A Survey of Agentic Workflow Optimization

Anonymous ACL submission

Abstract

Multi-agent systems have emerged as a powerful paradigm for solving complex tasks by decomposing them into coordinated subtasks conducted by specialized agents. However, designing effective multi-agent workflows is challenging and requires strong domain expertise, making them brittle, inefficient, and difficult to adapt across tasks and resource constraints. Addressing these challenges, this survey systematically review the emerging field of Multi-Agent Workflow Optimization (MAWO), treating the workflow itself as the object of automatic improvement. We formalize MAWO as a constrained optimization problem and organize existing methods along four dimensions: workflow representation, optimization targets, reward signals, and optimization algorithms. By reviewing and categorizing a wide range of recent works, we provide a unified framework for understanding automated workflow improvement and outline open challenges in this field. This survey serves as a foundational resource for researchers and practitioners advancing self-improving multi-agent systems.

1 Introduction

Large language models (LLMs) have significantly advanced the capabilities of artificial intelligence systems, demonstrating strong performance in reasoning, planning, and tool use across a wide range of tasks (Chen et al., 2021; Ren et al., 2025; Ke et al., 2025). Building upon these capabilities, recent research has increasingly explored LLM-based agents that can perceive their environment, make decisions, and take actions autonomously. While early works focused on single-agent settings (Yao et al., 2022), recent research has shown growing interest in multi-agent systems, where a complex objective is decomposed into a set of subtasks, each delegated to a specialized LLM-based agent that executes in coordination with others.

Multi-agent workflows provide a principled paradigm for structured agent collaborations (Li et al., 2025b; Hong et al., 2023). A workflow specifies how a high-level task is decomposed into interdependent subtasks, how information flows between agents, and how intermediate results are validated and aggregated. However, most existing workflows are manually designed or rely on heuristics and ad hoc prompt engineering, requiring substantial human expertise and iterative tuning. Optimizing the workflow configuration is a complex, multi-objective search problem, balancing task performance against critical constraints like computational budget, latency, and API costs. As a result, manually constructed workflows often suffer from brittleness, inefficiency, and poor generalization across tasks.

Motivated by these challenges, we present a systematic and optimization-centric review of Multi-Agent Workflow Optimization (**MAWO**). MAWO focuses on automatic improvement of agent workflows, where the workflow itself becomes the object of optimization. These methods aim to search, learn, or evolve workflow structures, agent configurations, orchestration strategies, etc. In this work, we unify the design space of workflows and formalize workflow design as a constrained optimization problem. Building on this formulation, we organize existing methods along four dimensions: workflow representation, optimization targets, reward and evaluation signals, and optimization algorithms.

Existing surveys on LLM-based agents and multi-agent systems primarily focus on constructing multi-agent architectures (Li et al., 2024b), inter-agent communication mechanisms (Khan et al., 2023), and applications (Bennai et al., 2023). Most of them take workflows as descriptive system components, and discussion of agent evolution are often limited to isolated components such as prompts or tools (Fang et al., 2025). In contrast, to the best of our knowledge, this survey is the

first to give a systematic and formalized view over workflow optimization.

Specifically, our contributions are threefold:

- Formalization: We provide a unified definition of multi-agent workflows and formulate workflow design as a constrained optimization problem, clarifying the objectives and trade-offs involved.
- Taxonomy: We present a comprehensive taxonomy of existing methods along four dimensions: workflow representations, optimization targets, reward designs, and algorithmic strategies, highlighting connections and distinctions among existing approaches.
- Analysis and Outlook: After reviewing existing approaches, we further discuss open challenges and future directions, outlining promising research directions for scalable, efficient, and general workflow auto-improvement.

2 Workflow Structure and Formulation

Components. The core components of a multi-agent workflow (Chen et al., 2024; Yamamoto et al., 2025; Zhang et al., 2025b) include: **(1) Nodes:** The basic units of work. Each node performs a specific task, such as running code, calling a tool, or retrieving information. **(2) Edges:** Directed connections between nodes. They define the execution order and dependencies, and specify how outputs from one node are passed to the next. **(3) Orchestrator:** The controller that runs the workflow. It triggers nodes according to the edges, passes data and state between steps, and manages logic such as branching, parallelism, and retries so the workflow executes as an ordered, reproducible process.

Notations. To formulate MAW problem, we first define the space of possible workflows, $\mathbb{W}$. A workflow can be characterized by several components: **(1)** Graph Structure $\mathbb{G}$: specifies which nodes the workflow contains and how they are connected, i.e., the overall process structure and information flow. **(2)** Agent Parameters $\mathbb{P}$: Specifies how each node is instantiated, such as the model choice, prompts, and key inference or generation hyperparameters. **(3)** Orchestration Policy $\mathbb{C}$: Specifies how the workflow is executed at runtime, including routing decisions, branching or iteration, stopping criteria, and how outputs from different nodes are combined. We model the workflow space as

$$\mathbb{W} := \mathbb{G} \times \mathbb{P} \times \mathbb{C} \tag{1}$$

meaning each workflow $W = (G, P, C) \in \mathbb{W}$ is a specific combination of a graph structure, agent parameter settings, and an orchestration policy.

Goal. Let $\mathcal{T}$ denote a distribution over task instances τ. Each instance specifies an input x, auxiliary context κ, and an evaluation specification that defines how the workflow output is scored, such as a reference output y or a task-specific scoring rule. The goal of workflow optimization is to find a workflow W^* that maximizes the expected utility over $\mathcal{T}$:

$$W^* = \arg\max_{W \in \mathbb{W}} \mathcal{U}(W), \tag{2}$$

where the utility of a workflow is defined as the expected task performance

$$\mathcal{U}(W) := \mathbb{E}_{\tau \sim \mathcal{T}}\big[\mathrm{Perf}(W, \tau)\big], \tag{3}$$

and $\mathrm{Perf}(W, \tau)$ measures how well workflow W performs on task instance τ. It can be instantiated as accuracy, success rate, pass@k, or a judge-based score.

