Evolving Agentic Workflow Driven by Human-Agent Collaboration

Yuxin Liu[1,2]* **Jinxuan Zhang**[2]* **Yuezhang Peng**[1] **Hefeng Zhou**[1,2]
Xiangfeng Wang[3] **Jiong Lou**[1] **Chentao Wu**[1] **Jie Li**[1]† **Jingjing Qu**[2]† **Chaochao Lu**[2]
[1]Shanghai Jiao Tong University
[2]Shanghai Artificial Intelligence Laboratory [3]East China Normal University
aph.xin@gmail.com jlby1770445165@gmail.com

Abstract

Agentic workflows, composed of multiple collaborating Large Language Models (LLMs), have become a key paradigm for complex problem-solving. However, their effectiveness is often hindered by three critical challenges: high manual design costs, inefficient agentic search, and poor dynamic adaptability to new tasks and human preferences. To address these limitations, we propose HFlow, an evolutionary framework for generating agentic workflows through human-agent collaboration. HFlow employs an evolutionary algorithm to automate the search for optimal workflows by mutating and crossing over their structures, prompts, and LLM backbones. This process is guided by human preferences to ensure rapid convergence, while a hierarchical experience memory enables the generalization of learned strategies. Extensive experiments on math and code generation benchmarks show HFlow surpasses other automated baselines by up to 27.34%, while achieving comparable performance to o1-preview at only one-fourth of the cost. Our work introduces a new paradigm for workflow design that produces cost-effective and adaptive solutions, better aligning automated agentic systems with dynamic human needs.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable potential in addressing tasks across diverse domains like code generation and mathematical reasoning (Shinn et al., 2023; Romera-Paredes et al., 2024). Early forms of these LLM-driven agentic systems operate via single-agent paradigms, which executed predefined tasks in a sequential, manually-defined manner (Wei et al., 2022; Fu et al., 2022; Wang et al., 2022; Yao et al., 2023). More recently, the field has shifted towards a structured workflow paradigm

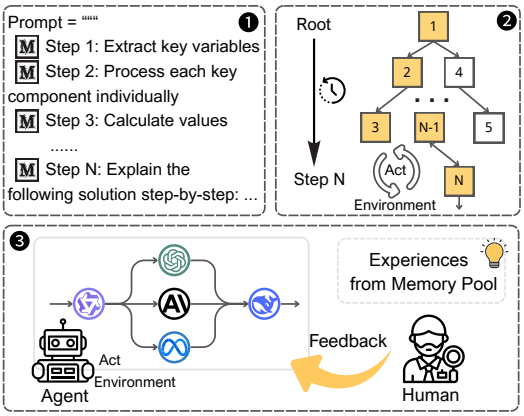

Figure 1: Comparison of current methods: ❶ Manually-defined workflows, ❷ Automated workflows, and ❸ our proposed HFlow by human-agent collaboration.

composed of multiple agents, showcasing impressive collaborative intelligence (Wang et al., 2023; Khan et al., 2024; Jiang et al., 2023; Liu et al., 2023; Chen et al., 2023b; Qian et al., 2024). These agentic workflows, guided by human requirements, employ automated search to decompose complex, reasoning-intensive tasks into sub-tasks assigned to different agents (Chen et al., 2023a; Zhuge et al., 2024; Hu et al., 2024; Shang et al., 2024; Zhang et al., 2024). This evolution underscores the importance of agentic design, highlighting the need for systems that can fully comprehend task requirements and effectively automate exploration.

Despite these advancements, the design of agentic workflows faces three critical challenges, shown in Fig. 1. First, high manual design cost: To cater to varied human needs, agentic systems still require manual exploration of their prompts and execution sequences. This significant human intervention hinders their generalizability and scalability. Second, inefficient agentic search: Existing automated search methods primarily rely on heuristic approaches like Monte Carlo tree search (Zhang et al., 2024; Li et al., 2025). These methods require prolonged, iterative feedback collection and evaluation, resulting in high LLM API costs and

*Equal contribution.
†Corresponding authors are Jie Li and Jingjing Qu.

Findings of the Association for Computational Linguistics: ACL 2026, pages 24960–24969

lengthy search cycles. Third, poor dynamic adaptability: Current workflow designs are largely unidirectional. An agentic system generates a fixed execution plan based on a complex task goal provided by a human. However, it cannot dynamically adjust or comprehend human implicit knowledge and preferences when encountering new environments, thus failing to efficiently leverage historical experiences and requiring re-initialization.

These considerations raise a critical question for the current paradigm of agentic system design: *How can we automatically optimize a set of agentic workflows that are efficient, cost-effective, and adaptive to human needs, thereby providing diverse solutions for varied queries?*

To this end, we propose HFlow, a framework that addresses these challenges through three core designs: 1) We employ an evolutionary algorithm to expand the agentic search space, automating the generation of diverse workflows through crossover and mutation of LLMs, structures, and prompts. 2) We construct an efficient human-agent collaboration framework where human preferences guide the iterative generation of workflows, enabling a cost-effective solution with rapid convergence. 3) We introduce a hierarchical memory across various modules that enables adaptation to new tasks by leveraging accumulated experiences from diverse problem-solving strategies.

We validate our framework on four widely-used benchmarks spanning domains. Experimental results demonstrate that our method outperforms other automated baselines by 4.63%∼27.34%, while achieving comparable performance to o1-preview model at only one-fourth of the cost. Furthermore, our approach exhibits a steeper search trajectory, discovering high-performance agentic designs in fewer iterations. We also investigate the impact of different forms of human feedback on the evolutionary generation of workflows and prove the beneficial role of this collaborative approach. Ablation validates hierarchical experience memory.

