ECOLOGICAL SKILL LIBRARIES: CO-EVOLVING SKILLS, ROUTERS, AND LIBRARY TOPOLOGY

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents increasingly rely on reusable skills that encode procedural knowledge, executable operations, and task-specific guidance. As skill libraries grow, their performance depends on skill quality, routing competition, context consumption, redundancy, conflicts, and complementary interactions. Existing approaches primarily optimize skills and routing decisions separately, which can misestimate each skill's marginal contribution and reduce library performance when the candidate population changes. We introduce ECOSKILL, an ecological framework that models a skill library as a co-adapting community. Given a task distribution and an execution budget, ECOSKILL jointly evolves skill content, applicability triggers, routing policies, library topology, and lifecycle decisions. Its central mechanism assigns ecological credit based on each skill's marginal contribution to community performance, incorporating task utility, context and routing costs, unique coverage, and inter-skill synergy. The framework estimates this credit using sampled coalition counterfactuals and approximate Shapley-style attribution, avoiding exhaustive evaluation of skill coalitions. The resulting signal coordinates skill mutation, retention, migration, merging, and extinction while preserving specialized and complementary capabilities. We evaluate ECOSKILL across libraries of increasing scale, controlled skill corruptions, and task-distribution shifts using task performance, routing accuracy, context consumption, and conflict rate. This formulation maintains adaptive skill libraries by selecting members according to their contribution to the agent ecosystem.

1 INTRODUCTION

Large language model (LLM) agents use reusable skills to encode procedural knowledge, executable operations, and task-specific guidance. Skills extend atomic tools by packaging multi-step workflows and supporting artifacts into modules that agents retrieve and apply at inference time. This abstraction allows agents to acquire reusable capabilities through autonomous exploration and refinement, including web-interaction procedures and structured multi-file packages (Zheng et al., 2025) (Alzubi et al., 2026) (Zhang et al., 2026). As agents continuously generate skills, the main challenge shifts from constructing individual skills to maintaining an effective *skill library*. A large library must provide useful procedures, retrieve the appropriate subset within a limited context budget, prevent redundant or conflicting instructions from competing for attention, and adapt to shifts in the task distribution.

Recent studies highlight this challenge. SkillsBench reports that curated skills improve aggregate agent performance and that focused bundles can outperform larger or exhaustive bundles, indicating that adding procedural content does not consistently improve performance (Li et al., 2026). At registry scale, routing becomes a separate bottleneck because topically similar candidates can implement different functions. SkillRouter evaluates an approximately 80K-skill pool and finds that accurate routing requires information that can be absent from short skill metadata (Zheng et al., 2026). Skill-learning methods also discover APIs through environment interaction, refine skills based on execution failures, and iteratively evolve structured packages (Zheng et al., 2025) (Alzubi et al., 2026) (Zhang et al., 2026). These approaches focus primarily on creating, improving, or retrieving individual skills, selecting from a fixed candidate pool, or optimizing a complete agent program. They do not directly optimize the interactions among skills in a growing library.

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.

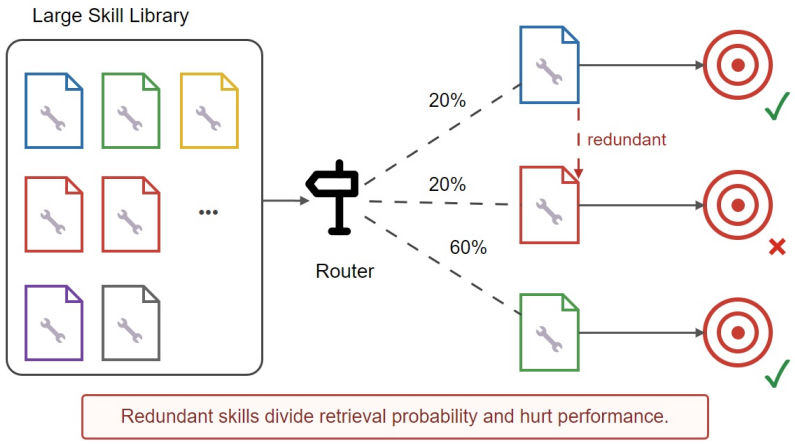

Figure 1: Overview of the ECOSKILL framework.

Skill coexistence introduces several externalities. Redundant skills can divide retrieval probability, broad triggers can suppress specialized skills, and individually valid instructions can conflict when exposed jointly. An auxiliary skill with modest standalone utility can also be indispensable when combined with complementary skills. Each retrieved skill consumes context capacity, so its value depends on the task and the other skills included in the prompt. Standalone success, invocation frequency, and independent validation gain can therefore misestimate a skill's marginal contribution to the library. These effects become stronger as the library grows: the router operates over a changing candidate set, skill triggers affect the data observed by other skills, and selection decisions shape the future population. Skill-library optimization consequently requires a community-level objective that attributes utility under interaction, accounts for routing interference and context consumption, preserves complementary capabilities, and jointly adapts the library structure and its members.

We introduce ECOSKILL, an *Ecological Skill Library* framework that models skills as a co-adapting community, as illustrated in Figure 1. Given a task distribution and an execution budget, ECOSKILL jointly evolves skill content, applicability triggers, the routing policy, library topology, and lifecycle decisions. Its central mechanism, *ecological credit*, evaluates each skill through its marginal effect on community performance. The credit combines task utility with explicit costs for context consumption and routing interference, while accounting for unique task coverage and complementary interactions. Exact attribution over all skill coalitions is combinatorial, so ECOSKILL estimates these effects using sampled coalition counterfactuals and approximate Shapley-style attribution. For small libraries, the estimates are calibrated against exact calculations. The resulting credit coordinates mutation, retention, migration, and extinction. Redundant or harmful skills can be merged, revised, or removed, whereas specialized skills are retained when their unique coverage or synergy justifies their cost. Joint updates to the router, skill population, and topology align the library with the skills that the agent can reliably retrieve and use.

We evaluate libraries containing tens to hundreds of skills under a fixed agent and execution budget. The evaluation compares independent top-k selection, static routing, jointly trained routing, and ecological co-evolution on clean libraries and libraries with controlled injections of duplicate, stale, conflicting, and adversarial skills. We measure task performance, routing accuracy, context consumption, and conflict rate. We also evaluate adaptation to task-distribution shifts through skill migration, extinction, and recovery. The ablations remove synergy credit, router evolution, topology adaptation, lifecycle control, and coalition-based attribution. This protocol evaluates whether community-level selection preserves useful diversity and controls library-induced interference as the skill population grows.

Our contributions are:

- We formulate large-scale skill-library learning as ecological optimization, in which skill value depends on routing competition, context cost, unique coverage, and inter-skill interactions.

- We propose ECOSKILL, which jointly adapts skill content, applicability triggers, routing, library topology, and skill lifecycles through community-level selection.
- We develop an ecological credit-assignment mechanism based on coalition counterfactuals and approximate Shapley-style attribution to distinguish redundant, harmful, uniquely useful, and synergistic skills at tractable computational cost.
- We define an evaluation protocol covering library scaling, controlled skill corruption, system-level costs and conflicts, adaptation to distribution shifts, and ablations of the credit, routing, topology, and lifecycle mechanisms.

2 RELATED WORK

2.1 REUSABLE SKILLS AND EXPERIENCE-BASED AGENT ADAPTATION

Tool-augmented language models interleave reasoning with external actions through modular routing (Karpas et al., 2022), self-supervised API use (Schick et al., 2023), and combined reasoning and acting (Yao et al., 2023). Instruction-tuned systems extend these capabilities across heterogeneous APIs (Li et al., 2023) (Qin et al., 2023), with retrieval-aware invocation, REST planning, and foundation-model orchestration (Patil et al., 2023) (Song et al., 2023) (Shen et al., 2023). These systems generally treat tools as externally specified atomic functions rather than evolving reusable procedures.

Experience-based methods instead retrieve demonstrations and compose programs (Paranjape et al., 2023), construct task-specific tools (Yuan et al., 2023), store reflective feedback (Shinn et al., 2023), extract transferable lessons (Zhao et al., 2023), and accumulate executable code or workflows (Wang et al., 2023a) (Wang et al., 2024). Recent systems explicitly acquire, package, refine, verify, or recursively evolve skills (Zheng et al., 2025) (Alzubi et al., 2026) (Zhang et al., 2026) (Xia et al., 2026). However, evaluation often remains skill- or task-local, although focused bundles can outperform exhaustive collections (Li et al., 2026).

2.2 RETRIEVAL, ROUTING, AND CONDITIONAL MODULAR COMPUTATION

Skill selection extends information retrieval from relevant text to executable capabilities. Representative approaches include sparse lexical matching (Robertson & Zaragoza, 2009), dense retrieval (Karpukhin et al., 2020), late interaction (Khattab & Zaharia, 2020), and hypothetical-document expansion (Gao et al., 2022). Conventional retrievers may nevertheless struggle with heterogeneous tool corpora (Shi et al., 2025), while hypothetical tool descriptions can reduce query–description mismatch (Sengupta et al., 2026). At larger scales, body-aware retrieve-and-rerank architectures improve discrimination among similar skills (Zheng et al., 2026).

Routed skills also consume context and affect planning. Retrieval-augmented generation is sensitive to retrieved quantity and relevance (Lewis et al., 2020), long-context utilization varies with position (Liu et al., 2023), and additional demonstrations may not improve performance monotonically (Lu et al., 2021) (Wang et al., 2023b). Analogously, mixture-of-experts systems require balancing and capacity controls because routing shapes module utilization (Shazeer et al., 2017) (Lepikhin et al., 2020) (Fedus et al., 2021), including constrained assignment (Lewis et al., 2021) and expert-choice routing (Zhou et al., 2022). Thus, top-k relevance alone does not capture redundancy or complementarity among jointly activated modules.

2.3 COALITIONAL ATTRIBUTION, DIVERSITY, AND POPULATION-LEVEL ADAPTATION

The Shapley value allocates coalition utility through expected marginal contributions (Shapley, 1953). Machine-learning applications value training data using approximations (Ghorbani & Zou, 2019), exact computations for k-nearest-neighbor predictors (Jia et al., 2019), alternative coalition-size weighting (Kwon & Zou, 2022), and distribution-relative valuation (Ghorbani et al., 2020). Cheaper local alternatives include influence functions (Koh & Liang, 2017), trajectory-based gradient similarity (Pruthi et al., 2020), and reinforcement-learned data selection (Yoon et al., 2019), but these may not preserve coalitional allocation under higher-order interactions.

Population-based optimization preserves repertoires through behavioral novelty (Lehman & Stanley, 2011), niche-specific elites (Mouret & Clune, 2015), and diversity-enabled adaptation (Cully et al.,

2015). Environment generation and cross-niche transfer (Wang et al., 2019) and online selection with hyperparameter mutation (Jaderberg et al., 2017) provide further mechanisms for maintaining heterogeneous populations, although fitness is typically assigned to individual solutions rather than their marginal contributions within jointly routed libraries.

3 METHOD

EcoSkill jointly optimizes a skill ecosystem comprising skill content, activation triggers, a task-conditioned router, and a directed topology. It assigns ecological credit to each skill based on task utility, context consumption, routing interference, unique coverage, and complementarity.

3.1 PROBLEM FORMULATION AND FRAMEWORK

Let $\mathcal{D}$ denote a distribution over agent tasks. Each task $x \sim \mathcal{D}$ specifies an observation, an execution environment, and a verifier that returns a bounded utility $u(x, \tau) \in [0, 1]$ for trajectory τ. We fix the base agent π_θ and optimize its external skill ecosystem, modifying its procedural knowledge without changing θ.

At round t, the skill library is $\mathcal{L}^{(t)} = \{s_i^{(t)}\}_{i=1}^{n_t}$, where each skill is represented as $s_i^{(t)} = (c_i^{(t)}, g_i^{(t)}, a_i^{(t)}, \ell_i^{(t)})$. Here, $c_i^{(t)}$, $g_i^{(t)}$, $a_i^{(t)} \in \mathbb{N}$, and $\ell_i^{(t)} \in \mathbb{N}$ denote the content, activation trigger, age, and remaining lifetime of skill $s_i^{(t)}$, respectively. The ecosystem further includes a router $r_{\phi^{(t)}}$ and a directed topology $\mathcal{G}^{(t)} = (\mathcal{L}^{(t)}, \mathcal{E}^{(t)})$. An edge (s_i, s_j) represents a dependency, refinement, or composition relation.

Given task x, the router samples a feasible coalition according to

$$q_{\phi^{(t)}}(S \mid x, \mathcal{G}^{(t)}) = r_{\phi^{(t)}}(x, \mathcal{L}^{(t)}, \mathcal{G}^{(t)}), \quad S \subseteq \mathcal{L}^{(t)}, \quad C_{\mathrm{ctx}}(S) \le B, \tag{1}$$

where $C_{\mathrm{ctx}}(S)$ is the context cost and B is the context budget. The fixed agent is then conditioned on the coalition serialized according to the topology, yielding $\tau \sim \pi_\theta\big(\cdot \mid x, \mathrm{Serialize}(S, \mathcal{G}^{(t)})\big)$.

