0	**94.0**	62.0	72.0	78.0	70.8
Gemini-2.5-Pro (zero-shot)	60.3	91.5	58.0	69.5	74.5	70.8
ReAct	35.0	71.3	33.5	52.0	55.0	49.4
Reflexion	42.7	76.5	39.0	56.5	60.0	54.9
Mem0+ReAct	33.6	79.0	42.5	59.0	63.5	55.5
GRPO (vanilla)	77.6	86.0	50.0	63.0	67.0	68.7
SkillRL (static skills)	89.9	89.5	55.0	67.5	71.5	74.7
EvoSkill (content only)	88.0	90.5	57.5	70.0	74.0	76.0
ToRA (math only)	–	84.0	58.0	–	–	–
FINDER (finance only)	–	–	–	75.3	81.9	–
CoEvoSkill (ours)	**92.5**	92.5	**63.5**	**77.0**	**83.0**	**81.7**

without changing the library. **Co-evolution:** from this checkpoint we run GRPO ($G = 8$, temperature 0.7) for 3 phases of 5 epochs, the graph fixed within and evolved between epochs, progressively unlocking L_0–$L_1/L_2/L_3$, starting from 45 domain-general and 40 task-specific seeds.

Reward. All RL methods share an outcome reward: $r = 1$ when the task is completed (goal condition in ALFWorld; exact answer match on the four reasoning benchmarks) and $r = 0$ otherwise, with a -0.01 term per invalid action in ALFWorld to discourage malformed commands. The same reward is used for cold-start filtering and GRPO, so comparisons are not confounded by reward design.

Evaluation protocol. Each result uses one episode per task at a fixed temperature with no answer-level majority voting or self-consistency (Wang et al., 2023b; Wei et al., 2022), and the same instruction for all models; within an episode the guarded MCTS controller uses a fixed per-decision budget. Closed models are zero-shot and open models share the cold-start checkpoint and GRPO budget. We report execution accuracy on the financial datasets, their standard metric, which keeps columns comparable even where they differ from vendor numbers that use model-specific templates and repeated sampling. Learned results average three seeds with standard deviation below 1.5, so the gaps exceed run-to-run noise.

Baselines. We compare with the main alternatives: (i) GPT-4o (OpenAI, 2024) and Gemini-2.5-Pro (Google DeepMind, 2025), zero-shot; (ii) ReAct (Yao et al., 2023), reasoning and acting without skills; (iii) Reflexion (Shinn et al., 2023), verbal self-reflection across attempts; (iv) Mem0+ReAct (Mem0, 2025), structured memory; (v) GRPO (vanilla), policy RL with the same reward but no skills; (vi) SkillRL, skill-augmented RL with a static set; (vii) EvoSkill (Alzubi et al., 2026), evolving content without a graph; and (viii) domain-specific ToRA (Gou et al., 2024) (math) and FINDER (Khatuya et al., 2025) (finance).

4.2 Main Results

Table 1 reports the results. CoEvoSkill attains the highest score on four of five benchmarks and the highest average of 81.7, which is $+5.7$ over the strongest learnable baseline EvoSkill, $+7.0$ over SkillRL, and $+13.0$ over vanilla GRPO. GPT-4o keeps the lead on GSM8K, where single-trajectory accuracy is already near saturation. The per-benchmark breakdown for the learned-skill methods is given in Appendix F.

Three observations follow. First, ALFWorld shows the limits of zero-shot prompting on a non-restartable long-horizon task: GPT-4o and Gemini-2.5-Pro reach 48.0 and 60.3 and the prompt-/memory-based methods 33.6–42.7, as one mistaken action is unrecoverable, whereas every environment-trained policy clears 77 (vanilla GRPO 77.6, CoEvoSkill 92.5). This column reflects in-environment learning rather than model scale.

Table 2: **Ablation study (five-benchmark average, %).** The right column is the drop from the full system.