Our contributions are summarized as follows:

- **New Paradigm.** We introduce a new paradigm for agentic workflow design, an evolutionary process driven by human-agent collaboration. This treats human preferences as key guiding signals, explicitly framing workflow generation as an optimization problem aimed at efficiently and cost-effectively satisfying dynamic human needs.

- **Practical Framework.** We propose HFlow, a novel framework that automates workflow generation. HFlow utilizes an evolutionary algorithm guided by human feedback and a hierarchical experience memory to autonomously evolve diverse agentic workflows.

- **Comprehensive Evaluation.** We empirically demonstrate that HFlow is: (I) Effective, outperforming baselines by up to 27.34%; (II) Economical, achieving performance comparable to the expensive o1-preview model at a fraction of the cost; and (III) Adaptive, capable of evolving workflows to complex structures based on the guidance.

2 Related Work

2.1 Agentic Workflow Methods

Foundational work in agentic systems includes manually-defined workflows (Wu et al., 2024; Li et al., 2023; Hong et al., 2023), established paradigms for complex problem-solving through structured agent collaboration. These frameworks rely on manual design of agent roles, communication protocols, and workflow logic. Researchers then seek to partially automate agent design (Khattab et al., 2023; Yuan et al., 2024; Zhuge et al., 2024). More recent efforts have shifted towards end-to-end automated workflows. These systems treat agent design as a search problem over a modular design space (Zhang et al., 2024; Hu et al., 2024; Shang et al., 2024), enabling the automated discovery of entire configurations. However, the common limitations are the failure to account for dynamic preferences in real-world scenarios, leading to costly and time-consuming iterations.

2.2 Learning from Accumulated Experience

An agentic system enables to learn from past experiences. Rather than solving each new problem from scratch, this paradigm involves creating and maintaining a persistent knowledge base of effective solutions. These methods build a repository of successful components (Wang et al., 2024; Romera-Paredes et al., 2024; Novikov et al., 2025; Li et al., 2025). These methods typically treat the experience memory as a task-specific repository. They lack a hierarchical management to isolate different types of knowledge, thereby limiting the transferability of experience across diverse contexts.

2.3 Human-Agent Collaboration

Human-Agent Collaboration (HAC) leverages the synergy between users and agents for complex problem-solving. A key aspect of HAC is the human's role in providing personalized guidance (Wang et al., 2021). Rather than simply providing a final answer, the human guides the agent's reasoning process by expressing preferences, such as a desire for workflow simplicity over complexity, and correcting reasoning steps to better align with their unique intent. To overcome high costs and scalability limitations of direct human involvement, a strategy involves using powerful oracle models to act as proxies for human feedback (Caetano et al., 2025; Feng et al., 2024; Don-Yehiya et al., 2025).

3 Preliminary

3.1 Problem Formulation

Agentic Workflow. Following AFlow (Zhang et al., 2024), we define an agentic workflow W as a directed acyclic graph, $W = (\mathcal{N}, \mathcal{E})$, where nodes $\mathcal{N} = \{N_1, N_2, \ldots, N_n\}$ represent concrete operations and edges $\mathcal{E}$ define the data flow. Each node N_i is an operator, which is a specific, executable implementation characterized by the LLM M it invokes, the input prompt P provided to the model, and its parameters θ (e.g., temperature). Thus, an operator node can be represented as $N_i = (M_i, P_i, \theta_i)$. The structure of the workflow, defined by the edges $\mathcal{E}$, shows the graph topology represented in Fig. 2.

Human-Agent Collaborative Evolution. Traditional automated workflow optimization seeks a single optimal workflow W^* that maximizes a static evaluation function G for a given task T: $W^* = \arg\max_{W \in \mathcal{S}} G(W, T)$, where $\mathcal{S}$ is the entire search space of possible workflows. This formulation is insufficient for complex problems where human preferences, performance, and cost trade-offs are critical.

We reformulate the problem as a workflow evolution process guided by human-agent collaboration. The objective is not to find a single W^*, but to evolve a population of diverse and effective workflows $\mathcal{W}^*$, that continuously adapts to human feedback. Given a set of tasks T and a human feedback mechanism $\mathcal{H}$, the goal is to find the workflow that maximizes a unified objective function which

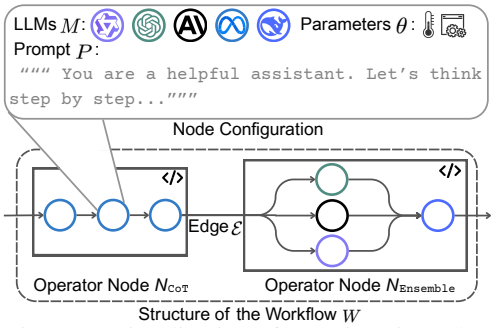

Figure 2: Visualization of Notations in HFlow.

incorporates both utility and cost, formulated as:

$$\begin{aligned} \mathcal{W}^* &= \arg\max_{\mathcal{W} \in \mathcal{S}_{\text{HFlow}}} \left(u(\mathcal{W}, \mathcal{H}, T) - \lambda \cdot c(\mathcal{W}, T)\right) \\ &= \arg\max_{\mathcal{W} \in \mathcal{S}_{\text{HFlow}}} \left(p(\mathcal{W}, T) + h(\mathcal{W}, T) - \lambda \cdot c(\mathcal{W}, T)\right) \end{aligned} \quad (1)$$

where $u(\cdot)$ is a composite utility function that measures a workflow's overall quality by combining its performance $p(\cdot)$ with preferences $h(\cdot)$, $c(\cdot)$ evaluates the system LLM API cost, and λ is a balancing coefficient that controls the trade-off between maximizing utility and minimizing cost.