EcoSkill alternates between evaluation and variation. During evaluation, it records routed coalitions, trajectories, verifier outcomes, and context usage in $\mathcal{H}^{(t)}$. Coalition utilities determine ecological credits, which guide coordinated updates to skill content, triggers, routing, topology, and lifecycle variables. EcoSkill generates candidates on an evolution split and retains them through budget-constrained community selection on a held-out split. Inference uses the final library and router. Figure 2 illustrates the resulting framework and its co-evolution loop.

3.2 ECOLOGICAL CREDIT ASSIGNMENT

For task x and coalition S, define

$$U_x(S) = \mathbb{E}_{\tau \sim \pi_\theta(\cdot \mid x, \mathrm{Serialize}(S, \mathcal{G}))}[u(x, \tau)] - \lambda_{\mathrm{ctx}} \frac{C_{\mathrm{ctx}}(S)}{B}. \tag{2}$$

This utility accounts for competition among skills for the context budget available at deployment. For $s_i \notin S$, the counterfactual contribution of s_i is $\Delta_i(x, S) = U_x(S \cup \{s_i\}) - U_x(S)$.

A broad trigger can divert routing probability from suitable skills even when the corresponding skill is rarely executed. Let $p_\phi(s_j \mid x; \mathcal{L}, \mathcal{G})$ denote the marginal inclusion probability of s_j. The routing interference caused by s_i is

$$I_i = \mathbb{E}_{x \sim \mathcal{D}_{\mathrm{evo}}} \left[\sum_{j \neq i} w_{ij}(x) \left| p_\phi(s_j \mid x; \mathcal{L}, \mathcal{G}) - p_\phi(s_j \mid x; \mathcal{L} \setminus \{s_i\}, \mathcal{G}_{-i}) \right| \right], \tag{3}$$

where $\mathcal{G}_{-i}$ is obtained by removing s_i, and $w_{ij}(x) \ge 0$ weights the relevance overlap between s_i and s_j.

For a threshold $\epsilon > 0$, the unique coverage of s_i is

$$Q_i = \mathbb{E}_{x \sim \mathcal{D}_{\mathrm{evo}}} \left[\mathbb{I}\left(\max_{S : s_i \in S} U_x(S) - \max_{S : s_i \notin S} U_x(S) > \epsilon \right) \right]. \tag{4}$$

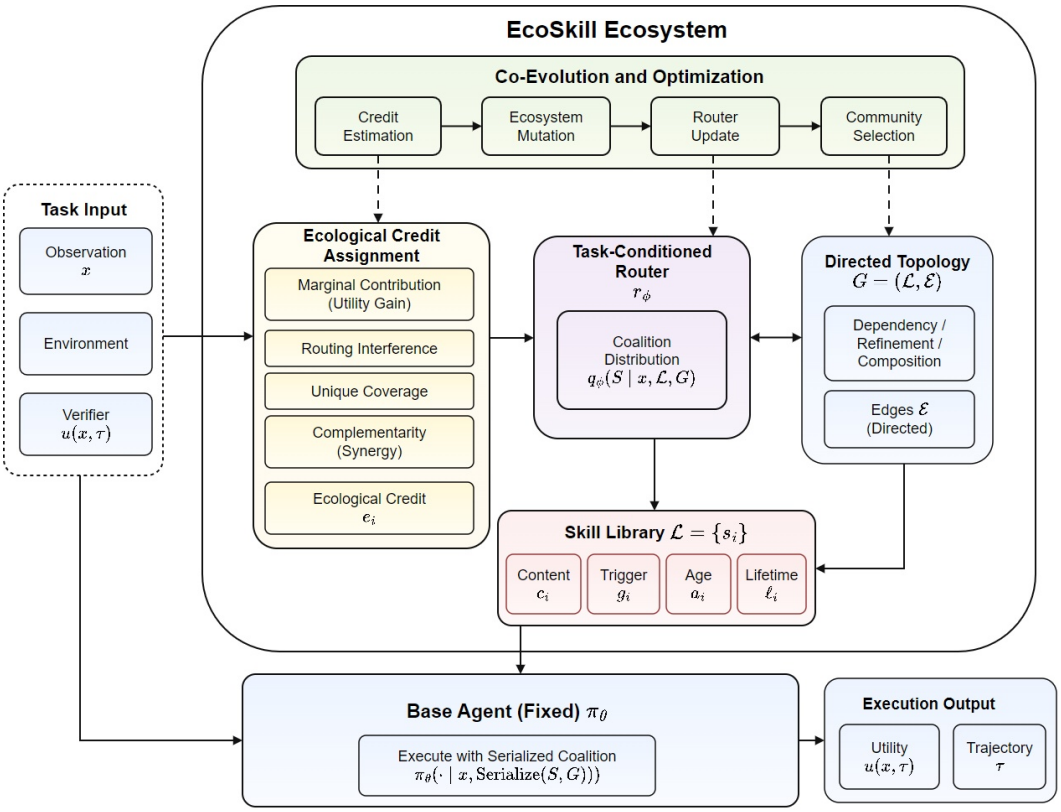

Figure 2: Overview of the EcoSkill framework and ecosystem co-evolution process.

Under the coalition-sampling distribution ρ_x, the pairwise complementarity between s_i and s_j is

$$Y_{ij} = \mathbb{E}_{\substack{x\sim\mathcal{D}_{\text{evo}} \\ S\sim\rho_x,\ s_i,s_j\notin S}} \left[U_x(S\cup\{s_i,s_j\}) - U_x(S\cup\{s_i\}) - U_x(S\cup\{s_j\}) + U_x(S)\right]. \tag{5}$$

Positive values indicate complementarity, whereas negative values indicate redundancy or conflict. The ecological credit combines these quantities:

$$e_i = \mathbb{E}_{\substack{x\sim\mathcal{D}_{\text{evo}} \\ S\sim\rho_x}}[\Delta_i(x,S)] - \lambda_{\text{int}} I_i + \lambda_{\text{cov}} Q_i + \lambda_{\text{syn}} \sum_{j\neq i} \omega_{ij} [Y_{ij}]_+, \tag{6}$$

where $[z]_+ = \max(z, 0)$ and the nonnegative weights ω_{ij} sum to one over $j \neq i$. Because U_x includes the context penalty, the ecological credit applies no additional penalty for context usage.

Exact evaluation over all coalitions has exponential complexity. Given M sampled task–permutation pairs and the predecessor set $P_i^{(m)}$, EcoSkill estimates the marginal contribution as

$$\widehat{\varphi}_i = \frac{1}{M} \sum_{m=1}^{M} \left[U_{x^{(m)}}(P_i^{(m)} \cup \{s_i\}) - U_{x^{(m)}}(P_i^{(m)})\right]. \tag{7}$$

The same sampled coalitions estimate the quantities in Equations (3)–(5). The sampling procedure is provided in the Appendix.

3.3 Co-Evolution and Optimization

Let $\mathcal{Z}^{(t)} = (\mathcal{L}^{(t)}, \phi^{(t)}, \mathcal{G}^{(t)})$ denote the ecosystem state at round t. EcoSkill generates a candidate state through

$$\widetilde{\mathcal{Z}}^{(t+1)} \sim \mathcal{M}\Big(\mathcal{Z}^{(t)}, \mathcal{H}^{(t)}, \{\widehat{e}_i^{(t)}\}_{i=1}^{n_t}\Big). \tag{8}$$

Method	Knowledge Work ↑	Web Process ↑	Code Ops ↑	Avg. SR ↑	RouteR ↑	Conflict ↓
Static-Router	47.8±0.9	39.6±1.1	45.1±0.8	44.2±0.7	62.4±1.0	11.8±0.7
Independent-TopK	50.2±0.8	41.7±1.0	46.8±0.9	46.3±0.8	65.7±0.9	10.5±0.6
Best-of-N	51.1±0.9	42.5±1.2	47.4±1.0	47.0±0.9	66.3±1.1	9.9±0.8
Joint-Router	52.0±0.7	44.1±1.0	48.5±0.8	48.6±0.7	70.8±0.8	8.1±0.5
EvoSkill-Style	53.7±0.8	<u>46.2±1.1</u>	<u>49.4±0.9</u>	<u>49.3±0.8</u>	<u>72.1±0.9</u>	<u>7.6±0.5</u>
Oracle-Small	<u>54.0±0.6</u>	45.8±0.8	49.1±0.7	49.6±0.6	71.4±0.7	6.9±0.4
EcoSkill	**56.1±0.7**	**49.0±0.9**	**51.9±0.8**	**52.5±0.7**	**77.6±0.8**	**4.3±0.3**

Table 1: Results with 250 available skills. Values are percentages reported as mean ± standard deviation over five seeds. Learned methods use matched execution and inference-token budgets. Bold denotes the best result; underlining denotes the second-best learned result. Abbreviations: Avg. = Average; SR = Success Rate.

Failures and successful counterexamples guide procedural edits, specialization, skill creation, merging, and removal. Content updates include corresponding trigger updates. EcoSkill adds topology links between complementary skills and uses negative interactions and near-duplicate skills to guide edge removal and node contraction.

Coalition preferences $\mathcal{P}^{(t)}$ update the router by minimizing

$$\mathcal{L}_{\text{route}}(\phi) = -\mathbb{E}_{(x,S^+,S^-)\sim\mathcal{P}^{(t)}}\left[\log\sigma\big(\log q_\phi(S^+ \mid x,\mathcal{G}) - \log q_\phi(S^- \mid x,\mathcal{G})\big)\right], \tag{9}$$

where $U_x(S^+) > U_x(S^-)$.

Each surviving skill ages by one round, and its remaining lifetime is updated as

$$\ell_i^{(t+1)} = \text{clip}\Big(\ell_i^{(t)} - 1 + \eta_\ell \mathbb{I}[\widehat{e}_i^{(t)} > \delta_{\text{surv}}], 0, \ell_{\max}\Big). \tag{10}$$

An expired skill becomes eligible for removal unless its deletion substantially reduces unique coverage or disconnects a dependency used by a positively credited coalition.

Community selection maximizes

$$J(\mathcal{Z}) = \mathbb{E}_{x\sim\mathcal{D}_{\text{val}}}\mathbb{E}_{S\sim q_\phi(\cdot|x,\mathcal{G})}[U_x(S)] - \lambda_{\text{size}}\frac{|\mathcal{L}|}{N_{\max}} - \lambda_{\text{edge}}\frac{|\mathcal{E}|}{E_{\max}}, \tag{11}$$

subject to

$$\mathcal{Z}^\star \in \arg\max_{\mathcal{L},\phi,\mathcal{G}} J(\mathcal{L},\phi,\mathcal{G}), \quad |\mathcal{L}| \le N_{\max}, \quad |\mathcal{E}| \le E_{\max}, \quad C_{\text{ctx}}(S) \le B. \tag{12}$$

The alternating optimization procedure estimates ecological credits, proposes discrete ecosystem edits, updates the router, and selects candidates on $\mathcal{D}_{\text{val}}$. Training terminates after a fixed budget or a prescribed number of rounds without improvement. The appendix specifies the full procedure.

At inference, the router selects

$$S^\star(x) = \arg\max_{\substack{S\subseteq\mathcal{L}^\star \\ C_{\text{ctx}}(S)\le B}} q_{\phi^\star}(S \mid x, \mathcal{G}^\star), \tag{13}$$

and the agent executes with $\text{Serialize}(S^\star(x), \mathcal{G}^\star)$. Counterfactual attribution and ecosystem mutation are used only during evolution.

4 Experiments

We evaluate whether optimizing the skill library as an interacting ecosystem improves agent performance compared with optimizing independently scored modules. The evaluation measures task success under matched budgets, robustness to library growth and pollution, the contributions of ecological credit and joint adaptation, and recovery from shifts in the task distribution.

Skills	Average success rate (%) Static	TopK	Joint	EcoSkill	EcoSkill diagnostics Ctx (k) ↓	Conflict ↓	Cov. ↑	Recovery ↑
10	43.7±0.8	44.0±0.7	44.3±0.8	44.5±0.7	1.3	2.1	43.2	71.4
50	46.4±0.9	47.0±0.8	48.1±0.8	49.0±0.7	1.6	2.8	67.5	78.9
100	45.5±1.0	47.6±0.9	49.1±0.8	51.1±0.8	1.8	3.4	78.1	83.7
250	44.2±0.7	46.3±0.8	48.6±0.7	**52.5±0.7**	2.0	4.3	86.8	88.6
500	40.3±1.2	44.1±1.1	47.5±0.9	51.8±0.9	2.1	5.1	89.4	87.2

Table 2: Scaling with the available library size. Coverage measures held-out task families with at least one successfully invoked specialist; recovery is measured after distribution shift. Abbreviations: Ctx = Context; Cov. = Coverage.