Variant	Avg	Δ
Full CoEvoSkill	**81.7**	–
w/o niche decomposition	77.1	-4.6
w/o symbiosis edges (co-occurrence)	78.8	-2.9
w/o co-evolution scheduler (fixed 50/50)	79.2	-2.5
w/o niche-vacancy insertion	79.9	-1.8
graph evolution only (no micro layer)	74.1	-7.6
content refinement only (no macro layer)	73.1	-8.6

Second, the edge over content-only refinement grows with chained skills: EvoSkill margins are $+9.0/+7.0/+6.0$ on ConvFinQA/FinQA/MATH but $+2.0/+4.5$ on GSM8K/ALFWorld. Third, GRPO $<$ SkillRL $<$ EvoSkill $<$ CoEvoSkill shows static skills help, evolving content more, both most, while CoEvoSkill beats ToRA and FINDER in one transferable library.

4.3 Ablation Study

Evolving only the graph costs -7.6: static text leaves edges pointing to skills that cannot fulfill the assigned role, and content only costs slightly more (-8.6), as a fixed graph caps gains. Both single-layer drops exceed the largest component drop (-4.6), so the layers contribute through interaction rather than additively, consistent with Theorem 1.

Removing niche decomposition is costliest (-4.6): uniform budgets edit workflows like wrappers, over-editing primitives. Raw co-occurrence costs -2.9 through substitute edges, a fixed scheduler -2.5, and vacancy insertion -1.8 but populates high-level roles.

4.4 Mechanism Analysis

Ecosystem evolution. Over 15 epochs levels diverge from $\ell = 1$-clustered seeds (entropy $0.62 \to 1.41$): $\ell = 0$ wrappers (epochs 1–5), $\ell = 2$ workflows (6–10), $\ell = 3$ meta-skills (after 10); 11 substituting finance pairs ($\sigma = 0.9$) stay unlinked while complementary pairs ($\sigma = 3.2$) are kept.

Convergence. Δ_t peaks at the $\ell = 2$ transition and falls below 10^{-3} by epoch 12; five more epochs change the score by only 0.3 with fewer than two edges each, while a fixed schedule delays convergence by about four epochs.

Library properties. The library holds 78 ± 6 skills (below its 85 start and EvoSkill's 128 ± 12), about 30% shorter (180 ± 40 vs. 260 ± 55 tokens). The MATH chain normalization ($\ell = 1$) $\to$ substitution ($\ell = 2$) $\to$ validation ($\ell = 1$), $\sigma = 2.9/3.1$, is retrieved in part on simpler problems, unlike EvoSkill's monolithic solver.

5 Conclusion

We presented CoEvoSkill, which evolves skill content and graph topology as one coupled system via niche-conditioned refinement, symbiosis-driven edges, and a tension-based scheduler, with a provable stationary point under bounded edits and strict gates. Across five benchmarks in three domains it raises the average to 81.7, beating the strongest content-evolving baseline by 5.7 and vanilla RL by 13.0, with the largest gains on compositional tasks; agents gain as much from organizing knowledge as from accumulating it.

Limitations. The proposer and gates add LLM cost; the guarantee assumes a contractive topology and may fail under aggressive insertion, the niche is hand-designed, and simulated benchmarks leave non-stationary deployment open.

Broader impact. CoEvoSkill is a compact, model-agnostic text artifact; gates and human review remain important to avoid propagating bias.

Reproducibility Statement

We describe the complete training and evaluation pipeline, including all hyperparameters and the data-processing protocol, in Appendix A and Appendix B. The assumptions and a full proof of Theorem 1 are in Appendix E, and representative retrieval and edit prompts are in Appendix G. The skill library, task adapters, guards, and co-evolution code will be released as an anonymous supplementary archive, and all benchmarks are public with fixed test splits never used for adaptation.

Ethics Statement

This work uses only public, previously released benchmarks and collects no personal data. The deployed artifact is text, and the framework neither calls nor gains access to restricted resources beyond the tools defined by each task adapter. Because evolved skills could encode errors or domain bias, all content revisions pass validation gates and are logged for inspection, and we recommend human review before deployment. We do not foresee adverse environmental or safety impacts beyond those standard for LLM-agent research.