3.2 HFlow Search Space

To manage the immense complexity, we define the workflow search space $\mathcal{S}$. A complete agentic workflow W is a combination of its structure and the set of its operator nodes. The full search space $\mathcal{S}_{\text{HFlow}}$ is thus the set of all possible workflows, composed of structure space and operator space:

$$\begin{aligned} \mathcal{S}_{\text{HFlow}} &= \{(\mathcal{N}, \mathcal{E}) \mid \mathcal{N} \in \mathcal{S}_{\text{op}}, \mathcal{E} \in \mathcal{S}_{\text{struct}}\} \\ &= \{(\mathcal{N}, \mathcal{E}) \mid N_i = (M_i, P_i, \theta_i), M_i \in \mathcal{M}, \\ &\qquad P_i \in \mathcal{P}, \theta_i \in \Theta, \mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}\}. \end{aligned} \quad (2)$$

$\mathcal{S}_{\text{struct}}$ is the space of all valid workflow topologies, defined by the set of edges $\mathcal{E}$. Evolution in this space optimizes the high-level strategy and data flow. Concurrently, the operator space $\mathcal{S}_{\text{op}}$ is the space of all possible operator configurations, where each operator N_i is a tuple $(\mathcal{M}_i, \mathcal{P}_i, \theta_i)$, and evolution in this space optimizes the low-level tactics by finding the best LLM, prompt, and parameters.

The overall evolutionary process is driven by HFlow framework, which explores the full search space to find a set of optimal workflows that maximizes the objective:

$$\mathcal{W}^* = \text{HFlow}(\mathcal{S}_{\text{HFlow}}, \mathcal{H}, T) \quad (3)$$

4 HFlow Framework

4.1 HFlow Overview

HFlow begins by creating an initial population of workflows among the operator and structure search

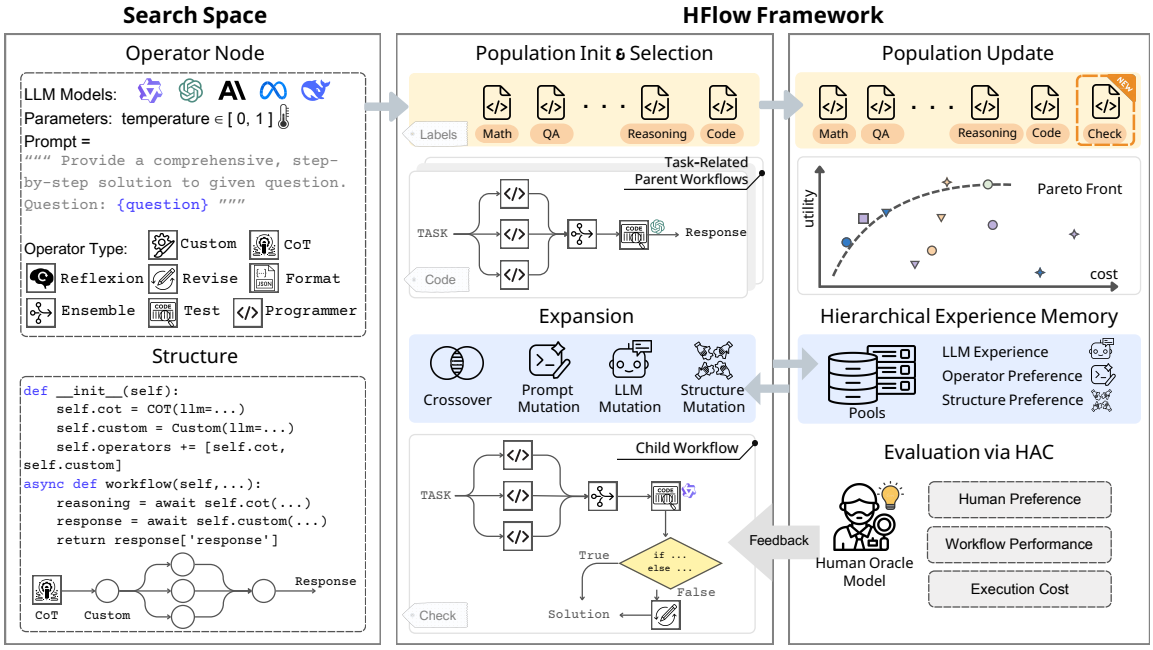

Figure 3: Overview of HFlow framework.

spaces. As new tasks are introduced, the system selects the most relevant parent workflows based on a similarity score of the profiles. These parents then undergo genetic operations, crossover & mutation, to generate a new child workflow. The performance of this child is evaluated through a human-agent collaboration feedback mechanism, which provides a rich signal encompassing performance, cost, and qualitative insights. Finally, the system updates its workflow populations and hierarchical memory modules based on this feedback, ensuring that the population continuously improves over time, as Fig. 3 and Algorithm 1 illustrate.

