MAS-SKILL: SKILL-GUIDED MULTI-AGENT SYSTEMS VIA COLLABORATIVE TOPOLOGY TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM)-based multi-agent systems (MASs) depend on collaboration topologies that assign roles and govern information exchange. Effective topologies vary across tasks and queries, yet selecting them from a combinatorial design space is costly. Rule-based systems reuse fixed templates with limited adaptability. Search-based systems repeatedly explore configurations and often fail to preserve useful collaboration experience. To better address these limitations, we introduce **MAS-Skill**, which converts execution trajectories into reusable skills that record topology, roles, tools, synthesis rules, applicability conditions, and execution evidence. During offline learning, it compares matched combinations of topologies and agent counts, selects Pareto-optimal base skills by accuracy and cost, and uses genetic evolution to fuse supported structures with lessons from successful and failed trajectories. At inference time, the main agent retrieves a type-aware skill, instantiates its agents and communication structure, and uses a one-pass fallback planner when no reliable match exists. MAS-Skill outperforms strong baseline methods on six general benchmarks and a benchmark designed to evaluate multi-agent collaboration. Moreover, the learned skills transfer effectively to out-of-domain datasets and across model backbones[1].

1 INTRODUCTION

Large language model (LLM)-based agents have shown strong performance in question answering, code generation, mathematical reasoning, and web navigation Xi et al. (2025); Wang et al. (2024a). Building on these advances, multi-agent systems (MASs) coordinate several agents to generate complementary solutions, critique intermediate results, and synthesize a final answer Hong et al. (2024); Wu et al. (2024); Chen et al. (2024b). MAS performance depends on both agent capability and the collaboration topology that governs information exchange and division of responsibility. Selecting an appropriate topology is therefore central to system design.

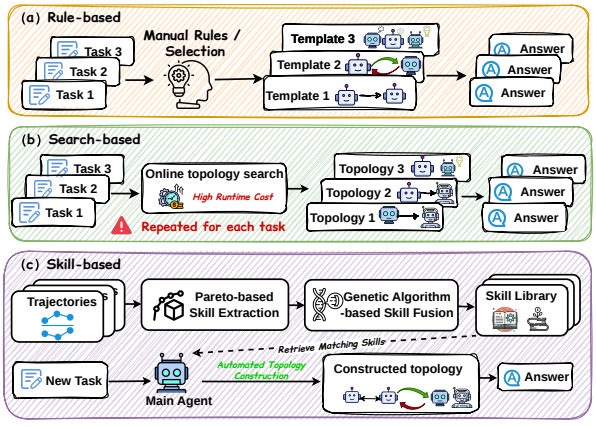

Figure 1: Three paradigms for MAS topology construction. Rule-based methods reuse fixed topologies, whereas search-based methods optimize candidate structures during system design or inference. MAS-Skill follows a **skill-based** paradigm that retrieves reusable collaboration experience.

A practical MAS must adapt its collaboration structure to the task at hand. Existing architectures include debate Du et al. (2024), brainstorming Liang et al. (2024), actor-critic Shinn et al. (2023), divide-and-conquer Qian et al. (2025), and

[1]Code are available in supplementary materials.

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.

centrally orchestrated workflows Hong et al. (2024); Wu et al. (2024); Zhuge et al. (2024). These structures have different strengths. A mathematical proof may benefit from decomposition, whereas an exploratory task may favor parallel brainstorming. The preferred interaction pattern can also change with query difficulty within the same domain. In practice, human teams rarely design their collaboration from scratch for every new problem. They reuse effective roles and interaction patterns from prior experience and adapt them to the current task. Ideally, an adaptive MAS should learn to collaborate in the same manner.

Existing methods approach this problem through two broad paradigms (Figure 1). Rule-based methods use fixed topologies and incur little or no topology-selection cost at inference time, but their handcrafted roles and communication patterns may not adapt well across queries. Search-based methods explore a broader design space during development or inference Liu et al. (2024); Zhuge et al. (2024); Zhang et al. (2025a). Query-dependent search can improve adaptability, but repeated candidate evaluation adds runtime cost. Methods that optimize a global workflow may instead require substantial design-time search. In both paradigms, the resulting configuration is often not retained in a structured form that can be retrieved for a later query together with its roles, tool assignments, and execution evidence.

Building such reusable records is nontrivial because the collaboration design space is combinatorial. Each configuration jointly specifies the agent count, role assignment, interaction graph, model choice, and synthesis strategy. Comparing two configurations reliably requires executing both on representative queries, while evaluating and storing every possible configuration is prohibitively expensive. The problem is therefore to identify a compact set of transferable collaboration patterns from limited execution trajectories.

To address this problem, we propose **MAS-Skill**, a skill-based framework that learns reusable collaboration patterns from execution trajectories and applies them to new queries. MAS-Skill consists of two stages (Figure 2). During offline skill learning, MAS-Skill compares predefined topology and agent-count configurations under matched conditions. It selects non-dominated configurations according to accuracy and token cost and stores them with their execution evidence as *base skills*. A genetic algorithm then recombines and mutates their topologies, roles, and synthesis strategies, using both successful and failed trajectories to produce *fused skills*. During skill-guided inference, the main agent retrieves a skill with TF-IDF and instantiates its topology, roles, tools, and synthesis rule when the match is reliable. Otherwise, it constructs a collaboration plan in one pass.

We summarize our contributions as follows:

1. **MAS-Skill Orchestration Framework**: We introduce MAS-Skill, a skill-guided multi-agent orchestration framework that distills successful collaboration trajectories into reusable skill records and retrieves them at inference time, shifting the optimization burden from per-query runtime search to offline skill learning.
2. **Base Skill Extraction and Genetic Fusion**: We develop a two-stage learning procedure that borrows the controlled-comparison idea of A/B testing to evaluate multi-parameter configuration combinations, extracts base skills through accuracy-cost Pareto selection, and then uses genetic evolution and trajectory experience to obtain fused skills beyond the initial grid.
3. **Accuracy Gains across Diverse Benchmarks**: Across six general benchmarks, MAS-Skill reaches an average score of 85.17%, 1.58 percentage points above MaAS (83.59%), the strongest baseline evaluated on all six tasks. On the collaboration benchmark, it obtains an average Overall GSR of 0.444, compared with 0.300 for MaAS, while using 5.5% fewer online inference tokens.

2 Related Work

LLM Agents and Agentic Systems. LLM-based agents combine role prompting Wang et al. (2024b), tools Shinn et al. (2023), planning Zhou et al. (2024a), and memory Xi et al. (2025) to solve complex tasks. Multi-agent frameworks extend these capabilities through specialized roles and structured communication. AutoGen Wu et al. (2024) supports flexible conversational patterns. MetaGPT Hong et al. (2024) encodes standard operating procedures as role-based workflows.

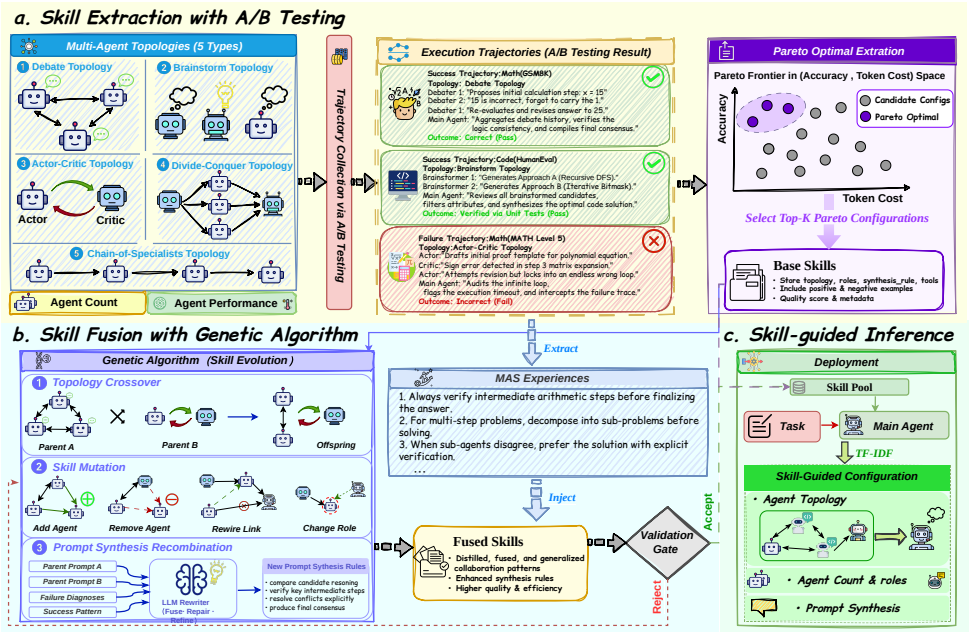

Figure 2: Overview of MAS-Skill. Offline skill learning extracts base skills through A/B testing and Pareto selection, then evolves fused skills with a genetic algorithm. Skill-guided inference retrieves a relevant skill and instantiates its collaboration topology.

AgentVerse Chen et al. (2024b) studies coordinated agent groups, and ChatDev Qian et al. (2024) models software development as a virtual company. Debate-based methods further use iterative critique to improve reasoning and factuality Du et al. (2024); Liang et al. (2024). These systems establish useful collaboration primitives, but their interaction patterns are generally specified at design time rather than retrieved from a library of prior executions.

Automated MAS Design. Automated MAS design has been studied through communication optimization, agent selection, and workflow search. GPTSwarm Zhuge et al. (2024) and DyLAN Liu et al. (2024) optimize inter-agent communication, while AutoAgents Chen et al. (2024a) generates agent teams. MaAS Zhang et al. (2025a) parameterizes a distribution over agentic architectures with an agentic supernet, and AFlow Zhang et al. (2025b) applies Monte Carlo tree search to workflow generation. Related adaptive reasoning methods Wei et al. (2022); Yao et al. (2023); Zhou et al. (2024b); Besta et al. (2024) optimize the structure of intermediate reasoning rather than the multi-agent interaction graph. MAS-Skill is complementary to these approaches. It uses offline exploration to construct executable collaboration records and retrieves those records for later queries instead of treating each selected structure as an isolated outcome.

