A Comprehensive Survey of Self-Evolving AI Agents

A New Paradigm Bridging Foundation Models and Lifelong Agentic Systems

Jinyuan Fang[*1], **Yanwen Peng**[*2], **Xi Zhang**[*1], **Yingxu Wang**[*3], **Xinhao Yi**[1], **Guibin Zhang**[4], **Yi Xu**[5], **Bin Wu**[6], **Siwei Liu**[7], **Zihao Li**[1], **Zhaochun Ren**[8], **Nikos Aletras**[2], **Xi Wang**[2], **Han Zhou**[5], **Zaiqiao Meng**[1]✉

[1]University of Glasgow, [2]University of Sheffield, [3]Mohamed bin Zayed University of Artificial Intelligence, [4]National University of Singapore, [5]University of Cambridge, [6]University College London, [7]University of Aberdeen, [8]Leiden University

[*]Equal Contributor, ✉Corresponding Author

Recent advances in large language models (LLMs) have sparked growing interest in AI agents capable of solving complex, real-world tasks. However, most existing agent systems rely on manually crafted configurations that remain static after deployment, limiting their ability to adapt to dynamic and evolving environments. To address this limitation, recent research has explored agent *evolution* techniques that aim to automatically enhance agent systems based on interaction data and environmental feedback. This emerging direction lays the foundation for *self-evolving AI agents*, which bridge the static capabilities of foundation models with the continuous adaptability required by *lifelong agentic systems*. In this survey, we provide a comprehensive review of existing techniques for self-evolving agentic systems. Specifically, we first introduce a *unified conceptual framework* that abstracts the feedback loop underlying the design of self-evolving agentic systems. The framework highlights four key components: *System inputs*, *Agent System*, *Environment*, and *Optimisers*, serving as a foundation for understanding and comparing different strategies. Based on this framework, we systematically review a wide range of self-evolving techniques that target different components of the agent system, including foundation models, agent prompts, memory, tools, workflows, and communication mechanisms across agents. We also investigate domain-specific evolution strategies developed for specialised fields such as biomedicine, programming, and finance, where agent behaviour and optimisation objectives are tightly coupled with domain constraints. In addition, we provide a dedicated discussion on the *evaluation, safety, and ethical considerations* for self-evolving agentic systems, which are critical to ensuring their effectiveness and reliability. This survey aims to provide researchers and practitioners with a systematic understanding of self-evolving AI agents, laying the foundation for the development of more adaptive, autonomous, and lifelong agentic systems.

Github: https://github.com/EvoAgentX/Awesome-Self-Evolving-Agents

1 Introduction

Recent progress in large language models (LLMs) has significantly advanced the development of artificial intelligence (AI). Owing to progress in large-scale pretraining, supervised fine-tuning and reinforcement learning, LLMs have demonstrated remarkable capabilities in planning, reasoning, and natural language understanding (Zhao et al., 2023; Grattafiori et al., 2024; Yang et al., 2025a; Guo et al., 2025). These advances have sparked growing interest in *LLM-based agents* (a subclass of AI agents in which an LLM serves as the decision/policy module) (Wang et al., 2024c; Luo et al., 2025a), which are *autonomous systems that leverage LLMs as the core reasoning components for understanding inputs, planning actions, and generating outputs in open-ended, real-world environments* (Wang et al., 2024c; Xi et al., 2025; Luo et al., 2025a). A typical AI agent consists of several components that enable it to perform complex, goal-oriented tasks in an autonomous manner. The foundation model (e.g. an LLM) is the core, responsible for interpreting goals, making plans, and executing actions. To support these capabilities, additional modules, such as perception (Shridhar et al., 2021; Zheng et al., 2024), planning (Yao et al., 2023a,b; Besta et al., 2024), memory (Modarressi et al., 2023; Zhong et al., 2024), and tools (Schick et al., 2023; Gou et al., 2024; Liu et al., 2025d), are integrated to help the agent perceive inputs, decompose tasks, retain contextual information, and interact with tools (Wang et al., 2024c).

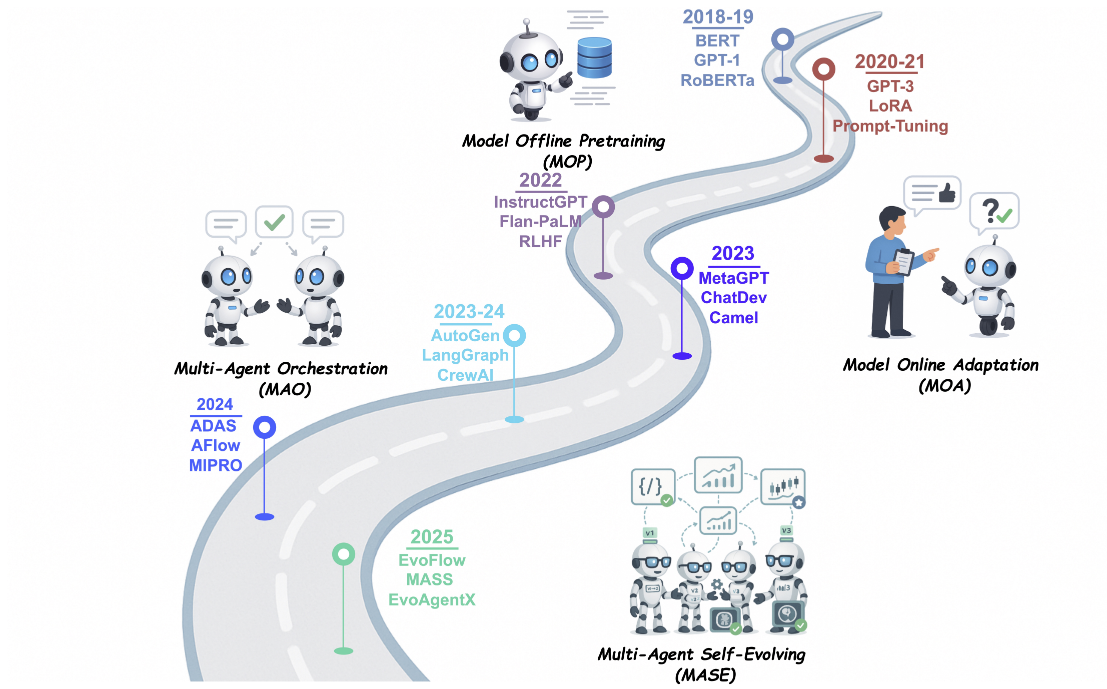


Figure 1 LLM-centric learning is evolving from learning purely from static data, to interacting with dynamic environments, and ultimately towards lifelong learning through multi-agent collaboration and self-evolution.

While single-agent systems have demonstrated strong generalisation and adaptability in various tasks, they often struggle with task specialisation and coordination in dynamic and complex environments (Wu et al., 2024a; Qian et al., 2024). These limitations have led to the development of multi-agent systems (MAS) (Hong et al., 2024; Guo et al., 2024c; Zhou et al., 2025a), where multiple agents collaborate to solve complex problems. Compared with single-agent systems, MAS enables functional specialisation, with each agent designed for a specific subtask or domain of expertise. Moreover, agents can interact, exchange information, and coordinate their behaviour to achieve shared goals. Such collaboration enables the system to tackle tasks beyond the capability of a single agent, while simulating more realistic, dynamic, and interactive environments. LLM-based agent systems have been successfully applied to a wide range of real-world tasks, ranging from code generation (Jiang et al., 2024), scientific research (Lu et al., 2024a), web navigation (Lai et al., 2024a), to domain-specific applications in biomedicine (Kim et al., 2024) and finance (Tian et al., 2025).

Despite the notable progress in agent systems, most of them, whether single- or multi-agent, continue to rely extensively on manually designed configurations. Once deployed, these systems typically maintain static architectures and fixed functionalities. However, real-world environments are dynamic and continuously evolving, e.g., user intents shift, task requirements change, and external tools or information sources may vary over time. For instance, an agent assisting in customer service may need to handle newly introduced products, updated company policies, or unfamiliar user intents. Similarly, a scientific research assistant may be required to incorporate a newly published algorithm, or integrate a novel analysis tool. In such settings, manually reconfiguring the agent system is time-consuming, labour-intensive, and difficult to scale.