4.2 Population Initialization

The evolutionary process commences by seeding the structure pool $\mathcal{P}_{\texttt{struct}}$ and the operator pool $\mathcal{P}_{\texttt{op}}$. The operator pool is populated with a diverse set of fundamental implementations for common operator types, such as `CoT` (Wei et al., 2022), `Reflexion` (Wang et al., 2022),`Ensemble` (Jiang et al., 2023), `Test` (Zhong et al., 2024), and `Custom`, etc, each with different models, prompts, and parameters.

$$\mathcal{P}_{\texttt{op}}^{(0)} = \{N_{\texttt{CoT}}, N_{\texttt{Reflexion}}, \ldots, N_{\texttt{Custom}}\}. \tag{4}$$

Notably, the operator pool is denoted as a finite set. During the evolutionary process, $\mathcal{P}_{\texttt{op}}^{(0)}$ is dynamically expanded with new operators. Moreover, each operator N can be individually configured with customized node settings.

The structure pool is initialized with a set of basic workflow topologies representing common patterns like linear sequences, generate-and-review loops. To facilitate adaptation to relevant domains, we assign each structure a set of profile labels, indicating the likely areas of application where a structure excels. This can be formalized as: $F_{\texttt{profile}}(\mathcal{E}_k) = \{\tau_{k1}, \tau_{k2}, \ldots, \tau_{ki}\}$. Here, τ_{ki} represents the i-th profile label for structure $\mathcal{E}_k$, and $F_{\texttt{profile}}$ is an LLM-based classifier function that generates labels based on the structure's functionality. Formally, we initialize the structure pool:

$$\mathcal{P}_{\texttt{struct}}^{(0)} = \{\mathcal{E}_{\tau_1}, \mathcal{E}_{\tau_2}, \ldots, \mathcal{E}_{\tau_N}\}, \tag{5}$$

The initial population $\mathbf{P}^{(0)}$ is formed by a set of base structure topologies with concrete operator configurations drawn from the operator pool:

$$\mathbf{P}^{(0)} = \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_N\}, \tag{6}$$

where N is the population size. This initial population $\mathbf{P}^{(0)}$ serves as the starting point.

4.3 Retrieval-Based Selection

For query q_t, HFlow selects a subset of K parent workflows from current $\mathbf{P}^{(t)}$ that are most relevant to the task. This selection is performed via label-based retrieval, where relevance score $S(\mathcal{W}_i|q_t)$ is computed based on the cosine similarity between query and workflow's descriptive profile labels.

$$S(\mathcal{W}_i|q_t) = \frac{V(\tau_i) \cdot V(q_t)}{\|V(\tau_i)\|\|V(q_t)\|} \tag{7}$$

where $V(\cdot)$ is an embedding function that maps text to a vector representation. The top-K workflows are chosen as parents for the next stage. This targeted selection strategy ensures that children are generated from individuals already specialized for similar tasks, improving search efficiency.

4.4 Crossover & Mutation Expansion

Expansion aims to generate novel children through crossover and mutation applied to the selected parents based on existing experiences. This will modify or create both the high-level structure and the low-level operator.

Crossover. This operation generates a child workflow by combining the characteristics of multiple parent workflows. After identifying the K most suitable parents from the population $\{\mathcal{W}_1, \ldots, \mathcal{W}_K\}$, they are passed to the crossover function to generate a child: $F_{\texttt{crossover}}(\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_K) = \mathcal{W}_{\text{child}}$. Here, $\mathcal{W}_{\text{child}}$ represents the newly generated workflow. The crossover function, $F_{\texttt{crossover}}$, is facilitated by an LLM designed to merge the structural and operational traits of the parents. To further enhance population diversity, this child then undergoes a series of mutation operations.

Structure Mutation. Guided by structural preferences from the memory pool, $\pi_s(\cdot)$ alters the workflow topology $\mathcal{E}$ by adding or removing nodes or edges. This allows for the targeted evolution of entirely new strategic approaches.

Prompt Mutation. The prompt $\mathcal{P}$ of an operator is refined by $\pi_p(\cdot)$. Drawing on successful prompt formulations in the experience pool, this can range from simple modifications to sophisticated rewriting of the prompt.

LLMs Mutation. The LLM models $\mathcal{M}$ of an operator is modified by $\pi_l(\cdot)$, guided by the model performance and cost in the experience pool. This involves swapping the current LLM for another, including leading closed-source and open-source models. This workflow is then denoted as $\mathcal{W}'_{\text{child}}$.

4.5 Evaluation via Human-Agent Collaboration

New children introduction presents the challenge of designing an efficient mechanism that evolves the population towards higher diversity, preference, and performance. While prior works often struggle with escalating costs and complexity with low efficiency (Zhang et al., 2024; Li et al., 2025; Shang et al., 2024; Chen et al., 2023a), we integrate an evaluation method centered on human-agent collaboration. This method is engineered to leverage human feedback as a guiding signal for the iterative evolution of workflows, achieving rapid convergence towards the objective.