Skill Learning for LLM Agents. Voyager Wang et al. (2024a) stores automatically discovered embodied behaviors as executable code, Reflexion Shinn et al. (2023) accumulates verbal feedback, and ExpeL Zhao et al. (2024) extracts reusable insights from prior attempts. SkillOrchestra Wang et al. (2026) builds skill handbooks that describe agent competence and cost for routing. MAS-Skill extends experience reuse from individual-agent behavior or capability routing to a complete collaboration protocol. A skill jointly specifies the topology, roles, tool bindings, synthesis strategy, applicability conditions, and execution evidence needed to instantiate a multi-agent workflow.

3 METHOD

Figure 2 illustrates how MAS-Skill learns collaboration skills from solved tasks and reuses them through online inference. We first formulate the skill library, then describe offline learning and skill-guided execution.

3.1 MAS Formulation and Skill Library

Let $\mathcal{D} = \{(x_i, y_i, t_i)\}_{i=1}^{M}$ denote M training tasks with query x_i, reference y_i, and optional type t_i. A sub-agent $a_j = (m_j, r_j, u_j)$ specifies its model preference, role, and tools. A multi-agent system (MAS) configuration is

$$g = (\tau, n, \mathbf{a}, s), \qquad \mathbf{a} = (a_1, \ldots, a_n), \tag{1}$$

where τ, n, $\mathbf{a}$, and s denote the topology, agent count, agent definitions, and synthesis instruction. Executing g on x_i yields $\hat{y}_{i,g}$ at token cost $c_{i,g}$ and records the planning, sub-agent interaction, and synthesis steps in trajectory $\pi_{i,g}$.

MAS-Skill considers five topologies: **Debate** Du et al. (2024) alternates independent proposals and critiques; **Brainstorm** Liang et al. (2024) synthesizes diverse solutions generated in parallel; **Actor-Critic** Shinn et al. (2023) revises an actor's draft from critic feedback; **Divide-and-Conquer** Qian et al. (2025) assigns subproblems to different agents and merges their solutions; and **Chain-of-Specialists** passes each agent's output to the next in an ordered pipeline.

MAS-Skill converts a useful configuration and its execution evidence into a skill

$$\sigma = \left(g, \kappa, d, \mathcal{E}^{+}, \mathcal{E}^{-}, \mathbf{e}, q\right). \tag{2}$$

Here, κ and d describe suitable queries and the collaboration pattern; $\mathcal{E}^{+}$ and $\mathcal{E}^{-}$ store successful and failed tasks; and $\mathbf{e}$ summarizes effective strategies and failure diagnoses. The quality $q \in [0, 1]$ comes from a base skill's Pareto score or a fused skill's validation fitness. Thus, g remains executable, while the other fields provide evidence for retrieval, adaptation, and validation. The library contains extracted base skills and evolved fused skills.

3.2 Learning Reusable Collaboration Skills

MAS-Skill constructs the library in two stages. It first extracts non-dominated base skills through controlled comparisons over a configuration grid, which we retain under the name A/B testing, and then applies a genetic algorithm to recombine their structures and trajectory-derived experience beyond that grid.

Skill Extraction with A/B Testing. We use the controlled-comparison principle of A/B testing for matched multi-parameter combination tests, not as a binary causal intervention. For each query, we fix the backbone, decoding, and evaluator while complete configurations jointly vary topology, agent count, roles, tools, and synthesis. End-to-end comparison is appropriate because a skill is an executable combination of these interdependent components.

Let $\mathcal{T}$ and $\mathcal{N}$ denote the topology set and candidate agent counts. Template $\Phi(\tau, n)$ assigns agent definitions $\mathbf{a}_{\tau,n}$ and synthesis rule $s_{\tau,n}$, yielding

$$\mathcal{G} = \{g_{\tau,n} = (\tau, n, \mathbf{a}_{\tau,n}, s_{\tau,n}) : \tau \in \mathcal{T},\ n \in \mathcal{N}\}. \tag{3}$$

For each x_i, MAS-Skill executes every $g \in \mathcal{G}$ under matched conditions, producing

$$z_{i,g} = (x_i, t_i, g, \ell_{i,g}, c_{i,g}, \pi_{i,g}), \tag{4}$$

where $\ell_{i,g}$ is the correctness label, $c_{i,g}$ is token cost, and $\pi_{i,g}$ records planning, interaction, tool use, and synthesis. These records identify reliable, efficient combinations and provide success and failure evidence for fusion.

Pareto extraction of base skills. Because the grid trades accuracy against cost, we aggregate trajectories by task type and optimize both. Let $\mathcal{I}_t = \{i : t_i = t\}$ denote the indices of type t; without labels, all tasks form one group. We compute

$$\mathrm{Acc}_t(g) = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \ell_{i,g},\ \ \mathrm{Tok}_t(g) = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} c_{i,g}, \tag{5}$$

Here, $\mathrm{Acc}_t(g)$ is the success rate and $\mathrm{Tok}_t(g)$ is the mean online token cost. We remove g if another configuration is no less accurate, no more expensive, and strictly better on either metric. The remaining Pareto configurations preserve distinct trade-offs between accuracy and cost and are ranked by

$$Q_t(g) = w_{\mathrm{acc}} \widetilde{\mathrm{Acc}}_t(g) + w_{\mathrm{tok}} \left(1 - \widetilde{\mathrm{Tok}}_t(g)\right), \tag{6}$$

Both metrics are min-max normalized among Pareto candidates. We set $w_{\text{acc}} = 0.6$ and $w_{\text{tok}} = 0.4$ and retain at most two configurations per task type. Their correct and failed trajectories populate $\mathcal{E}^+$ and $\mathcal{E}^-$; task type, roles, tools, and trajectory summaries determine κ, d, and $\mathbf{e}$. These records form $\mathcal{S}_{\text{base}}$ and provide diverse, empirically supported seeds for genetic evolution.

Skill Fusion with Genetic Algorithm. The initial grid varies topology and agent count but covers only a fraction of the role, tool, edge, and synthesis space. A genetic algorithm (GA) expands this space by encoding each candidate as

$$\gamma = (\tau, n, \mathbf{a}, s)\,, \tag{7}$$

following Equation 1. Base skills initialize the population, and each generation applies the three operations in Figure 2.

An offspring inherits topology and agent count from one parent and compatible roles, tools, or synthesis components from another. This searches beyond the template grid while preserving supported parent structures, for example by combining one skill's interaction graph with another's specialist assignments.

Crossover can only rearrange existing components. MAS-Skill therefore adds or removes an agent, rewires an edge, changes a role or model preference, or adjusts a tool profile. By modifying only part of the genome, mutation repairs over-specialized roles, redundant agents, and missing dependencies while limiting premature convergence.

A suitable topology may still fail under an underspecified synthesis rule. An LLM rewriter combines parent prompts with successful patterns and failure diagnoses in $\mathbf{e}$. It retains effective strategies, adds verification and conflict resolution, and removes failure-associated behavior, transferring trajectory evidence rather than simply concatenating prompts.

After the three operations, MAS-Skill evaluates each offspring on a development set $\mathcal{D}_{\text{dev}}$ using

$$F(\gamma) = \text{Acc}_{\text{dev}}(\gamma) - \lambda \frac{\text{Tok}_{\text{dev}}(\gamma)}{Z}, \tag{8}$$

where Acc_{dev} and Tok_{dev} are validation accuracy and mean cost, Z normalizes cost, and λ controls its penalty. We use $\lambda = 0.05$, $Z = 8000$, 12 genomes, crossover probability 0.85, and mutation probability 0.65. Over five generations, two elites survive per generation, and only offspring improving this objective pass the validation gate.

Accepted genomes become fused skills whose development trajectories populate $\mathcal{E}^+$ and $\mathcal{E}^-$ and whose summaries combine parent evidence with new failures. They form $\mathcal{S}_{\text{fused}}$, giving

$$\mathcal{S} = \mathcal{S}_{\text{base}} \cup \mathcal{S}_{\text{fused}}. \tag{9}$$

3.3 MAS Inference with Skills

For query x, the main agent coordinates skill selection and collaboration planning. It forms $\mathcal{C}(x) = \mathcal{S}_{t_x}$ when metadata provides type t_x; otherwise, it predicts a coarse type and uses the full library if that group is empty. Each candidate document combines its description, trigger, roles, tools, and positive examples. Term frequency-inverse document frequency (TF-IDF) encodes the query and skill documents without an embedding model. The score is

$$\begin{aligned} s^+(x,\sigma) &= \cos(\mathbf{v}_x, \mathbf{v}_\sigma), \\ s^-(x,\sigma) &= \max_{e \in \mathcal{E}_\sigma^-} \cos(\mathbf{v}_x, \mathbf{v}_e), \\ S(x,\sigma) &= s^+(x,\sigma)\big(1 - \lambda_-(\sigma) s^-(x,\sigma)\big)\big(1 + \lambda_q q_\sigma\big). \end{aligned} \tag{10}$$

Here, $\mathbf{v}_x$, $\mathbf{v}_\sigma$, and $\mathbf{v}_e$ are TF-IDF vectors for the query, skill, and failed example. The term s^+ measures similarity between the query and skill; s^- is the maximum failed-example similarity, or zero if none exists. The coefficients $\lambda_-(\sigma)$ and λ_q control the failure penalty and quality boost. We set λ_- to 0.6 for base and 0.7 for fused skills, with $\lambda_q = 0.1$. The main agent selects $\sigma^* = \arg\max_{\sigma \in \mathcal{C}(x)} S(x,\sigma)$.