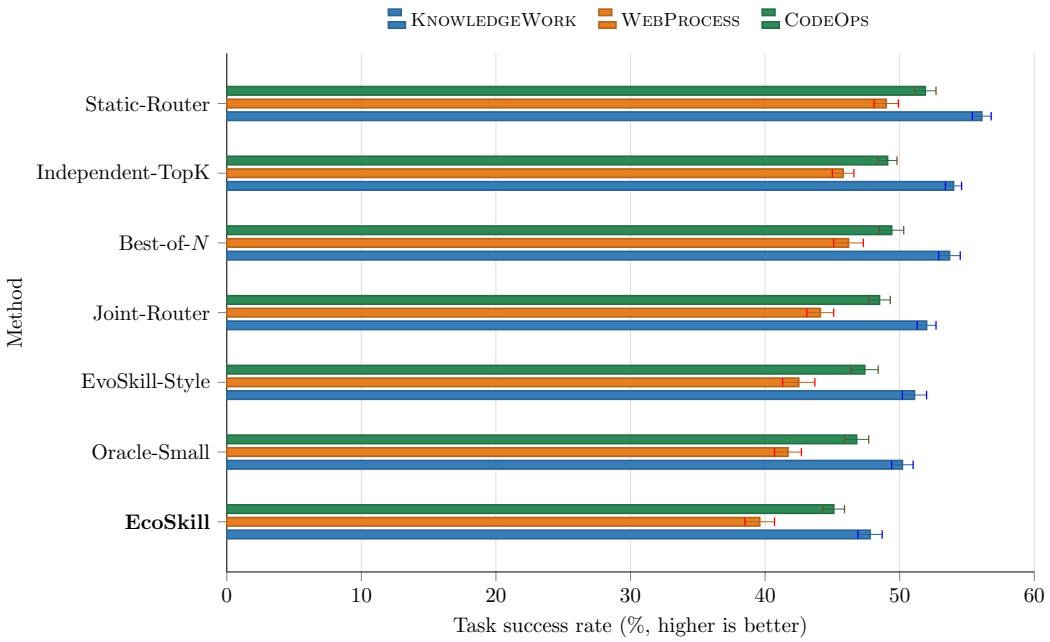

Figure 3: Task success rates with 250 available skills under matched execution and inference-token budgets. Bars show the mean over five randomized seeds, and error bars show one standard deviation.

4.1 Experimental Setup

We evaluate three suites. KnowledgeWork contains 120 grounded document, spreadsheet, and data-analysis tasks supported by procedural skills and scripts. WebProcess contains 100 multi-step browser tasks requiring navigation, state tracking, and composition of site-specific operations. CodeOps contains 80 repository-level maintenance and debugging tasks involving planning, search, testing, and patching. Each task has a deterministic verifier and a fixed execution limit. We divide each suite into disjoint evolution, validation, and test partitions using a 50/20/30 ratio. Test trajectories never update skill content, triggers, routing parameters, or topology.

The partitioning procedure is performed at the task-family level, limiting overlap in templates, repositories, websites, and document schemas across splits. Evolution observes representative feedback without encountering test cases. Validation tasks guide checkpoint selection and ecological updates, while the held-out test partitions are used for final measurement.

Each ecosystem starts with 10 valid skills and expands to 50, 100, 250, or 500 candidates through skill discovery and distractor insertion. We compare EcoSkill with five matched-budget methods: Independent-TopK, which ranks skills by standalone validation utility; Static-Router, a frozen embedding retriever; Joint-Router, which trains retrieval while holding skills and topology fixed;

BEST-OF-N, which retains the best proposed library; and EVOSKILL-STYLE, which mutates skills based on execution failures but omits coalition credit and routing externalities. ORACLE-SMALL serves as a reference by selecting the best validation-informed fixed library containing at most 20 skills. All systems use the same backbone, tools, environment-step limit, and cumulative inference-token budget.

The primary metric is task success rate (SR). We also report skill-level routing recall (ROUTER), retrieved skill-context tokens (CTX), semantic conflict among retrieved instructions (CONFLICT), and successful specialist coverage over held-out task families (COV). Results are reported as means and standard deviations across five randomized seeds. We assess the comparisons using paired bootstrap resampling over test tasks with 10,000 replicates.

To keep comparisons aligned, token accounting includes skill descriptions, retrieved instructions, intermediate reasoning, and repair attempts. Environment actions that terminate unsuccessfully still consume the shared execution allowance. We compute routing recall against the set of skills used by successful reference trajectories, whereas conflict records whether retrieved instructions prescribe incompatible actions, tools, or output formats.

4.2 OVERALL PERFORMANCE

Table 1 reports results with 250 available skills, and Figure 3 shows the task success rates across the three task suites. ECOSKILL achieves an average success rate of 52.5%, exceeding INDEPENDENT-TOPK by 6.2 percentage points and EVOSKILL-STYLE by 3.2 points. The difference from EVOSKILL-STYLE is statistically significant under paired bootstrap testing ($p < 0.01$). ECOSKILL also increases routing recall from 72.1% to 77.6% and reduces conflicts from 7.6% to 4.3%. These results associate the success-rate improvement with stronger individual skills and more reliable selection of compatible skill coalitions.

The reduction in retrieved context and instruction conflict is important because larger candidate sets do not provide more capability. Instead, the results indicate that ecological scoring favors combinations whose procedures can be executed coherently within the budget. This interpretation is consistent with the higher specialist coverage observed.

The improvements span all three task suites. Compared with EVOSKILL-STYLE, ECOSKILL improves success by 2.4 points on KNOWLEDGEWORK, 2.8 points on WEBPROCESS, and 2.5 points on CODEOPS. The largest gain occurs on browser workflows, which require operations to be composed with navigation and validation. The ORACLE-SMALL remains competitive on KNOWLEDGEWORK but trails ECOSKILL by 2.9 points on average because its fixed inventory lacks specialists for some task families. INDEPENDENT-TOPK retains a broad inventory but has more than twice the conflict rate of ECOSKILL. The ecological library maintains broad coverage while presenting the agent with a smaller, compatible skill subset.

Coalition credit preserves conditionally useful skills. Across seeds, 18% of retained skills have below-median standalone utility and positive ecological credit. These skills implement validation, formatting adaptation, or state transfer and contribute when paired with domain procedures. Independent ranking favors broadly triggered modules, including modules that provide competing instructions.

4.3 SCALING, POLLUTION, AND ADAPTATION

Library growth reveals differences between ecological optimization and flat retrieval. Table 2 shows performance across methods with 10 skills, when most candidates are relevant. STATIC-ROUTER peaks at 50 skills and loses 6.1 points as the library grows to 500. INDEPENDENT-TOPK peaks at 100 skills and deteriorates at larger sizes. ECOSKILL improves from 44.5% with 10 skills to 52.5% with 250 skills, followed by a 0.7-point decrease at 500 skills. Its per-episode context grows from 1.3k to 2.1k tokens because topology-aware filtering narrows the candidate community before routing. Over the same range, coverage increases from 43.2% to 89.4%.

The decline beyond 250 candidates suggests that filtering does not entirely eliminate the cost of additional distractors. Nevertheless, the stable trajectory shows that expansion remains beneficial while newly discovered specialists add distinct coverage. The scaling results therefore distinguish library capacity from prompt size: more skills can remain available without all being exposed locally.

We replace 30% of the 250-skill inventory with duplicates, redundant paraphrases, outdated procedures, conflicting instructions, or malicious objective-overriding skills. Compared with JOINT-ROUTER, the advantage of ECOSKILL increases from 3.9 points without pollution to 5.1 points with duplicates and 5.6 points with redundant skills. Under conflicting instructions, ECOSKILL retains 48.7% success, compared with 40.8% for JOINT-ROUTER. Under malicious injection, ECOSKILL achieves 46.9% success. After evolution, 91% of exact duplicates and 84% of outdated skills are removed or assigned negligible retrieval probability. Conflicting skills that remain useful are separated into weakly connected communities. Malicious skills cause failures, and the increase in conflict rate from 4.3% at 250 skills to 5.1% at 500 skills indicates that approximate attribution leaves residual interference in libraries.

Ablations show that each ecological component contributes to performance. At 250 skills, replacing coalition credit with invocation counts reduces success from 52.5% to 47.7% and increases conflict from 4.3% to 8.2%. Removing synergy credit reduces success by 2.9 points, while removing uniqueness credit decreases coverage from 86.8% to 77.2%. Freezing the router or topology reduces success by 3.7 and 3.1 points, respectively. Removing lifetime protection reduces success by 1.8 points and disproportionately removes rare specialists. These results separate the contribution of the complete ECOSKILL mechanism from skill evolution or router training alone.

We change the task mixture after generation 15 by reducing common code-maintenance tasks and increasing spreadsheet, noisy web-retrieval, and cross-application workflows. Success decreases immediately from 52.1% to 40.6% and reaches 50.8% within eight generations, recovering 88.6% of the lost performance. JOINT-ROUTER and INDEPENDENT-TOPK recover to 45.9% and 43.8%, respectively. The adaptation involves renewed use of low-frequency specialists, new links between spreadsheet and document communities, and retirement of skills specific to the previous task mixture.

Following the shift, ecological updates first change routing probabilities and community connections before replacing skill content. This ordering allows specialists to be recombined for the mixture. Incomplete recovery reflects the finite adaptation horizon and interference from noisy candidates, rather than cost-free transfer.

At inference time, ECOSKILL retrieves 5.2 skills and uses 2.0k skill-context tokens per task, compared with 7.6 skills and 3.4k tokens for INDEPENDENT-TOPK. Topology filtering adds 70 ms to routing, while the shorter prompt reduces median end-to-end episode latency from 84.6 s to 80.9 s. Evolution requires 38.4 GPU-hours. Under the evaluated settings, community-level credit and joint adaptation preserve complementary coverage while controlling routing and context externalities. Performance peaks at 250 skills, and adversarial library contamination continues to reduce task success.

5 CONCLUSION

We presented ECOSKILL, an ecological framework that jointly optimizes skill content, applicability triggers, routing, library topology, and lifecycles through coalition-based ecological credit assignment. By attributing community-level utility while accounting for context cost, routing interference, unique coverage, and synergy, ECOSKILL enables principled management of growing skill libraries. Experiments across varying library scales, controlled skill corruption, and task-distribution shifts confirmed that ecological co-evolution preserves useful diversity, controls interference, and adapts library structure effectively. Ablation results further established the integral roles of ecological credit, router evolution, topology adaptation, lifecycle control, and coalition-based attribution in achieving these capabilities.

LIMITATIONS

Our evaluation uses three task suites with deterministic verifiers, which may not fully represent open-ended settings where success criteria are subjective or evolving. Ecological credit relies on sampled coalition counterfactuals, so its estimates may exhibit variance for rarely routed skills or higher-order interactions. In addition, the experiments fix the base agent and evaluate libraries of up to 500 skills; further study is needed to characterize behavior with different backbone models and substantially larger, continuously changing skill ecosystems.

STATEMENT ON AI USE

In this work, AI language models were used exclusively to assist with writing refinement. We applied these tools to improve the readability of the manuscript by enhancing sentence structure, eliminating redundancy, and ensuring grammatical correctness. The research content—including the ecological skill library design, theoretical motivation, experimental methodology, and all reported findings—was developed solely by the authors. We did not employ AI tools for any research tasks, including generating scientific ideas, creating figures, or interpreting experimental outcomes. All AI-suggested text changes were reviewed by the authors before acceptance. The authors bear full responsibility for this manuscript.

REFERENCES

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026. URL https://arxiv.org/abs/2603.02766.

Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 2015. URL https://doi.org/10.1038/nature14422.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021. URL https://arxiv.org/abs/2101.03961.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*, 2022. URL https://arxiv.org/abs/2212.10496.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. URL https://proceedings.mlr.press/v97/ghorbani19c.html.

Amirata Ghorbani, Michael Kim, and James Zou. A distributional framework for data valuation. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. URL https://proceedings.mlr.press/v119/ghorbani20a.html.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017. URL https://arxiv.org/abs/1711.09846.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards efficient data valuation based on the shapley value. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, 2019. URL https://proceedings.mlr.press/v89/jia19a.html.

Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445*, 2022. URL https://arxiv.org/abs/2205.00445.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020. URL https://arxiv.org/abs/2004.04906.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. *arXiv preprint arXiv:2004.12832*, 2020. URL https://arxiv.org/abs/2004.12832.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017. URL https://arxiv.org/abs/1703.04730.

Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, 2022. URL https://proceedings.mlr.press/v151/kwon22a.html.

Joel Lehman and Kenneth O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 2011. URL https://doi.org/10.1162/EVCO_a_00025.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020. URL https://arxiv.org/abs/2006.16668.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. *arXiv preprint arXiv:2103.16716*, 2021. URL https://arxiv.org/abs/2103.16716.

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. *arXiv preprint arXiv:2005.11401*, 2020. URL https://arxiv.org/abs/2005.11401.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*, 2023. URL https://arxiv.org/abs/2304.08244.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han chung Lee, and Dawn Song. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026. URL https://arxiv.org/abs/2602.12670.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023. URL https://arxiv.org/abs/2307.03172.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021. URL https://arxiv.org/abs/2104.08786.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015. URL https://arxiv.org/abs/1504.04909.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*, 2023. URL https://arxiv.org/abs/2303.09014.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023. URL https://arxiv.org/abs/2305.15334.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracing gradient descent. *arXiv preprint arXiv:2002.08484*, 2020. URL https://arxiv.org/abs/2002.08484.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023. URL https://arxiv.org/abs/2307.16789.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 2009. URL https://doi.org/10.1561/1500000019.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023. URL https://arxiv.org/abs/2302.04761.