Algorithm 1: Evolving Agentic Workflow by HAC

Input: Initial pools $\mathcal{P}^{(0)}_{\texttt{op}}$, $\mathcal{P}^{(0)}_{\texttt{struct}}$, $\mathcal{T}_{\text{train}}$, N, K
Output: Well-evolved workflow population $\mathbf{P}_{\text{final}}$

$\mathbf{P}^{(0)} = \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_N\} \leftarrow \{\mathcal{P}^{(0)}_{\texttt{struct}}, \mathcal{P}^{(0)}_{\texttt{op}}, N\}$
Initialize memory: $\{\mathcal{M}^{(0)}_{\texttt{struct}}, \mathcal{M}^{(0)}_{\texttt{op}}, \mathcal{M}^{(0)}_{\texttt{LLM}}\}$
for each task q_t in $\mathcal{T}_{\text{train}}$ **do**
  $\{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_K\} \leftarrow \arg\max_{\substack{\mathbf{P}' \subseteq \mathbf{P}^{(t)} \\ |\mathbf{P}'|=K}} \sum_{\mathcal{W} \in \mathbf{P}'} S(\mathcal{W}|q_t)$ by Eq. 7
  $\mathcal{W}_{\text{child}} \leftarrow F_{\texttt{crossover}}(\{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_K\})$
  $\mathcal{W}'_{\text{child}} \leftarrow \pi_s(\cdot) \cup \pi_p(\cdot) \cup \pi_l(\cdot)$
  $\mathbf{P}_{\text{candidate}} \leftarrow \mathbf{P}^{(t)} \cup \{\mathcal{W}'_{\text{child}}\}$
  for each $\mathcal{W}_i$ in $\mathbf{P}_{\text{candidate}}$ **do**
    Execute $\mathcal{W}_i$ on query q_t
    Obtain human feedback:
    $\{H_{\text{struct}}, H_{\text{op}}, H_{\text{LLM}}\} \leftarrow F_{\text{human}}(\mathcal{W}_i, q_t)$
    Update cumulative metrics for $\mathcal{W}_i$:
    $\bar{p}_t(\mathcal{W}_i) \leftarrow (\bar{p}_{t-1}(\mathcal{W}_i)(n_t - 1) + p_t(\mathcal{W}_i, q_t))/n_t$
    $\bar{h}_t(\mathcal{W}_i) \leftarrow (\bar{h}_{t-1}(\mathcal{W}_i)(n_t - 1) + h_t(\mathcal{W}_i, q_t))/n_t$
    $\bar{c}_t(\mathcal{W}_i) \leftarrow (\bar{c}_{t-1}(\mathcal{W}_i)(n_t - 1) + c_t(\mathcal{W}_i, q_t))/n_t$
  end for
  if $|\mathbf{P}_{\text{candidate}}| > N$ **then**
    $\mathcal{W}_{\text{worst}} \leftarrow \arg\max_{\mathcal{W} \in \mathbf{P}_{\text{candidate}}} F(\mathcal{W})$ by Eq. 9
    $\mathbf{P}^{(t+1)} \leftarrow \mathbf{P}_{\text{candidate}} \setminus \{\mathcal{W}_{\text{worst}}\}$
  else
    $\mathbf{P}^{(t+1)} \leftarrow \mathbf{P}_{\text{candidate}}$
  end if
  Update memory: $\{\mathcal{M}^{(t+1)}_{\texttt{struct}}, \mathcal{M}^{(t+1)}_{\texttt{op}}, \mathcal{M}^{(t+1)}_{\texttt{LLM}}\}$
end for
return $\mathbf{P}^{(t+1)}$

Multi-Objective Metrics. To assess the quality of a workflow $\mathcal{W}$, we apply three core metrics. The first is the performance score, denoted as $p(\cdot)$, which provides a success measure of a verifiable task. This is complemented by the human preference score $h(\cdot)$ as well as various types of feedback $f(\cdot)$, formally defined by the function $F_{\text{human}}(\mathcal{W}, q_t)$. Finally, the cost, $c(\cdot)$, accounts for resource consumption (e.g., API usage).

To ensure a stable assessment, we maintain cumulative metrics over time. For $\mathcal{W}_i$ on query q_t, its cumulative records are updated as follows:

$$\bar{p}_t(\mathcal{W}_i) = (\bar{p}_{t-1}(\mathcal{W}_i) \cdot (n_t - 1) + p(\mathcal{W}_i, q_t))/n_t, \quad (8)$$

where n_t is the total number of times workflow $\mathcal{W}_i$ has been evaluated at iteration t. The term $\bar{p}_{t-1}$ represents the cumulative performance scores from

the previous iteration. The updates for $\bar{h}_t(\mathcal{W})$ and $\bar{c}_t(\mathcal{W})$ follow the same formulation.

Evolution Mechanism. To maintain the population size, we employ a fitness assignment scheme based on Pareto dominance of the multi-objective.

Each workflow $\mathcal{W}$ is represented by its objective vector $\big(\bar{p}(\mathcal{W}), \bar{h}(\mathcal{W}), \bar{c}(\mathcal{W})\big)$. A workflow $\mathcal{W}_1$ is said to dominate $\mathcal{W}_2$ if it is no worse than $\mathcal{W}_2$ on any objective and strictly better on at least one. The fitness $F(\mathcal{W})$ used for survival selection is then calculated based on these relationships:

$$F(\mathcal{W}) = \sum_{\mathcal{W} \in \mathcal{P}^{(t)} \cup \mathcal{W}'_{\text{child}}} \exp\left(\frac{I(\mathcal{W}', \mathcal{W})}{\varphi \cdot I_{\max}}\right), \quad (9)$$

where $\mathcal{P}^{(t)}$ is the set of competing workflows. The term $I(\mathcal{W}', \mathcal{W})$ is a dominance indicator function that returns a positive value if $\mathcal{W}'$ dominates $\mathcal{W}$, a negative value if $\mathcal{W}$ dominates $\mathcal{W}'$, and zero otherwise. $I_{\max}$ is the maximum absolute indicator value, and φ is a scaling factor.