These challenges have motivated recent efforts to explore the new paradigm of **Self-Evolving AI Agents**, a novel class of agent systems capable of autonomous adaptation and continuous self-improvement, bridging foundation models with lifelong learning agentic systems.

Definition

Self-evolving AI agents are autonomous systems that continuously and systematically optimise their internal components through interaction with environments, with the goal of adapting to changing tasks, contexts and resources while preserving safety and enhancing performance.

Inspired by Isaac Asimov's *Three Laws of Robotics*[1], we propose a set of guiding principles for safe and effective self-evolution of AI agents:

Three Laws of Self-Evolving AI Agents

I. *Endure (Safety Adaptation)*
Self-evolving AI agents must maintain safety and stability during any modification;

II. *Excel (Performance Preservation)*
Subject to the First law, self-evolving AI agents must preserve or enhance existing task performance;

III. *Evolve (Autonomous Evolution)*
Subject to the First and Second law, self-evolving AI agents must be able to autonomously optimise their internal components in response to changing tasks, environments, or resources.

We characterise the emergence of self-evolving AI agents as part of a broader paradigm shift in the development of LLM-based systems. This shift spans from early-stage Model Offline Pretraining (MOP) and Model Online Adaptation (MOA), to more recent trends in Multi-Agent Orchestration (MAO), and ultimately, to Multi-Agent Self-Evolving (MASE). As summarised in Figure 1 and Table 1, each paradigm builds on the previous one, moving from a static, frozen foundation model to fully autonomous, self-evolving agentic systems.

- **MOP (Model Offline Pretraining)**. The initial stage focuses on pretraining foundation models on large-scale, static corpora and then deploying them in a fixed, frozen state, without further adaptation.
- **MOA (Model Online Adaptation)**. Building on MOP, this stage introduces post-deployment adaptation, where the foundation models can be updated through techniques such as supervised fine-tuning, low-rank adapters (Pfeiffer et al., 2021; Hu et al., 2022), or reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022), using labels, ratings, or instruction prompts.
- **MAO (Multi-Agent Orchestration)**. Extending beyond a single foundation model, this stage coordinates multiple LLM agents that communicate and collaborate via message exchange or debate prompts (Li et al., 2024g; Zhang et al., 2025h), to solve complex tasks without modifying the underlying model parameters.
- **MASE (Multi-Agent Self-Evolving)**. Finally, MASE introduces a lifelong, self-evolving loop where a population of agents continually refines their prompts, memory, tool-use strategies and even their interaction patterns based on environmental feedback and meta-rewards (Novikov et al., 2025; Zhang et al., 2025i).

The evolution from MOP to MASE represents a fundamental shift in the development of LLM-based systems, from static, manually configured architectures to adaptive, data-driven systems that can evolve in response to changing requirements and environments. *Self-evolving AI agents* bridge the static capabilities of foundation models with the continuous adaptability required by *lifelong agentic systems*, offering a path toward more autonomous, resilient, and sustainable AI.

Despite self-evolving AI agents representing an ambitious vision for future AI systems, achieving this level of autonomy remains a long-term goal. Current systems are still far from exhibiting the full capabilities required for safe, robust and open-ended self-evolution. In practice, current progress towards this vision is achieved through *agent evolution and optimisation* techniques, which provide practical means for enabling agent systems to iteratively refine their components based on interaction data and environmental feedback, thereby enhancing their effectiveness in real-world tasks. Recent research has explored several key directions in this area. One line of work focuses on enhancing the underlying LLM itself to improve the core capabilities, such as planning (Qiao et al., 2024), reasoning (Zelikman et al., 2022; Tong et al., 2024), and tool use (Feng et al., 2025a). Another line of research targets the optimisation of auxiliary components within agent systems, including prompts (Xu et al., 2022; Prasad et al., 2023; Yang et al., 2024a; Wang et al., 2025i), tools (Yuan et al., 2025b; Qu et al., 2025),

[1]Introduced in his stories "Runaround" (1942) and "I, Robot" (1950). These laws are hierarchical: the Second cannot override the First, and the Third cannot override the First or Second. Although conceived as fictional moral constraints, they have become influential in AI ethics research. Therefore, we articulate the "Three Laws of Self-Evolving AI Agents", advocating that AI agents, as the core of embodied AI, prioritise compliance and safety before pursuing autonomous evolution.

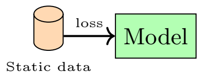
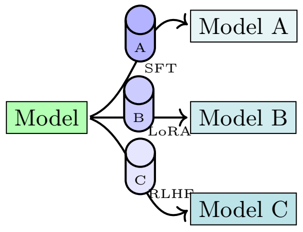
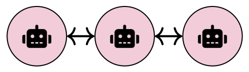
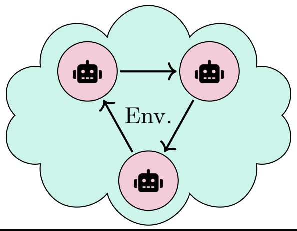
Paradigm	Interaction & Feedback	Key Techniques	Diagram
Model Offline Pretraining (MOP)	Model ⇔ Static data (loss/backprop)	• Transformer Pretraining (Causal LM, Masked LM, NSP) • BPE / SentencePiece • MoE & Pipeline Parallelism	
Model Online Adaptation (MOA)	Model ⇔ Supervision (labels/scores/rewards)	• Task Fine-tuning • Instruction Tuning • LoRA / Adapters / Prefix-Tuning • RLHF (RLAIF, DPO, PPO) • Multi-Modal Alignment • Human Alignment	 Model A A SFT Model B LoRA Model B C RLHF Model C
Multi-Agent Orchestration (MAO)	Agent$_1$ ⇔ Agent$_2$ (message exchange)	• Multi-Agent Systems • Self-Reflection • Multi-Agent Debate • Chain-of-Thought Ensemble • Function / Tool Calling / MCP	
Multi-Agent Self-Evolving (MASE)	Agents ⇔ Environment (signals from env.)	• Behaviour Optimisation • Prompt Optimisation • Memory Optimisation • Tool Optimisation • Agentic Workflow Optimisation	

Table 1 Comparison of four LLM-centric learning paradigms – Model Offline Pretraining (MOP), Model Online Adaptation (MOA), Multi-Agent Orchestration (MAO), and Multi-Agent Self-Evolving (MASE), highlighting each paradigm's interaction & feedback mechanisms, core techniques, and illustrative diagrams to trace the progression from static model training to dynamic, autonomous agent evolution.

memory (Zhong et al., 2024; Lee et al., 2024d), and etc., allowing the agents to better generalise to new tasks and dynamic environments. Furthermore, in multi-agent systems, recent work investigates the optimisation of agent topologies and communication protocols (Bo et al., 2024; Chen et al., 2025h; Zhang et al., 2025j; Zhou et al., 2025a), aiming to identify agent structures that are best suited to the current task and improve the coordination and information sharing among agents.

Existing surveys on AI agents either focus on the general introduction of agent architectures and functionalities (Wang et al., 2024c; Guo et al., 2024c; Xi et al., 2025; Luo et al., 2025a; Liu et al., 2025a,c), or target specific components such as planning (Huang et al., 2024b), memory (Zhang et al., 2024d), collaboration mechanism (Tran et al., 2025), and evaluation (Yehudai et al., 2025). Other surveys investigate domain-specific applications of agents, such as operating system agents (Hu et al., 2025b) and healthcare agents (Sulis et al., 2023). While these surveys provide valuable insights into various aspects of agent systems, recent advances in agent self-evolution and continual adaptation have not been sufficiently covered, which corresponds to the capabilities of agents that are central to the development of lifelong, autonomous AI systems. This leaves a critical gap in the literature for researchers and practitioners seeking a holistic understanding of the new learning paradigm that underpins adaptive and self-evolving agentic systems.