A confidence gate prevents a weak match from imposing an unsuitable structure. It uses $l = 0.12$ and $h = 0.42$: scores at or above h activate the skill, those below l reject it, and intermediate

Method	GSM8K	MATH	MultiArith	MBPP	HumanEval	HotpotQA	Avg.
Standard prompting methods							
Vanilla	87.45	46.29	96.85	71.83	87.08	64.10	75.60
CoT Wei et al. (2022)	87.10	46.40	96.31	71.83	88.13	73.40	77.20
SC (CoT×5) Wang et al. (2023)	87.57	47.91	96.58	73.60	88.60	74.00	78.04
Rule-based multi-agent systems							
MultiPersona Wang et al. (2024b)	87.50	45.43	97.49	73.19	88.32	73.10	77.51
LLM-Debate Du et al. (2024)	89.47	48.54	97.33	70.29	88.68	66.90	76.87
LLM-Blender Jiang et al. (2023)	88.35	46.92	97.29	77.05	88.80	70.40	78.14
Search-based multi-agent systems							
DyLAN Liu et al. (2024)	89.98	48.63	97.12	77.30	90.42	80.80	80.71
AgentVerse Chen et al. (2024b)	89.91	47.35	97.50	74.28	89.29	72.50	78.47
MacNet Qian et al. (2025)	87.95	45.18	96.03	65.28	84.57	56.10	72.52
AutoAgents Chen et al. (2024a)	87.69	45.32	96.42	71.95	87.64	66.80	75.97
GPTSwarm Zhuge et al. (2024)	89.14	47.88	96.79	77.43	89.32	68.10	78.11
ADAS Hu et al. (2025)	86.12	43.18	96.02	68.13	84.19	78.50	76.02
AgentSquare Shang et al. (2025)	87.62	48.51	97.77	78.46	89.08	71.70	78.86
AFlow Zhang et al. (2025b)	91.16	51.28	96.22	81.67	90.93	77.90	81.53
MaAS Zhang et al. (2025a)	92.30	51.82	98.80	82.17	92.85	83.60	83.59
Skill-based multi-agent systems							
MAS-Skill	**92.80**	**52.88**	**99.00**	**84.20**	**94.66**	**87.50**	**85.17**

Table 1: Performance on six general benchmarks with GPT-4o-mini. Values are accuracy (%) except pass@1 (%) for code generation. Baseline results on the first five benchmarks follow Zhang et al. (2025a); HotpotQA uses exact match. Avg. is the macro-average over all six benchmarks.

scores invoke an LLM judge. Accepted skills instantiate their topology, roles, tools, experience, and synthesis rule. If no candidate is accepted, the main agent constructs a one-shot plan from the same design space. This avoids iterative topology evaluation for familiar tasks while retaining an adaptive fallback.

4 Experiments

4.1 Experimental Setup

Datasets and metrics. We evaluate on seven benchmarks covering four task types. GSM8K Cobbe et al. (2021), MATH Hendrycks et al. (2021), and MultiArith Roy & Roth (2015) measure mathematical reasoning with exact-match accuracy. MBPP Austin et al. (2021) and HumanEval Chen et al. (2021) measure code generation with pass@1. HotpotQA Yang et al. (2018) measures multi-hop question answering with exact match. We use all 800 items in the HotpotQA test split provided by the MaAS data manifest. The multiagent-collab-scenario-benchmark Shu et al. (2024) evaluates specialist collaboration with Overall Goal Success Rate (GSR). A scenario receives a score of one only when the output satisfies every user-facing and system-facing assertion. Its Travel, Mortgage, and Software domains each contain 30 scenarios and provide 10, 6, and 8 available agents, respectively. We report mean online inference tokens as the cost metric and exclude offline skill construction.

Baselines. We compare MAS-Skill with standard prompting baselines, hand-crafted MAS methods, and automated topology and workflow methods. For GSM8K, MATH, MultiArith, MBPP, and HumanEval, we follow the evaluation subsets and published baseline results reported by MaAS Zhang et al. (2025a). We evaluate HotpotQA on the complete 800-item test split with the same backbone. For the specialist benchmark, we reproduce AFlow and MaAS with their standard multi-agent workflows on all 90 scenarios, using the same scenario context, backbone, evaluator, and test protocol as MAS-Skill.

Implementation details. All experiments except cross-model transfer use GPT-4o-mini as the main-agent backbone. MAS-Skill evaluates five predefined topologies with three agent counts (2 to 4), selects configurations on the accuracy-token Pareto frontier, and applies genetic fusion. Skill construction and validation use examples disjoint from every reported test set, and GA development

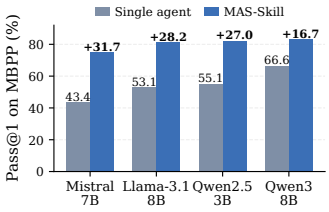

Figure 3: Cross-model transfer on MBPP.

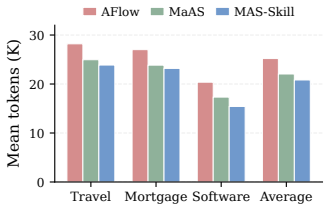

Figure 4: Online tokens per scenario.

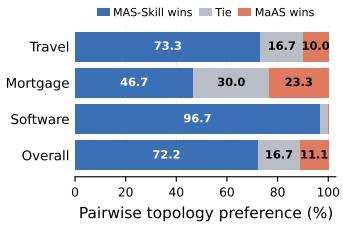

Figure 5: Pairwise topology preference.

remains restricted to the skill-learning split. For each specialist domain, we conduct three-fold cross-validation with 20 scenarios for skill learning and 10 disjoint scenarios for testing per fold. The GPT-4o-mini evaluator scores every specialist-collaboration run. We set the temperature to 0, allow at most five concurrent API calls with deterministic answer extraction.

4.2 Main Results

Tables 1 and 2 evaluate reusable skills on general tasks and specialist collaboration. MAS-Skill leads every general-benchmark column and reaches an 85.17% macro-average, 1.58 percentage points above MaAS, with gains across reasoning, code generation, and multi-hop question answering. The largest gain on HotpotQA is consistent with invoking a relevant skill when matched and retaining the main agent's normal planning process otherwise.

Method	Travel	Mortgage	Software	**Avg.**
AFlow	0.100	0.067	**0.533**	0.233
MaAS	0.300	0.133	0.467	0.300
MAS-Skill	**0.367**	**0.500**	0.467	**0.444**

Table 2: Overall GSR on the multiagent-collab-scenario-benchmark with GPT-4o-mini. AFlow and MaAS are reproduced with their standard workflows. Each domain aggregates 30 held-out predictions from three folds.

MAS-Skill also improves specialist collaboration, reaching an average GSR of 0.444 compared with 0.300 for MaAS and 0.233 for AFlow. The largest gain appears in Mortgage, where tasks often require ordered retrieval, validation, and domain-specific interpretation. This pattern matches the role dependencies stored in a collaboration skill. The result is weaker on Software, where MAS-Skill matches MaAS and trails AFlow. Software scenarios combine a wider range of coding, testing, review, infrastructure, and deployment roles, so performance remains sensitive to specialist selection.

5 Analysis

5.1 Cross-Model Skill Transferability

We next test whether a precomputed skill library remains useful after replacing the main-agent backbone. We reuse four Pareto-selected MBPP skills learned from 86 GPT-4o-mini training items and 1,290 exploration runs without additional extraction or fusion. Evaluation uses all 341 items in the MaAS MBPP test split. We replace the main reasoning backbone and the `strong_logic` slot with Llama-3.1-8B, Mistral-7B, Qwen2.5-3B, or Qwen3-8B. Other supporting roles remain on GPT-4o-mini. Figure 3 shows gains from 16.71 to 31.67 percentage points over direct inference with the substituted backbones. The precomputed guidance therefore remains effective after these main-agent substitutions. Because the comparison adds fixed GPT-4o-mini supporting roles, it does not isolate how much of each gain comes from the transferred skill rather than the supporting model pool.

5.2 Ablation Study

We isolate whether the skill pool, genetic fusion, and A/B-testing-inspired configuration comparison contribute to the final system. Table 3 uses the same full-system scores as the main results. Removing the complete skill pool causes the largest reduction, lowering GSM8K accuracy by 0.86 percentage points and HumanEval pass@1 by 3.06 points. Removing GA fusion reduces the two scores by 0.67 and 2.29 points, respectively, while removing A/B testing results in smaller drops of 0.38 and 0.77 points. These results show that all three components contribute to performance: the controlled combination tests provide empirically grounded seeds, and genetic fusion extends them beyond the initial grid. HumanEval benefits especially from the learned coder, reviewer, and tester structures. The full system costs 0.31 dollars on HumanEval, compared with 0.47 to 0.55 dollars for the ablations, which is consistent with reusing learned collaboration plans rather than repeatedly constructing them online.

Dataset	**GSM8K**		**HumanEval**	
Metric	**Cost**	**Acc.**	**Cost**	**Pass@1**
MAS-Skill	**4.53**	**92.80**	**0.31**	**94.66**
w/o skill pool	3.52	91.94	0.47	91.60
w/o GA fusion	4.52	92.13	0.53	92.37
w/o A/B testing	4.54	92.42	0.55	93.89

Table 3: Component ablation on GSM8K and HumanEval with GPT-4o-mini. Values within each dataset come from the same dedicated ablation run.

5.3 Inference Cost on the Collaboration Benchmark

We compare online inference cost with AFlow and MaAS under the reported implementations. Figure 4 reports mean tokens per specialist-collaboration scenario and excludes offline skill learning for every method. MAS-Skill uses 20,824 tokens on average, 5.5% fewer than MaAS and 17.4% fewer than AFlow. The reduction reaches 24.3% relative to AFlow in Software. This pattern is consistent with reusing a stored topology and role assignment, but the aggregate token counts do not isolate planning from agent execution.

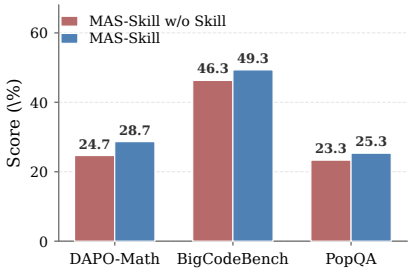

Figure 7: Zero-shot cross-dataset skill transfer. Each source-domain skill library is frozen and applied to an unseen target dataset without target-domain training.

5.4 Pairwise Topology Quality

We examine whether a fixed rubric favors MAS-Skill structures independently of final task success. A GPT-5.6-sol judge scores each MAS-Skill and MaAS structure with a 10-point rubric covering capability coverage, dependency structure, interaction fit, and parsimony. A score difference within 0.5 counts as a tie. Figure 5 shows that the judge prefers MAS-Skill in 72.2% of all comparisons, records ties in 16.7%, and prefers MaAS in 11.1%.

The preference rises to 96.7% on Software, whose tasks often require explicit code, test, review, infrastructure, and deployment roles. The rubric rewards this capability coverage even when execution errors limit end-to-end GSR. On Mortgage, the judge prefers MAS-Skill in 46.7% because many scenarios are simple operations or out-of-scope rejections that need little collaboration. We retained per-item topology metadata only for Fold 1. For Folds 2-3, we reconstruct structures from the saved fold-level skill and task-type router. We therefore use this single-judge comparison as exploratory structural evidence.