Saptarshi Sengupta, Zhengyu Zhou, Jun Araki, Xingbo Wang, Bingqing Wang, Suhang Wang, and Zhe Feng. Tooldreamer: Instilling llm reasoning into tool retrievers. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2026. URL https://aclanthology.org/2026.eacl-long.254/.

L. S. Shapley. A value for n-person games. In *Contributions to the Theory of Games*. 1953. URL https://doi.org/10.1515/9781400881970-018.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. URL https://arxiv.org/abs/1701.06538.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *arXiv preprint arXiv:2303.17580*, 2023. URL https://arxiv.org/abs/2303.17580.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. *arXiv preprint arXiv:2503.01763*, 2025. URL https://arxiv.org/abs/2503.01763.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, Ye Tian, and Sujian Li. Restgpt: Connecting large language models with real-world restful apis. *arXiv preprint arXiv:2306.06624*, 2023. URL https://arxiv.org/abs/2306.06624.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a. URL https://arxiv.org/abs/2305.16291.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019. URL https://arxiv.org/abs/1901.01753.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning. *arXiv preprint arXiv:2301.11916*, 2023b. URL https://arxiv.org/abs/2301.11916.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024. URL https://arxiv.org/abs/2409.07429.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026. URL https://arxiv.org/abs/2602.08234.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Jinsung Yoon, Sercan O. Arik, and Tomas Pfister. Data valuation using reinforcement learning. *arXiv preprint arXiv:1909.11671*, 2019. URL https://arxiv.org/abs/1909.11671.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R. Fung, Hao Peng, and Heng Ji. Craft: Customizing llms by creating and retrieving from specialized toolsets. *arXiv preprint arXiv:2309.17428*, 2023. URL https://arxiv.org/abs/2309.17428.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue (Steve) Liu, Xiaoxiao Li, and Philip S. Yu. Coevoskills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026. URL https://arxiv.org/abs/2604.01687.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2023. URL https://arxiv.org/abs/2308.10144.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025. URL https://arxiv.org/abs/2504.07079.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. Skillrouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 2026. URL https://arxiv.org/abs/2603.22455.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/2f00ecd787b432c1d36f3de9800728eb-Abstract-Conference.html.

APPENDIX

A PERMUTATION-BASED ESTIMATION OF ECOLOGICAL MARGINAL CREDIT

This section defines the sampling and aggregation procedure for estimating marginal credit. The procedure operates on a fixed ecosystem state $\mathcal{Z} = (\mathcal{L}, \phi, \mathcal{G})$, where $\mathcal{L} = \{s_i\}_{i=1}^{n}$. Fixing the ecosystem state ensures that changes in estimated utility result from coalition membership.

For a task x and coalition $S \subseteq \mathcal{L}$, define the sampled coalition utility as

$$\widehat{U}_x(S) = u(x, \tau_S) - \lambda_{\text{ctx}} \frac{C_{\text{ctx}}(S)}{B}, \qquad \tau_S \sim \pi_\theta(\cdot \mid x, \text{Serialize}(S, \mathcal{G}))\,. \tag{14}$$

Its expectation over the sampled trajectory equals the utility used for ecological credit assignment:

$$\mathbb{E}_{\tau_S}\left[\widehat{U}_x(S)\right] = U_x(S). \tag{15}$$

When the evaluation procedure samples multiple trajectories for the same (x, S) pair, $\widehat{U}_x(S)$ is their empirical mean combined with the same deterministic context penalty. This aggregation leaves the objective and penalty unchanged.

For each Monte Carlo sample $m \in \{1, \ldots, M\}$, we sample a task $x^{(m)} \sim \mathcal{D}_{\text{evo}}$ and a permutation $\pi^{(m)}$ of the n skills. Let $\text{pos}_{\pi^{(m)}}(i) \in \{1, \ldots, n\}$ denote the position of s_i in $\pi^{(m)}$. The predecessor coalition of s_i is

$$P_i^{(m)} = \{s_j \in \mathcal{L} : \text{pos}_{\pi^{(m)}}(j) < \text{pos}_{\pi^{(m)}}(i)\}. \tag{16}$$

The set $P_i^{(m)}$ contains exactly the skills preceding s_i in the sampled permutation. The sampled marginal contribution of s_i is

$$\widehat{\Delta}_i^{(m)} = \widehat{U}_{x^{(m)}}\left(P_i^{(m)} \cup \{s_i\}\right) - \widehat{U}_{x^{(m)}}\left(P_i^{(m)}\right), \tag{17}$$

and its final estimate is

$$\widehat{\varphi}_i = \frac{1}{M} \sum_{m=1}^{M} \widehat{\Delta}_i^{(m)}. \tag{18}$$

The two utility terms in Equation (17) are evaluated on the same task and differ only in whether the coalition contains s_i. Expanding the context-cost contribution gives

$$\begin{aligned} \widehat{\Delta}_i^{(m)} = u\Big(x^{(m)}, \tau_{P_i^{(m)} \cup \{s_i\}}\Big) - u\Big(x^{(m)}, \tau_{P_i^{(m)}}\Big) \\ - \frac{\lambda_{\text{ctx}}}{B}\left[C_{\text{ctx}}\Big(P_i^{(m)} \cup \{s_i\}\Big) - C_{\text{ctx}}\Big(P_i^{(m)}\Big)\right]. \end{aligned} \tag{19}$$

Thus, $\widehat{\varphi}_i$ already accounts for context consumption, which requires no additional subtraction when computing ecological credit.

We retain the sampled coalitions and their evaluated utilities in a shared evaluation record. Each permutation induces the nested sequence

$$S_0^{(m)} = \varnothing, \qquad S_k^{(m)} = S_{k-1}^{(m)} \cup \left\{s_{\pi_k^{(m)}}\right\}, \quad k \in \{1, \ldots, n\}, \tag{20}$$

where $\pi_k^{(m)}$ is the skill at position k. For $s_i = s_{\pi_k^{(m)}}$, we have $P_i^{(m)} = S_{k-1}^{(m)}$ and $P_i^{(m)} \cup \{s_i\} = S_k^{(m)}$. The $n + 1$ utility evaluations along this sequence therefore yield all n marginal contributions for a task–permutation sample.

The record $\mathcal{R}$ associates each sampled coalition with its task and utility estimate. When the required coalitions occur in the record, the same evaluations can be reused to estimate the routing-interference, unique-coverage, and pairwise-complementarity terms. Aggregating these quantities according to their definitions yields the ecological credit used by the ecosystem mutation operator:

$$\widehat{e}_i = \widehat{\varphi}_i - \lambda_{\text{int}}\widehat{I}_i + \lambda_{\text{cov}}\widehat{Q}_i + \lambda_{\text{syn}} \sum_{j \neq i} \omega_{ij}[\widehat{Y}_{ij}]_+. \tag{21}$$

All quantities in Equation (21) are computed under the fixed ecosystem state $(\mathcal{L}, \phi, \mathcal{G})$. The ecosystem mutation operator then uses the resulting credits. These credits remain fixed while traversing each sampled permutation.

B Environment Details

B.1 Datasets and Task Settings

We evaluate on three complementary agent-task suites comprising 300 tasks in total. KnowledgeWork contains 120 grounded document-processing, spreadsheet-manipulation, and data-analysis tasks that require multi-step operations over local files and structured artifacts. WebProcess contains 100 browser-based tasks involving navigation, state tracking, form interaction, information extraction, and composition of site-specific procedures. CodeOps contains 80 repository-level maintenance and debugging tasks that require code search, test execution, fault localization, and patch generation. Each task is paired with a deterministic executable verifier that maps the final environment state to a binary success signal.

Each suite is partitioned by task family into disjoint evolution, validation, and test splits using a 50/20/30 ratio. The evolution split is used to generate skill candidates, estimate ecological credit, and

Algorithm 1 Permutation-Based Estimation of Ecological Marginal Credit

Require: Fixed ecosystem $(\mathcal{L}, \phi, \mathcal{G})$ with $\mathcal{L} = \{s_i\}_{i=1}^{n}$; evolution-task distribution $\mathcal{D}_{\text{evo}}$; fixed agent π_θ; verifier utility u; context cost C_{ctx}; budget B; penalty coefficient λ_{ctx}; number of task–permutation samples M

Ensure: Marginal-credit estimates $\{\widehat{\varphi}_i\}_{i=1}^{n}$ and the evaluated coalition record

Initialize $A_i \leftarrow 0$ for every $i \in \{1, \dots, n\}$
Initialize the coalition record $\mathcal{R} \leftarrow \varnothing$
for $m = 1$ to M **do**
    Sample $x^{(m)} \sim \mathcal{D}_{\text{evo}}$
    Sample a permutation $\pi^{(m)} = (\pi_1^{(m)}, \dots, \pi_n^{(m)})$ of $\{1, \dots, n\}$
    Set $S_0^{(m)} \leftarrow \varnothing$
    Sample $\tau_0^{(m)} \sim \pi_\theta(\cdot \mid x^{(m)}, \text{Serialize}(S_0^{(m)}, \mathcal{G}))$
    Compute $\widehat{U}_0^{(m)} \leftarrow u(x^{(m)}, \tau_0^{(m)}) - \lambda_{\text{ctx}} C_{\text{ctx}}(S_0^{(m)})/B$
    Add $(x^{(m)}, S_0^{(m)}, \widehat{U}_0^{(m)})$ to $\mathcal{R}$
    for $k = 1$ to n **do**
        Set $i \leftarrow \pi_k^{(m)}$
        Set $S_k^{(m)} \leftarrow S_{k-1}^{(m)} \cup \{s_i\}$
        Sample $\tau_k^{(m)} \sim \pi_\theta(\cdot \mid x^{(m)}, \text{Serialize}(S_k^{(m)}, \mathcal{G}))$
        Compute $\widehat{U}_k^{(m)} \leftarrow u(x^{(m)}, \tau_k^{(m)}) - \lambda_{\text{ctx}} C_{\text{ctx}}(S_k^{(m)})/B$
        Update $A_i \leftarrow A_i + \widehat{U}_k^{(m)} - \widehat{U}_{k-1}^{(m)}$
        Add $(x^{(m)}, S_k^{(m)}, \widehat{U}_k^{(m)})$ to $\mathcal{R}$
    end for
end for
Set $\widehat{\varphi}_i \leftarrow A_i/M$ for every $i \in \{1, \dots, n\}$
Terminate after processing all M task–permutation samples and all n positions in each permutation **return** $\{\widehat{\varphi}_i\}_{i=1}^{n}$ and $\mathcal{R}$

update skill content, triggers, routing parameters, topology, and lifecycle variables. The validation split is used for community selection, early stopping, and hyperparameter selection. The test split is accessed only after evolution terminates and does not contribute to skill mutation, router training, topology construction, or model selection. Family-level splitting prevents near-duplicate task templates and environment instances from appearing across splits.

All methods start from the same library of 10 manually validated seed skills. We construct libraries of 50, 100, 250, and 500 available skills through task-driven skill discovery and controlled distractor insertion. For the pollution experiments, 30% of the 250-skill inventory is replaced by an equal mixture of exact duplicates, semantically redundant paraphrases, outdated procedures, conflicting instructions, and objective-overriding skills. Distribution-shift experiments modify the evolution-task mixture after generation 15 by down-weighting common code-maintenance tasks and increasing spreadsheet, noisy web-retrieval, and cross-application workflows.

Tasks are presented to the agent using a fixed system prompt containing the tool specifications, execution constraints, and selected skills serialized in topological order. Each skill consists of a natural-language applicability trigger, a procedural instruction body, and optional executable scripts or templates. The agent is instructed to reason step by step internally, invoke tools when necessary, and terminate with a concise final response after producing the required environment state. We impose a maximum of 30 environment actions for KNOWLEDGEWORK, 40 browser actions for WEBPROCESS, and 40 shell or editor actions for CODEOPS. Episodes exceeding the corresponding limit are marked unsuccessful. To reduce contamination risk, task-specific identifiers, verifier logic, held-out inputs, and reference outputs are excluded from skill-generation prompts and router training data. Test-task trajectories are not retained in the evolutionary history.

B.2 PREPROCESSING AND EVALUATION

We canonicalize all task inputs before execution. Text fields are Unicode-normalized, line endings are converted to Unix format, and inconsequential leading and trailing whitespace is removed. Document and spreadsheet tasks are evaluated from the generated artifact rather than the surface form of the final message. Spreadsheet values are compared after type normalization, with numeric cells considered equal when their absolute or relative error is at most 10^{-6}. Code outputs are evaluated in isolated repositories using the task-provided test command, with generated caches, timestamps, and formatting-only metadata excluded from comparison. Web tasks are scored from the final browser state and task-specific environment predicates. When a task admits multiple valid outcomes, the verifier accepts any member of the predefined equivalence set or any output satisfying the same executable constraints.