To bridge this gap, this survey provides a focused and systematic review of techniques that enable agents to evolve and improve themselves based on interaction data and environmental feedback. Specifically, we introduce a *unified conceptual framework* that abstracts the feedback loop underlying the design of self-evolving agentic systems. This framework identifies four core components: *System Inputs*, *Agent System*, *Environment*, and *Optimisers*, highlighting the evolution loop of agent systems. Building on this framework, we systematically examine a wide range of evolution and optimisation techniques that target different components of the agent systems, including the LLM, prompts, memory, tools, workflow topologies, and communication mechanisms. Moreover, we also investigate domain-specific evolution strategies developed for specialised fields. In addition, we provide a dedicated discussion on the *evaluation*, *safety*, and *ethical considerations* for self-evolving agentic

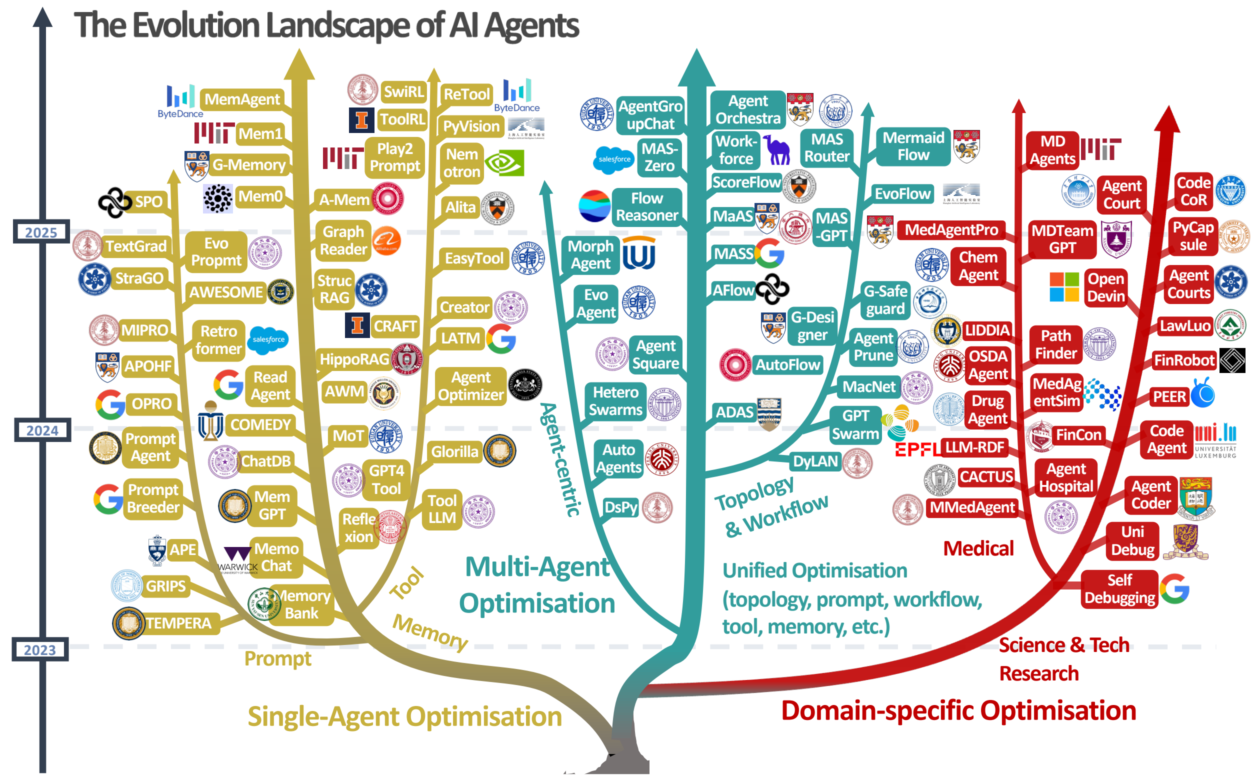


Figure 2 A visual taxonomy of AI agent evolution and optimisation techniques, categorised into three major directions: single-agent optimisation, multi-agent optimisation, and domain-specific optimisation. The tree structure illustrates the development of these approaches from 2023 to 2025, including representative methods within each branch.

systems, which are critical to ensuring their effectiveness and reliability. As a concurrent work, Gao et al. (2025b) surveys self-evolving agents organised around three foundational dimensions: what to evolve, when to evolve, and how to evolve. While their taxonomy offers valuable insights, our survey aims to provide a more comprehensive and integrative perspective, i.e., the unified conceptual framework, on the mechanisms and challenges associated with building lifelong, self-evolving agentic systems.

This survey aims to provide a comprehensive and systematic review of existing techniques for self-evolving agentic systems, thereby offering researchers and practitioners valuable insights and guidelines for developing more effective and sustainable agentic systems. Figure 2 presents a visual taxonomy of existing agent evolution strategies across single-agent, multi-agent, and domain-specific optimisation, highlighting representative approaches in each direction. Our main contributions are as follows:

- We formalise the Three Laws of Self-Evolving AI Agents and map the evolution of LLM-centric learning paradigms from static pretraining to fully autonomous, lifelong self-evolving agentic systems.
- We introduce a unified conceptual framework that abstracts the feedback loop underlying self-evolving agentic systems, and provides a foundation for systematically understanding and comparing different evolution and optimisation approaches.
- We conduct a systematic review of existing evolution and optimisation techniques across single-agent, multi-agent, and domain-specific settings.
- We provide a comprehensive review of evaluation, safety, and ethical considerations for self-evolving agentic systems, emphasising their critical role in ensuring the effectiveness, safety, and responsible deployment of these systems.
- We identify key open challenges and outline promising research directions in agent self-evolution, aiming to facilitate future exploration and advance the development of more adaptive, autonomous, and self-evolving

agentic systems.

The remainder of this survey is organised as follows. Section 2 presents preliminaries on AI agents and multi-agent systems, including their definitions, key components, representative architectures, and the broader vision of autonomous and self-evolving agent systems. Section 3 introduces a unified conceptual framework for agent evolution approaches, outlining the key elements such as system inputs, evolution objectives, agent structures, and optimisers. Section 4 focuses on the optimisation of single-agent systems. It discusses several key aspects such as the optimisation of reasoning strategies, prompt formulation, memory mechanisms, and tool usage. Section 5 focuses on multi-agent systems and review methods for optimising agent workflows, topologies, and inter-agent communication strategies. Section 6 highlights domain-specific agent optimisation techniques and their applications, while Section 7 discusses evaluation methodologies and benchmarks for assessing agent systems. Section 8 presents existing challenges in the agent evolution and optimisation field and outlines some promising future research directions. Finally, we conclude the survey in Section 9.

2 Foundation of AI Agent Systems

To facilitate a clear understanding of agent evolution and optimisation, this section provides an overview of existing AI agent systems. We begin by introducing single-agent systems in Section 2.1, outlining their definitions and core components. We then turn to multi-agent systems (MAS) in Section 2.2, highlighting their motivations, structural paradigms, and collaboration mechanisms. Finally, we present the vision of lifelong, self-evolving agentic systems in Section 2.3.

2.1 AI Agents

An AI agent refers to an autonomous system capable of perceiving its inputs, reasoning about goals, and interacting with the environment to complete tasks (Luo et al., 2025a). In this section, we focus on single-agent systems, which serve as the foundation of AI agent research. While our goal here is to provide only a brief overview, readers may refer to existing surveys for more comprehensive discussions of AI agent architectures and capabilities (Guo et al., 2024c; Xi et al., 2025; Luo et al., 2025a; Liu et al., 2025a).

An AI agent is typically composed of multiple components that work together to enable autonomous decision-making and execution. The core component of an agent is the **Foundation Model**, most commonly an **LLM**[2], which serves as the central reasoning engine responsible for interpreting instructions, generating plans, and producing actionable responses. In addition, there are also some supporting modules that enhance the agent's ability in complex and dynamic environments:

(1) **Perception Module.** The perception module is responsible for acquiring and interpreting information from the environment (Li et al., 2024f). This includes processing textual inputs, audio signals, video frames, or other sensory-like data to build a representation suitable for reasoning.