5.5 Cross-Dataset Skill Transfer

To test whether the learned skills transfer beyond their source benchmarks, we conduct a zero-shot cross-dataset study. We freeze the skill libraries from MATH, HumanEval, and HotpotQA and apply them to held-out samples from DAPO-Math Yu et al. (2025), BigCodeBench Zhuo et al. (2025),

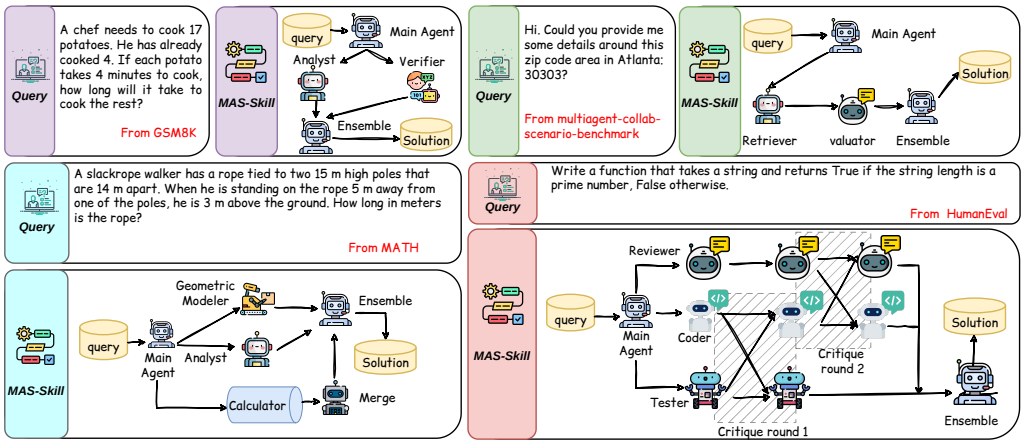

Figure 6: Representative MAS-Skill cases include verification-oriented arithmetic on GSM8K, divide-and-conquer geometric reasoning on MATH, multi-round code critique on HumanEval, and chain-of-specialists tool use on the collaboration benchmark.

and PopQA Mallen et al. (2023), respectively. No target example is used to extract or fuse skills, calibrate retrieval, or adapt the model. We compare MAS-Skill with *MAS-Skill w/o Skill*, which uses the same GPT-4o-mini backbone but disables skill retrieval and performs direct inference. Figure 7 shows consistent improvements in all three domains, including code generation with broader library dependencies and open-domain QA with a different answer distribution from HotpotQA. These gains indicate that the library preserves transferable interaction structures and role assignments that benefit unseen datasets without target-domain training.

5.6 Case Study

Figure 6 illustrates the retrieved collaboration structures.

Arithmetic reasoning. An actor–critic skill pairs an analyst deriving the solution with a verifier checking interpretation and calculation order. The main agent resolves their feedback, adding targeted verification without unnecessary parallel agents.

Geometric reasoning. A divide-and-conquer skill separates modeling from calculation. A modeler decomposes the scene, an analyst and calculator solve and cross-check subproblems, and a merge agent combines verified results to contain modeling errors.

Code generation. A debate skill combines coder drafts, reviewer logic checks, and tester boundary checks. Exchanged critiques guide synthesis of agreed corrections, providing complementary evidence.

Domain collaboration. A Chain-of-Specialists skill routes neighborhood evidence from a tool-using retriever to a valuator for interpretation. The main agent synthesizes both outputs into a report, preserving the dependency between tool use and domain interpretation.

6 Conclusion

MAS-Skill represents multi-agent collaboration as reusable skills. It compares multi-parameter configurations with an A/B-testing-inspired protocol, selects Pareto-optimal skills by accuracy and cost, and evolves them from trajectory evidence. At inference, the main agent retrieves a matched skill or plans directly. MAS-Skill averages 85.17% across six benchmarks, 1.58 percentage points above MaAS, and attains 0.444 Overall GSR with 5.5% fewer online tokens than MaAS. Transfer and ablation results support skill reuse. Together, these findings show that a skill library can amortize offline topology discovery while preserving adaptation to each query. Evaluation remains limited to the tested models and tasks. Online cost excludes offline skill construction, retrieval relies on lexical similarity, and topology quality is assessed by a single judge.

AI USE STATEMENT

In this work, we used generative AI tools only for final-stage language polishing and grammar refinement of the manuscript. We have not used generative AI tools for generating research ideas, designing methods, conducting experiments, implementing code, analyzing data, interpreting results, or producing scientific claims, and other generative AI-assisted tasks requiring disclosure are not applicable to this work. We have reviewed all AI-assisted revisions to ensure their correctness and consistency with the authors' intended content. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have made efforts to ensure the reproducibility of this work by providing detailed descriptions of the proposed method, experimental settings, implementation details, and evaluation protocols in the main paper and supplementary materials. The supplementary materials include the source code required to reproduce our experiments, and we will publicly release the code after the review process to further facilitate verification and future research. The content of this paper can be completely reproduced.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024. doi: 10.1609/aaai.v38i16.29720. URL https://ojs.aaai.org/index.php/AAAI/article/view/29720.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F. Karlsson, Jie Fu, and Yemin Shi. AutoAgents: A framework for automatic agent generation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 22–30, 2024a. doi: 10.24963/ijcai.2024/3. URL https://www.ijcai.org/proceedings/2024/3.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 20094–20136, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/578e65cdee35d00c708d4c64bce32971-Paper-Conference.pdf.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John

Schulman. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, zili wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/6507b115562bb0a305f1958ccc87355a-Paper-Conference.pdf.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/36b7acf6f6010652b3f2a433774a66fe-Abstract-Conference.html.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, 2023. doi: 10.18653/v1/2023.acl-long.792. URL https://aclanthology.org/2023.acl-long.792/.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17889–17904, 2024. doi: 10.18653/v1/2024.emnlp-main.992. URL https://aclanthology.org/2024.emnlp-main.992/.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *Proceedings of the First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=XII0Wp1XA9.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822, 2023. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, 2024. doi: 10.18653/v1/2024.acl-long.810. URL https://aclanthology.org/2024.acl-long.810/.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*, volume 2025, pp. 41488–41505, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/66a026c0d17040889b50f0dfa650e5e0-Abstract-Conference.html.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, 2015. doi: 10.18653/v1/D15-1202. URL https://aclanthology.org/D15-1202/.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. AgentSquare: Automatic LLM agent search in modular design space. In *International Conference on Learning Representations*, volume 2025, pp. 3841–3865, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/0ae94013da7cd459402fd77874e09ee3-Abstract-Conference.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Raphael Shu, Nilaksh Das, Michelle Yuan, Monica Sunkara, and Yi Zhang. Towards effective GenAI multi-agent collaboration: Design and evaluation for enterprise applications. arXiv preprint arXiv:2412.05449, 2024. URL https://arxiv.org/abs/2412.05449.

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

Jiayu Wang, Yifei Ming, Zixuan Ke, Shafiq Joty, Aws Albarghouthi, and Frederic Sala. SkillOrchestra: Learning to route agents via skill transfer. arXiv preprint arXiv:2602.19672, 2026. URL https://arxiv.org/abs/2602.19672.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 257–279, 2024b. doi: 10.18653/v1/2024.naacl-long.15. URL https://aclanthology.org/2024.naacl-long.15/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Proceedings of the First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, Qi Zhang, and Tao Gui. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025. doi: 10.1007/s11432-024-4222-0. URL https://doi.org/10.1007/s11432-024-4222-0.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 75834–75852, 2025a. URL https://proceedings.mlr.press/v267/zhang25bi.html.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html. Oral.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160, 2024a. URL https://proceedings.mlr.press/v235/zhou24r.html.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. SELF-DISCOVER: Large language models self-compose reasoning structures. In *Advances in Neural Information Processing Systems*, volume 37, pp. 126032–126058. Curran Associates, Inc., 2024b. doi: 10.52202/079017-4004. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e41efb03e20ca3c231940a3c6917ef6f-Abstract-Conference.html.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767, 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, James Hoang, Armel Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro von Werra. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, volume 2025, pp. 66602–66656, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a6a90bcc2aa470c3871b2d39a67d26e8-Abstract-Conference.html. Oral.

A Supplementary Overview

This appendix provides implementation details, formal definitions, experimental protocols, and extended numerical results for MAS-Skill. It is intended to make the design choices in the main paper easier to inspect. The main paper is self-contained: its method, evaluation protocol, and claims do not depend on this supplement, and no additional result introduced here is required to interpret the main findings.

The supplement follows the two phases of the framework. Appendices B–D expand the offline construction of a reusable collaboration-skill library. Appendix E describes online retrieval, gating, and execution. Appendix F documents the datasets and evaluation controls. Appendix G reproduces complete skill records from the benchmark libraries. Appendix H converts results already shown in the main paper and its figures into exact tables and clarifies the scope of the analysis. We introduce no new experimental runs or unverified values.

B Detailed Skill Representation

B.1 Executable Skill Record

For a query x, a multi-agent configuration is

$$g = (\tau, n, \mathbf{a}, s), \qquad \mathbf{a} = (a_1, \ldots, a_n), \tag{11}$$

where τ is a communication topology, n is the agent count, $a_j = (m_j, r_j, u_j)$ gives the model preference, role, and tools of agent j, and s is the final synthesis instruction. Executing g produces an answer, an online token cost, and a trajectory that records planning, messages, tool calls, and synthesis.

MAS-Skill stores an executable configuration together with the evidence needed to decide when it should be reused:

$$\sigma = (g, \kappa, d, \mathcal{E}^+, \mathcal{E}^-, \mathbf{e}, q). \tag{12}$$

The value $q \in [0, 1]$ is an empirical quality signal produced by Pareto ranking for a base skill or by validation fitness for a fused skill. It is not treated as a probability. Table 4 lists the fields exposed by the review package.

B.2 Five Collaboration Topologies

The same role set can behave differently under different information flows. MAS-Skill therefore stores the topology explicitly rather than treating a skill as a prompt collection. Table 5 gives the operational semantics used by the execution layer.

Topology handlers preserve a common trace interface. Every message is associated with a source role, destination or stage, token usage, and optional tool observation. This shared representation makes skills comparable even when their graphs differ. It also lets the main agent synthesize a result from the complete evidence rather than only from the final sub-agent message.