Objective. In MAW, resources are often limited and are typically treated as explicit constraints, typically including: **(1) Budget constraint:** Limits the overall computation budget of the workflow, such as total token usage, API calls, or monetary cost, so that the workflow does not exceed a fixed budget B. **(2) Latency constraint:** Requires the workflow to finish within a time bound $\mathrm{T}_{\max}$. The constrained optimization problem can be written as

$$\begin{aligned} \arg\max_{W \in \mathbb{W}} \quad & \mathbb{E}_{\tau \sim \mathcal{T}}\big[\mathrm{Perf}(W, \tau)\big] \\ \text{s.t.} \quad & \mathbb{E}_{\tau \sim \mathcal{T}}\big[\mathrm{Cost}_b(W, \tau)\big] \le \mathrm{B}, \\ & \mathbb{E}_{\tau \sim \mathcal{T}}\big[\mathrm{Cost}_t(W, \tau)\big] \le \mathrm{T}_{\max}. \end{aligned} \tag{4}$$

The objective in (4) searches for the optimal W that maximizes expected task performance while keeping the expected budget-related cost and expected runtime within the limits B and $\mathrm{T}_{\max}$, thereby making the performance-efficiency trade-off explicit. This is a basic formulation and can be extended with additional constraints, such as an instantaneous (peak) resource constraint that bounds the maximum memory or CPU usage during execution to prevent system overload when multiple agents run concurrently.

Failure modes of MAW. Designing effective multi-agent workflows is a challenging task, as there are multiple potential failure modes. MAW often struggle due to **(1) poor or ambiguous prompt** specifications, which can make coordination strategies

underspecified and force repeated clarification (Pan et al., 2024; Wang et al., 2024a). In particular, **(2) unclear agent roles** and responsibilities undermine division of labor and can lead to redundancy of roles, reducing collaboration efficiency (Li et al., 2025a; Zhu et al., 2025). Another major failure mode arises from **(3) inter-agent interaction and coordination breakdowns** during execution (Zhu et al., 2025; Agashe et al., 2023). Agents may **(4) omit essential context**, leading to lost details and downstream decisions based on partial information (Pan et al., 2024; Han et al., 2024). They may also face **(5) inadequate tools or access**, preventing key actions or verification and causing the workflow to stall (Cemri et al., 2025). Recent work has begun to systematically characterize these failures (Cemri et al., 2025), identifying 14 failure modes and showing that straightforward fixes struggle to succeed. These challenges underscore the urgent need for a systematic review of automatic optimization strategies to improve MAW design.

3 Automatic Workflow Improvement

This section focuses on how automatic MAW optimization (MAWO) were improved. We summarize the key design dimensions of MAWO, including workflow representations, which components to optimize, workflow topology, optimization signals or rewards, and the optimization algorithms used to search the workflow space.

3.1 Workflow Modeling

Most MAWO modeling approaches can be categorized into the following two paradigms:

- **Textual-instruction modeling.** This paradigm formulate MAWO in the space of natural language: defining it as a textual object that can be generated, edited, and compared. For example, a collection of agent role descriptions and system prompts, message templates and interaction protocols, and communication topology expressed as a graph encoded in text. Under this formulation, the search space is the set of such textual specifications, and the optimization problem is to find the specification that maximizes end-task utility under constraints such as token budget, latency, or safety. Typically, MASS frames the design variables explicitly as prompts and topologies, and defines the goal as selecting the best-performing configuration in that combined space (Zhou et al., 2025). GPTSwarm models the workflow as a graph structured prompting scheme. Node behaviors are language operations parameterized by prompt templates. Edges specify how intermediate textual contexts are passed and transformed to compose multi-step reasoning patterns (Zhuge et al., 2024). EvoAgentX extends multi-agent workflows into a modular system. Each agent is configured with a prompt template that specifies its role and task in natural language, and these prompts are iteratively refined by the system's optimization algorithms so the workflow is composed and updated through textual instructions (Wang et al., 2025b).

- **Programming Modeling.** An alternative paradigm implements agentic workflow procedure through explicit programming. AFlow models workflow discovery as search in a program space where workflows are represented as executable code with node parameters and control-flow edges (Zhang et al., 2024d). ADAS treats agent invention itself as program synthesis, letting a meta-agent write agents in code and search the program space (Hu et al., 2024). SEW specializes in code generation, automatically generating and optimizing multi-agent workflows and studying textual encodings of workflow representations (Liu et al., 2025). VFlow casts Verilog generation as graph-structured workflow search under a code-level multi-objective combining compile and simulation correctness with area, power and timing (Wei et al., 2025). MetaGPT formalizes collaborative software engineering as SOP-encoded multi-agent processes that produce and verify code artifacts, including tests and debugging (Hong et al., 2023).These works define either the search space as programs or the objective signals via compile, test and execution feedback, grounding optimization in coding.

3.2 What to Optimize

The design space of LLM-based agent workflow systems encompasses multiple interconnected components, including: (1) **prompts** that define agent instructions and provide task context, (2) **workflow topology** that determines the structural arrangement and connectivity of agents, (3) **communication protocols** that regulate inter-agent information exchange, (4) **operators** that serve as reusable building blocks for common agent operations, (5) **tools and APIs** that extend agent capabilities.

Table 1: Main optimization components. Legend: ✓✓: primary target; ✓: secondary or considered; –: not a focus.

Method	Prompts	Workflow Topology	Comm.	Operators	Hyper-parameters	Memory	Tools
Reflexion (Shinn et al., 2023)	✓✓	–	–	–	–	–	–
FlowMind (Zeng et al., 2023)	–	✓	–	–	–	–	–
MetaGPT (Hong et al., 2023)	–	✓	✓	–	–	–	–
PromptBreeder (Fernando et al., 2024)	✓✓	–	–	–	–	–	–
G-Designer (Zhang et al., 2024c)	–	✓	✓✓	–	–	–	–
GPTSwarm (Zhuge et al., 2024)	–	✓✓	✓	✓	–	–	–
EvoAgentX (Yuan et al., 2024)	✓✓	✓✓	–	–	–	✓	✓
AFlow (Zhang et al., 2024d)	✓	✓✓	–	✓✓	–	–	–
ADAS (Hu et al., 2024)	✓	✓✓	–	✓	–	–	✓
MaAS (Zhang et al., 2025b)	✓✓	✓✓	–	✓	–	–	✓
ScoreFlow (Wang et al., 2025c)	✓✓	✓✓	–	✓	–	–	–
OPTIMAS (Wu et al., 2025)	✓✓	–	–	–	✓✓	–	–
VFlow (Wei et al., 2025)	✓	✓✓	–	✓	–	–	–
W4S (Nie et al., 2025)	✓	✓✓	–	✓	–	–	–
DebFlow (Su et al., 2025)	–	✓✓	–	✓	–	–	–
FlowReasoner (Gao et al., 2025)	✓	✓✓	–	–	–	–	–
MermaidFlow (Zheng et al., 2025a)	–	✓✓	–	✓	–	–	–
ARG-Designer (Li et al., 2025b)	–	✓	✓✓	–	–	–	–
SEW (Liu et al., 2025)	✓	✓✓	–	–	–	–	–
Topological Learning (Yang et al., 2025)	–	✓✓	✓✓	–	–	–	–

3.2.1 Prompt

In agent workflow systems, prompts can play various roles. The prompt provides persona, instruction, and may include examples, to empower role assignment, task assignment, and chain-of-thought. These applications have made prompt one of the key optimization targets in MAWO.