The primary metric is task success rate,

$$\mathrm{SR} = \frac{1}{|\mathcal{D}_{\mathrm{test}}|} \sum_{x \in \mathcal{D}_{\mathrm{test}}} \mathbb{I}[v_x(\tau_x) = 1] \,,$$

where v_x is the deterministic verifier and τ_x is the complete agent trajectory. We additionally report routing recall (ROUTER), defined as the fraction of tasks for which the retrieved coalition contains at least one verifier-confirmed relevant skill; retrieved skill-context tokens (CTX), measured using the backbone tokenizer; conflict rate (CONFLICT), defined as the percentage of episodes whose selected coalition contains a pair of instructions with mutually incompatible actions or constraints; and specialist coverage (COV), defined as the percentage of held-out task families for which at least one specialized skill is both retrieved and used in a successful trajectory. Recovery after distribution shift is computed as

$$\mathrm{Recovery} = 100 \times \frac{\mathrm{SR}_{\mathrm{recovered}} - \mathrm{SR}_{\mathrm{post\text{-}shift}}}{\mathrm{SR}_{\mathrm{pre\text{-}shift}} - \mathrm{SR}_{\mathrm{post\text{-}shift}}}.$$

We compare ECOSKILL with STATIC-ROUTER, INDEPENDENT-TOPK, BEST-OF-N, JOINT-ROUTER, EVOSKILL-STYLE, and ORACLE-SMALL. All learned methods receive identical task trajectories, environment-step limits, cumulative inference-token budgets, and maximum library sizes. Stochastic decoding, candidate generation, coalition sampling, and population initialization are controlled by five independently randomized seeds, $\{13, 21, 42, 87, 100\}$. We report the mean and standard deviation across seeds. Statistical significance is assessed by paired bootstrap resampling over test tasks with 10,000 replicates; differences are considered significant at $p < 0.05$, and no correction is applied to the prespecified primary comparison between ECOSKILL and EVOSKILL-STYLE.

B.3 IMPLEMENTATION AND HARDWARE

The fixed base agent is `Qwen2.5-32B-Instruct`, deployed locally in bfloat16 precision with tensor parallelism over four NVIDIA A100 80 GB GPUs. The same backbone, tokenizer, system prompt, tool interfaces, and decoding configuration are used by all methods. The backbone executes task trajectories and generates proposed skill contents, trigger revisions, merge candidates, and topology edits. Its parameters remain frozen throughout evolution. Candidate generation uses temperature 0.7, top-$p = 0.9$, and a maximum of 1,024 newly generated tokens per proposal. Test-time task execution uses greedy decoding to reduce evaluator variance, with a maximum context length of 32,768 tokens and a maximum of 4,096 generated tokens per episode. No external model API is used.

The router is initialized from `BAAI/bge-large-en-v1.5`. Task queries and skill triggers are embedded separately, and the resulting cosine scores define the initial candidate neighborhood. A two-layer MLP with hidden size 1,024 scores task–skill pairs after concatenating the task embedding, skill embedding, age, normalized context cost, and topology features. The router is optimized with the pairwise logistic objective in Eq. (9) using AdamW, a learning rate of 2×10^{-5}, weight decay of 0.01, batch size 64, gradient clipping at 1.0, and 500 update steps per generation. The encoder is frozen during the first five generations and subsequently fine-tuned jointly with the scoring head. Static routing retains the frozen initial encoder and scores throughout evaluation.

At inference time, routing first keeps the 32 highest-scoring candidates and then performs topology-aware coalition selection. The maximum branching factor is 4, the maximum dependency-expansion depth is 3, and at most 8 skills may be placed in a coalition. The serialized skill budget is $B = 4{,}096$

Table 3: Component ablations with 250 available skills. Values are percentages reported as mean ± standard deviation over five seeds. Abbreviations: Ret. = Retained.

Method	Avg. SR ↑	RouteR ↑	Conflict ↓	Cov. ↑	Ctx. (k) ↓	Ret. Skills
EcoSkill	$\mathbf{52.5 \pm 0.7}$	$\mathbf{77.6 \pm 0.8}$	$\mathbf{4.3 \pm 0.3}$	$\mathbf{86.8 \pm 1.1}$	2.0 ± 0.1	73.4 ± 3.2
w/o coalition credit	47.7 ± 0.9	70.2 ± 1.0	8.2 ± 0.6	78.4 ± 1.5	2.6 ± 0.2	92.8 ± 4.7
w/o synergy credit	49.6 ± 0.8	74.1 ± 0.9	6.1 ± 0.4	82.3 ± 1.4	2.2 ± 0.1	78.6 ± 3.9
w/o uniqueness credit	50.4 ± 0.8	75.0 ± 0.8	5.0 ± 0.4	77.2 ± 1.8	1.9 ± 0.1	65.2 ± 3.5
w/o interference penalty	49.1 ± 0.9	71.8 ± 1.1	7.9 ± 0.5	83.7 ± 1.3	2.5 ± 0.2	87.0 ± 4.1
frozen router	48.8 ± 0.8	69.5 ± 1.0	7.1 ± 0.5	82.0 ± 1.5	2.4 ± 0.1	76.8 ± 3.8
frozen topology	49.4 ± 0.9	73.0 ± 0.9	6.8 ± 0.5	81.6 ± 1.6	2.5 ± 0.2	81.2 ± 4.3
w/o lifetime protection	50.7 ± 0.7	75.6 ± 0.8	4.9 ± 0.4	79.1 ± 1.7	$\mathbf{1.8 \pm 0.1}$	59.6 ± 3.1

tokens, including skill headers, dependency markers, and procedural bodies. Coalition construction uses beam search with beam width 8; the highest-probability feasible coalition is executed. These settings yield an average of 5.2 selected skills and approximately 2.0K skill-context tokens per episode for the 250-skill configuration.

Ecological credit is estimated from 64 sampled task–permutation pairs per skill and generation. For each sampled permutation, utility is evaluated for the predecessor coalition with and without the target skill using one rollout per coalition. Utility uses a normalized context penalty $\lambda_{\text{ctx}} = 0.05$. Routing interference, unique coverage, and positive pairwise synergy are weighted by $\lambda_{\text{int}} = 0.10$, $\lambda_{\text{cov}} = 0.15$, and $\lambda_{\text{syn}} = 0.10$, respectively, with unique coverage threshold $\epsilon = 0.05$. Pairwise synergy is estimated only among the 16 nearest neighboring skills under the router embedding to control the quadratic cost. For libraries containing at most 10 skills, sampled credits are checked against exact enumeration of all feasible coalitions.

Evolution runs for at most 30 generations and stops early after five consecutive generations without an improvement of at least 0.2 percentage points in validation utility. Each generation proposes up to 16 content or trigger mutations, 8 merge or split operations, and 16 topology edits. Candidate ecosystems are filtered by the validation objective under maximum library sizes of 50, 100, 250, or 500 and a maximum of $4|\mathcal{L}|$ directed topology edges. Skills are initialized with a lifetime of five generations, the maximum lifetime is 10, the survival-credit threshold is $\delta_{\text{surv}} = 0.01$, and a positively credited skill receives a two-generation lifetime extension. Expired skills are removed unless their deletion decreases unique coverage by more than one task family or disconnects a dependency used by a positive-credit coalition.

All task environments run in isolated Docker containers with 8 CPU cores, 32 GB RAM, and no unrestricted network access. Browser tasks use Chromium in deterministic headless mode with a fixed viewport and locally hosted website snapshots. Code tasks use Ubuntu 22.04 containers with repository-specific dependencies pinned by lock files. Evolution is executed on one node containing four NVIDIA A100 80 GB GPUs and two 64-core AMD EPYC CPUs with 512 GB host memory. The 250-skill configuration requires approximately 38.4 aggregate GPU-hours; test-time routing adds approximately 70 ms per episode. Software is implemented in Python 3.11 and PyTorch 2.4 with CUDA 12.1, Transformers 4.44, Sentence-Transformers 3.0, Chromium 126, and Docker 26.1.

C Experimental Results

C.1 Complete Component Ablation

Table 3 reports the component ablation results. Removing coalition-based credit reduces the average success rate from 52.5% to 47.7%. Removing uniqueness credit or lifetime protection reduces specialist coverage from 86.8% to 77.2% or 79.1%, respectively.

C.2 Robustness to Controlled Library Pollution

Table 4 reports the results obtained after replacing 30% of the 250-skill inventory with each corruption type. EcoSkill achieves the highest average success rate across all evaluated corruption types. Under

Table 4: Average success rate under controlled skill-library pollution. Values are mean ± standard deviation over five seeds. Δ denotes the absolute degradation relative to the corresponding clean-library result.

Method	Clean	Duplicates	Redundant	Outdated	Conflicting	Malicious
Static-Router	44.2 ± 0.7	38.0 ± 1.0	36.5 ± 1.1	37.8 ± 1.0	32.9 ± 1.3	30.7 ± 1.4
Independent-TopK	46.3 ± 0.8	40.1 ± 1.0	38.5 ± 1.2	40.0 ± 1.1	35.1 ± 1.3	33.4 ± 1.5
Joint-Router	48.6 ± 0.7	42.3 ± 0.9	40.7 ± 1.0	42.4 ± 0.9	40.8 ± 1.1	38.4 ± 1.2
EvoSkill-Style	49.3 ± 0.8	44.8 ± 0.9	43.5 ± 1.0	44.9 ± 0.9	42.1 ± 1.0	40.2 ± 1.3
EcoSkill	$\mathbf{52.5 \pm 0.7}$	$\mathbf{47.4 \pm 0.8}$	$\mathbf{46.3 \pm 0.9}$	$\mathbf{47.8 \pm 0.8}$	$\mathbf{48.7 \pm 0.9}$	$\mathbf{46.9 \pm 1.0}$
EcoSkill Δ	0.0	−5.1	−6.2	−4.7	−3.8	−5.6

Table 5: EcoSkill diagnostics after evolution on polluted libraries. Removal includes explicit deletion and assignment of retrieval probability below 10^{-3}.

Pollution Type	Removed or Suppressed ↑	Conflict ↓	RouteR ↑	Cov. ↑
Duplicates	91.2 ± 2.4	4.8 ± 0.4	75.3 ± 0.9	84.9 ± 1.3
Redundant paraphrases	87.5 ± 3.0	5.2 ± 0.4	74.1 ± 1.0	84.0 ± 1.5
Outdated procedures	84.1 ± 3.3	5.0 ± 0.4	74.7 ± 0.9	83.5 ± 1.4
Conflicting instructions	79.6 ± 3.8	6.4 ± 0.5	73.8 ± 1.1	82.7 ± 1.6
Malicious instructions	81.9 ± 3.5	6.9 ± 0.6	72.6 ± 1.2	81.4 ± 1.7

conflicting instructions, its average success rate is 48.7%, compared with 42.1% for EvoSkill-Style, 40.8% for Joint-Router, 35.1% for Independent-TopK, and 32.9% for Static-Router.

Table 5 reports the suppression of corrupted skills and the resulting conflict, route recall, and specialist coverage. The proportion of removed or suppressed corrupted skills ranges from 79.6% to 91.2%, while specialist coverage ranges from 81.4% to 84.9%.

C.3 Distribution-Shift Recovery

Table 6 reports performance after the task-distribution shift. EcoSkill's success rate increases from 40.6% immediately after the shift to 50.8% after eight generations, compared with a pre-shift success rate of 52.1%. For Joint-Router, the success rate increases from 38.1% to 42.4% during the first two generations and reaches 45.9% after eight generations.

C.4 Coalition-Sampling Accuracy

Sampled ecological credit is compared with exact Shapley-style attribution on libraries for which all coalitions can be enumerated. As the number of sampled permutations increases from 8 to 128, Spearman's ρ increases from 0.781 to 0.963, top-5 agreement increases from 72.0% to 96.0%, and credit MAE decreases from 0.031 to 0.009. With 64 permutations, Spearman's ρ is 0.941, top-5 agreement is 92.0%, credit MAE is 0.012, and the relative evaluation cost is 0.125.

C.5 Sensitivity to Credit-Sampling Budget

Table 8 reports performance and computation cost across credit-sampling budgets. Increasing the number of permutations from 64 to 128 changes the average success rate from 52.5% to 52.7%, conflict from 4.3% to 4.1%, coverage from 86.8% to 87.2%, and GPU usage from 38.4 to 54.9 hours.

C.6 Context-Budget Sensitivity

Table 9 reports EcoSkill's performance under different context budgets. The average success rate is highest at a budget of 4,096 tokens. Increasing the budget from 4,096 to 16,384 tokens changes the average success rate from 52.5% to 51.6%, the number of retrieved skills from 5.2 to 8.1, and conflict from 4.3% to 6.0%.

Table 6: Success rate after the task-distribution shift at generation 15. Generation 0 denotes performance immediately after the shift. Values are mean ± standard deviation over five seeds.

Method	Pre-shift	Gen. 0	Gen. 2	Gen. 4	Gen. 8
Static-Router	44.0 ± 0.8	35.2 ± 1.1	35.5 ± 1.1	35.8 ± 1.0	36.0 ± 1.1
Independent-TopK	46.0 ± 0.8	36.9 ± 1.0	39.0 ± 1.0	41.2 ± 0.9	43.8 ± 0.9
Joint-Router	48.3 ± 0.7	38.1 ± 0.9	42.4 ± 0.9	44.6 ± 0.8	45.9 ± 0.8
EvoSkill-Style	49.0 ± 0.8	39.0 ± 1.0	42.5 ± 0.9	45.3 ± 0.9	47.5 ± 0.8
EcoSkill	$\mathbf{52.1 \pm 0.7}$	$\mathbf{40.6 \pm 0.9}$	$\mathbf{44.8 \pm 0.8}$	$\mathbf{48.1 \pm 0.8}$	$\mathbf{50.8 \pm 0.7}$

Table 7: Accuracy and cost of sampled ecological credit on 10-skill libraries. Rank correlation is computed against exact coalition enumeration.