To normalize trajectories, MAS-Skill maps their topology-specific event names into a common sequence of planning, agent-message, tool, revision, and synthesis events. Each event retains its role and stage, but transport-level request data and repeated prompt prefixes are discarded. Correctness and usage are stored once at the execution-record level so that a long debate does not receive more statistical weight merely because it produces more messages.

Positive and negative evidence is also bounded. The skill record keeps compact task descriptions that characterize the local region in which the structure was observed. It does not copy every training prompt or every message from a successful run. Experience summaries describe reusable actions, such as checking a boundary condition or passing retrieved evidence to a downstream specialist. They avoid instance-specific answers. Failed runs are summarized by the condition associated with failure and the relevant diagnostic, which supports a retrieval penalty without making the skill document a list of memorized outputs.

This normalization serves two purposes. Pareto statistics remain tied to complete executions, while textual retrieval receives a concise and comparable record. The executable plan is also separated

Table 4: Normalized collaboration-skill record used by MAS-Skill. Long raw traces are summarized before storage; the record retains the fields required for retrieval and execution.

Field	Symbol / type	Purpose	Execution use
Topology	τ / categorical	One of five supported information-flow structures.	Selects the topology handler and determines which agents may exchange intermediate results.
Agent count	n / integer	Number of participating sub-agents, excluding the coordinating main agent.	Instantiates the corresponding number of role specifications.
Roles and models	$\mathbf{a}$ / list	Role instruction and model preference for each participant.	Binds responsibilities such as solving, checking, testing, retrieving, or merging.
Tools	$u_1, \ldots, u_n$ / lists	Tools allowed for each role.	Restricts tool access and keeps tool evidence associated with its responsible specialist.
Synthesis	s / text	Rule for resolving, ordering, or merging agent outputs.	Prompts the main agent to produce the final response from the complete interaction trace.
Applicability	κ / text	Trigger that characterizes suitable queries, task types, and constraints.	Contributes to the searchable skill document and the confidence decision.
Description	d / text	Human-readable summary of the collaboration pattern.	Supports retrieval and review of the selected structure.
Positive evidence	$\mathcal{E}^+$ / examples	Compact examples of tasks on which the configuration succeeded.	Adds task vocabulary and successful patterns to retrieval and adaptation.
Negative evidence	$\mathcal{E}^-$ / examples	Compact examples of failures or unsuitable cases.	Penalizes candidates that resemble known failures.
Experience	$\mathbf{e}$ / text or list	Successful strategies, failure diagnoses, and repair guidance summarized from trajectories.	Guides synthesis rewriting and provides execution reminders.
Quality	q / scalar	Pareto score or validation fitness.	Applies a small multiplicative quality boost during retrieval.

Table 5: Operational information flow for the five topology families.

Topology	Information flow
Debate	Agents propose solutions, inspect competing proposals, and exchange targeted critiques before synthesis.
Brainstorm	Agents generate diverse proposals in parallel without requiring early agreement; the main agent consolidates complementary evidence.
Actor–Critic	An actor drafts a solution, a critic identifies defects, and the actor or synthesizer incorporates the feedback.
Divide-and-Conquer	A decomposer assigns subproblems, specialists solve them independently or with limited dependencies, and a merge step reconciles the parts.
Chain-of-Specialists	Outputs pass through an ordered sequence. Later roles receive the evidence produced by earlier roles, which supports tool-dependent workflows.

from empirical evidence: topology, roles, tools, and synthesis define what the system will do; examples and experience explain when the plan has or has not worked. Genetic fusion may modify the executable portion, but an offspring receives new quality and evidence only after development-set validation.

C Controlled Configuration Comparison and Base-Skill Extraction

C.1 Matched Multi-Parameter Comparisons

We retain the name A/B testing because the construction stage borrows its controlled-comparison principle. Here an "arm" is a complete multi-parameter collaboration configuration, not one isolated scalar change. For every query, the backbone, decoding setting, evaluator, and input are held fixed. The varied arm jointly determines topology and agent count. A template $\Phi(\tau, n)$ then specifies the compatible roles, tool profiles, and synthesis rule:

$$\mathcal{G} = \{g_{\tau,n} = (\tau, n, \mathbf{a}_{\tau,n}, s_{\tau,n}) : \tau \in \mathcal{T}, n \in \mathcal{N}\}. \tag{13}$$

Consequently, the observed difference compares complete end-to-end designs under matched conditions. It does not claim that the effect of topology can be identified independently from the roles and synthesis rule supplied by its template.

For training item i and configuration g, the stored record is

$$z_{i,g} = (x_i, t_i, g, \ell_{i,g}, c_{i,g}, \pi_{i,g}), \tag{14}$$

where t_i is an optional task type, $\ell_{i,g}$ is correctness, $c_{i,g}$ is online token cost, and $\pi_{i,g}$ is the execution trajectory. These records provide matched outcome and cost measurements and preserve both positive and negative execution evidence.

C.2 Accuracy–Cost Pareto Selection

Let $\mathcal{I}_t$ be the training indices associated with task type t. If type labels are unavailable, all items are assigned to one group. For each configuration,

$$\mathrm{Acc}_t(g) = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \ell_{i,g}, \tag{15}$$

$$\mathrm{Tok}_t(g) = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} c_{i,g}. \tag{16}$$

A configuration is removed when another is no less accurate, no more expensive, and strictly better on at least one measure. Among the remaining Pareto candidates, accuracy and token cost are independently min–max normalized:

$$\widetilde{v}(g) = \begin{cases} \dfrac{v(g) - v_{\min}}{v_{\max} - v_{\min}}, & v_{\max} > v_{\min}, \\ 1, & v_{\max} = v_{\min}. \end{cases} \tag{17}$$

The tie case prevents an arbitrary division by zero and gives all candidates the same contribution on an uninformative dimension. Candidates are ranked by

$$Q_t(g) = 0.6\, \widetilde{\mathrm{Acc}}_t(g) + 0.4 \left(1 - \widetilde{\mathrm{Tok}}_t(g)\right). \tag{18}$$

The library retains at most two base skills per task type. This cap preserves different accuracy–cost trade-offs while limiting redundant retrieval candidates. Positive and failed trajectories populate $\mathcal{E}^+$ and $\mathcal{E}^-$, and concise trajectory summaries populate $\mathbf{e}$.

D Genetic Skill Fusion

D.1 Genome and Operators

The template grid varies topology and agent count, but it does not enumerate all compatible role, tool, edge, and synthesis combinations. Genetic fusion expands this design space while starting from empirically supported base skills. Each genome has the executable form

$$\gamma = (\tau, n, \mathbf{a}, s). \tag{19}$$

Crossover takes the topology and agent count from one parent and compatible roles, tools, or synthesis components from another. Compatibility checks prevent references to absent roles and remove tools that are unavailable to the new owner. Mutation changes one bounded component: it can add or remove an agent, rewire an edge, replace a role or model preference, adjust a tool profile, or revise the synthesis rule. These local changes target redundant agents, missing dependencies, and over-specialized instructions without discarding the entire parent structure.

Structural recombination alone cannot express lessons found in trajectories. An LLM-based rewriting step therefore receives the parent synthesis instructions, successful patterns, and failure diagnoses. It produces a coherent rule that preserves supported strategies, adds verification or conflict resolution when needed, and removes behavior associated with failure. The rewriter is constrained to the new genome: it cannot invent a role or tool that the candidate does not contain.

Each offspring is evaluated on a development split disjoint from the reported test set. Its validation fitness is

$$F(\gamma) = \mathrm{Acc}_{\mathrm{dev}}(\gamma) - 0.05\frac{\mathrm{Tok}_{\mathrm{dev}}(\gamma)}{8000}. \tag{20}$$

The population contains 12 genomes. We run five generations, preserve two elites per generation, and use crossover and mutation probabilities of 0.85 and 0.65. A candidate enters the fused library only when it passes the validation gate and improves the fitness criterion used for its comparison. The accepted development trajectories refresh its positive and negative evidence rather than copying parent outcomes unchanged.

D.2 Validation Invariants

The validation gate checks structural validity before measuring fitness. The agent count must match the number of role specifications, every topology edge must reference an existing role, and each assigned tool must be present in the allowed registry. A chain must have a valid first and final stage; an actor–critic plan must identify the draft and review responsibilities; and a divide-and-conquer plan must specify how partial results return to synthesis. The synthesis instruction may refer only to roles and evidence available in the candidate.

Candidates that pass these checks are executed on the same development items used for their comparison. Correctness is evaluated with the dataset-specific metric and token cost is aggregated from the complete trace. The gate therefore does not accept a syntactically plausible genome on prompt appearance alone. Conversely, a rejected offspring does not alter its parents or their evidence. Elitism keeps the two best existing genomes available while the remaining population explores bounded changes.

E Retrieval and Skill-Guided Inference

E.1 Evidence-Aware Retrieval

Candidate generation first uses task metadata when available. For an inferred or supplied type t_x, the main agent searches the corresponding group $\mathcal{S}_{t_x}$; if that group is empty, it searches the full library. A skill document concatenates its description, applicability condition, roles, tools, synthesis cues, and compact positive examples. Query and skill documents are encoded by TF–IDF, avoiding a separate embedding model.

For query x and skill σ,

$$s^+(x,\sigma) = \cos(\mathbf{v}_x, \mathbf{v}_\sigma), \tag{21}$$

$$s^-(x,\sigma) = \max_{e \in \mathcal{E}_\sigma^-} \cos(\mathbf{v}_x, \mathbf{v}_e), \tag{22}$$

$$S(x,\sigma) = s^+(x,\sigma)\left(1 - \lambda_-(\sigma)s^-(x,\sigma)\right)(1 + 0.1q_\sigma). \tag{23}$$

If a skill has no failed example, s^- is zero. The failure coefficient is 0.6 for base skills and 0.7 for fused skills. The larger fused-skill penalty is conservative because a recombined candidate may cover a narrower region than its parents. The quality coefficient is 0.1, so lexical relevance remains the principal signal and quality only breaks otherwise similar matches.