To improve prompts' contextualization capabilities, Zhang et al. (2025b) build blocks with agents divided into five types, with prompt differs across agent types. Each prompt includes a persona, a task description, and examples of this task. In ScoreFlow (Wang et al., 2025c), example tasks are enriched including code generation, debugging, and formatting answers. OPTIMAS further includes tasks such as query rewriting, answer generation, and context analysis. FlowReasoner builds a prompt library to store various prompts.

One important direction is experience collection and self-evolution of prompts. In EvoAgentX (Wang et al., 2025b), the prompt is initialized by a minimalist task description. But with various-level optimization, the prompt is fulfilled with a more detailed description, demonstrations, etc. Reflexion (Shinn et al., 2023) enhances the prompt with the self-reflection of trial results. In ADAS, the prompt also evolves from a zero-shot chain of thought prompt to including feedback and human-like insights. In Aflow (Zhang et al., 2024d), the initial prompt is defined as an Operator prompt, including a simple task description. Later, with evolving, customizable prompts, such as the experience of avoiding format errors and test failures. SEW (Liu et al., 2025) evolves the prompt by adding the experience of various perspectives. First, requirements are added to avoid test failures. Then, meta-analysis is included. Finally, format requirements are added.

To develop a pricipled prompt evolution framework, PromptBreeder (Fernando et al., 2024) designs a three-level hierarchical prompt system: the task prompt, the mutation prompt, and the hyper-mutation prompt. The task prompt provides direct instructions for solving the downstream task, the mutation prompt describes how to evolve the task prompt. The hyper-mutation prompt, as it is named, decides how to update the mutation prompt.

Besides, MAWO is a multi-objective optimization task. To prevent the dominance of objective subsets, VFlow (Wei et al., 2025) proposes population-specific refinement for prompt evolution. Each population focuses on a specific objective. As a result, each population-specific refinement guides the prompt to evolve towards one of the directions to the final goal.

3.2.2 Workflow Topology

Creating new agents is common during the construction of an agent system. When multiple agents are created, their topological relations in the agent system are non-trivial for the system's performance. These topological relations are defined in a manner that can be optimized. These are graph-based, code-based, and specification-based representations.

Graph Generation. By modeling the workflow topology as DAGs, the topology optimization problem is transformed into a graph generation problem. The agent workflow in GPTSwarm (Zhuge et al., 2024) is modeled as DAGs with a tuple (N, E, F, o), consisting of nodes N, edges E, computational routines F, and output node o. G-Designer (Zhang et al., 2024c) uses a variational graph auto-encoder to construct task-adaptive dynamic topologies represented as graphs. MaAS defines an agentic supernet and samples its topologies from an architecture distribution. EvoAgentX (Lee et al., 2025) implements a modular architecture with five layers (basic components, agent, workflow, evolving, evaluation), representing workflows as graphs with agents and execution topologies. EvoFlow (Zhang et al., 2025a) defines hierarchical workflows, where invoking nodes and operator nodes are connected by intra-operator and inter-operator connectivity.

Code Generation. Code-based methods (Zeng et al., 2023; Zhang et al., 2024d; Hu et al., 2024; Wang et al., 2025c; Wei et al., 2025; Su et al., 2025; Nie et al., 2025; Gao et al., 2025) represent agent workflows as executable Python code. Differing from the graph-based representation, the code-based representation implicitly embeds the DAG in the code execution flow. I.e., the code itself **is** the agent workflow topology. With code-based representation, the coding ability of the LLMs can be applied to modify and refine the current topology (code). Once generated, this code can be executed to verify its correctness and can be interpreted when it is not too complex.

Specification Representation Specification-based representation models the topological information with formal languages such as markup languages and Domain-Specific Languages (DSLs). MermaidFlow (Zheng et al., 2025a) uses Mermaid markup language to create typed, declarative workflow graphs with role semantics, data flow, and structural constraints. Later, the mermaid specification is compiled into executable code. SEW (Liu et al., 2025) investigates five representation schemas (BPMN, CoRE, Python, YAML, pseudo-code) for self-evolution. MetaGPT implements role-based orchestration with SOPs (Standard Operating Procedures), where agent workflows are defined through templates.

3.2.3 Agents Communication

The communication connects the downstream task solving and the workflow topology. MetaGPT (Hong et al., 2023) proposes to establish schemas and mechanisms for agent communication. Specifically, each agent is expected to generate content with a specified format. Their communication mechanism is a subscription system to a shared message pool, which aggregates messages from all agents.

Many works explore automatic communication optimization. G-Designer (Zhang et al., 2024c), the task-specific communication is initialized as a simple chain structure and is refined by the optimizer. GPTSwarm (Zhuge et al., 2024) initializes the communication by assigning a pre-defined probability to each possible edge. In ARG-Designer (Li et al., 2025b), the edge is generated following the creation of the corresponding node. Specifically, Topology Learning (Yang et al., 2025) proposes a two-hierarchy communication mechanism. The optimizer first selects macro-level communication patterns such as chain, star, etc. Subsequently, the micro-level communication graphs are generated.

3.2.4 Operators

Operators are reusable modules or abstractions. Some examples are building blocks and agentic operations. These operators enable efficient workflow construction and reduce the search space of workflow optimization.