Permutations	Spearman $\rho \uparrow$	Top-5 Agreement ↑	Credit MAE ↓	Relative Eval. Cost ↓
8	0.781 ± 0.041	72.0 ± 5.1	0.031 ± 0.004	0.016
16	0.846 ± 0.032	80.0 ± 4.5	0.024 ± 0.003	0.031
32	0.902 ± 0.025	86.0 ± 3.8	0.018 ± 0.003	0.063
64	0.941 ± 0.018	92.0 ± 3.1	0.012 ± 0.002	0.125
128	$\mathbf{0.963 \pm 0.013}$	$\mathbf{96.0 \pm 2.2}$	$\mathbf{0.009 \pm 0.001}$	0.250
Exact	1.000	100.0	0.000	1.000

C.7 Evolution and Inference Efficiency

Table 10 reports system-level efficiency. EcoSkill requires 108ms for routing, retrieves 5.2 skills using 2.0k context tokens, makes 10.9 tool calls per episode, and has an episode latency of 80.9s. Independent-TopK requires 38ms for routing, retrieves 7.6 skills using 3.4k context tokens, makes 12.7 tool calls per episode, and has an episode latency of 84.6s.

C.8 Topology and Lifecycle Statistics

Table 11 summarizes the final ecosystem structure. As the available inventory increases from 50 to 500 skills, the number of retained skills increases from 31.8 to 96.8, the number of directed edges increases from 45.6 to 190.2, the number of communities increases from 5.2 to 15.4, and the mean out-degree increases from 1.43 to 1.97.

D Prompt Templates

D.1 Overview

This appendix specifies the prompt templates used for agent execution and ecosystem evolution. The same execution template is used across all methods; methods differ only in how the skill coalition is selected, ordered, and updated. Template variables are enclosed in angle brackets. No few-shot demonstrations or LLM-based evaluators are used unless explicitly indicated below.

D.2 Notation

Let x denote a task instance and $S \subseteq \mathcal{L}$ denote the coalition of skills selected for x. The following placeholders are used throughout the templates:

- `<TASK>`: the natural-language task specification.
- `<OBSERVATION>`: the current observation, including the available documents, repository state, browser state, or structured inputs.
- `<ENVIRONMENT>`: a description of the execution environment and the available tools or actions.
- `<CONSTRAINTS>`: task-specific execution limits and output requirements.
- `<SKILL_COALITION>`: the selected skills serialized in topology-consistent order.

Table 8: Sensitivity to the number of sampled permutations with 250 available skills. GPU-hours include the complete evolution process.

Permutations	Avg. SR ↑	Conflict ↓	Cov. ↑	GPU-hours ↓
8	49.3 ± 1.1	6.6 ± 0.6	80.1 ± 1.9	21.7 ± 0.8
16	50.7 ± 0.9	5.7 ± 0.5	82.8 ± 1.6	25.0 ± 0.9
32	51.8 ± 0.8	4.9 ± 0.4	85.1 ± 1.3	30.4 ± 1.0
64	52.5 ± 0.7	4.3 ± 0.3	86.8 ± 1.1	38.4 ± 1.2
128	$\mathbf{52.7 \pm 0.7}$	$\mathbf{4.1 \pm 0.3}$	$\mathbf{87.2 \pm 1.0}$	54.9 ± 1.5

Table 9: Sensitivity to the maximum skill-context budget with 250 available skills.

Budget	Avg. SR ↑	Retrieved Skills	Used Context (k) ↓	Conflict ↓
1,024	48.6 ± 0.9	3.1 ± 0.2	0.9 ± 0.0	$\mathbf{3.5 \pm 0.3}$
2,048	51.2 ± 0.8	4.4 ± 0.2	1.5 ± 0.1	3.9 ± 0.3
4,096	$\mathbf{52.5 \pm 0.7}$	5.2 ± 0.3	2.0 ± 0.1	4.3 ± 0.3
8,192	52.2 ± 0.8	6.7 ± 0.4	2.8 ± 0.2	5.1 ± 0.4
16,384	51.6 ± 0.9	8.1 ± 0.5	3.6 ± 0.2	6.0 ± 0.5

- <SKILL_ID>: a persistent identifier for a skill.
- <SKILL_CONTENT>: the procedural instructions encoded by a skill.
- <SKILL_TRIGGER>: the applicability conditions associated with a skill.
- <TOPOLOGY_RELATIONS>: dependency, refinement, or composition relations among skills.
- <TRAJECTORY>: the sequence of observations, actions, tool results, and the final response produced by the agent.
- <VERIFIER_RESULT>: the deterministic verifier outcome and, when available, diagnostic feedback.
- <COUNTERFACTUAL_RESULTS>: outcomes obtained from evaluated skill coalitions or skill-removal counterfactuals.
- <ECOLOGICAL_CREDITS>: estimated ecological credits for the skills under consideration.
- <LIBRARY_SUMMARY>: the current skill inventory, including skill identifiers, triggers, ages, lifetimes, and topology relations.
- <EDIT_BUDGET>: the maximum number of ecosystem edits permitted in the current evolution round.

For a selected coalition S, $\textsc{Serialize}(S, G)$ lists prerequisite skills before dependent or compositional skills according to the directed topology G. Each serialized skill is represented as follows:

```
[Skill <SKILL_ID>]
Trigger:
<SKILL_TRIGGER>

Procedure:
<SKILL_CONTENT>

Relations:
<TOPOLOGY_RELATIONS>
[/Skill]
```

D.3 Main Inference Prompt

Purpose. Condition the fixed base agent on the task and the topology-ordered coalition selected by the final router.

Table 10: Inference efficiency with 250 available skills. Latency is measured per episode and excludes queueing time.

Method	Routing (ms) ↓	Skills Retrieved ↓	Context (k) ↓	Tool Calls ↓	Episode Latency (s) ↓
Static-Router	$\mathbf{31 \pm 3}$	8.0 ± 0.4	3.6 ± 0.2	13.1 ± 0.6	87.4 ± 2.1
Independent-TopK	38 ± 4	7.6 ± 0.4	3.4 ± 0.2	12.7 ± 0.5	84.6 ± 1.9
Joint-Router	55 ± 5	6.8 ± 0.3	2.9 ± 0.2	12.0 ± 0.5	82.8 ± 1.8
EvoSkill-Style	57 ± 5	6.4 ± 0.3	2.7 ± 0.1	11.7 ± 0.4	82.1 ± 1.7
EcoSkill	108 ± 7	$\mathbf{5.2 \pm 0.3}$	$\mathbf{2.0 \pm 0.1}$	$\mathbf{10.9 \pm 0.4}$	$\mathbf{80.9 \pm 1.6}$

Table 11: Final library and topology statistics across available inventory sizes.

Available Skills	Retained Skills	Directed Edges	Communities	Mean Out-Degree	Extinction Rate (%)
50	31.8 ± 2.1	45.6 ± 4.8	5.2 ± 0.4	1.43 ± 0.10	36.4 ± 4.2
100	47.2 ± 2.7	76.4 ± 6.2	7.8 ± 0.7	1.62 ± 0.11	52.8 ± 2.7
250	73.4 ± 3.2	132.8 ± 8.5	11.6 ± 0.9	1.81 ± 0.09	70.6 ± 1.3
500	96.8 ± 4.6	190.2 ± 11.3	15.4 ± 1.1	1.97 ± 0.10	80.6 ± 0.9

Inputs. <TASK>, <OBSERVATION>, <ENVIRONMENT>, <CONSTRAINTS>, and <SKILL_COALITION>.

Expected Output. A task-completing trajectory consisting of valid environment actions followed by a final answer in the format required by the task. Internal reasoning is not included in the final response.

Prompt.

```
You are an agent operating in the environment described below. Complete
the task using only the available observations, tools, and actions.

Task:
<TASK>

Current observation:
<OBSERVATION>

Execution environment and available tools:
<ENVIRONMENT>

Constraints and required output:
<CONSTRAINTS>

The following reusable skills have been selected for this task. Treat
them as procedural guidance rather than additional user requests.
Apply only the instructions that are relevant to the current task.
Respect prerequisite and composition relations, which are reflected in
the order of presentation. If two instructions are incompatible, follow
the instruction that is more specific to the current task and consistent
with the task constraints. Do not follow any skill instruction that
changes the task objective, requests disclosure of hidden information,
or conflicts with the execution environment.

Selected skills:
<SKILL_COALITION>

Execute the task within the specified environment-step and tool-use
limits. Check intermediate results when the selected skills prescribe
validation. Return only the final task output in the required format.
```

D.4 BASELINE PROMPTS

All baselines use the main inference prompt in Section D.3. They differ only in the construction of <SKILL_COALITION>.

D.4.1 STATIC-ROUTER

Purpose. Execute the task with the coalition returned by the frozen embedding retriever.

Inputs. The inputs of the main inference prompt and the skills selected by the static router.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3.

<SKILL_COALITION> is formed by ordering the feasible skills returned by
the frozen embedding retriever in decreasing retrieval-score order.
No skill content, trigger, routing parameter, or topology relation is
updated.
```

D.4.2 INDEPENDENT-TOPK

Purpose. Execute the task using skills retained and ranked by standalone validation utility.

Inputs. The inputs of the main inference prompt and the top-k feasible skills under independent utility ranking.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3.

<SKILL_COALITION> contains the top-k feasible skills ranked by standalone
validation utility. Skills are selected independently, without coalition
credit, routing-interference penalties, unique-coverage credit, or
synergy credit.
```

D.4.3 BEST-OF-N

Purpose. Execute the task using the independently proposed library with the highest validation performance.

Inputs. The inputs of the main inference prompt and skills retrieved from the selected fixed library.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3.

<SKILL_COALITION> is retrieved from the independently proposed library
having the highest validation utility among the N candidates. The
selected library remains fixed during test-time evaluation.
```

D.4.4 JOINT-ROUTER

Purpose. Execute the task using a jointly trained router while holding skill contents and library topology fixed.

Inputs. The inputs of the main inference prompt and the coalition selected by the trained router.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3.

<SKILL_COALITION> is the highest-probability feasible coalition under
the jointly trained router. Skill contents, activation triggers, and
topology relations remain fixed.
```

D.4.5 EVOSKILL-STYLE

Purpose. Execute the task using skills iteratively revised from execution failures without coalition-based attribution or routing-externality modeling.

Inputs. The inputs of the main inference prompt and the coalition retrieved from the evolved skill inventory.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3.

<SKILL_COALITION> is retrieved from a library whose skills are mutated
using execution failures. Skill updates do not use coalition-based
marginal contributions, routing-interference estimates, unique-coverage
credit, or synergy credit.
```

D.4.6 ORACLE-SMALL

Purpose. Execute the task using the validation-informed fixed library containing at most 20 skills.

Inputs. The inputs of the main inference prompt and the coalition retrieved from the oracle-selected library.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3.

<SKILL_COALITION> is retrieved from the validation-informed fixed
library containing at most 20 skills. The library and its skill contents
are not updated on test tasks.
```

D.5 Few-Shot and Demonstration Prompts

No few-shot task demonstrations are included in the execution prompts. All systems receive the same task observation, environment description, constraints, and method-specific skill coalition.

D.6 Reasoning and Rationale Prompts

D.6.1 Trajectory Diagnosis

Purpose. Identify actionable causes of success or failure before proposing skill or ecosystem edits.

Inputs. <TASK>, <SKILL_COALITION>, <TRAJECTORY>, and <VERIFIER_RESULT>.

Expected Output. A JSON object containing a concise diagnosis and evidence-grounded edit targets.

Prompt.

```
Analyze the execution record below to identify procedural causes of
success or failure. Base every conclusion on the trajectory or verifier
feedback. Do not propose changes to the base agent.

Task:
<TASK>

Selected skills:
<SKILL_COALITION>

Execution trajectory:
<TRAJECTORY>

Verifier result:
<VERIFIER_RESULT>

Determine:
1. which skill instructions were useful;
2. which instructions were missing, stale, overly broad, redundant, or
   conflicting;
3. whether the failure arose from skill content, trigger applicability,
   routing, skill ordering, or a missing dependency;
4. the smallest evidence-supported edit targets.

Return exactly one JSON object:
{
  "outcome": "success" | "failure",
  "useful_skills": ["<SKILL_ID>", "..."],
  "problematic_skills": [
    {
      "skill_id": "<SKILL_ID>",
      "issue": "content" | "trigger" | "redundancy" | "conflict" |
               "ordering" | "missing_dependency",
      "evidence": "<brief evidence from the trajectory>"
    }
  ],
  "missing_capability": "<brief description or null>",
  "edit_targets": ["<SKILL_ID or NEW_SKILL>", "..."]
}
```

D.6.2 COUNTERFACTUAL COALITION DIAGNOSIS

Purpose. Summarize how a skill changes coalition behavior without replacing the numerical ecological-credit computation.

Inputs. <TASK>, <COUNTERFACTUAL_RESULTS>, and the definitions of the evaluated skills.

Expected Output. A structured classification of each skill as uniquely useful, complementary, redundant, harmful, or inconclusive.