AI Use Statement

We used a generative language assistant to help draft prose and write utility scripts. The research questions, method and its formalization, experimental design, and analysis were developed by the authors, and all AI-assisted text and code were independently checked. The authors take full responsibility for the final content of this work.

References

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems, 2026.

Akhil Bagaria, J. K. Senthil, and George Konidaris. Skill discovery for exploration and planning using deep skill graphs. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 521–531, 2021.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Samana Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Kenneth Huang, Bryan R. Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3697–3711, 2021.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. ConvFinQA: Exploring the chain of numerical reasoning in conversational finance question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6279–6292, 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Google DeepMind. Gemini 2.5: Our most intelligent AI model. Google DeepMind Blog, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujie Yang, Minlie Huang, Nan Duan, and Weizhu Chen. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations (ICLR)*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Subhendu Khatuya, Shashwat Naidu, Pawan Goyal, and Niloy Ganguly. Program of thoughts for financial reasoning: Leveraging dynamic in-context examples and generative retrieval. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025. arXiv:2510.13157.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Xiaoyuan Li, Moxin Li, Keqin Bao, Yubo Ma, Wenjie Wang, Dayiheng Liu, and Fuli Feng. SKILLGRAPH: Skill-augmented reinforcement learning for agents via evolving skill graphs, 2026.

Mem0. Mem0: Building production-ready AI agents with scalable long-term memory, 2025.

OpenAI. Hello GPT-4o. OpenAI Blog, 2024.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller LLMs stronger problem-solvers. In *International Conference on Learning Representations (ICLR)*, 2025.

Lingfei Qian, Yan Wang, Xueqing Peng, Weipeng Zhou, Yi Han, Yilun Zhao, Jimin Huang, Qianqian Xie, and Jian-Yun Nie. Fin-o1: On the transferability of reasoning-enhanced LLMs and reinforcement learning to finance, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a.

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. In *International Conference on Learning Representations (ICLR)*, 2023b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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, 2026.

Shunyu Yao, Yu Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. Plan4MC: Skill reinforcement learning and planning for open-world minecraft tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

A Hyperparameters and Implementation Details

Table 3 lists the full configuration. Training-stage values follow Section 4.1; controller and gate parameters were selected on the adaptation-selection set and held fixed across benchmarks and seeds.

Table 3: Full configuration for CoEvoSkill.

Component	Symbol	Value
Base policy	π_θ	Qwen2.5-7B-Instruct
Editor / proposal model	—	Qwen2.5-14B-Instruct
Cold-start tasks per domain	—	300
Rollouts per cold-start task	—	4
Retained demonstrations	—	≈ 1.8k
SFT epochs / learning rate	—	1 / 5×10^{-6}
LR schedule / warm-up	—	cosine / 0.1
GRPO group size	G	8
Sampling temperature	—	0.7
Phases $\times$ evolution epochs	—	3×5
Policy learning rate	—	1×10^{-6}
PPO clip range	ε_c	0.2
KL coefficient	λ_{KL}	0.04
Initial general / task seeds	—	45 / 40
Edit budget intercept / slope	b_0/b_1	24 / 48 tokens
Maximum edit budget	$B_{\max}$	96 tokens
Evaluation count intercept	m_0	4
Evaluation count slopes	m_1/m_2	4 / 4
Aggregate gain / regression tolerance	δ/ξ	0.5 / 0.3 points
Edge EMA rate / cap	$\eta/\sigma_{\max}$	0.2 / 4.0
Edge add / remove thresholds	—	1.5 / 0.6
Minimum joint successes	—	8
Ordering threshold for prerequisite	θ_{order}	0.8
Drift EMA rate	κ	0.3
Scheduler gain / offset	β/γ	10 / -1
Scheduler probability bounds	$f_{\min}/f_{\max}$	0.25 / 0.85
Search simulations per decision	B_s	12
Local / explore candidate caps	$k_{\mathrm{local}}/k_{\mathrm{explore}}$	6 / 3
Traversal depth (hops)	—	2
PUCT constant	c_{puct}	1.5
Graph-weight rank coefficient	λ_g	0.3
Smoothing constant	ϵ	10^{-3}

Progressive unlocking. The three GRPO phases unlock skill levels in order. Phase 1 admits only L_0, L_1 nodes, phase 2 adds L_2 workflows, and phase 3 adds L_3 cross-domain meta-skills. This prevents high-level compositions from being assembled before the atomic and single-step procedures they rely on are stable, keeping early guard failures interpretable.