A smaller fitness value indicates a superior individual that dominates its peers. Consequently, the workflow with the largest fitness value is eliminated. We prioritize Pareto optimization over weighted sums as agentic workflows inherently involve conflicting objectives, such as performance versus cost. While weighted sums require manual tuning of hyperparameters that generalize poorly across varying task difficulties, Pareto optimization preserves a Pareto front of diverse optimal trade-offs. This mechanism prevents premature convergence to local optima and maintains a robust gene pool for subsequent crossover and mutation, ensuring the population explores a wide range of cost-effective solutions.

4.6 Hierarchical Experience Memory

To enable long-term learning, HFlow incorporates a hierarchical experience memory that acts as an active retrieval tool to guide workflow evolution across three stages: (1) Initialization, where profile similarity is used to retrieve Top-K successful structures as parents; (2) Mutation, where memory directs the selection of successful templates and optimal LLM backbones; and (3) Population Update, where performance metrics and human preferences are written back to refine future retrievals.

Structure Preference Memory. $\mathcal{M}_{\texttt{struct}}$ tracks historical performance of architectures by storing triplets $(\mathcal{W}_i, q_j, H_{\text{struct}})$, where H_{struct} represents human feedback, including quantitative preference scores and qualitative analysis. During initialization and structure mutation, the system retrieves high-performing templates from this pool to steer the search toward proven architectural patterns.

Prompt Preference Memory. $\mathcal{M}_{\text{op}}$ records the triplets $(\mathcal{W}_i, q_j, H_{\text{op}})$, where H_{op} evaluates the contribution of specific prompts to task success and human feedback for the individual operators. During the expansion phase, this memory provides successful prompt variants, allowing the system to reuse this logic across similar task types.

LLMs Experience Memory. $\mathcal{M}_{\texttt{LLM}}$ monitors the performance of specific LLMs within different operator contexts, storing correctness and cost metrics H_{LLM}. In the mutation stage, the system queries this memory to intelligently select the most cost-effective backbone.

5 Experiments

5.1 Experiment Setup

Baselines. We compared manually-defined and automated workflows. The former include **Single-Agent Methods** like direct Vanilla, Chain-of-Thought (Wei et al., 2022), ComplexCoT (Fu et al., 2022), Self-Consistency (Wang et al., 2022), and **Multi-Agent Methods** like LLM-Debate (Du et al., 2023), LLM-Blender (Jiang et al., 2023), DyLAN (Liu et al., 2023), AgentVerse (Chen et al., 2023b), and MacNet (Qian et al., 2024). **Automated workflows** include AutoAgents (Chen et al., 2023a), GPTSwarm (Zhuge et al., 2024), ADAS (Hu et al., 2024), AgentSquare (Shang et al., 2024), AFlow (Zhang et al., 2024), and MASS (Zhou et al., 2025).

Datasets. Our experiments investigate the mathematical, coding, and reasoning capabilities. For math, we utilize GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), with performance evaluated by solve rate. For code generation, we employ HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), measured by the pass@1 metric. For the embodied AI domain, we utilize ALFWorld (Shridhar et al., 2020), evaluated based on its success rate. All datasets were partitioned into train and test sets at a 1:4 ratio. Following the setup in the work (Hong et al., 2024), we retained only the Level 5 difficulty problems from the MATH dataset.

Category	Method	HumanEval	MBPP	ALFWorld	GSM8K	MATH	Avg.
	Vanilla (Direct Inference)	87.08	71.83	39.89	87.45	46.29	66.50
Single Agent	CoT (Wei et al., 2022)	88.13	71.83	39.92	87.10	46.40	66.68
	ComplexCoT (Fu et al., 2022)	87.49	72.36	41.68	86.89	46.53	66.99
	Self-Consistency (Wang et al., 2022)	88.60	73.60	40.55	87.57	47.91	67.65
Multi Agent	MultiPersona (Wang et al., 2023)	88.32	73.19	39.10	87.50	45.43	66.71
	LLM-Debate (Khan et al., 2024)	88.68	70.29	44.68	89.47	48.54	68.33
	LLM-Blender (Jiang et al., 2023)	88.80	77.05	43.79	88.35	48.92	69.38
	DyLAN (Liu et al., 2023)	90.42	77.30	53.32	89.98	48.63	71.93
	AgentVerse (Chen et al., 2023b)	89.29	74.28	45.03	89.91	47.35	68.77
	MacNet (Qian et al., 2024)	84.57	65.28	43.66	87.95	45.18	65.33
Automated Workflows	AutoAgents (Chen et al., 2023a)	87.64	71.95	46.15	87.69	45.32	67.75
	GPTSwarm (Zhuge et al., 2024)	89.32	77.43	53.19	89.14	47.88	71.39
	ADAS (Hu et al., 2024)	84.19	68.13	47.66	86.12	43.18	65.86
	AgentSquare (Shang et al., 2024)	89.08	78.46	66.42	87.62	48.51	74.02
	AFlow (Zhang et al., 2024)	90.93	81.67	59.16	91.16	51.28	74.84
	MASS (Zhou et al., 2025)	91.67	**86.50**	-	-	-	-
	HFlow (Ours)	**91.71**$_{+4.63}$	84.02$_{+12.19}$	**67.23**$_{+27.34}$	**94.03**$_{+6.58}$	**57.20**$_{+10.91}$	**78.84**$_{+12.34}$

Table 1: Performance of all methods across five datasets. Only MASS methods utilize the Gemini-1.5 Pro series for evaluation. To maintain consistency, GPT-4o-mini serves as the foundational model of HFlow and the other baseline methods. The best results are shown in **Bold** and the runner-ups are in underline.