(2) **Planning Module.** The planning module enables the agent to decompose complex tasks into actionable sub-tasks or sequences of operations and guide their execution across multiple steps (Huang et al., 2024b). This process facilitates hierarchical reasoning and ensures coherent task completion. One of the simplest forms of planning involves linear task decomposition, where a problem is broken down into multiple intermediate steps, and the LLM follows these steps to address the problem. This is exemplified by methods such as chain-of-thought prompting (Wei et al., 2022). Beyond static planning, more dynamic approaches interleave planning and execution in an iterative loop. For instance, the ReAct (Yao et al., 2023b) framework combines reasoning with actions, allowing the agent to revise its plans based on real-time feedback. In addition to linear planning, some methods adopt a branching strategy, where each step may lead to multiple possible continuations. Representative examples are Tree-of-Thought (Yao et al., 2023a) and Graph-of-Thought (Besta et al., 2024), which enable the agent to explore multiple reasoning paths.

(3) **Memory Module.** The memory module enables the agent to retain and recall past experience, enabling context-aware reasoning and long-term consistency. Broadly, memory can be categorised into short-term and long-term memory. Short-term memory typically stores the context and interactions generated during

[2]While this survey focuses on LLMs, the backbone can be any foundation model (e.g., vision–language models, protein sequence/structure models), and the core agentic principles we discuss readily generalise to such backbones.

the execution of the current task. Once the task is completed, the short-term memory will be removed. In contrast, long-term memory persists over time and may store accumulated knowledge, past experiences, or reusable information across tasks. To access relevant long-term memory, many agent systems adopt a retrieval-augmented generation (RAG) module (Zhang et al., 2024d), where the agent retrieves relevant information from the memory and incorporates them into the input context for the LLM. Designing an effective memory module involves several challenges, including how to structure memory representations, when and what to store, how to retrieve relevant information efficiently, and how to integrate it into the reasoning process Zeng et al. (2024a). For a more comprehensive review of memory mechanisms in AI agents, we refer readers to the survey by Zhang et al. (2024d).

(4) **Tool Use.** The ability to use external tools is a key factor for AI agents to effectively operate in real-world scenarios. While LLMs are powerful in language understanding and generation, their capabilities are inherently limited by their static knowledge and reasoning capabilities. By using external tools, agents can extend their functional scope, allowing them to better interact with real-world environments. Typical tools include web search engines (Li et al., 2025c), code interpreters or execution environments (Islam et al., 2024), and browser automation framework (Müller and Žunič, 2024). The design of the tool-use component often involves selecting tools, constructing tool-specific inputs, invoking API calls, and integrating tool outputs back into the reasoning process.

2.2 Multi-Agent Systems

While single-agent systems have demonstrated strong capabilities in various tasks, many real-world tasks demand specialisation and coordination that exceed the capabilities of a single agent. This limitation has motivated the development of *Multi-Agent Systems* (MAS), which mirror the distributed intelligence found in biological and social systems.

MAS are formally defined as a collection of autonomous agents that interact within a shared environment to achieve goals that are beyond the capabilities of a single agent. In contrast to single-agent systems that rely solely on individual reasoning and capabilities, MAS focuses on achieving collective intelligence through structured coordination and collaboration among different agents (Tran et al., 2025). A fundamental mechanism enabling such coordination is the concept of *agent topology*, the structural configuration that defines how agents are connected and communicate within the system. The topology determines the information flow and collaboration strategies among agents, directly influencing how tasks are distributed and executed. Therefore, MAS is often realised as a *multi-agent workflow*, where the system's topology orchestrates the interactions among agents to accomplish complex, shared goals. The key insight is that when multiple agents collaborate through such workflows, the system's overall performance can exceed the sum of the individual capabilities of all agents within the system (Lin et al., 2025; Luo et al., 2025a).

MAS brings several notable advantages over single-agent systems. First, MAS can decompose complex tasks into manageable sub-tasks and assign them to specialised agents, which is helpful to improve the overall performance (Krishnan, 2025; Sarkar and Sarkar, 2025). This approach mirrors human organisational collaboration, enabling MAS to handle tasks that are beyond the capacity of a single agent. Second, MAS supports parallel execution, allowing multiple agents to work simultaneously to complete the task. This feature is particularly advantageous for time-sensitive applications, as it greatly accelerates the problem-solving process (Zhang et al., 2025k; Liu et al., 2025a; Li et al., 2025d). Third, the decentralized nature of MAS enhances robustness: when one agent fails, other agents can dynamically redistribute tasks and compensate for the failure, ensuring graceful degradation rather than a complete system breakdown (Huang et al., 2024a; Yang et al., 2025b). Fourth, MAS offers inherent scalability, as new agents can be seamlessly integrated without redesigning the entire system (Han et al., 2024; Chen et al., 2025g). Finally, collaborative mechanisms like debate and iterative refinement enable MAS to generate more innovative and reliable solutions by leveraging diverse perspectives and critical evaluation among agents (Guo et al., 2024c; Lin et al., 2025). Frameworks such as CAMEL and AutoGen have further streamlined the development of MAS by providing modular architectures, role-playing patterns, and automated orchestration capabilities that reduce engineering overhead (Li et al., 2023a; Wu et al., 2024a).

2.2.1 System Architecture

The architectural design of MAS fundamentally determines how agents organise, coordinate, and execute tasks. These structures range from rigid hierarchies to flexible peer-to-peer networks, each embodying different philosophies about control, autonomy, and collaboration.

(1) **Hierarchical Structure.** These systems employ static hierarchical organisations, typically linear or tree-based, where tasks are explicitly decomposed and sequentially assigned to specific agents. For instance, MetaGPT (Hong et al., 2024) introduces Standard Operating Procedures (SOPs) to streamline software development, while HALO (Hou et al., 2025) incorporates Monte Carlo Tree Search to enhance reasoning performance. This highly customised approach offers modularity, ease of development, and domain-specific optimisation, making it prevalent in software development, medicine, scientific research, and social sciences (Zheng et al., 2023b; Park et al., 2023; Qian et al., 2024; Li et al., 2024c; Cheng et al., 2025).

(2) **Centralised Structure.** This architecture follows a manager-follower paradigm where a central agent or higher-level coordinator handles planning, task decomposition, and delegation, while subordinate agents execute assigned subtasks. This design effectively balances global planning with specific task execution (Fourney et al., 2024; Roucher et al., 2025; CAMEL-AI, 2025). However, the central node creates performance bottlenecks and introduces single-point-of-failure vulnerabilities that compromise system robustness (Ko et al., 2025).

(3) **Decentralised Structure.** In this architecture, agents collaborate as peers in a distributed network, widely adopted in world simulation applications. The absence of central control prevents single-point failures—damage to any node does not paralyse the entire system, eliminating bottlenecks and enhancing robustness (Lu et al., 2024b; Yang et al., 2025b). However, this introduces challenges in information synchronisation, data security, and increased collaboration costs (Ko et al., 2025). Recent work explores blockchain technology to address these coordination challenges (Geren et al., 2024; Yang et al., 2025d).

2.2.2 Communication Mechanisms

The effectiveness of MAS largely depends on how agents exchange information and coordinate actions. Communication methods in MAS have evolved from simple message passing to sophisticated protocols that balance expressiveness, efficiency, and interoperability.

(1) **Structured Output.** This approach employs formats like JSON (Li et al., 2024e; Chen et al., 2025g), XML (Zhang et al., 2025b; Kong et al., 2025), and executable code (Roucher et al., 2025) for inter-agent communication. The explicit structure and well-defined parameters ensure high machine readability and interpretability, while standardised formats facilitate cross-platform collaboration (Chen et al., 2025g). These characteristics make structured communication ideal for applications demanding precision and efficiency, such as problem-solving and reasoning tasks. The compact information representation further enhances computational efficiency (Wang et al., 2024h).

(2) **Natural Language.** Natural language communication preserves rich contextual and semantic details, making it particularly suitable for creative tasks, world simulation, and creative writing scenarios (Liu et al., 2025a). This expressiveness enables nuanced interactions that capture subtle meanings and intentions. However, it introduces challenges including ambiguity, potential misinterpretation, and reduced execution efficiency compared to structured formats (Guo et al., 2024c; Yang et al., 2025c; Kong et al., 2025).