B Data Processing and Evaluation Protocols

Table 4 summarizes each benchmark, including split sizes, the success criterion, and representative seeds. Training tasks come from the official training partitions; a fixed adaptation-selection subset is split off before training and is used only to accept revisions and select checkpoints.

Table 4: Benchmark details and representative seed skills.

Benchmark	Test size	Success criterion	Representative seeds
ALFWorld	134	Goal condition met	navigate-to, pick-and-place, clean-and-place
GSM8K	1,319	Exact final number	decompose-quantity, arithmetic-check
MATH	5,000	Exact `\boxed` answer	equation-normalization, systematic-substitution, solution-validation
FinQA	1,147	Program execution match	table-extraction, numerical-reasoning, calculator-execution
ConvFinQA	1,151	Program execution match	conversation-state-tracking, table-extraction, numerical-reasoning

Financial execution accuracy. For FinQA and ConvFinQA the model emits the operation sequence over grounded table cells, and the prediction is scored by executing that sequence and comparing the result to the annotated program output. This metric tolerates different surface forms that compute the same value and is the standard metric for these datasets.

Embodied evaluation. Each ALFWorld test task runs once with a fixed action budget and succeeds only if the simulator goal condition is satisfied. Invalid commands incur the small shaping penalty but do not end the episode; guard rejections prevent illegal state changes.

C Niche and Symbiosis Computation

Dependency depth and influence. $\mathrm{dep}(v)$ is the length of the longest directed prerequisite path ending at v, computed by memoized traversal of the acyclic prerequisite subgraph. $\mathrm{inf}(v)$ counts distinct prerequisite descendants within two hops along outgoing edges. Both are recomputed after each accepted graph revision and clipped to fixed scales before normalization.

Co-activation diversity. Let $d(u) \in \mathcal{D}_{\mathrm{dom}}$ be the fixed domain label of a co-activation neighbor u. Then

$$\mathrm{div}(v) = -\sum_{d} \widehat{p}(d \mid \mathrm{Nbr}_{\mathrm{co}}(v)) \log \widehat{p}(d \mid \mathrm{Nbr}_{\mathrm{co}}(v)), \tag{13}$$

which is zero for an empty neighborhood and is normalized by $\log \max(2, |\mathcal{D}_{\mathrm{dom}}|)$.

Symbiosis estimation. Counts $n_t^+(u)$, $n_t^+(u, v)$, and N_t^+ accumulate over committed, successful trajectories within the current phase window. Pairs below the minimum joint successes do not update an edge. The edge weight follows Eq. equation 7, which smooths the estimate and caps the contribution of a single window.

D Search Controller and Training Details

This section collects the value-head, GRPO, and auxiliary-learning details kept compact in Section 3.6.

Value heads and backup. The value model has separate continuation and terminal heads, $q_{\mathrm{cont}}(s,v), q_{\mathrm{stop}}(s,v) \in [0,1]$. A newly expanded continuation uses the leaf estimate

$$\begin{aligned} V_\psi(s') &= \sum_{u \in C(s')} P_\phi(u \mid s') q_{\mathrm{type}(u)}(s', u), \\ z_{\mathrm{leaf}} &= 2V_\psi(s') - 1. \end{aligned} \tag{14}$$

A guard-valid terminal action uses $z = 2r - 1$ when task feedback $r \in [0,1]$ is available through the permitted environment interface, and $z = 2q_{\mathrm{stop}}(s,v) - 1$ otherwise. An empty-support leaf or exhausted budget receives -1. Each state–skill edge represents one sampled realization, so its value refers to that realization rather than an average over all realizations; search statistics are empirical estimates, not certified values.