Algorithm 1 Offline Collaboration-Skill Learning

Require: Training set $\mathcal{D}$, development set $\mathcal{D}_{\text{dev}}$, topologies $\mathcal{T}$, counts $\mathcal{N}$, templates Φ
Ensure: Skill library $\mathcal{S}$

```
Construct complete arms G ← {Φ(τ, n)}
for each matched query x_i ∈ D do
    for each g ∈ G do
        Execute g and record correctness, cost, and trajectory
    end for
end for
for each task type t do
    Compute accuracy and mean online tokens for each g
    Remove dominated configurations
    Min–max normalize the Pareto candidates
    Retain at most two candidates ranked by 0.6/0.4
    Build base records with positive and negative evidence
end for
Initialize 12 genomes from the base records
for generation = 1, ..., 5 do
    Preserve the two highest-fitness elites
    Apply crossover with probability 0.85
    Apply bounded mutation with probability 0.65
    Rewrite synthesis rules from parent experience
    Validate offspring on D_dev
    Admit only candidates that pass the validation gate
end for
return S_base ∪ S_fused
```

E.2 Confidence Gate and Execution

The highest-scoring candidate is evaluated by a three-region gate. A score at or above 0.42 activates the skill. A score below 0.12 rejects it. The intermediate region invokes an LLM judge that receives the query, applicability condition, topology summary, and evidence but not the reference answer. This judge resolves ambiguous lexical matches, such as queries that share surface terms but require different tool dependencies.

An accepted record instantiates its topology, roles, tools, experience, and synthesis rule. The main agent gathers the outputs exposed by the selected handler and then performs one final synthesis. When no skill is accepted, the main agent constructs a collaboration plan once from the same topology design space. The fallback does not perform iterative candidate search or compare multiple candidate executions online.

E.3 Decision Boundary and Traceability

The confidence regions separate retrieval from execution. The high region is intended for a candidate whose trigger and evidence provide a direct match. The low region avoids imposing a stored structure when lexical overlap is weak or resembles known failure cases. Only the middle region incurs an additional judgment call. The judge returns an applicability decision; it does not rewrite the skill, evaluate multiple structures, or see a gold answer.

For inspection, the online trace records the candidate set, component scores, threshold region, optional judge decision, selected skill identifier, and instantiated plan. It then records topology events and token usage through the same interface used in offline exploration. This bookkeeping makes it possible to distinguish a retrieval rejection, a binding error, a tool failure, and an incorrect synthesis. Such distinctions are used for analysis and later offline skill construction, not for repeated optimization of the current query.

Algorithm 2 Online Skill-Guided Inference

Require: Query x, skill library $\mathcal{S}$, thresholds $l = 0.12$, $h = 0.42$
Ensure: Final answer $\hat{y}$

```
Infer or read coarse task type t_x
Form type-aware candidates C(x)
for each σ ∈ C(x) do
    Compute positive similarity s^+
    Compute failed-example similarity s^−
    Apply failure penalty and quality boost to obtain S
end for
σ* ← arg max_σ S(x, σ)
if S(x, σ*) ≥ h then
    accept σ*
else if S(x, σ*) < l then
    reject σ*
else
    ask the LLM judge whether σ* applies
end if
if a skill is accepted then
    Instantiate its agents, tools, topology, and synthesis rule
    Execute the selected topology and synthesize ŷ
else
    Construct one collaboration plan and execute it once
end if
return ŷ
```

F Experimental Protocol and Dataset Details

F.1 Benchmarks, Metrics, and Evaluation Sizes

Table 6 lists every benchmark used in the main paper. The six general benchmarks cover mathematical reasoning, code generation, and multi-hop question answering. The collaboration benchmark tests whether a system satisfies all user-facing and system-facing assertions while routing among domain specialists and tools. DAPO-Math, BigCodeBench, and PopQA are used only for zero-shot cross-dataset transfer. Their target examples are not used to extract, fuse, or calibrate the source-domain skill libraries.

F.2 Controls and Specialist Cross-Validation

Except in the cross-model study, GPT-4o-mini is the main-agent backbone. Decoding temperature is zero, answer extraction is deterministic, and at most five API calls are issued concurrently. Matched comparisons use the same query, backbone, evaluator, and decoding controls. Reported cost is online inference cost only; it excludes one-time exploration, base-skill extraction, fusion, and validation.

For each collaboration domain, the 30 scenarios are partitioned into three folds. In every fold, 20 scenarios support skill learning and 10 disjoint scenarios support evaluation. The partitions rotate so that each scenario is used as a test case exactly once. Domain results therefore aggregate 30 held-out predictions, and the overall result aggregates 90. A scenario is successful only if every required assertion passes. A GPT-4o-mini evaluator scores these runs under the same scenario context and evaluation protocol used for the reproduced AFlow and MaAS workflows.

In cross-model transfer, the source library is fixed. The substituted model is assigned to the main agent and the primary reasoning role. Other supporting roles continue to use GPT-4o-mini. The comparison therefore measures whether the stored collaboration skills remain useful after the main reasoning backbone changes. It does not isolate a fully homogeneous deployment of the substituted model.

Table 6: Evaluation protocol. Counts refer to the fixed evaluation manifests used in the paper. The code-and-data attachment contains three format examples per dataset, not the complete evaluation sets.

Benchmark	Task family	Metric	Items	Role in the paper
GSM8K	Mathematical reasoning	Exact-match accuracy	1,055	General benchmark and component ablation
MATH	Competition mathematics	Exact-match accuracy	486	General benchmark; source skills for DAPO-Math
MultiArith	Arithmetic word problems	Exact-match accuracy	180	General benchmark
MBPP	Python code generation	pass@1 from executable tests	341	General benchmark and cross-model transfer
HumanEval	Python code generation	pass@1 from executable tests	131	General benchmark; source skills for BigCodeBench
HotpotQA	Multi-hop QA	Exact match	800	General benchmark; source skills for PopQA
Collaboration benchmark	Specialist and tool collaboration	Overall Goal Success Rate	90	30 scenarios in each of Travel, Mortgage, and Software
DAPO-Math	Out-of-domain mathematics	Exact-match accuracy	150	Zero-shot transfer from MATH skills
BigCodeBench	Library-aware code generation	pass@1 from executable tests	150	Zero-shot transfer from HumanEval skills
PopQA	Open-domain factual QA	Alias-normalized exact match	150	Zero-shot transfer from HotpotQA skills

F.3 Evaluation Bookkeeping

Every evaluated item has a stable identifier and one terminal prediction. General mathematical tasks use the dataset answer normalizer followed by exact matching. Code tasks preserve the required entry point and are scored from executable tests; explanatory prose outside the requested code is not counted as a second candidate. HotpotQA and the mathematical OOD set use their respective exact-answer normalization, while PopQA accepts the documented gold aliases. BigCodeBench keeps public instructions separate from executable gold tests.

For specialist collaboration, the evaluator receives the scenario assertions, the final response, and the recorded tool actions. User-facing content and required system actions are both necessary for a successful scenario. Fold membership is fixed before skill learning, and results are appended only for the held-out portion. Token totals include main-agent planning, sub-agent messages, tool-related model interactions, and final synthesis when applicable. They exclude the offline configuration grid and genetic construction. This boundary is applied consistently to MAS-Skill, AFlow, and MaAS in the reported online-cost comparison.

G Representative Complete Skill Records

This section reproduces verbatim one complete `SKILL.md` record from each stored benchmark library. Dataset labels are added only for navigation; relative references remain part of the original skill packages. No user name, affiliation, email address, or machine-specific absolute path is included.

GSM8K: complete SKILL.md

```
---
name: maas-math-gsm8k-fused
description: "MATH type GSM8K; recommended topology collab.debate
    with 4 sub-agents; use when the problem matches this type and you
     want this collaboration layout. Tool profile calculator;
    optional math_local TOOL_CALL/CODE_EXEC per references/
    tool_protocol.md."
---

## Workflow

Use the fixed topology and role slots below. The main agent applies
    the synthesis rule after sub-agents finish.

## Topology and scale

- **MATH type**: `GSM8K`
- **topology**: `collab.debate`
- **sub_agent_count**: 4
- **historical_outcome** (from traces): `success`

## Role slots (model_preference per slot)

- **Agent_1**: Initial Calculation Expert -- preference `strong_logic
    `
- **Agent_2**: Verification Specialist -- preference `strong_logic`
- **Agent_3**: Alternative Approach Analyst -- preference `
    strong_logic`
- **Agent_4**: Final Consensus Evaluator -- preference `strong_logic`

## Synthesis rule for the main agent

Verify each sub-agent's arithmetic independently. Check that the word
     problem was interpreted correctly (quantities, relationships,
    units). Prefer the solution that shows all intermediate steps. If
     all agents agree, confirm with one verification step.

## Tools

- **tool_profile**: `calculator`

Use explicit arithmetic verification where applicable; align with the
     topology's division of labor.
## Notes

{"winner": {"accuracy": 0.9431818181818182, "mean_total_tokens":
    52040.85227272727, "topology": "collab.debate", "n": 4}, "
    candidates_evaluated": 88, "suffix": "maas_math_gsm8k_01"}

## Experience (from exploration / post-mortem)

- Always verify each sub-agent's arithmetic independently to catch
    potential errors early.
- Ensure that the word problem is interpreted correctly by confirming
     quantities, relationships, and units before proceeding.
- Show all intermediate steps in calculations to facilitate easier
    verification and understanding of the solution process.
- Use collaborative strategies, such as brainstorming or debating, to
     explore different approaches and validate solutions.
- Identify and define variables clearly before setting up equations
    to avoid confusion later in the problem-solving process.
```

GSM8K: complete SKILL.md (continued)

```
## Bundled references

- `references/cluster_manifest.json` -- machine-readable skill record
    for `topology_plan_from_skill`
- `references/examples.md` -- example questions from this cluster
- `references/tool_protocol.md` -- TOOL_CALL / CODE_EXEC protocol for
    `math_local`
```