(3) **Standardised Protocols.** Recent advances have introduced specialised protocols designed to standardise MAS communication, creating more inclusive and interoperable agent ecosystems: A2A (LLC and Contributors) standardises horizontal communication through a structured, peer-to-peer task delegation model, enabling agents to collaborate on complex, long-running tasks while maintaining execution opacity. ANP (Chang and Contributors) implements secure, open horizontal communication for a decentralised "agent internet" through a hierarchical architecture with built-in Decentralised Identity (DID) and dynamic protocol negotiation. MCP (PBC and Contributors) standardises vertical communication between individual agents and external tools or data resources through a unified client-server interface. Agora (Marro and Contributors) functions as a meta-protocol for horizontal communication, enabling agents to dynamically negotiate and evolve their communication methods, seamlessly switching between flexible natural language and efficient structured routines.

2.3 The Vision of Lifelong, Self-Evolving Agentic Systems

The trajectory from Model Offline Pretraining (MOP) through Model Online Adaptation (MOA) and Multi-Agent Orchestration (MAO) has steadily reduced the degree of manual configuration in LLM-based systems. Yet, even the most advanced multi-agent frameworks today often depend on handcrafted workflows, fixed communication protocols, and human-curated toolchains (Talebirad and Nadiri, 2023; Zhao et al., 2024; Luo et al., 2025a; Tran et al., 2025). These static elements constrain adaptability, making it difficult for agents to sustain performance in dynamic, open-ended environments where requirements, resources, and goals evolve over time.

The emerging paradigm of Multi-Agent Self-Evolving (MASE) systems addresses these limitations by closing the loop between deployment and continual improvement. In a MASE system, a population of agents is equipped to autonomously refine their prompts, memory, tool-use strategies, and even their interaction topology – guided by feedback from the environment and higher-level meta-rewards (Novikov et al., 2025; Zhang et al., 2025i). This continuous optimisation process enables agents not merely to adapt once, but to evolve over their lifetime in response to shifting tasks, domains, and operational constraints.

Lifelong, self-evolving agentic systems aim to overcome these constraints by embedding a continuous improvement loop into the core of the architecture. Guided by the Three Laws of Self-Evolving AI Agents – Endure (safety adaptation), Excel (performance preservation), and Evolve (autonomous optimisation) – these systems are designed to:

(I) Monitor their own performance and safety profile during operation;

(II) Preserve or enhance capabilities through controlled, incremental updates;

(III) Autonomously adapt prompts, memory structures, tool-use strategies, and even inter-agent topologies in response to shifting tasks, environments, and resources.

Rather than requiring human designers to handcraft every interaction pattern, a lifelong self-evolving system can generate, evaluate, and refine its own agentic configurations, closing the loop between environment feedback, meta-level reasoning, and structural adaptation. This transforms agents from static executors into continually learning, co-evolving participants in their operational ecosystems.

This vision has far-reaching implications. In scientific discovery, self-evolving agent ecosystems could autonomously generate hypotheses, design experiments, and iterate on research workflows. In software engineering, they could co-evolve development pipelines, integrating new tools as they emerge. In human–AI collaboration, they could learn individual preferences and continually personalise interaction styles. Extending beyond the digital realm, such systems could interface with the physical world through robotics, IoT devices, and cyber–physical infrastructures, perceiving environmental changes, acting upon them, and incorporating real-world feedback into their evolutionary loop. By treating agents as reconfigurable computational entities capable of self-evolving, coordination, and long-term adaptation, MASE offers a pathway toward scalable, sustainable, and trustworthy AI – AI that is not just trained once, but that *lives*, *learns*, and *lasts*.

3 A Conceptual Framework of MASE

To provide a comprehensive overview of self-evolving agentic systems, we propose a high-level conceptual framework that abstracts and summarises the key elements underlying the design and implementation of agent evolution and optimisation methods. This framework provides an abstract yet generalisable view of most existing optimisation approaches, thereby enabling a comprehensive understanding of the field and facilitating comparative analysis across different approaches.

3.1 Overview of the Self-Evolving Process

We begin with an overview of the self-evolving process in agent systems, which in practice is often realised through iterative optimisation. In this process, the agent system is iteratively updated based on feedback signals obtained from performance evaluations and environmental interactions. As illustrated in Figure 3, the process begins with a task specification, which may include a high-level description, input data, contextual information, or concrete examples. These elements constitute the **system inputs**, which define the problem

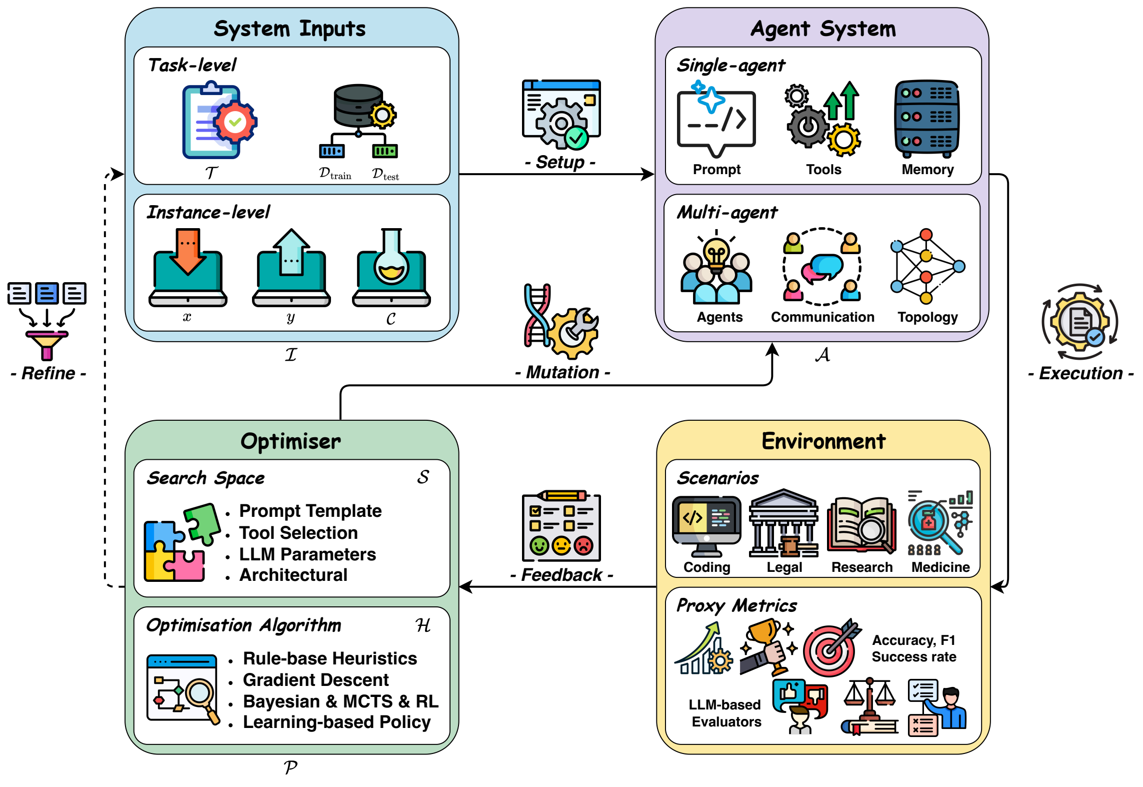


Figure 3 Conceptual framework of the self-evolving process in agent systems. The process forms an iterative optimisation loop comprising four components: *System Inputs, Agent System, Environment*, and *Optimiser*. System inputs define the task setting (e.g., task-level or instance-level). The agent system (in single- or multi-agent form) executes the specified task. The environment (depending on different scenarios) provides feedback via proxy metrics. The optimiser updates the agent system through a defined search space and optimisation algorithm until performance goals are met.

setting for the agent system. The **agent system**, either following a single-agent or multi-agent architecture, is then deployed to perform the task within an **environment**. The environment provides the operating context and generates feedback signals, which are derived from predefined evaluation metrics, that measure the system's effectiveness and guide subsequent optimisation. Based on feedback from the environment, the **optimiser** applies specific algorithms and strategies to update the agent system, such as adjusting the LLM parameters, modifying prompts, or refining the system's structure. In some cases, the optimiser may also refine the system inputs by synthesising training examples to augment existing datasets, thereby expanding the data available for subsequent optimisation cycle. The updated agent system is then redeployed to the environment, initialising the next iteration. This process forms an iterative, closed feedback loop in which the agent system is progressively refined and optimised over multiple iterations. The loop terminates once a predefined performance threshold is reached or convergence criteria are satisfied. Building on the conceptual framework of MASE, **EvoAgentX** is the first open-source framework to apply this self-evolving agent process, designed to automate the generation, execution, evaluation, and optimisation of agentic systems (Wang et al., 2025i).