Implementation Details. Our experiments utilize Claude-3.5-Sonnet, GPT-4o-mini, DeepSeek-V2.5, Qwen-2.5-72B, and Llama-3.1-70B for node configuration, workflow generation, and answer synthesis. To balance cost and precision, we employ a human oracle powered by GPT-4o to simulate collaboration. This oracle integrates insights from three human experts and an automated agent, the latter being continuously calibrated against expert judgments to ensure fidelity. For evaluation, workflows are first evolved on the training set. During testing, the optimal workflow is dynamically selected for each task based on label embedding similarity retrieval. Reported metrics represent the average performance generated by the optimally selected workflow for each test task after respective retrieval-based selection.

5.2 Overall Performance

HFlow achieves superior results across all benchmarks, demonstrating a 12.34% improvement in average accuracy. In Table 1, we present a comparative analysis of HFlow against 16 baselines. For math datasets, HFlow exhibits significant performance gains compared to Vanilla, achieving a 6.58% and 10.91% increase, respectively. HFlow's notable advantage in challenging MATH underscores its exceptional ability to tackle complex problems. On HumanEval datasets, HFlow's improvement over other baselines is less significant; a potential reason is that the base models already exhibit strong performance, thereby narrowing the scope for further enhancement. However, it is worth noting that while the latest MASS method is not open-source, our comparison with its reported results reveals that HFlow outperforms MASS (driven by Gemini-1.5-pro) on HumanEval, despite HFlow utilizing a much lighter backbone (GPT-4o-mini). In contrast, on the more challenging embodied task ALFWorld, HFlow achieves a remarkable 67.23% success rate, surpassing the Vanilla baseline by a substantial 27.34%. This significant margin demonstrates that for high-complexity, multi-step tasks, HFlow's automatic feedback mechanism is far more effective than single-agent systems or static workflows.

5.3 Cost Analysis

HFlow achieves leading performance while maintaining superior cost-efficiency. As illustrated in Table 2, HFlow outperforms DyLAN and AFlow, achieving results comparable to o1-preview at one-fourth of the cost. While HFlow incurs higher costs than vanilla baseline, this is a strategic trade-off justified by three key factors: (1) Performance Gains: The increased cost translates directly into capability, yielding up to a 27.34% improvement and solving complex reasoning problems where a single agent fails. (2) Relative Efficiency: Benchmarked against the closed models like o1-preview, HFlow proves highly economical. (3) One-Time Cost: Our evolutionary search is a one-time investment; once optimal workflow is discovered and stored in hierarchical memory, it is reused for subsequent inference, amortizing the cost over long-term deployment. Furthermore, this efficiency is bolstered by our human-agent collaboration and dynamic LLM switching, which

Method	MATH		MBPP	
	Cost ($)	Acc	Cost ($)	Acc
o1-preview	7.84	70.20	3.21	89.65
$\text{Vanilla}_{(\text{Qwen})}$	0.03	63.80	0.01	69.76
$\text{Vanilla}_{(\text{Llama})}$	0.02	31.93	0.01	65.11
$\text{DyLAN}_{(\text{Qwen})}$	16.86	64.17	8.79	75.63
$\text{DyLAN}_{(\text{Llama})}$	11.99	38.19	6.78	69.92
$\text{AFlow}_{(\text{Qwen})}$	3.95	66.38	1.60	80.84
$\text{AFlow}_{(\text{Llama})}$	1.96	36.97	0.74	67.42
HFlow	2.11	68.32	0.67	84.75

Table 2: Cost and accuracy comparison. Qwen and Llama refer to Qwen2.5-72B and Llama3.1-70B. HFlow automatically selects backbones, with costs representing LLM API expenditures across training and inference.

Method	MATH	MBPP
w/o feedback	48.51	74.27
w/ random feedback	45.23	76.50
w/ wrong feedback	42.17	70.21
w/ score	50.42	79.59
w/ score + summary	52.67	83.14
w/ score + reason	54.62	83.63
w/ multiscore + summary	55.44	83.45
w/ multiscore + reason	**57.20**	**84.02**

Table 3: Ablation study of different feedback settings on MATH and MBPP. The best results are shown in **Bold**.

minimize redundant iterations and optimize model selection based on task complexity.

5.4 Evolution Efficiency

HFlow demonstrates the most rapid increase on the MATH and ultimately achieves the best performance. In Figure 4, we compare the performance of AFlow and HFlow across iterations. HFlow converges faster and secures the highest final solve rate, which can be attributed to its human-agent collaboration design and experience memory. The human oracle model provides high-quality guidance during the training iterations, leading to a more direct convergence path at the beginning. Besides, the experience memory fosters a cumulative learning effect, ensuring that the evolutionary process is consistently steered in a beneficial direction, thus further improving final convergence.

5.5 Feedback Impact

Accurate feedback is crucial for steering HFlow's evolutionary trajectory toward optimal workflows. As shown in Table 3, the absence of feedback results in a nearly 10% accuracy drop. Notably, HFlow exhibits high **sensitivity to feedback quality**: introducing wrong or random feedback degrades performance even below the baseline. This proves that HFlow actively learns from human signals rather than relying on random exploration. Furthermore, **feedback granularity** significantly impacts efficiency; a reason-style feedback that includes error identification and actionable suggestions provides clearer optimization direction than one-sentence summaries. Consequently, HFlow achieves its peak performance when utilizing fine-grained multi-score feedback.