MATH: complete SKILL.md

```
---
name: math-type-counting-and-probability-main-fused
description: "MATH type Counting & Probability; recommended topology
    collab.divide_and_conquer with 3 sub-agents; use when the problem
     matches this type and you want this collaboration layout. Tool
    profile math_local; optional math_local TOOL_CALL/CODE_EXEC per
    references/tool_protocol.md."
---

## Workflow

Use the fixed topology and role slots below. The main agent applies
    the synthesis rule after sub-agents finish.

## Topology and scale

- **MATH type**: `Counting & Probability`
- **topology**: `collab.divide_and_conquer`
- **sub_agent_count**: 3
- **historical_outcome** (from traces): `success`

## Role slots (model_preference per slot)

- **Agent_1**: Total Outcomes Calculator -- preference `
    high_creativity`
- **Agent_2**: Successful Outcomes Analyzer -- preference `
    strong_logic`
- **Agent_3**: Probability Combiner -- preference `general_purpose`

## Synthesis rule for the main agent

Calculate the total outcomes using agent_1. Then, agent_2 computes
    the successful outcomes based on the specific problem conditions.
     Finally, agent_3 combines the probabilities and formats the
    answer as a common fraction.

## Tools

- **tool_profile**: `math_local`
## Bundled references

- `references/cluster_manifest.json` -- machine-readable skill record
    for `topology_plan_from_skill`
- `references/examples.md` -- example questions from this cluster
- `references/tool_protocol.md` -- TOOL_CALL / CODE_EXEC protocol for
    `math_local`
```

MultiArith: complete SKILL.md

```
---
name: multiarith-arithmetic-ga-best
description: "MATH type arithmetic; recommended topology collab.
    brainstorm with 2 sub-agents; use when the problem matches this
    type and you want this collaboration layout. Tool profile
    calculator; optional math_local TOOL_CALL/CODE_EXEC per
    references/tool_protocol.md."
---

## Workflow

Use the fixed topology and role slots below. The main agent applies
    the synthesis rule after sub-agents finish.

## Topology and scale

- **MATH type**: `arithmetic`
- **topology**: `collab.brainstorm`
- **sub_agent_count**: 2
- **historical_outcome** (from traces): `success`

## Role slots (model_preference per slot)

- **Agent_1**: Calculate Remaining Students -- preference `
    high_creativity`
- **Agent_2**: Distribute Students into Groups -- preference `
    strong_logic`

## Synthesis rule for the main agent

Verify each sub-agent's arithmetic independently. Prefer the solution
     that shows all intermediate steps and verifies the answer. If
    all agree, confirm with one verification step.

## Tools

- **tool_profile**: `calculator`

Use explicit arithmetic verification where applicable; align with the
     topology's division of labor.

## Notes

{"winner": {"accuracy": 0.852112676056338, "mean_total_tokens":
    11371.38028169014, "topology": "collab.brainstorm", "n": 2}, "
    candidates_evaluated": 142, "suffix": "brainstorm2_pareto0"}

## Bundled references

- `references/cluster_manifest.json` -- machine-readable skill record
     for `topology_plan_from_skill`
- `references/examples.md` -- example questions from this cluster
- `references/tool_protocol.md` -- TOOL_CALL / CODE_EXEC protocol for
     `math_local`
```

MBPP: complete SKILL.md

```
---
name: maas-math-mbpp-fused
description: "MATH type MBPP; recommended topology collab.brainstorm
    with 2 sub-agents; use when the problem matches this type and you
     want this collaboration layout. Tool profile none; optional
    math_local TOOL_CALL/CODE_EXEC per references/tool_protocol.md."
---

## Workflow

Use the fixed topology and role slots below. The main agent applies
    the synthesis rule after sub-agents finish.

## Topology and scale

- **MATH type**: `MBPP`
- **topology**: `collab.brainstorm`
- **sub_agent_count**: 2
- **historical_outcome** (from traces): `success`

## Role slots (model_preference per slot)

- **Agent_1**: Code Generator -- preference `strong_logic`
- **Agent_2**: Code Generator -- preference `strong_logic`

## Synthesis rule for the main agent
Review each sub-agent's code for correctness. Verify the function
    signature matches the problem exactly. If all agents produce
    similar correct code, adopt the cleanest version. If agents
    disagree, construct a mental test case from the function name to
    determine which is correct.

## Tools

- **tool_profile**: `none`

Use explicit arithmetic verification where applicable; align with the
     topology's division of labor.

## Notes

{"winner": {"accuracy": 0.7727272727272727, "mean_total_tokens":
    9489.59090909091, "topology": "collab.brainstorm", "n": 2}, "
    candidates_evaluated": 22, "suffix": "brainstorm2_pareto1"}

## Experience (from exploration / post-mortem)

- Always verify that the function signature matches the problem
    requirements exactly before proceeding with further checks.
- Use a collaborative approach where multiple agents generate code
    and review each other's work for correctness.
- Adopt the cleanest version of the code when multiple agents produce
     similar correct solutions to enhance readability and
    maintainability.
- Construct mental test cases based on the function name to resolve
    disagreements between agents on code correctness.
- Implement a structured decomposition strategy by breaking down the
    problem into sub-tasks handled by specialized agents.

## Bundled references
```

MBPP: complete SKILL.md (continued)

```
- `references/cluster_manifest.json` -- machine-readable skill record
    for `topology_plan_from_skill`
- `references/examples.md` -- example questions from this cluster
- `references/tool_protocol.md` -- TOOL_CALL / CODE_EXEC protocol for
    `math_local`
```

HumanEval: complete SKILL.md

```
---
name: maas-math-humaneval-fused
description: "HumanEval code generation; recommended topology collab.
    brainstorm with 3 sub-agents; tool profile code_local with auto-
    verification."
---

## Workflow

Use the fixed topology and role slots below. The main agent applies
    the synthesis rule after sub-agents finish. Each sub-agent's code
     is auto-verified against test assertions; if tests fail, the sub
    -agent receives feedback and revises.

## Topology and scale

- **MATH type**: `HumanEval`
- **topology**: `collab.brainstorm`
- **sub_agent_count**: 3
- **historical_outcome** (from traces): `success`

## Role slots (model_preference per slot)

- **Agent_1**: Code Generator 1 -- preference `strong_logic`
- **Agent_2**: Code Generator 2 -- preference `strong_logic`
- **Agent_3**: Code Generator 3 -- preference `strong_logic`

## Synthesis rule for the main agent

Review each sub-agent's code for correctness. Verify the function
    signature matches the problem exactly. If all agents produce
    similar correct code, adopt the cleanest version. If agents
    disagree, trace through a test case to determine which is correct
    . Prefer code that has been auto-verified (passed test assertions
    ).

## Tools

- **tool_profile**: `code_local`
Auto-verification runs automatically: the system extracts your code
    and runs it against test assertions. If tests fail, you will
    receive feedback with error messages. Fix the code and output the
     corrected version in a fenced python block.

## Notes

{"winner": {"accuracy": 0.8484848484848485, "mean_total_tokens":
    12920.484848484848, "topology": "collab.brainstorm", "n": 3}, "
    candidates_evaluated": 33, "suffix": "brainstorm3_pareto0"}
```

HumanEval: complete SKILL.md (continued)

```
## Experience (from exploration / post-mortem)

- Always verify that the function signature matches the problem
    statement exactly to avoid discrepancies.
- Pay close attention to edge cases: negative numbers, zero, empty
    inputs, boundary values.
- For math operations, be careful with integer vs float division and
    rounding.
- When checking conditions like "cube of integer", use round(x**(1/3)
    )**3 == x for positive, round((-x)**(1/3))**3 == -x for negative.
- For counting problems, use inclusion-exclusion principle carefully:
     |A ∪B| = |A| + |B| - |A ∩B|.
- For string/list problems, handle empty inputs gracefully but don't
    add unnecessary guards.
- Test your logic mentally with the provided examples before
    outputting code.
- If auto-verification reports test failures, read the error messages
     carefully and fix the specific failing cases.

## Bundled references

- `references/cluster_manifest.json` -- machine-readable skill record
     for `topology_plan_from_skill`
- `references/examples.md` -- example questions from this cluster
```

HotpotQA: complete SKILL.md

```
---
name: maas-math-bridge-fused
description: "MATH type bridge; recommended topology collab.
    actor_critic with 2 sub-agents; use when the problem matches this
     type and you want this collaboration layout. Tool profile
    calculator; optional math_local TOOL_CALL/CODE_EXEC per
    references/tool_protocol.md."
---

## Workflow

Use the fixed topology and role slots below. The main agent applies
    the synthesis rule after sub-agents finish.

## Topology and scale

- **MATH type**: `bridge`
- **topology**: `collab.actor_critic`
- **sub_agent_count**: 2
- **historical_outcome** (from traces): `success`

## Role slots (model_preference per slot)

- **Agent_1**: Information Extractor -- preference `strong_logic`
- **Agent_2**: Context Verifier -- preference `strong_logic`

## Synthesis rule for the main agent

Verify each sub-agent used the correct context passages. Check that
    the reasoning chain is complete (all hops traced). Prefer the
```

HotpotQA: complete SKILL.md (continued)

```
    answer that is most concise and directly addresses the question.
    If agents agree, confirm by checking one key passage.

## Tools

- **tool_profile**: `calculator`

Use explicit arithmetic verification where applicable; align with the
     topology's division of labor.

## Notes

{"winner": {"accuracy": 1.0, "mean_total_tokens": 16965.0, "topology
    ": "collab.actor_critic", "n": 2}, "candidates_evaluated": 3, "
    suffix": "actor_critic2_pareto1"}

## Experience (from exploration / post-mortem)

- Ensure each sub-agent utilizes the correct context passages to
    maintain relevance.
- Verify that the reasoning chain is complete by tracing all hops in
    the answer.
- Prioritize answers that are concise and directly address the
    question to enhance clarity.
- If agents reach an agreement, confirm the validity by checking one
    key passage for accuracy.
- Implement a systematic error-checking method by reviewing each
    agent's output against the context.

## Bundled references

- `references/cluster_manifest.json` -- machine-readable skill record
     for `topology_plan_from_skill`
- `references/examples.md` -- example questions from this cluster
- `references/tool_protocol.md` -- TOOL_CALL / CODE_EXEC protocol for
     `math_local`
```

DAPO-Math transfer library: complete SKILL.md

```
---
name: maas-math-prealgebra-fused
description: "MATH type Prealgebra; recommended topology collab.
    debate with 4 sub-agents; use when the problem matches this type
    and you want this collaboration layout. Tool profile math_local;
    optional math_local TOOL_CALL/CODE_EXEC per references/
    tool_protocol.md."
---

## Workflow

Use the fixed topology and role slots below. The main agent applies
    the synthesis rule after sub-agents finish.

## Topology and scale

- **MATH type**: `Prealgebra`
- **topology**: `collab.debate`
- **sub_agent_count**: 4
```