Building on the overview above, there are four key components within the agent optimisation process: *system inputs*, *agent systems*, *environment* and *optimisers*. In what follows, we provide an introduction to each component, highlighting their individual roles, characteristics and interactions within the optimisation framework.

3.2 System Inputs

System inputs refer to the contextual information and data provided to the optimisation process. Formally, we denote the set of system inputs as $\mathcal{I}$, which may consist of one or more elements that specify task requirements, constraints, and available data. These inputs define the problem setting for the agent system and determine the scope of optimisation. Depending on the scenario, $\mathcal{I}$ can take different forms:

- **Task-Level Optimisation**. The most common setting in existing research focuses on improving the agent system's overall performance on a specific task. In this case, the system inputs $\mathcal{I}$ may include a task description $\mathcal{T}$ and a training dataset $\mathcal{D}_{\text{train}}$ used for training or validation: $\mathcal{I} = \{\mathcal{T}, \mathcal{D}_{\text{train}}\}$. A separate test dataset $\mathcal{D}_{\text{test}}$ can also be incorporated to evaluate the optimised agent's performance. In some scenarios, task-specific labeled data, i.e., $\mathcal{D}_{\text{train}}$, are unavailable. To enable optimisation in such settings, recent approaches (Huang et al., 2025; Zhao et al., 2025a) propose to dynamically *synthesise* training examples, often through LLM-based data generation, to create a surrogate dataset for iterative improvement.
- **Instance-Level Optimisation**. Recent studies also explore a more fine-grained setting, where the objective is to enhance the agent system's performance on a specific example (Sun et al., 2024a; Novikov et al., 2025). In this case, the system inputs may consist of an input-output pair (x, y), along with optional contextual information $\mathcal{C}$, i.e., $\mathcal{I} = \{x, y, \mathcal{C}\}$.

3.3 Agent Systems

The agent system is the core component within the feedback loop that is subject to optimisation. It defines the decision-making process and functionality of the agent(s) in response to given inputs. Formally, we denote the agent system as $\mathcal{A}$, which may consist of a single agent or a collection of collaborating agents. The agent system $\mathcal{A}$ can be further decomposed into several components, such as the underlying LLM, prompting strategy, memory module, tool-use policy, etc. Optimisation methods may focus on one or more of these components depending on the intended scope. In most existing works, optimisation is performed on a single component of $\mathcal{A}$, such as finetuning the LLM to enhance reasoning and planning capabilities (Zelikman et al., 2022; Tong et al., 2024; Lai et al., 2024b), or tuning the prompts and selecting appropriate tools to improve task-specific performance without modifying the LLM itself (Yang et al., 2024a; Yuan et al., 2025b). Moreover, recent research has also explored joint optimisation of multiple components with $\mathcal{A}$. For example, in single-agent systems, some approaches jointly optimise LLM and prompting strategy to better align model behaviour with task requirements (Soylu et al., 2024). In multi-agent systems, existing studies have explored the joint optimisation of prompts and inter-agent topology to improve overall effectiveness (Zhang et al., 2025j; Zhou et al., 2025a).

3.4 Environments

The environment serves as the external context in which the agent system operates and generates outputs. Specifically, the agent system interacts with the environment by perceiving its inputs, executing actions, and receiving corresponding outcomes. Depending on the task, the environment can vary from a benchmark dataset to a fully dynamic, real-world setting (Liu et al., 2023a). For example, in code generation tasks, the environment may include code execution and verification components such as compilers, interpreters, and test cases. In scientific research, it may consist of literature databases, simulation platforms, or laboratory equipment.

Beyond providing the operational context, the environment also plays a critical role in generating *feedback signals* that inform and guide the optimisation process. These signals are typically derived from *evaluation metrics* that quantify the effectiveness or efficiency of the agent system. In most cases, such metrics are task-specific, e.g., accuracy, F1, or success rate, which provide quantitative measures of performance. However, in settings where labelled data or ground-truth are unavailable, LLM-based evaluators are often employed to estimate performance (Yehudai et al., 2025). These evaluators can generate proxy metrics or provide textual feedback by assessing aspects such as correctness, relevance, coherence, or alignment with task instructions. A more detailed discussion of evaluation strategies across different applications is presented in Section 7.

3.5 Optimisers

Optimisers ($\mathcal{P}$) are the core component of the self-evolving feedback loop, responsible for refining the agent system $\mathcal{A}$ based on performance feedback from the environment. Their objective is to search, via specialised algorithms and strategies, for the agent configuration that achieves the best performance under the given evaluation metric. Formally, this can be expressed as:

$$\mathcal{A}^* = \arg\max_{\mathcal{A} \in \mathcal{S}} \mathcal{O}(\mathcal{A}; \mathcal{I}), \tag{1}$$

where $\mathcal{S}$ denotes the search space of configurations, $\mathcal{O}(\mathcal{A}; \mathcal{I}) \in \mathbb{R}$ is the evaluation function that maps the performance of $\mathcal{A}$ on the given system inputs $\mathcal{I}$ to a scalar score, and $\mathcal{A}^*$ denotes the optimal agent configuration.

An optimiser is typically defined by two core components: (1) **search space ($\mathcal{S}$)**: This defines the set of agent configurations that can be explored and optimised. The granularity of $\mathcal{S}$ depends on which part(s) of the agent system are subject to optimisation, ranging from agent prompts or tool selection strategies to continuous LLM parameters or architectural structures. (2) **optimisation algorithm ($\mathcal{H}$)**: This specifies the strategy used to explore $\mathcal{S}$ and select or generate candidate configurations. It can include rule-based heuristics, gradient descent, Bayesian optimisation, Monte Carlo Tree Search (MCTS), reinforcement learning, evolutionary strategies, or learning-based policies. Together, the pair $(\mathcal{S}, \mathcal{H})$ defines the behaviour of the optimiser and determines how efficiently and effectively it can adapt the agent system toward better performance.

In the following sections, we introduce typical optimisers in three different settings: single-agent systems (Section 4), multi-agent systems (Section 5), and domain-specific agent systems (Section 6). Each setting exhibits distinct characteristics and challenges, leading to different designs and implementations of optimisers. In single-agent optimisation, the focus is on improving an individual agent's performance by tuning LLM parameters, prompts, memory mechanisms, or tool-use policies. In contrast, multi-agent optimisation extends the scope to optimising not only individual agents but also their structural designs, communication protocols, and collaboration capabilities. Domain-specific agent optimisation presents additional challenges, where optimisers must account for specialised requirements and constraints inherent to particular domains, leading to tailored optimiser designs. A comprehensive hierarchical categorisation of these optimisation settings and representative methods is provided in Figure 5.

4 Single-Agent Optimisation

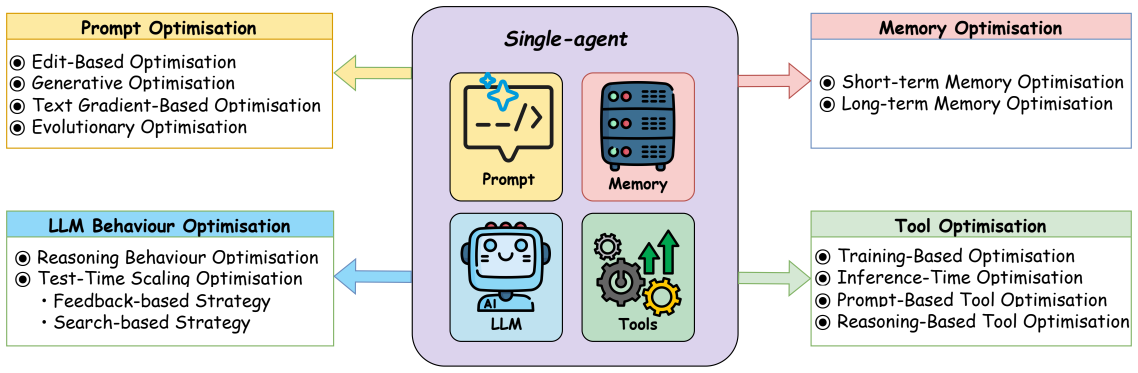

Prompt Optimisation
Edit-Based Optimisation
Generative Optimisation
Text Gradient-Based Optimisation
Evolutionary Optimisation
Single-agent
Prompt
Memory
LLM
Tools
AI
Memory Optimisation
Short-term Memory Optimisation
Long-term Memory Optimisation
LLM Behaviour Optimisation
Reasoning Behaviour Optimisation
Test-Time Scaling Optimisation
• Feedback-based Strategy
• Search-based Strategy
Tool Optimisation
Training-Based Optimisation
Inference-Time Optimisation
Prompt-Based Tool Optimisation
Reasoning-Based Tool Optimisation

Figure 4 An overview of single-agent optimisation approaches, categorised by the targeted component within the agent system: prompt, memory, and tool.