Depending on the target domains and task scales, the operators considered by the workflow system vary. Aflow (Zhang et al., 2024d), Scoreflow (Wang et al., 2025c), and Debflow (Su et al., 2025) have identical 7 operators: *Generate, CodeGenerate, Format, Review, Revise, Ensemble, Test, Programmer*. Other works that apply operators each have a different operator setting. Some operators are task-agnostic and are commonly used, such as *ensemble* (Zhang et al., 2024d; Wang et al., 2025c; Su et al., 2025; Gao et al., 2025; Zheng et al., 2025a; Zhang et al., 2025a) that outputs the best solution among all generated solutions, *Test* (Zhang et al., 2024d; Wang et al., 2025c; Su

et al., 2025; Gao et al., 2025; Zheng et al., 2025a) that provides success or failure information, and *Review/Reflect* (Zhang et al., 2024d; Wang et al., 2025c; Su et al., 2025; Zhang et al., 2025b; Gao et al., 2025; Zhang et al., 2025a) that provides feedback. In addition, some operators are task-dependent, such as Circuit Optimizer (Wei et al., 2025) for the downstream task Verilog generation.

3.2.5 Tool Integration

External tools, such as Web search, APIs, memory, and code execution, enable the agent to access external information, verify solutions, and interact with the physical world. How to use these tools is a non-trivial problem in workflow optimization.

Code execution tools are used in workflow, either to verify the generated workflow topology (Zhang et al., 2024d; Wang et al., 2025c; Gao et al., 2025; Zhang et al., 2025b), or to verify the generated code in the coding domain (Gao et al., 2025; Wei et al., 2025). Information retrieval tools such as RAG retrievers (Wu et al., 2025; Zhang et al., 2025b) are used to retrieve related documents.Web search tools are invoked when real-time information is necessary for downstream tasks (Zhang et al., 2025b; Zhuge et al., 2024).

3.3 Optimization Reward

The efficacy of agentic workflow optimization critically depends on the design of appropriate reward signals. Unlike traditional optimization where objective functions or rewards are readily computable, agentic workflows present unique challenges: their execution involves multiple steps, partial outputs, and complex dependencies. This section examines two fundamental paradigms for reward formulation in workflow optimization: intermediate rewards for partial evaluation and long-term rewards for holistic assessment.

3.3.1 Intermediate reward

Intermediate rewards are widely used by simplifying the complex workflow evaluation task into smaller, more manageable units. By evaluating partial progress, these rewards offer immediate feedback to guide the optimization of individual components, without waiting for the entire workflow to complete. The design of intermediate rewards typically involves: **(1) Component-level Evaluators**: These are specialized functions designed to assess the output of individual workflow components or agents. For example, in a code generation workflow, a syntax checker can provide early feedback on code validity, allowing immediate corrections before the execution phase begins. **(2) Step-wise Performance Metrics**: These rewards assess success at each step of the workflow, offering real-time evaluation of specific aspects of the process. For instance, a chain-of-thought agent might be evaluated on reasoning consistency at each reasoning step, or in a retrieval-augmented generation (RAG) workflow, the relevance of retrieved documents can be scored before they are used in the final output.

3.3.2 Long-term Reward

Long-term rewards provide a holistic evaluation of the entire workflow's performance, aligning the agent's objectives with the ultimate task goal. These rewards typically come with the challenge of reward sparsity and delayed feedback, where feedback is provided only after the full task is completed and it is difficult for agents to assign it across the action trajectory. Key formulations of long-term rewards include: **(1) End-to-End Task Success**: This type of reward provides a binary or scalar measure indicating whether the workflow successfully achieved its intended goal. It's commonly used in benchmark evaluations and serves as a clear indicator of overall success or failure. **(2) Cost-Aware Utility Functions**: These rewards balance task performance with resource consumption. For example, the utility function might incorporate factors such as token usage, execution latency, and financial costs (e.g., computational resource costs or cloud service fees). This encourages workflows that optimize not just for accuracy or speed, but for efficiency in terms of resources used. **(3) Human Preference Signals**: As workflows become more complex and nuanced, automated metrics may fail to capture quality dimensions valued by humans, such as creativity or user satisfaction. In these cases, rewards may be designed based on human feedback, where a reward model is learned from human judgments to guide optimization towards more qualitative, user-preferred outcomes.

3.4 Optimization Methods

The efficacy of a multi-agent workflow is determined not only by the capabilities of its individual agents but also by the optimization of its structure, parameters, and dynamic execution policies. Optimizing these complex systems involves navigating a vast search space of possible agent configurations, interaction patterns, and operational parame-

ters. This section surveys the primary algorithmic families employed for this optimization challenge, ranging from direct construction methods to sophisticated learning and search-based meta-heuristics.

3.4.1 Zero-shot Generation

Zero-shot generation leverages LLMs to directly generate valid multi-agent workflow structures, task allocation rules, or coordination logics. It excels at rapid workflow prototyping and adapting to novel multi-agent scenarios. In foundational approaches (Yao et al., 2022; Liu et al., 2024; Wang et al., 2025a), a human designer explicitly defines the agents, their roles, their tools, and the graph structure governing their interactions. These methods rely heavily on domain expertise and iterative prompt engineering. A survey on agentic retrieval-augmented generation (Singh et al., 2025) systematically reviews how zero-shot prompt design for the agent enables dynamic information integration in multi-agent workflows, implicitly highlighting design patterns for these workflows, which serve as heuristics for effective manual construction. While not an automated optimization algorithm in the traditional sense, this expert-driven design serves as the essential baseline and starting point for more advanced automated methods.

3.4.2 Reinforcement Learning

Reinforcement Learning offers a formal mathematical framework for optimizing LLM-based agents in dynamic environments (Xu et al., 2023). Typical multi-agent RL methods employ an actor-critic framework like MADDPG (Lowe et al., 2017) and MAPPO (Yu et al., 2022), and value-based ones like VDN (Sunehag et al., 2017), QMIX (Rashid et al., 2020), and NA2Q (Liu et al., 2023c). This simplified and memory-efficient structure makes GRPO (Shao et al., 2024) particularly well-suited for the joint training of complex, heterogeneous multi-agent systems (Ke et al., 2025; Li et al., 2024c; Zhuge et al., 2024).

Concretely, Yue et al. (2025) targeted complex trajectories in knowledge-intensive tasks by decomposing them into subtasks. It put forward a co-training paradigm for multi-agent frameworks, which not only secures synergy among agents but also preserves the fine-grained performance of each individual agent. Liu et al. (2023b) proposed an alignment scheme which effectively mitigates the issues of instability and reward gaming that are typically associated with reward-based RL optimization. JoyAgents-R1 (Han et al., 2025) introduced an adaptive memory evolution mechanism that repurposed GRPO rewards as cost-free supervisory signals to eliminate repetitive reasoning and accelerate convergence.