DAPO-Math transfer library: complete SKILL.md (continued)

```
- **historical_outcome** (from traces): `success`

## Role slots (model_preference per slot)

- **Agent_1**: Angle Sum Validator -- preference `strong_logic`
- **Agent_2**: Equation Derivation Specialist -- preference `
    high_creativity`
- **Agent_3**: Intermediate Step Verifier -- preference `strong_logic
    `
- **Agent_4**: Final Answer Formatter -- preference `high_creativity`

## Synthesis rule for the main agent

Each agent presents their findings. Agent 1 checks the angle sums or
    equation constraints. Agent 2 derives equations step-by-step,
    ensuring logical flow. Agent 3 verifies intermediate steps and
    checks for common errors. Agent 4 formats the final answer in \
    boxed{...} notation, ensuring all steps are clear and concise.
    Verification includes plugging answers back into the original
    equations, checking units, and confirming that all constraints
    are satisfied.

## Tools

- **tool_profile**: `math_local`
## Experience (from exploration / post-mortem)

- Always verify that the sum of angles in polygons matches the known
    total for that shape, such as 540 degrees for a pentagon.
- Break down complex problems into smaller, manageable parts by
    identifying relationships between different elements, such as
    angles or equations.
- Use consistent formatting for final answers, such as boxed notation
    , to clearly present solutions.
- Check the validity of each calculated angle by ensuring it adheres
    to the properties of the geometric figure involved.
- Involve multiple roles or agents in the problem-solving process to
    cover different aspects of the solution and catch potential
    errors.
- Regularly review and critique intermediate steps to ensure logical
    consistency and accuracy before arriving at the final answer.
- Establish clear relationships between variables early in the
    problem-solving process to simplify calculations later on.
- Utilize verification methods, such as checking sums or conditions,
    to confirm the correctness of the solution.
- When dealing with equations, derive them systematically and ensure
    each step logically follows from the previous one.

## Bundled references

- `references/cluster_manifest.json` -- machine-readable skill record
     for `topology_plan_from_skill`
- `references/examples.md` -- example questions from this cluster
- `references/tool_protocol.md` -- TOOL_CALL / CODE_EXEC protocol for
     `math_local`
```

BigCodeBench transfer library: complete SKILL.md

```
---
name: maas-math-humaneval-fused
description: "HumanEval code generation; recommended topology collab.
    brainstorm with 3 sub-agents; tool profile code_local with auto-
    verification."
---

## Workflow

Use the fixed topology and role slots below. The main agent applies
    the synthesis rule after sub-agents finish. Each sub-agent's code
     is auto-verified against test assertions; if tests fail, the sub
    -agent receives feedback and revises.

## Topology and scale

- **MATH type**: `HumanEval`
- **topology**: `collab.brainstorm`
- **sub_agent_count**: 3
- **historical_outcome** (from traces): `success`

## Role slots (model_preference per slot)

- **Agent_1**: Code Generator 1 -- preference `strong_logic`
- **Agent_2**: Code Generator 2 -- preference `strong_logic`
- **Agent_3**: Code Generator 3 -- preference `strong_logic`

## Synthesis rule for the main agent

Review each sub-agent's code for correctness. Verify the function
    signature matches the problem exactly. If all agents produce
    similar correct code, adopt the cleanest version. If agents
    disagree, trace through a test case to determine which is correct
    . Prefer code that has been auto-verified (passed test assertions
    ).

## Tools

- **tool_profile**: `code_local`

Auto-verification runs automatically: the system extracts your code
    and runs it against test assertions. If tests fail, you will
    receive feedback with error messages. Fix the code and output the
     corrected version in a fenced python block.

## Notes

{"winner": {"accuracy": 0.8484848484848485, "mean_total_tokens":
    12920.484848484848, "topology": "collab.brainstorm", "n": 3}, "
    candidates_evaluated": 33, "suffix": "brainstorm3_pareto0"}

## Experience (from exploration / post-mortem)

- Always verify that the function signature matches the problem
    statement exactly to avoid discrepancies.
- Pay close attention to edge cases: negative numbers, zero, empty
    inputs, boundary values.
- For math operations, be careful with integer vs float division and
    rounding.
- When checking conditions like "cube of integer", use round(x**(1/3)
    )**3 == x for positive, round((-x)**(1/3))**3 == -x for negative.
```

BigCodeBench transfer library: complete SKILL.md (continued)

```
- For counting problems, use inclusion-exclusion principle carefully:
     |A ∪B| = |A| + |B| - |A ∩B|.
- For string/list problems, handle empty inputs gracefully but don't
    add unnecessary guards.
- Test your logic mentally with the provided examples before
    outputting code.
- If auto-verification reports test failures, read the error messages
     carefully and fix the specific failing cases.

## Bundled references

- `references/cluster_manifest.json` -- machine-readable skill record
     for `topology_plan_from_skill`
- `references/examples.md` -- example questions from this cluster
```

PopQA transfer library: complete SKILL.md

```
---
name: maas-math-bridge-fused
description: "MATH type bridge; recommended topology collab.
    actor_critic with 2 sub-agents; use when the problem matches this
     type and you want this collaboration layout. Tool profile
    calculator; optional math_local TOOL_CALL/CODE_EXEC per
    references/tool_protocol.md."
---

## Workflow

Use the fixed topology and role slots below. The main agent applies
    the synthesis rule after sub-agents finish.

## Topology and scale

- **MATH type**: `bridge`
- **topology**: `collab.actor_critic`
- **sub_agent_count**: 2
- **historical_outcome** (from traces): `success`

## Role slots (model_preference per slot)

- **Agent_1**: Information Extractor -- preference `strong_logic`
- **Agent_2**: Context Verifier -- preference `strong_logic`

## Synthesis rule for the main agent

Verify each sub-agent used the correct context passages. Check that
    the reasoning chain is complete (all hops traced). Prefer the
    answer that is most concise and directly addresses the question.
    If agents agree, confirm by checking one key passage.

## Tools

- **tool_profile**: `calculator`

Use explicit arithmetic verification where applicable; align with the
     topology's division of labor.

## Notes
```

PopQA transfer library: complete SKILL.md (continued)

```
{"winner": {"accuracy": 1.0, "mean_total_tokens": 16965.0, "topology
    ": "collab.actor_critic", "n": 2}, "candidates_evaluated": 3, "
    suffix": "actor_critic2_pareto1"}

## Experience (from exploration / post-mortem)

- Ensure each sub-agent utilizes the correct context passages to
    maintain relevance.
- Verify that the reasoning chain is complete by tracing all hops in
    the answer.
- Prioritize answers that are concise and directly address the
    question to enhance clarity.
- If agents reach an agreement, confirm the validity by checking one
    key passage for accuracy.
- Implement a systematic error-checking method by reviewing each
    agent's output against the context.

## Bundled references

- `references/cluster_manifest.json` -- machine-readable skill record
     for `topology_plan_from_skill`
- `references/examples.md` -- example questions from this cluster
- `references/tool_protocol.md` -- TOOL_CALL / CODE_EXEC protocol for
     `math_local`
```

H Extended Results and Analysis

H.1 Cross-Model Skill Transfer

Four Pareto-selected MBPP skills were learned from 86 GPT-4o-mini training items and 1,290 exploration runs. They are reused without additional extraction or fusion on all 341 MBPP evaluation items. Table 7 transcribes the values shown in the main-paper figure. The supporting GPT-4o-mini roles described above remain fixed.

Table 7: MBPP pass@1 (%) in cross-model transfer. Values are rounded to one decimal place as displayed in the main-paper figure.

Main reasoning backbone	Direct	MAS-Skill	Gain
Mistral-7B	11.7	43.4	+31.7
Llama-3.1-8B	24.9	53.1	+28.2
Qwen2.5-3B	28.1	55.1	+27.0
Qwen3-8B	49.9	66.6	+16.7

H.2 Cross-Dataset Transfer

Table 8 gives the exact values underlying the out-of-domain transfer figure. "Without skill" disables retrieval while preserving the GPT-4o-mini backbone and direct inference protocol. Each source library is frozen, and no target item is used for learning or calibration.

H.3 Component Ablation

The ablations preserve the backbone and evaluation protocol while disabling one component. "Without skill pool" removes reusable skill retrieval, "without GA fusion" retains extracted base skills

Table 8: Zero-shot cross-dataset transfer (%).

Target	Without skill	MAS-Skill	Gain
DAPO-Math	24.7	28.7	+4.0
BigCodeBench	46.3	49.3	+3.0
PopQA	23.3	25.3	+2.0

but removes evolved fused skills, and "without A/B testing" removes the matched configuration-comparison stage. The last variant therefore lacks the empirically selected seeds provided by controlled multi-parameter combination tests.

Table 9: Component ablation. Accuracy and pass@1 are percentages; cost values are those reported by the dedicated ablation run. Values within a dataset come from the same run.

	GSM8K		HumanEval	
Variant	Cost	Accuracy	Cost	pass@1
MAS-Skill	3.31	92.80	0.31	94.66
Without skill pool	3.52	91.94	0.47	91.60
Without GA fusion	4.52	92.13	0.53	92.37
Without A/B testing	4.54	92.42	0.55	93.89

Removing the complete skill pool produces the largest accuracy reduction: 0.86 percentage points on GSM8K and 3.06 points on HumanEval. Removing fusion reduces the scores by 0.67 and 2.29 points. Removing the controlled configuration comparison reduces them by 0.38 and 0.77 points. The results support complementary roles for empirically grounded base skills and trajectory-guided fusion.

H.4 Scope of Topology Preference Analysis

The main paper also reports a pairwise structural judgment between MAS-Skill and MaAS. A fixed rubric evaluates capability coverage, dependency structure, interaction fit, and parsimony. Across 90 comparisons, the judge prefers the MAS-Skill structure in 72.2% of cases, records a tie in 16.7%, and prefers MaAS in 11.1%. The preference for MAS-Skill is 73.3% on Travel, 46.7% on Mortgage, and 96.7% on Software.

This analysis is structural evidence only. A preferred topology can still fail because a specialist is bound incorrectly, a tool returns insufficient evidence, an intermediate answer is wrong, or synthesis omits an assertion. This distinction is especially visible on Software: its explicit code, testing, review, infrastructure, and deployment roles receive high structural scores, while end-to-end GSR does not exceed MaAS. In addition, per-item topology metadata was retained only for the first fold; later structures were reconstructed from the saved fold-level skill and task-type router. The single-judge result should therefore be read as exploratory support for the learned structures, not as a substitute for task success.