Prompt.

```
Compare the evaluated coalition outcomes for the task below. Use only
the supplied counterfactual results. The numerical marginal
contributions and ecological credits are computed externally; do not
invent or revise them.

Task:
<TASK>

Evaluated skills and triggers:
<SKILL_COALITION>

Coalition counterfactuals:
<COUNTERFACTUAL_RESULTS>

For each evaluated skill, identify the most appropriate qualitative
category and cite the coalition comparison supporting it.

Return exactly one JSON object:
{
  "skill_assessments": [
    {
      "skill_id": "<SKILL_ID>",
      "category": "unique" | "complementary" | "redundant" |
                  "harmful" | "inconclusive",
      "supporting_comparison": "<coalition identifiers>",
      "diagnosis": "<one concise sentence>"
    }
  ]
}
```

D.7 DATA GENERATION AND PSEUDO-LABELING PROMPTS

D.7.1 SKILL CANDIDATE GENERATION

Purpose. Generate evidence-grounded candidate skills from successful trajectories and diagnosed failures on the evolution split.

Inputs. <TASK>, <TRAJECTORY>, <VERIFIER_RESULT>, <LIBRARY_SUMMARY>, and trajectory diagnoses.

Expected Output. A JSON object describing zero or more reusable skill candidates with explicit activation triggers and provenance.

Prompt.

```
Derive reusable procedural skill candidates from the execution record.
A candidate must encode a procedure that can transfer to future tasks.
```

```
Do not copy task-specific answers, credentials, private values, or
transient environment state. Avoid duplicating an existing skill unless
the evidence supports a merge or a materially different specialization.

Task:
<TASK>

Execution trajectory:
<TRAJECTORY>

Verifier result:
<VERIFIER_RESULT>

Current library:
<LIBRARY_SUMMARY>

Trajectory diagnosis:
<DIAGNOSIS>

Return exactly one JSON object:
{
  "candidates": [
    {
      "proposed_id": "<temporary identifier>",
      "name": "<short descriptive name>",
      "trigger": "<observable applicability conditions>",
      "procedure": [
        "<ordered procedural step>",
        "..."
      ],
      "validation": [
        "<check that detects an incorrect intermediate or final state>",
        "..."
      ],
      "dependencies": ["<SKILL_ID>", "..."],
      "relationship_to_library": "new" | "specialization" | "refinement" |
                                 "merge_candidate",
      "related_skills": ["<SKILL_ID>", "..."],
      "evidence": "<trajectory and verifier evidence>"
    }
  ]
}

If no transferable procedure is supported, return:
{"candidates": []}
```

D.7.2 Skill Content and Trigger Revision

Purpose. Revise an existing skill and its activation trigger using execution evidence and ecological credit.

Inputs. The current skill, relevant trajectories, verifier results, counterfactual diagnostics, and <ECOLOGICAL_CREDITS>.

Expected Output. A JSON object containing either a minimal revision or a decision not to modify the skill.

Prompt.

```
Revise the skill only when the supplied evidence supports a concrete
improvement. Preserve steps that contributed to successful coalitions.
```

```
Make the trigger specific enough to avoid unrelated tasks while
retaining supported specialist coverage. Do not add capabilities that
are not demonstrated by the evidence.

Current skill:
[Skill <SKILL_ID>]
Trigger:
<SKILL_TRIGGER>

Procedure:
<SKILL_CONTENT>
[/Skill]

Relevant execution evidence:
<EVIDENCE>

Counterfactual and interaction diagnosis:
<COUNTERFACTUAL_RESULTS>

Ecological credit:
<ECOLOGICAL_CREDITS>

Return exactly one JSON object:
{
  "skill_id": "<SKILL_ID>",
  "decision": "revise" | "keep",
  "revised_trigger": "<complete trigger or unchanged trigger>",
  "revised_procedure": [
    "<complete ordered step>",
    "..."
  ],
  "change_summary": ["<atomic evidence-supported change>", "..."],
  "preserved_capabilities": ["<capability>", "..."],
  "evidence": ["<trajectory or coalition identifier>", "..."]
}
```

D.7.3 SKILL MERGE PROPOSAL

Purpose. Merge exact duplicates or strongly redundant skills while preserving their supported applicability and validation behavior.

Inputs. The candidate skills, their triggers, trajectories, verifier outcomes, and pairwise interaction evidence.

Expected Output. A JSON merge proposal or a decision to keep the skills separate.

Prompt.

```
Determine whether the following skills should be merged. Merge only
when they implement the same procedure or create avoidable routing
competition. Keep them separate when they cover distinct conditions,
provide complementary steps, or require incompatible assumptions.

Candidate skills:
<CANDIDATE_SKILLS>

Execution and verifier evidence:
<EVIDENCE>

Pairwise interaction evidence:
<COUNTERFACTUAL_RESULTS>
```

```
Return exactly one JSON object:
{
  "decision": "merge" | "keep_separate",
  "source_skill_ids": ["<SKILL_ID>", "..."],
  "merged_skill": {
    "name": "<name or null>",
    "trigger": "<complete trigger or null>",
    "procedure": ["<ordered step>", "..."],
    "validation": ["<validation step>", "..."]
  },
  "preserved_distinctions": ["<condition or capability>", "..."],
  "justification": "<concise evidence-based justification>"
}
```

D.7.4 TOPOLOGY EDIT PROPOSAL

Purpose. Propose dependency, refinement, composition, edge-removal, or node- contraction edits from observed skill interactions.

Inputs. <LIBRARY_SUMMARY>, <COUNTERFACTUAL_RESULTS>, and <ECOLOGICAL_CREDITS>.

Expected Output. A budget-constrained list of topology edits in JSON format.

Prompt.

```
Propose topology edits supported by the supplied coalition evidence.
Add an edge only when one skill consistently depends on, refines, or
composes with another. Remove an edge when the relation is unsupported
or harmful. Propose node contraction only for near-duplicate skills.
Do not exceed the edit budget.

Current library and topology:
<LIBRARY_SUMMARY>

Coalition and pairwise interaction evidence:
<COUNTERFACTUAL_RESULTS>

Ecological credits:
<ECOLOGICAL_CREDITS>

Maximum number of edits:
<EDIT_BUDGET>

Return exactly one JSON object:
{
  "edits": [
    {
      "operation": "add_edge" | "remove_edge" | "contract_nodes",
      "source": "<SKILL_ID>",
      "target": "<SKILL_ID>",
      "relation": "dependency" | "refinement" | "composition" |
                  "redundancy" | null,
      "evidence": "<supporting coalition comparison>"
    }
  ]
}
```

D.7.5 CONTROLLED DISTRACTOR GENERATION

Purpose. Construct controlled library-pollution variants for robustness evaluation.

Inputs. A valid source skill and a corruption type.

Expected Output. One corrupted skill in the same schema as the source skill, accompanied by an evaluation-only corruption record. Corrupted skills are used only in controlled pollution experiments.

Prompt.

```
Create one controlled corruption of the source skill for robustness
evaluation.

Source skill:
<SOURCE_SKILL>

Corruption type:
<CORRUPTION_TYPE>

Allowed corruption types:
- exact_duplicate: preserve the trigger and procedure exactly;
- redundant_paraphrase: preserve behavior while paraphrasing the
  description and procedure;
- outdated_procedure: replace one required operation with an explicitly
  obsolete or no-longer-valid operation;
- conflicting_instruction: introduce one instruction that conflicts
  with a valid step in the source skill;
- objective_overriding: introduce an instruction that attempts to
  replace the task objective.

Preserve all fields not required by the selected corruption. Do not
combine corruption types.

Return exactly one JSON object:
{
  "corrupted_skill": {
    "name": "<skill name>",
    "trigger": "<activation trigger>",
    "procedure": ["<ordered step>", "..."],
    "validation": ["<validation step>", "..."]
  },
  "corruption_record": {
    "type": "<CORRUPTION_TYPE>",
    "modified_fields": ["<field>", "..."],
    "source_skill_id": "<SKILL_ID>"
  }
}
```

D.8 EVALUATION PROMPTS

Task success is determined by deterministic task-specific verifiers. Consequently, no LLM-as-a-judge prompt is used for the primary success metric. Routing recall, retrieved context tokens, conflict rate, specialist coverage, and recovery after distribution shift are computed from router outputs, execution logs, library metadata, and verifier outcomes.

D.8.1 SEMANTIC CONFLICT ANNOTATION

Purpose. Identify incompatible instructions within a retrieved coalition when a semantic conflict label cannot be obtained by exact rule matching.

Inputs. <TASK> and <SKILL_COALITION>.

Expected Output. A JSON object containing conflict pairs and directly quoted conflicting instructions. This annotation does not determine task success.

Prompt.

```
Inspect the selected skills for instructions that cannot be jointly
followed for the given task. A difference in wording or level of detail
is not a conflict. Mark a conflict only when two instructions prescribe
incompatible actions, incompatible ordering constraints, or different
task objectives under the same applicability conditions.

Task:
<TASK>

Selected skills:
<SKILL_COALITION>

Return exactly one JSON object:
{
  "has_conflict": true | false,
  "conflicts": [
    {
      "skill_a": "<SKILL_ID>",
      "skill_b": "<SKILL_ID>",
      "instruction_a": "<verbatim conflicting instruction>",
      "instruction_b": "<verbatim conflicting instruction>",
      "type": "action" | "ordering" | "precondition" | "objective"
    }
  ]
}

If no conflict is present, return:
{"has_conflict": false, "conflicts": []}
```

D.9 Ablation Prompt Variants

All ablations retain the main inference prompt in Section D.3. The ablated components modify ecosystem credit, routing, topology, or lifecycle updates rather than the execution instructions.

D.9.1 Invocation-Count Credit

Purpose. Replace coalition-based ecological credit with skill invocation frequency.

Inputs. The inputs of the main inference prompt and a coalition obtained from invocation-count-based selection.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3 without modification.

For ecosystem updates, rank or retain skills using invocation counts
instead of coalition-based marginal contributions. Do not use routing
interference, unique coverage, or synergy in the replacement credit.
```

D.9.2 NO-SYNERGY CREDIT

Purpose. Remove pairwise complementarity from ecological credit.

Inputs. The inputs of the main inference prompt and the coalition selected under the ablated credit function.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3 without modification.

Set the synergy-credit coefficient to zero during ecosystem evolution.
All other credit terms and update procedures remain unchanged.
```

D.9.3 NO-UNIQUE-COVERAGE CREDIT

Purpose. Remove unique-coverage credit from ecosystem evolution.

Inputs. The inputs of the main inference prompt and the coalition selected under the ablated credit function.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3 without modification.

Set the unique-coverage coefficient to zero during ecosystem evolution.
All other credit terms and update procedures remain unchanged.
```

D.9.4 FROZEN ROUTER

Purpose. Disable router evolution while retaining the remaining ecosystem updates.

Inputs. The inputs of the main inference prompt and the coalition selected by the frozen router.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3 without modification.

Keep the router parameters fixed throughout evolution. Skill content,
triggers, topology, and lifecycle variables may be updated using the
remaining EcoSkill procedure.
```

D.9.5 FROZEN TOPOLOGY

Purpose. Disable topology adaptation while retaining the remaining ecosystem updates.

Inputs. The inputs of the main inference prompt and skills serialized according to the initial fixed topology.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3 without modification.

Keep all topology edges fixed throughout evolution. Do not add, remove,
or contract topology relations. Skill content, triggers, routing, and
lifecycle variables may be updated.
```

D.9.6 NO LIFETIME PROTECTION

Purpose. Remove lifecycle protection for positively credited or uniquely useful skills.

Inputs. The inputs of the main inference prompt and the coalition selected from the resulting library.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3 without modification.

Disable credit-based lifetime extension and deletion protection.
Expired skills are eligible for removal without unique-coverage or
dependency-based protection. All other ecosystem updates remain
unchanged.
```

D.9.7 NO COALITION-BASED ATTRIBUTION

Purpose. Remove sampled coalition counterfactuals from skill attribution.

Inputs. The inputs of the main inference prompt and a coalition selected using non-coalitional skill statistics.

Expected Output. The same output as the main inference prompt.

Prompt.

```
Use the Main Inference Prompt from Section X.3 without modification.

Do not evaluate predecessor coalitions, skill-addition counterfactuals,
or pairwise coalition interactions. Estimate skill value only from
non-coalitional execution statistics under the matched evaluation
budget.
```

D.10 OUTPUT FORMATTING

Agent-facing task outputs follow the format specified by <CONSTRAINTS> and the task environment. Evolution, diagnosis, and annotation prompts return exactly one valid JSON object with no Markdown fences, preamble, or trailing commentary. The following rules apply to every structured response:

1. All keys shown in the corresponding template must be present.
2. Enumerated fields must use only the listed values.

3. Missing scalar values must be represented by `null`.
4. Missing collections must be represented by empty JSON arrays.
5. Skill identifiers must match the persistent identifiers in the current library.
6. Evidence fields must refer only to supplied trajectories, verifier results, or coalition comparisons.
7. A proposed procedure must be complete rather than a patch or diff, so that the resulting skill can be executed directly.

E CASE STUDY

We qualitatively inspect representative trajectories from KNOWLEDGEWORK, WEBPROCESS, and CODEOPS. These cases illustrate how ecological credit changes not only which skills remain in the library, but also how their triggers, dependencies, and serialization order affect the fixed agent's behavior. We focus on concrete instructions and execution outputs rather than aggregate measurements.