3.4.3 Evolutionary Computing

Researchers have extensively explored the interaction between evolutionary algorithms (EAs) and LLM workflows. This exploration covers diverse areas, including prompting (Xu et al., 2022), code generation (Romera-Paredes et al., 2024), project planning (Tao et al., 2023), and evolutionary scaling during inference (Lee et al., 2025).

Early efforts, such as EvoAgent (Yuan et al., 2024) and EvoPrompt (Guo et al., 2023), made use of basic genetic algorithms to optimize individual agent components, specifically agent profiles and prompts. To evolve at the workflow level, Evoflow (Zhang et al., 2025a) automatically searches a population of heterogeneous, complexity-adaptive agentic workflows. It utilizes tag-based retrieval alongside core evolutionary mechanics such as crossover, mutation, and niching-based selection. Furthermore, the open-source platform EvoAgentX (Wang et al., 2025b) automates the generation, execution, and evolutionary optimization of multi-agent workflows. It employs a modular five-layer evolving architecture to iteratively refine agent prompts, tool configurations, and the overall workflow topologies.

3.4.4 Monte Carlo Tree Search

With the great success of CoT in enhancing the ability of LLM reasoning, MCTS has recently emerged as a powerful technique to conduct multi-hop reasoning (Zhang et al., 2024a; Feng et al., 2023). These methods mainly construct MCTS in two ways, representing each node with a complete answer refined from the parents (Zhou et al., 2023; Zhang et al., 2024b) or a reasoning step following its parents (Qi et al., 2024; Antoniades et al., 2024). Recently, MCTS-AHD (Zheng et al., 2025b) organizes LLM-generated heuristics in a tree structure and can better develop the potential of temporarily underperforming heuristics. AFlow (Zhang et al., 2024d) formulates the workflow generation as a search problem over a code-based operator space, using Monte Carlo evaluations to prune the search tree. AgentSwift (Li et al., 2025c) proposes an uncertainty-guided hierarchical expansion strategy based on MCTS, incorporating recombination,

mutation, and refinement over components of the workflow. This evolution positions MCTS not just as a reasoning tool, but as a key enabler for complex, adaptive agentic tree-search and optimization.

3.4.5 Others

A growing body of research adapts neural architecture search (NAS) (Ren et al., 2021) in agentic workflows, such as specifically *meta learning* and *bayesian optimization*, to the automating the design of agentic systems. Works like ADAS (Hu et al., 2024) and FlowReasoner (Gao et al., 2025) utilize techniques ranging from meta-learning to jointly fine-tune agent prompts, configurations, and the execution graph itself. AutoAgents (Chen et al., 2023) generates task-adaptive agent teams and execution graphs by searching through a space of possible roles effectively performing instance-level architecture search. More recently, MaAS (Zhang et al., 2025b) introduces a differentiable search strategy using probabilistic supernets by learning a query-conditioned distribution over agentic architectures, which shifts the objective from finding a single static workflow to learning an adaptive policy. While meta learning focuses on structure, bayesian optimization (Zhou et al., 2019) addresses the challenge of tuning agent hyperparameters and modulars. For example, MoLAS (Shang et al., 2024) searches by LLM agents modular design space bayesian combination for novel and good performance. Cognify (He et al., 2025) automatically optimizes complex gen-AI workflows by dynamically allocating a search budget across different tuning layers based on workflow-level evaluation results. These methods generally treat workflow optimization as a hyperparameter or topology search problem, employing LLMs not just as actors, but as optimizers themselves.

4 Applications

Due to space limitation, we put application and benchmark summaries in App. A.

5 Discussions

5.1 Challenges

Despite the promise of MAWO, several critical hurdles remain due to the stochastic nature of LLMs and the complexity of multi-agent environments.

Optimization Complexity and Search Space. The search space $\mathbb{W} = \mathbb{G} \times \mathbb{P} \times \mathbb{C}$ is high-dimensional, discrete, and non-differentiable. Small changes in prompts or topology can lead to non-monotonic performance shifts. Furthermore, evaluating the true utility function $\mathcal{U}(W)$ requires expensive, repeated execution of agent chains, making exhaustive search infeasible and necessitating more sample-efficient optimization strategies.

Correctness, Verification, and Safety. Validating workflows is harder than modular software because upstream hallucinations can propagate through a chain, leading to "silent failures." Current optimization methods largely optimize for final metrics, lacking the granularity to verify intermediate reasoning steps or implement “quality gateways" effectively. Without formal verification mechanisms or reliable confidence estimation at the node level, optimized workflows are prone to malfunction amplification, where minor errors in early stages cascade into catastrophic failures in the final output.

Generalization to Structural and Semantic Constraints in Open-ended Domains. Beyond binary tasks like coding, where success is measurable via unit tests, generalizing optimization to open-ended domains remains difficult due to implicit constraints and interface mismatches. In reality, nodes and edges are governed by latent semantic constraints and logical dependencies, such as prior knowledge dependencies or data format requirements. Agents often produce outputs that mismatch the input schema of downstream agents, leading to structural failures. An effective optimizer must treat workflows as constrained state machines, ensuring that state transitions respect both the rigid logic of the task and the soft constraints.

5.2 Future Directions

To further advance MAWO, research should transition from heuristic, black-box search toward principled, transparent frameworks. Promising directions include: (1) Improving sample efficiency. For example, using differentiable components or surrogate models to reduce the cost of evaluating complex workflows. (2) Adaptive optimization. Workflows can be improved by dynamically updating their topology and strategies over time as environments and user intents shift. (3) Neuro-symbolic integration. Incorporating formal constraints and human-in-the-loop feedback would benefit MAW’s reliability and controllability. By bridging the gap between stochastic generation and rigorous logic, optimized workflows can be not only high-performing but also interpretable, safe, and robust.

Limitations

Our survey is subject to a few limitations. First, we primarily focus on the optimization of workflows driven by LLMs, and thus do not extensively cover classical multi-agent reinforcement learning unless explicitly adapted for generative agents. Second, given the extremely rapid pace of research in this domain, this review represents a snapshot of the current landscape; while we strive for comprehensiveness, new methods appear frequently. Finally, the absence of standardized benchmarks and unified evaluation protocols across the community restricts our ability to perform a rigorous quantitative meta-analysis, necessitating a reliance on qualitative taxonomy and conceptual comparison.