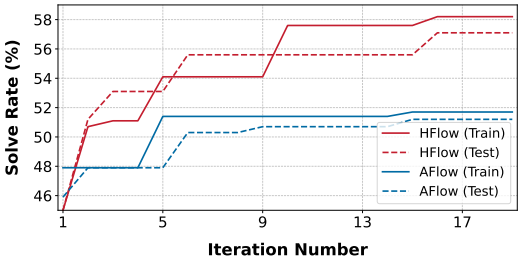

Figure 4: Solve rates across iterations on the MATH training set (119 problems) and test set (486 problems).

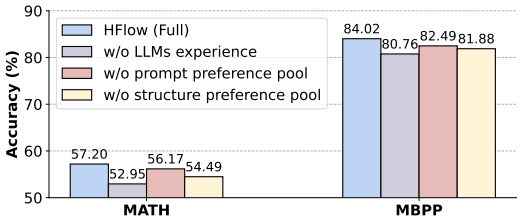

Figure 5: Ablation study of HFlow components.

5.6 Ablation Study

HFlow's performance significantly degrades without any of the layers in our hierarchical experience memory—the prompt preference pool, the structure preference pool, or the LLMs experience—as shown in Fig 5. Among these, the absence of LLMs experience has the most significant impact, followed closely by the structure preference pool. This suggests that selecting the appropriate LLM for a sub-task and defining a robust high-level workflow architecture are the most critical factors for success.

6 Conclusion

In this paper, we explore and demonstrate the value of human-agent collaboration for optimizing agentic workflows on complex tasks, addressing problems like high design cost, inefficient search, and poor adaptability. We propose HFlow, an evolutionary framework that leverages human preferences to guide the search over workflow structures,

prompts, and LLMs. This approach not only establishes a new, cost-effective paradigm for building adaptive agentic systems but also delivers superior performance, surpassing baselines by up to 27.34% and achieving comparable performance to the o1-preview model at only one-fourth of the cost.

Acknowledgments

This work was supported in part by National Key R&D Program of China No. 2024YFB2705300, NSFC Grant 62232011, 62402315, and the Shanghai Science and Technology Innovation Action Plan Grant 24BC3201200.

Limitations

In HFlow, we rely on a hybrid of human experts and a calibrated LLM model for scalability. While effective, we acknowledge that this LLM agent may not fully capture the diversity of real-world user feedback. We will address this limitation by conducting large-scale human studies in future work to validate our framework's generalizability.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Arthur Caetano, Kavya Verma, Atieh Taheri, Radha Kumaran, Zichen Chen, Jiaao Chen, Tobias Höllerer, and Misha Sra. 2025. Agentic workflows for conversational human-ai interaction design. *arXiv preprint arXiv:2501.18002*.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F Karlsson, Jie Fu, and Yemin Shi. 2023a. Autoagents: A framework for automatic agent generation. *arXiv preprint arXiv:2309.17288*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, and 1 others. 2023b. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*, 2(4):6.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Shachar Don-Yehiya, Ben Burtenshaw, Ramon Fernandez Astudillo, Cailean Osborne, Mimansa Jaiswal, Tzu-Sheng Kuo, Wenting Zhao, Idan Shenfeld, Andi Peng, Mikhail Yurochkin, and 1 others. 2025. The future of open human feedback. *Nature Machine Intelligence*, pages 1–11.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Xueyang Feng, Zhi-Yuan Chen, Yujia Qin, Yankai Lin, Xu Chen, Zhiyuan Liu, and Ji-Rong Wen. 2024. Large language model-based human-agent collaboration for complex task solving. *arXiv preprint arXiv:2402.12914*.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*.

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

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Chenxing Wei, Danyang Li, Jiaqi Chen, Jiayi Zhang, and 1 others. 2024. Data interpreter: An llm agent for data science. *arXiv preprint arXiv:2402.18679*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Shengran Hu, Cong Lu, and Jeff Clune. 2024. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*.

Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R Bowman, Tim Rocktäschel, and Ethan Perez. 2024. Debating with more persuasive llms leads to more truthful answers. *arXiv preprint arXiv:2402.06782*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, and 1 others. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Yu Li, Lehui Li, Zhihao Wu, Qingmin Liao, Jianye Hao, Kun Shao, Fengli Xu, and Yong Li. 2025. Agentswift: Efficient llm agent design via value-guided hierarchical search. *arXiv preprint arXiv:2506.06017*.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170*.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, and 1 others. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, and 1 others. 2024. Scaling large language model-based multi-agent collaboration. *arXiv preprint arXiv:2406.07155*.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, and 1 others. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. 2024. Agentsquare: Automatic llm agent search in modular design space. *arXiv preprint arXiv:2410.06153*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

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

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2023. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. *arXiv preprint arXiv:2307.05300*.

Zijie J Wang, Dongjin Choi, Shenyu Xu, and Diyi Yang. 2021. Putting humans in the natural language processing loop: A survey. *arXiv preprint arXiv:2103.04044*.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024. Agent workflow memory. *arXiv preprint arXiv:2409.07429*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. 2024. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. *arXiv preprint arXiv:2406.14228*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, and 1 others. 2024. Aflow: Automating agentic workflow generation. *URL https://arxiv. org/abs/2410.10762*.

Li Zhong, Zilong Wang, and Jingbo Shang. 2024. Debug like a human: A large language model debugger via verifying runtime execution step-by-step. *arXiv preprint arXiv:2402.16906*.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö Arık. 2025. Multi-agent design: Optimizing agents with better prompts and topologies. *arXiv preprint arXiv:2502.02533*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*.