GRPO objective. Let h_{ij} be the context preceding generated token a_{ij}, including the selected skill, and define $\varrho_{ij}(\theta) = \pi_\theta(a_{ij} \mid h_{ij}) / \pi_{\theta_{\mathrm{old}}}(a_{ij} \mid h_{ij})$. With $D_{ij}^{\mathrm{KL}} = D_{\mathrm{KL}}(\pi_\theta(\cdot \mid h_{ij}) \| \pi_{\mathrm{ref}}(\cdot \mid h_{ij}))$, the GRPO objective is

$$\begin{aligned} \ell_{ij}(\theta) &= \min\{\varrho_{ij} \widehat{A}_i, \mathrm{clip}(\varrho_{ij}, 1-\varepsilon_c, 1+\varepsilon_c) \widehat{A}_i\}, \\ \mathcal{J}_{\mathrm{GRPO}}(\theta) &= \mathbb{E}\left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{T_i} \sum_{j=1}^{T_i} \left(\ell_{ij}(\theta) - \lambda_{\mathrm{KL}} D_{ij}^{\mathrm{KL}} \right) \right], \end{aligned} \tag{15}$$

where T_i counts agent-generated tokens and π_{ref} is the fixed initial language policy. Tool observations and skill documents are conditioning context, not optimization targets, and since P_ϕ is frozen in this phase the skill-selection probabilities cancel in the likelihood ratio. The token-wise clipped expression is the GRPO surrogate rather than the full trajectory importance ratio. This trains the skill-conditioned reasoner under direct sampling and is not an unbiased estimator of the MCTS-mediated objective in Eq. equation 1.

Auxiliary policy and value learning. After the GRPO phase, freeze π_θ and collect fresh searches. Let $C_{\mathrm{live}}(s)$ contain root actions with guard-valid realizations; their visits provide policy targets

$$\begin{aligned} \widehat{\pi}(v \mid s) &= \frac{\mathbf{1}\{v \in C_{\mathrm{live}}(s)\} N(s,v)}{\sum_{u \in C_{\mathrm{live}}(s)} N(s,u)}, \\ \mathcal{L}_{\mathrm{policy}} &= -\mathbb{E}_s \sum_{v \in C(s)} \widehat{\pi}(v \mid s) \log P_\phi(v \mid s). \end{aligned} \tag{16}$$

States without a valid visited action or with fewer than two candidates are omitted. Rejected actions retain negative value targets; continuation targets map signed backed-up returns to $[0,1]$, which with bootstrapping are value targets rather than calibrated success probabilities. Terminal targets use the smoothed fraction of correct answers among K training realizations, $(n_{\mathrm{correct}} + 1)/(K+2)$, counting guard failures as unsuccessful. Each value head minimizes binary cross-entropy on its own targets, duplicate records are consolidated, and visit weights are capped. These losses update ϕ, ψ, not θ, and controller updates use the skill texts and candidate sets from the library version that generated each target.

E Proof of Theorem 1

We restate the assumptions in full and prove both parts for a fixed executor Ξ and panel within a single phase.

Assumptions. **(A1) Bounded edits.** Every micro edit has token-level edit distance at most $b_t(v)$, and every macro revision repairs at most a fixed, prespecified number of skills, so the per-step change to the library is bounded. **(A2) Strict gates.** A revision is accepted only under Eq. equation 5, which raises $\widehat{J}$ by at least $\delta > 0$ and bounds each affected-stratum regression by ξ. **(A3) Contractive topology.** The topology update is a contraction on the normalized niche metric: there exists $\rho < 1$ such that the niche displacement induced by a macro revision is at most ρ times the displacement that produced it.