Ethical considerations

The advancement of MAWO introduces significant ethical implications. A primary concern is dual-use risk; the same optimization algorithms that enhance productivity could be exploited to increase the efficiency and lethality of malicious workflows, such as automated cyber-attacks or disinformation campaigns. Additionally, as workflows become autonomously evolved and increasingly complex, there is a risk of reduced interpretability and accountability, creating black-box systems where the decision-making logic becomes opaque to human supervisors, complicating error diagnosis and safety enforcement in critical applications.

References

Saaket Agashe, Yue Fan, and Xin Eric Wang. 2023. Evaluating multi-agent coordination abilities in large language models.

Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Wang. 2024. Swe-search: Enhancing software agents with monte carlo tree search and iterative refinement. *arXiv preprint arXiv:2410.20285*.

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*.

M. Bennai, Z. Guessoum, S. Mazouzi, S. Cormier, and M. Mezghiche. 2023. Multi-agent medical image segmentation: A survey. *Computer methods and programs in biomedicine*, 232:107444.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, and 1 others. 2025. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*.

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

Junzhe Chen, Xuming Hu, Shuodi Liu, Shiyu Huang, Wei-Wei Tu, Zhaofeng He, and Lijie Wen. 2024. Llmarena: Assessing capabilities of large language models in dynamic multi-agent environments. *arXiv preprint arXiv:2402.16499*.

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, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, and Zihao Li. 2025. A comprehensive survey

of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, pages 13481–13544.

Hongcheng Gao, Yue Liu, Yufei He, Longxu Dou, Chao Du, Zhijie Deng, Bryan Hooi, Min Lin, and Tianyu Pang. 2025. Flowreasoner: Reinforcing query-level meta-agents. *arXiv preprint arXiv:2504.15257*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2023. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*.

Ai Han, Junxing Hu, Pu Wei, Zhiqian Zhang, Yuhang Guo, Jiawei Lu, and Zicheng Zhang. 2025. Joyagents-r1: Joint evolution dynamics for versatile multi-llm agents with reinforcement learning. *arXiv preprint arXiv:2506.19846*.

Shanshan Han, Qifan Zhang, Yuhang Yao, Weizhao Jin, and Zhaozhuo Xu. 2024. Llm multi-agent systems: Challenges and open problems. *arXiv preprint arXiv:2402.03578*.

Zijian He, Reyna Abhyankar, Vikranth Srivatsa, and Yiying Zhang. 2025. Cognify: Supercharging gen-ai workflows with hierarchical autotuning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 932–943.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

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*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 523–533.

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

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 2567–2577.

Yilun Jin, Zheng Li, Chenwei Zhang, Tianyu Cao, Yifan Gao, Pratik Jayarao, Mao Li, Xin Liu, Ritesh Sarkhel, Xianfeng Tang, and 1 others. 2024. Shopping mmlu: A massive multi-task online shopping benchmark for large language models. *Advances in Neural Information Processing Systems*, 37:18062–18089.

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, and 1 others. 2025. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *arXiv preprint arXiv:2504.09037*.

Rimsha Khan, Nageen Khan, and Tauqir Ahmad. 2023. Communication in multi-agent reinforcement learning: A survey. *The Nucleus*, 60(2):174–184.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. 2025. Evolving deeper llm thinking. *arXiv preprint arXiv:2501.09891*.

Haoran Li, Ziyi Su, Yun Xue, Zhiliang Tian, Yiping Song, and Minlie Huang. 2025a. Advancing collaborative debates with role differentiation through multi-agent reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22655–22666.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024a. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *Preprint*, arXiv:2402.19255.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. 2025b. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. *arXiv preprint arXiv:2507.18224*.

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. 2024b. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):1–43.

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

Zelong Li, Shuyuan Xu, Kai Mei, Wenyue Hua, Balaji Rama, Om Raheja, Hao Wang, He Zhu, and Yongfeng Zhang. 2024c. Autoflow: Automated workflow generation for large language model agents. *arXiv preprint arXiv:2407.12821*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Mingjie Liu, Nathaniel Pinckney, Brucek Khailany, and Haoxing Ren. 2023a. Verilogeval: Evaluating large language models for verilog code generation. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–8. IEEE.

Ruibo Liu, Ruixin Yang, Chenyan Jia, Ge Zhang, Denny Zhou, Andrew M Dai, Diyi Yang, and Soroush Vosoughi. 2023b. Training socially aligned language models on simulated social interactions. *arXiv preprint arXiv:2305.16960*.

Siwei Liu, Jinyuan Fang, Han Zhou, Yingxu Wang, and Zaiqiao Meng. 2025. Sew: Self-evolving agentic workflows for automated code generation. *arXiv preprint arXiv:2505.18646*.

Yuchi Liu, Jaskirat Singh, Gaowen Liu, Ali Payani, and Liang Zheng. 2024. Towards hierarchical multi-agent workflows for zero-shot prompt optimization. *arXiv preprint arXiv:2405.20252*.

Zichuan Liu, Yuanyang Zhu, and Chunlin Chen. 2023c. Na2q: Neural attention additive model for interpretable multi-agent q-learning. In *International Conference on Machine Learning*, pages 22539–22558. PMLR.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30.

Fan Nie, Lan Feng, Haotian Ye, Weixin Liang, Pan Lu, Huaxiu Yao, Alexandre Alahi, and James Zou. 2025. Weak-for-strong: Training weak meta-agent to harness strong executors. In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*.

Bo Pan, Jiaying Lu, Ke Wang, Li Zheng, Zhen Wen, Yingchaojie Feng, Minfeng Zhu, and Wei Chen. 2024. Agentcoord: Visually exploring coordination strategy for llm-based multi-agent collaboration. *arXiv preprint arXiv:2404.11943*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. 2021. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34.

Weijieying Ren, Tianxiang Zhao, Yuqing Huang, and Vasant Honavar. 2025. Deep learning within tabular data: Foundations, challenges, advances and future directions.

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.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1743–1752.

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*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2022. Language models are multilingual chain-of-thought reasoners. *Preprint*, arXiv:2210.03057.

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*.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*.