Single-agent optimisation focuses on improving the performance of a single-agent system. According to the optimisation feedback loop introduced earlier, the key challenge lies in the design of optimisers for updating the system. This involves identifying the specific components of the agent system to optimise (i.e., search space), determining the particular capabilities to enhance, and choosing appropriate optimisation strategies to effectively achieve these improvements (i.e., optimisation algorithm).

In this section, we organise single-agent optimisation approaches based on the targeted component within the agent system, as this determines both the structure of the search space and the choice of optimisation methods. Specifically, we focus on four major categories: (1) *LLM Behaviour optimisation*, which aims to improve the LLM's reasoning and planning capabilities through either parameter tuning or test-time scaling techniques; (2) *Prompt optimisation*, which focuses on adapting prompts to guide the LLM towards producing more accurate and task-relevant outputs; (3) *Memory optimisation*, which aims to enhance the agent's ability to store, retrieve, and reason over historical information or external knowledge; (4) *Tool optimisation*, which focuses on enhancing the agent's ability to effectively leverage existing tools, or autonomously create or configure new tools to accomplish complex tasks. Figure 4 shows the major categories of single-agent optimisation approaches.

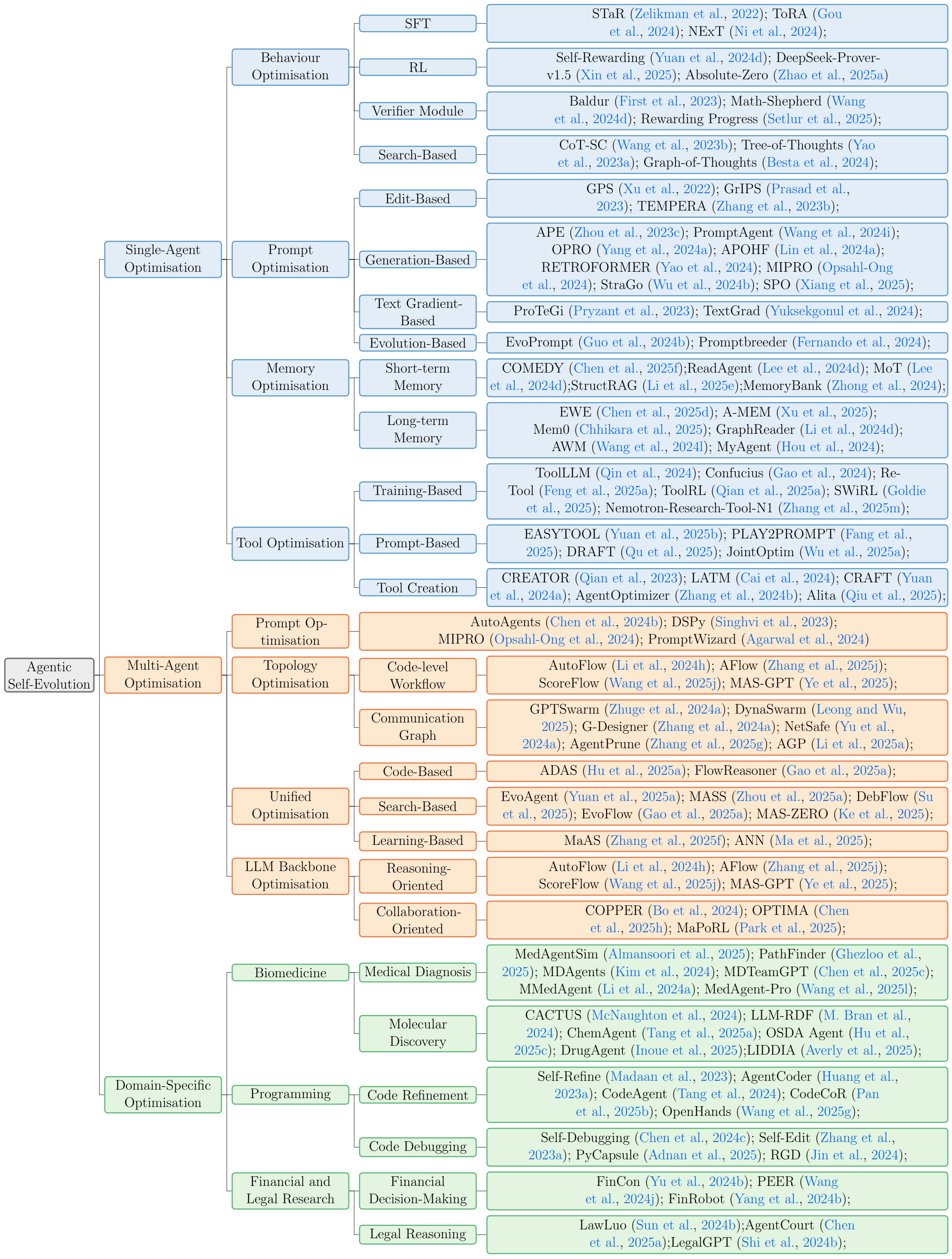

Agentic Self-Evolution
Single-Agent Optimisation
Behaviour Optimisation
SFT
STaR (Zelikman et al., 2022); ToRA (Gou et al., 2024); NExT (Ni et al., 2024);
RL
Self-Rewarding (Yuan et al., 2024d); DeepSeek-Prover-v1.5 (Xin et al., 2025); Absolute-Zero (Zhao et al., 2025a)
Verifier Module
Baldur (First et al., 2023); Math-Shepherd (Wang et al., 2024d); Rewarding Progress (Setlur et al., 2025);
Search-Based
CoT-SC (Wang et al., 2023b); Tree-of-Thoughts (Yao et al., 2023a); Graph-of-Thoughts (Besta et al., 2024);
Prompt Optimisation
Edit-Based
GPS (Xu et al., 2022); GrIPS (Prasad et al., 2023); TEMPERA (Zhang et al., 2023b);
Generation-Based
APE (Zhou et al., 2023c); PromptAgent (Wang et al., 2024i); OPRO (Yang et al., 2024a); APOHF (Lin et al., 2024a); RETROFORMER (Yao et al., 2024); MIPRO (Opsahl-Ong et al., 2024); StraGo (Wu et al., 2024b); SPO (Xiang et al., 2025);
Text Gradient-Based
ProTeGi (Pryzant et al., 2023); TextGrad (Yuksekgonul et al., 2024);
Evolution-Based
EvoPrompt (Guo et al., 2024b); Promptbreeder (Fernando et al., 2024);
Memory Optimisation
Short-term Memory
COMEDY (Chen et al., 2025f);ReadAgent (Lee et al., 2024d); MoT (Lee et al., 2024d);StructRAG (Li et al., 2025e);MemoryBank (Zhong et al., 2024);
Long-term Memory
EWE (Chen et al., 2025d); A-MEM (Xu et al., 2025); Mem0 (Chhikara et al., 2025); GraphReader (Li et al., 2024d); AWM (Wang et al., 2024l); MyAgent (Hou et al., 2024);
Tool Optimisation
Training-Based
ToolLLM (Qin et al., 2024); Confucius (Gao et al., 2024); Re-Tool (Feng et al., 2025a); ToolRL (Qian et al., 2025a); SWiRL (Goldie et al., 2025); Nemotron-Research-Tool-N1 (Zhang et al., 2025m);
Prompt-Based
EASYTOOL (Yuan et al., 2025b); PLAY2PROMPT (Fang et al., 2025); DRAFT (Qu et al., 2025); JointOptim (Wu et al., 2025a);
Tool Creation
CREATOR (Qian et al., 2023); LATM (Cai et al., 2024); CRAFT (Yuan et al., 2024a); AgentOptimizer (Zhang et al., 2024b); Alita (Qiu et al., 2025);
Multi-Agent Optimisation
Prompt Optimisation
AutoAgents (Chen et al., 2024b); DSPy (Singhvi et al., 2023); MIPRO (Opsahl-Ong et al., 2024); PromptWizard (Agarwal et al., 2024)
Topology Optimisation
Code-level Workflow
AutoFlow (Li et al., 2024h); AFlow (Zhang et al., 2025j); ScoreFlow (Wang et al., 2025j); MAS-GPT (Ye et al., 2025);
Communication Graph
GPTSwarm (Zhuge et al., 2024a); DynaSwarm (Leong and Wu, 2025); G-Designer (Zhang et al., 2024a); NetSafe (Yu et al., 2024a); AgentPrune (Zhang et al., 2025g); AGP (Li et al., 2025a);
Unified Optimisation
Code-Based
ADAS (Hu et al., 2025a); FlowReasoner (Gao et al., 2025a);
Search-Based
EvoAgent (Yuan et al., 2025a); MASS (Zhou et al., 2025a); DebFlow (Su et al., 2025); EvoFlow (Gao et al., 2025a); MAS-ZERO (Ke et al., 2025);
Learning-Based
MaAS (Zhang et al., 2025f); ANN (Ma et al., 2025);
LLM Backbone Optimisation
Reasoning-Oriented
AutoFlow (Li et al., 2024h); AFlow (Zhang et al., 2025j); ScoreFlow (Wang et al., 2025j); MAS-GPT (Ye et al., 2025);
Collaboration-Oriented
COPPER (Bo et al., 2024); OPTIMA (Chen et al., 2025h); MaPoRL (Park et al., 2025);
Domain-Specific Optimisation
Biomedicine
Medical Diagnosis
MedAgentSim (Almansoori et al., 2025); PathFinder (Ghezloo et al., 2025); MDAgents (Kim et al., 2024); MDTeamGPT (Chen et al., 2025c); MMedAgent (Li et al., 2024a); MedAgent-Pro (Wang et al., 2025l);
Molecular Discovery
CACTUS (McNaughton et al., 2024); LLM-RDF (M. Bran et al., 2024); ChemAgent (Tang et al., 2025a); OSDA Agent (Hu et al., 2025c); DrugAgent (Inoue et al., 2025);LIDDIA (Averly et al., 2025);
Programming
Code Refinement
Self-Refine (Madaan et al., 2023); AgentCoder (Huang et al., 2023a); CodeAgent (Tang et al., 2024); CodeCoR (Pan et al., 2025b); OpenHands (Wang et al., 2025g);
Code Debugging
Self-Debugging (Chen et al., 2024c); Self-Edit (Zhang et al., 2023a); PyCapsule (Adnan et al., 2025); RGD (Jin et al., 2024);
Financial and Legal Research
Financial Decision-Making
FinCon (Yu et al., 2024b); PEER (Wang et al., 2024j); FinRobot (Yang et al., 2024b);
Legal Reasoning
LawLuo (Sun et al., 2024b);AgentCourt (Chen et al., 2025a);LegalGPT (Shi et al., 2024b);

Figure 5 A comprehensive hierarchical categorisation of Agentic Self-Evolution methods, encompassing single-agent, multi-agent and domain-specific optimisation categories, illustrated with selected representative works.