Part (i): finite accepted revisions and a stationary library. Under A2 each accepted revision raises the deterministic score by at least δ. After K acceptances,

$$\widehat{J}(\mathcal{L}_K) \geq \widehat{J}(\mathcal{L}_0) + K\delta. \tag{17}$$

Because $\widehat{J} \leq 1$, we require $\widehat{J}(\mathcal{L}_0) + K\delta \leq 1$, hence

$$K \leq \left\lfloor \frac{1 - \widehat{J}(\mathcal{L}_0)}{\delta} \right\rfloor. \tag{18}$$

The number of accepted revisions is therefore finite. At this bound no revision satisfying the aggregate-gain condition remains; combined with the regression condition, no micro or macro proposal passes the gate, so the library is stationary. It remains to show the process does not move away through rejected or topological motion. Under A3, denote by z_t the niche displacement at step t; a macro revision induces $z_{t+1} \leq \rho z_t$ with $\rho < 1$, so the displacement contracts to zero and the updates settle rather than oscillate without bound.

Part (ii): weak dominance over single-layer alternatives. Let $\mathcal{P}_{\mathrm{ct}}$ be the content-only proposals and $\mathcal{P}_{\mathrm{gr}}$ the graph-only proposals. Every micro proposal is available to the joint process and every macro proposal is as well, so $\mathcal{P}_{\mathrm{ct}} \cup \mathcal{P}_{\mathrm{gr}}$ is contained in the joint proposal set. Because A2 makes the accepted score monotone and bounds regression on every affected stratum, the joint process can replay either single-layer sequence and reach a score at least as high. Thus, at the joint stationary point,

$$J(\mathcal{L}^\star) \geq J(\mathcal{L}^\star_{\mathrm{ct}}), \qquad J(\mathcal{L}^\star) \geq J(\mathcal{L}^\star_{\mathrm{gr}}). \tag{19}$$

If at least one coupled revision changing both content and topology is accepted, it realizes a gain unavailable to either single-layer process, making the corresponding inequality strict.

Stochastic-evaluation caveat. When evaluation is stochastic, $\widehat{J}$ is an empirical estimate and the monotonicity argument applies to that estimate rather than the population objective. Eq. equation 5, with stratified coverage and the regression tolerance, acts as an empirical safeguard rather than a population-level guarantee. The result is also stated for a fixed executor within a phase and does not assert convergence across GRPO rounds, when the executor changes.

F Per-Benchmark Ablations

Table 5 reports each variant on every benchmark, not only the average in Table 2. The pattern is consistent across domains: removing either full layer causes the largest drop on each benchmark, while component removals cause smaller, domain-dependent losses.

Table 5: Per-benchmark ablation results (%).

Variant	ALF.	GSM	MATH	FinQA	CFin
Full CoEvoSkill	92.5	92.5	63.5	77.0	83.0
w/o niche	87.9	88.2	58.6	72.6	78.2
w/o symbiosis	89.8	90.6	60.2	74.1	79.3
w/o scheduler	90.1	90.9	60.8	74.8	79.4
w/o vacancy	90.7	91.2	61.4	75.5	80.5
Graph only	85.2	85.0	54.0	69.6	76.8
Content only	84.0	84.6	52.6	68.2	76.0

G Prompts, Guard Checks, and Example Seeds

Editor prompt structure. The editor receives, in order: (1) the current skill text and identifier; (2) the level ℓ and niche vector, with the instruction to preserve the abstraction and input/output contract; (3) representative failed and successful realizations and the violated guard check; (4) the contracts of prerequisites and affected neighbors; and (5) the token edit budget. It returns only an insertion, replacement, or deletion within the budget.

Guard checklist. Each task adapter specifies the concrete checks in Eq. equation 12: schema and argument types; existence and type of any referenced object or table cell; executable preconditions; admissible state changes; duplicate actions under unchanged state; and mechanically checkable contract constraints. A realization is committed only if every check passes.

Example seed skill (MATH). The *equation-normalization* skill ($\ell = 1$) instructs the agent to rewrite the stated relations into a standard form, name each unknown once, and record which quantity the question asks for, before any substitution is attempted. It is a prerequisite for *systematic-substitution* and is retrieved independently on problems that require only normalization.