Jinwei Su, Yinghui Xia, Ronghua Shi, Jianhui Wang, Jianuo Huang, Yijin Wang, TIANYU SHI, Yang Jingsong, and Lewei He. 2025. Debflow: Automating agent creation via agent debate. In *ICML 2025 Workshop on Collaborative and Federated Agentic Workflows*.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, and 1 others. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Tao, Anthony Ventresque, and Takfarinas Saber. 2023. Program synthesis with generative pre-trained transformers and grammar-guided genetic programming grammar. In *2023 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pages 1–6. IEEE.

Qian Wang, Tianyu Wang, Zhenheng Tang, Qinbin Li, Nuo Chen, Jingsheng Liang, and Bingsheng He. 2025a. All it takes is one prompt: An autonomous llm-ma system. In *ICLR 2025 Workshop on Foundation Models in the Wild*.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024a. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? *arXiv preprint arXiv:2402.18272*.

Yingxu Wang, Siwei Liu, Jinyuan Fang, and Zaiqiao Meng. 2025b. Evoagentx: An automated framework for evolving agentic workflows. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 643–655.

Yinjie Wang, Ling Yang, Guohao Li, Mengdi Wang, and Bryon Aragam. 2025c. Scoreflow: Mastering llm agent workflows via score-based preference optimization. *arXiv preprint arXiv:2502.04306*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

Yangbo Wei, Zhen Huang, Huang Li, Wei W Xing, Ting-Jung Lin, and Lei He. 2025. Vflow: Discovering optimal agentic workflows for verilog generation. *arXiv preprint arXiv:2504.03723*.

Shirley Wu, Parth Sarthi, Shiyu Zhao, Aaron Lee, Herumb Shandilya, Adrian Mladenic Grobelnik, Nurendra Choudhary, Eddie Huang, Karthik Subbian, Linjun Zhang, and 1 others. 2025. Optimas: Optimizing compound ai systems with globally aligned local rewards. *arXiv preprint arXiv:2507.03041*.

Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. 2024. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. *Advances in Neural Information Processing Systems*, 37:127129–127153.

Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. 2022. Gps: Genetic prompt search for efficient few-shot learning. *arXiv preprint arXiv:2210.17041*.

Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. 2023. Language agents with reinforcement learning for strategic play in the werewolf game. *arXiv preprint arXiv:2310.18940*.

Atsushi Yamamoto, Takumi Iida, Taito Naruki, Akihiko Katagiri, Yudai Koike, Ryuta Shimogauchi, Kota Shimomura, Eri Onami, Koki Inoue, and Osamu Ito. 2025. Dynamic knowledge integration in multi-agent systems for content inference. In *Towards Agentic AI for Science: Hypothesis Generation, Comprehension, Quantification, and Validation*.

Jiaxi Yang, Mengqi Zhang, Yiqiao Jin, Hao Chen, Qingsong Wen, Lu Lin, Yi He, Weijie Xu, James Evans, and Jindong Wang. 2025. Topological structure learning should be a research priority for llm-based multi-agent systems. *arXiv preprint arXiv:2505.22467*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624.

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*.

Shengbin Yue, Siyuan Wang, Wei Chen, Xuanjing Huang, and Zhongyu Wei. 2025. Synergistic multi-agent framework with trajectory learning for knowledge-intensive tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25796–25804.

Zhen Zeng, William Watson, Nicole Cho, Saba Rahimi, Shayleen Reynolds, Tucker Balch, and Manuela Veloso. 2023. Flowmind: automatic workflow generation with llms. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 73–81.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024b. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*.

Guibin Zhang, Kaijie Chen, Guancheng Wan, Heng Chang, Hong Cheng, Kun Wang, Shuyue Hu, and Lei Bai. 2025a. Evoflow: Evolving diverse agentic workflows on the fly. *arXiv preprint arXiv:2502.07373*.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025b. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2024c. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, and 1 others. 2024d. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*.

Chengqi Zheng, Jianda Chen, Yueming Lyu, Wen Zheng Terence Ng, Haopeng Zhang, Yew-Soon Ong, Ivor Tsang, and Haiyan Yin. 2025a. Mermaidflow: Redefining agentic workflow generation via safety-constrained evolutionary programming. *arXiv preprint arXiv:2505.22967*.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. 2025b. Monte carlo tree search for comprehensive exploration in llm-based automatic heuristic design. *arXiv preprint arXiv:2501.08603*.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2023. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*.

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*.

Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. 2019. Bayesnas: A bayesian approach for neural architecture search. In *International conference on machine learning*, pages 7603–7613. PMLR.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Daisy Zhe Wang, Zhenhailong Wang, Cheng Qian, Robert Tang, Heng Ji, and 1 others. 2025. Multiagentbench: Evaluating the collaboration and competition of llm agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8580–8622.

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*.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, and 1 others. 2024. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv preprint arXiv:2406.15877*.

A Application Domains

Recent progress in MAWO has been empirically validated on a wide range of downstream tasks. To organize this literature, we group existing works by the evaluation tasks they adopt, which helps highlight domains that have been extensively studied as well as those that are less explored. Table 2 summarizes commonly used evaluation benchmarks and their corresponding domains. Below, we discuss the main application areas of MAWO.

Software Engineering. A dominant setting treats workflow optimization as improving code generation, debugging, or software-construction pipelines, typically assessed by functional correctness or task completion in programming environments. Representative works include Reflexion (Shinn et al., 2023), AFlow (Zhang et al., 2024d), G-Designer (Zhang et al., 2024c), GPTSwarm (Zhuge et al., 2024), SEW (Antoniades et al., 2024), FlowReasoner (Gao et al., 2025), EvoAgentX (Wang et al., 2025b), and OPTIMAS (Wu et al., 2025). These efforts typically instantiate multi agent pipelines that decompose software engineering tasks into specialized roles such as planning, code writing, debugging, and verification, and then optimize workflow components.

Mathematical Reasoning. Another major cluster of MAWO works focus on multi-step mathematical problem solving, emphasizing structured derivations, arithmetic reasoning, or competition-style math challenges (Zhang et al., 2024d; Nie et al., 2025; Wang et al., 2025c; Zhang et al., 2025b; Su et al., 2025; Li et al., 2025b; Wang et al., 2025b). These studies are evaluated on mathematical testbed where candidate workflows are compared primarily by final answer correctness and related automated signals, enabling scalable iteration and selection of improved multi-step solution procedures.

Knowledge-intensive Question Answering. MAWO is also commonly evaluated on question answering tasks that require multi-hop evidence aggregation or complex reasoning over provided context (Zhang et al., 2024d; Nie et al., 2025; Su et al., 2025; Wang et al., 2025b; Shinn et al., 2023). These efforts evaluate their methods on knowledge-intensive, open-domain multi-hop question answering tasks that require aggregating multiple pieces of evidence from encyclopedic sources to answer complex questions.