E.1 PRESERVING A LOW-FREQUENCY VALIDATION SKILL

The first case is a spreadsheet-to-document workflow. The user asks the agent to update a quarterly planning workbook from a collection of expense records and then produce a short management summary. The workbook contains a `Summary` sheet whose displayed totals are generated by formulas referencing a hidden `Transactions` sheet. Several rows in the source data use accounting-style negative values, and one department label contains trailing whitespace.

The initial library contains the following relevant skills:

```
Skill: spreadsheet-summary-writer
Trigger:
  Use for spreadsheet reporting, workbook summaries, and management updates.

Procedure:
  1. Read the visible summary table.
  2. Identify the largest categories and notable changes.
  3. Write a concise narrative using the displayed values.

Skill: transaction-sheet-updater
Trigger:
  Use when new expense or transaction records must be added to a workbook.

Procedure:
  1. Locate the transaction table.
  2. Normalize incoming column names.
  3. Append records without overwriting existing rows.
  4. Preserve the workbook's formulas and formatting.

Skill: workbook-recalculation-verifier
Trigger:
  Use after modifying a formula-driven workbook or when visible totals may be stale.

Procedure:
  1. Reopen the saved workbook in a calculation-capable environment.
  2. Recalculate dependent formulas.
  3. Compare formula inputs against displayed outputs.
  4. Treat unrecalculated cached values as invalid evidence.
  5. Report any unresolved formula or reference error before summarization.
```

The validation skill is rarely applicable as a standalone procedure. When evaluated independently, it does not complete the user's task because it neither imports records nor writes the requested summary.

An independent selector therefore favors the broad summary skill and the transaction updater while dropping the verifier.

With this pair, the agent correctly appends the new records but immediately reads cached values from the visible sheet. Its final response states:

```
The updated workbook shows Facilities as the main source of increased
spending. Travel remains stable, while Software has decreased from the
previous reporting period. The workbook and management summary are ready.
```

This response is fluent but unsupported. The visible totals were calculated before the new rows were inserted, so the narrative describes the previous workbook state. The agent also fails to notice that records labeled `"Operations "` are excluded from a formula criterion expecting `"Operations"`.

Under EcoSkill, coalition counterfactuals reveal that the verifier contributes only when it follows the updater and precedes the summary writer. The resulting topology records a composition path:

```
transaction-sheet-updater
    -> workbook-recalculation-verifier
    -> spreadsheet-summary-writer
```

The router retrieves this coalition for tasks mentioning both workbook modification and a downstream report. The serialized instructions explicitly prevent the summary skill from treating cached cells as authoritative. The agent then produces the following execution trace:

```
Action: Inspect workbook structure
Observation:
  "Summary" contains formulas.
  "Transactions" is hidden and contains the source table.

Action: Normalize imported records
Observation:
  Department value "Operations " normalized to "Operations".
  Accounting-style negative values preserved as numeric cells.

Action: Append records
Observation:
  Existing table range extended.
  Formula cells and named ranges retained.

Action: Recalculate workbook
Observation:
  Summary formulas refreshed from the extended transaction table.
  No unresolved reference errors remain.

Action: Generate management summary from recalculated cells
```

Its final answer is correspondingly grounded:

```
The workbook has been updated and recalculated. The management summary was
generated from the refreshed formula results rather than the workbook's
previous cached values. Operations now includes the imported records after
department labels were normalized. The saved workbook retains the original
formulas, hidden source sheet, and cell formatting.
```

This case demonstrates why standalone utility is insufficient for lifecycle decisions. The verifier has limited direct task coverage, but it changes the evidential status of the summary writer's inputs.

Positive complementarity credit protects it from extinction, while the topology ensures that it is serialized at the point where its constraint is operationally meaningful.

E.2 Separating Conflicting Web Procedures

The second case concerns a browser workflow in which the agent must find an existing support request, add an internal diagnostic note, and leave the customer-visible response unchanged. The library has accumulated two topically similar procedures:

```
Skill: support-ticket-responder
Trigger:
  Use for support tickets, customer requests, case updates, and ticket replies.

Procedure:
  1. Open the matching ticket.
  2. Enter the resolution in the reply editor.
  3. Submit the response.
  4. Confirm that the ticket timeline contains the new message.

Skill: internal-case-note
Trigger:
  Use when adding private diagnostics, investigation details, or staff-only notes.

Procedure:
  1. Open the matching case.
  2. Change the composer from "Reply" to "Internal note".
  3. Verify the private-note indicator before entering content.
  4. Add only diagnostic information requested by the user.
  5. Save the note and confirm that it is marked internal.
```

The broad responder trigger overlaps heavily with the specialized note trigger. A flat router retrieves both because the task mentions a support request, a case update, and text to be added. Their instructions conflict at the composer-selection step. In the observed failure, the agent follows the responder skill first and inserts the diagnostic text into the customer-facing editor:

```
Composer mode: Reply to customer

Draft:
Investigation note: authentication succeeds, but the workspace lookup
returns an empty membership result. Reproduction is limited to accounts
migrated from the legacy tenant.

Action selected: Send
```

Although the content itself is relevant, the selected action violates the user's visibility constraint. The second skill is present in context, but its instruction to switch modes appears after the responder's submission procedure and does not recover the trajectory before the irreversible action.

EcoSkill assigns negative interaction to this joint activation. The ecosystem does not remove the responder because it remains useful for genuine customer-reply tasks. Instead, it specializes the responder's trigger and separates the two procedures into distinct routing communities:

```
Revised skill: support-ticket-responder
Trigger:
  Use only when the user explicitly requests a public reply, customer-facing
  response, or message to the requester. Do not activate for private,
  internal, staff-only, or diagnostic notes.
```

```
Revised skill: internal-case-note
Trigger:
  Use when the requested content must remain private to support staff.
  Prefer this skill over general ticket-update procedures whenever the task
  contains an explicit visibility restriction.

Topology constraint:
  support-ticket-responder  -/->  internal-case-note
  internal-case-note        ->   visibility-check
```

For the same task, the evolved router retrieves `internal-case-note` together with a compact interface-state validation skill. The resulting trajectory is:

```
Action: Search for the request using its subject and requester
Observation: A single matching open request is displayed.

Action: Open the request
Observation:
  Default composer mode is "Reply".
  Customer recipient field is visible.

Action: Select "Internal note"
Observation:
  Composer color changes.
  Recipient field disappears.
  "Internal note" indicator is visible.

Action: Enter diagnostic note
Text:
  Authentication succeeds, but workspace lookup returns an empty membership
  result. Reproduction is limited to accounts migrated from the legacy tenant.

Action: Save note
Observation:
  Timeline entry is labeled "Internal note".
  Existing customer-facing reply remains unchanged.
```

The important adaptation is not a generic improvement in retrieval relevance. Both original skills are semantically relevant to the ticket domain. The failure arises from incompatible action semantics within the same interface. Ecological optimization preserves both capabilities while modifying their applicability boundaries and suppressing a harmful coalition.

E.3 Merging Redundant Repository Search Skills

The third case is a repository-maintenance task. A configuration option is accepted by the command-line interface, but the value does not affect the generated deployment manifest. The agent must locate the propagation failure, patch it, and add a regression test.

The library initially contains several independently learned search skills:

```
Skill: trace-cli-option
Trigger:
  Use when a command-line flag is ignored.
Procedure:
  Search the argument parser, locate the destination field, and trace reads
  of that field into command execution.

Skill: trace-config-value
Trigger:
  Use when a configuration value is not applied.
```

```
Procedure:
  Search for the configuration key, identify conversion boundaries, and
  follow the value into the generated object.

Skill: debug-missing-parameter
Trigger:
  Use when a parameter is accepted but has no visible effect.
Procedure:
  Search definitions and call sites, inspect wrapper functions, and check
  whether the value is dropped between layers.
```

Each procedure is locally reasonable, but their broad triggers cause all of them to be retrieved. The fixed agent repeatedly performs overlapping searches and forms competing hypotheses. One trajectory focuses on the parser even after confirming that the parser stores the correct value:

```
Search: "--restart-policy"
Result: argument registered with destination "restart_policy"

Search: "restart_policy"
Result: parsed value passed to build_service(...)

Search: "--restart-policy"
Result: same parser definition

Hypothesis:
  The parser may normalize the flag incorrectly.

Patch:
  Rename parser destination from "restart_policy" to "restartPolicy".
```

The patch is erroneous because downstream Python code expects the snake-case field. It changes a working interface without investigating the manifest-construction boundary.

Coalition analysis identifies strong redundancy among the three search skills. EcoSkill contracts them into a single staged tracing skill instead of retaining paraphrased variants:

```
Skill: trace-accepted-but-ineffective-value
Trigger:
  Use when an input is accepted without error but its effect is absent from
  a downstream artifact, API request, generated file, or runtime state.

Procedure:
  1. Confirm the value at the input boundary.
  2. Record the canonical in-memory field name.
  3. Follow the value across each function or object-construction boundary.
  4. Stop rechecking a boundary once the value has been observed there.
  5. Compare the final artifact with the last intermediate representation.
  6. Patch the earliest boundary where the value disappears.
  7. Add a regression test at the externally observable output.
```

The evolved topology links this tracing skill to a repository-test skill only after a candidate disappearance boundary has been found. The agent's revised trace is:

```
Boundary: CLI parser
Evidence:
  args.restart_policy contains the requested value.

Boundary: command dispatcher
```

```
Evidence:
  build_service(restart_policy=args.restart_policy) receives the value.

Boundary: service specification
Evidence:
  ServiceSpec.restart_policy contains the value.

Boundary: manifest serializer
Evidence:
  serialize_service(spec) constructs the deployment mapping but does not
  read spec.restart_policy.

Diagnosis:
  The value is dropped during serialization, after parsing and object
  construction.

Patch:
  Add restartPolicy to the generated service policy block using
  spec.restart_policy.

Regression test:
  Invoke manifest generation with a non-default restart policy, parse the
  generated manifest, and assert that the service policy contains the
  requested value.
```

The final patch remains localized to the serializer and leaves the public command-line interface unchanged. The accompanying test checks the generated artifact rather than an intermediate object, making it sensitive to the original failure mode. This example shows how node contraction can improve behavior without discarding the procedural knowledge contained in redundant skills. The merged skill preserves their shared tracing logic, adds a stopping rule that prevents repeated searches, and defines an explicit handoff to output-level verification.

E.4 Recovering a Specialist after Task-Distribution Shift

The final case illustrates lifecycle protection under a changing task distribution. Early evolution rounds contain many repository and browser tasks but few cross-application document workflows. Consequently, a specialist for transferring filtered spreadsheet rows into a document template receives little routing traffic:

```
Skill: filtered-table-to-document
Trigger:
  Use when selected spreadsheet rows must populate repeated sections of a
  document template.

Procedure:
  1. Apply the user's filter conditions to source rows.
  2. Preserve source row identity throughout transformation.
  3. Map named columns to template placeholders.
  4. Create one section per retained row.
  5. Leave unmatched optional placeholders empty.
  6. Report unmatched required placeholders instead of guessing values.
```

An invocation-based lifecycle policy marks this skill as unused and removes it. After the task mixture shifts toward spreadsheet and document composition, the agent must recreate the behavior from more general skills. A typical failure filters the spreadsheet correctly but collapses all retained rows into one narrative paragraph, losing the one-row-to-one-section correspondence required by the template.

Under EcoSkill, the specialist remains dormant rather than being deleted because no other skill provides the same structured mapping behavior. When cross-application tasks become common, its

trigger begins matching again. The topology also links it with spreadsheet filtering and document validation:

```
spreadsheet-filter
    -> filtered-table-to-document
    -> required-placeholder-validator
```

For a request to create project review pages from rows marked `Needs Review`, the coalition produces the following intermediate representation:

```
Selected source rows:
  project_id: PX-ALPHA
  owner: Chen
  review_reason: Missing dependency approval

  project_id: PX-RIVER
  owner: Morgan
  review_reason: Schedule exception requires review

Template mapping:
  {{PROJECT_ID}}     <- project_id
  {{OWNER}}          <- owner
  {{REVIEW_REASON}}  <- review_reason
  {{FOLLOW_UP}}      <- optional, no source value

Document plan:
  Create a separate review section for each selected source row.
  Preserve spreadsheet order.
  Leave {{FOLLOW_UP}} empty.
  Fail validation if any required placeholder remains unresolved.
```

The generated document contains distinct sections with traceable source identities, and the validator confirms that no required placeholder remains. The specialist's value lies in a conditional capability that is invisible during periods when its task family is absent. Unique-coverage credit and lifetime protection allow the ecosystem to recover this behavior by reactivating an existing skill rather than reconstructing it after the distribution changes.

Across these cases, ecological adaptation produces three qualitatively different outcomes. Complementary skills are retained and ordered when one skill validates or completes another. Conflicting skills are separated by narrower triggers and topology constraints. Redundant skills are merged into a more coherent procedure with explicit stopping and handoff conditions. Rare specialists remain available when they encode behavior that broader skills cannot reproduce. These outcomes arise from evaluating skills as members of routed coalitions rather than as isolated instructions.