Interactive Decision Making. Several works evaluate MAWO in interactive environments where agents must plan, act, and recover from failures, often under sparse or delayed feedback. For example, Reflexion (Shinn et al., 2023) explicitly evaluates sequential decision-making in interactive settings and web-interaction tasks; GPTSwarm (Zhuge et al., 2024) and MaAS (Zhang et al., 2025b) include agentic “knowledge discovery" style evaluations; DebFlow (Su et al., 2025) includes interactive task environments; and EvoAgentX (Wang et al., 2025b) assesses performance on real-world agent benchmarks.

Information Retrieval. A smaller but important set of MAWO studies target information retrieval, using agentic pipelines to coordinate stages such as query understanding, candidate retrieval, reranking, evidence consolidation, and final recommendation generation, while optimizing end-to-end system utility rather than the output of any single component. OPTIMAS (Wu et al., 2025) treats product recommendation and retrieval augmented QA as compound systems and improves end-to-end utility by jointly optimizing heterogeneous components. FlowReasoner (Gao et al., 2025) focuses on query level multi agent workflows and uses execution feedback to refine retrieval and reasoning. Recent agentic RAG pipelines surveyed in (Singh et al., 2025) also adopt multi agent decompositions to improve robustness in complex information retrieval.

Table 2: Commonly Used Evaluation Benchmarks for MAWO.

Domain	Datasets	Methods
Knowledge-intensive Question Answering	DROP (Dua et al., 2019)	AFlow (Zhang et al., 2024d); ScoreFlow (Wang et al., 2025c); DebFlow (Su et al., 2025); W4S (Nie et al., 2025); ADAS (Hu et al., 2024)
	HotpotQA (Yang et al., 2018)	Reflexion (Shinn et al., 2023); AFlow (Zhang et al., 2024d); ScoreFlow (Wang et al., 2025c); DebFlow (Su et al., 2025); EvoAgentX (Wang et al., 2025b); OPTIMAS (Wu et al., 2025)
	Natural Questions (NQ)	EvoAgentX (Wang et al., 2025b)
	StrategyQA (Geva et al., 2021)	PromptBreeder (Fernando et al., 2024)
	CommonsenseQA (Talmor et al., 2019)	PromptBreeder (Fernando et al., 2024)
	PubMedQA (Jin et al., 2019)	OPTIMAS (Wu et al., 2025)
	MMLU (Hendrycks et al., 2021a,b)	GPTSwarm (Zhuge et al., 2024); G-Designer (Zhang et al., 2024c); ARG-Designer (Li et al., 2025b)
	MMLU-Pro (Wang et al., 2024b)	W4S (Nie et al., 2025)
	GPQA (Rein et al., 2024)	W4S (Nie et al., 2025)
Software Engineering	HumanEval (Chen et al., 2021)	AFlow (Zhang et al., 2024d); G-Designer (Zhang et al., 2024c); GPTSwarm (Zhuge et al., 2024); FlowReasoner (Gao et al., 2025); SEW (Liu et al., 2025); ScoreFlow (Wang et al., 2025c); W4S (Nie et al., 2025); Reflexion (Shinn et al., 2023); DebFlow (Su et al., 2025); EvoAgentX (Wang et al., 2025b); MaAS (Zhang et al., 2025b)
	MBPP (Austin et al., 2021)	AFlow (Zhang et al., 2024d); ScoreFlow (Wang et al., 2025c); DebFlow (Su et al., 2025); SEW (Liu et al., 2025); FlowReasoner (Gao et al., 2025); W4S (Nie et al., 2025); EvoAgentX (Wang et al., 2025b)
	LeetCodeHardGym (Shinn et al., 2023)	Reflexion (Shinn et al., 2023)
	LiveCodeBench (Jain et al., 2024)	SEW (Liu et al., 2025); EvoAgentX (Wang et al., 2025b)
	BigCodeBench (Zhuo et al., 2024)	FlowReasoner (Gao et al., 2025)
	VerilogEval (Liu et al., 2023a)	VFlow (Wei et al., 2025)
Interactive Decision-Making	ALFWorld (Shridhar et al., 2020)	Reflexion (Shinn et al., 2023); DebFlow (Su et al., 2025)
Mathematical Reasoning	GSM8K (Cobbe et al., 2021)	PromptBreeder (Fernando et al., 2024); AFlow (Zhang et al., 2024d); ARG-Designer (Li et al., 2025b); ScoreFlow (Wang et al., 2025c); G-Designer (Zhang et al., 2024c); EvoAgentX (Wang et al., 2025b); W4S (Nie et al., 2025); ADAS (Hu et al., 2024); MaAS (Zhang et al., 2025b)
	GSM-Hard (Gao et al., 2022)	W4S (Nie et al., 2025); ADAS (Hu et al., 2024)
	GSM-Plus (Li et al., 2024a)	W4S (Nie et al., 2025)
	MGSM (Cobbe et al., 2021; Shi et al., 2022)	W4S (Nie et al., 2025); ADAS (Hu et al., 2024)
	SVAMP (Patel et al., 2021)	PromptBreeder (Fernando et al., 2024); W4S (Nie et al., 2025)
	MultiArith (Roy and Roth, 2015)	PromptBreeder (Fernando et al., 2024)
	AddSub (Hosseini et al., 2014)	PromptBreeder (Fernando et al., 2024)
	SingleEq (Koncel-Kedziorski et al., 2015)	PromptBreeder (Fernando et al., 2024)
	AQuA-RAT (Ling et al., 2017)	PromptBreeder (Fernando et al., 2024)
	MATH (Hendrycks et al., 2021c)	AFlow (Zhang et al., 2024d); ScoreFlow (Wang et al., 2025c); DebFlow (Su et al., 2025); W4S (Nie et al., 2025); EvoAgentX (Wang et al., 2025b); MaAS (Zhang et al., 2025b)
Information Retrieval	Amazon MMLU (Jin et al., 2024)	OPTIMAS (Wu et al., 2025)
	STARK-PRIME (Wu et al., 2024)	OPTIMAS (Wu et al., 2025)
Finance	NCEN-QA (Zeng et al., 2023)	FlowMind (Zeng et al., 2023)