4.1 LLM Behaviour Optimisation

Backbone LLMs lay the foundation for single-agent systems, serving as the primary component responsible for planning, reasoning, and task execution. Therefore, enhancing the planning and reasoning capabilities of the LLM is central to improving the overall effectiveness of the agent system. Recent efforts in this direction broadly fall into two categories: (1) **training-based methods**, which directly update the model's parameters to improve reasoning ability and task performance; (2) **test-time methods**, which aim to improve LLM's behaviour during inference without modifying its parameters. In the following, we review and summarise representative approaches from both categories.

4.1.1 Training-Based Behaviour Optimisation

While LLMs have demonstrated strong linguistic capabilities (Zhao et al., 2023), recent research (Wu et al., 2024c) highlights a notable gap between their fluency in natural language and their ability to perform complex reasoning. This discrepancy limits the effectiveness of LLM-based agents in tasks that require multi-step inference and complex decision-making. To address this, recent work has explored reasoning-oriented training methods, using supervised fine-tuning (SFT) and reinforcement learning (RL) to help models systematically evaluate and refine their responses.

Supervised Fine-tuning. The core idea of supervised fine-tuning is to train agents using annotated data that contains detailed reasoning steps, allowing the model to learn a complete mapping from the input question, through intermediate reasoning processes, to the final answer. This approach typically relies on carefully constructed reasoning trajectories, which can typically be constructed from (1) **rollouts generated by the agent itself during execution**, and (2) **demonstrations produced by stronger teacher agents**. By imitating these trajectories, the agent acquires the ability to perform step-by-step reasoning in a structured manner. STaR (Zelikman et al., 2022) proposes an iterative fine-tuning procedure, where the model is trained on instances it has solved correctly and refines incorrect traces to generate better trajectories. Based on this idea, NExT (Ni et al., 2024) uses self-generated trajectories filtered by unit test correctness to self-evolve agents for program repair tasks. Similarly, Deepseek-Prover (Xin et al., 2024) progressively evolve the agent by iteratively training the policy model with verified proofs, enabling it to generate increasingly accurate formal proofs for theorem-proving tasks. Another line of work fine-tunes agents on trajectories generated by proprietary LLMs, across domains such as mathematics (Gou et al., 2024; Yin et al., 2024) and science (Ma et al., 2024). Beyond agentic capability, Min et al. (2024); Huang et al. (2024c); Labs (2025) train models based on trajectories generated by OpenAI o1 (Jaech et al., 2024) to replicate its thinking capability, aiming to further improve the agent backbone's reasoning ability.

Reinforcement Learning. RL treats reasoning as a sequential decision-making process where the model is rewarded for producing correct or high-quality reasoning paths. One of the strategies is preference-based optimisation, where DPO (Rafailov et al., 2023) is applied using preference pairs generated from various sources, such as test case performance, correctness of final outcomes, or pseudo-labels from trained process reward models (PRMs) (Hui et al., 2024; Min et al., 2024; Jiao et al., 2024). Yuan et al. (2024d) further introduce a self-evolving framework where the policy model uses its own judgments to iteratively refine its reasoning ability. Similarly, Agent Q (Putta et al., 2024) combines MCTS-guided search and a self-critique mechanism to iteratively improve agents' decision making in web environments via DPO, leveraging both successful and failed trajectories. In another line of work, Tülu 3 (Lambert et al., 2024) applies reinforcement learning with verifiable rewards across mathematical and instruction-following tasks without any learned reward model. Notably, DeepSeek-R1 (Guo et al., 2025) further demonstrates the feasibility of pure RL with Group Relative Policy Optimisation (Shao et al., 2024) when ground-truth verification is possible. Building on this direction, Xin et al. (2025) extend the idea to enhance DeepSeek-Prover by incorporating reinforcement learning from proof assistant feedback. Beyond using verifiable rewards in a fixed dataset, Absolute Zero (Zhao et al., 2025a) trains a single model that alternates between task proposer and solver roles, self-evolving by generating and solving its own problems. Similarly, R-Zero (Huang et al., 2025) employs a dual-mode framework in which a challenger generates tasks tailored to the solver's current competence, enabling both to evolve iteratively without external supervision.

4.1.2 Test-Time Behaviour Optimisation

As training resources become increasingly constrained and API-based models cannot be fine-tuned, test-time compute emerges as a solution to these limitations by enabling models to refine or extend their reasoning capabilities during inference *without additional training*. By increasing the inference budget, models are able to "think longer".

Scaling test-time capabilities can be achieved through two primary strategies. The first involves *guiding inference through the incorporation of external feedback*, which facilitates the model's refinement of its responses. The second strategy focuses on *generating multiple candidate outputs using more efficient sampling algorithms*. This is followed by a selection process where a verifier identifies the most suitable output. Notably, these two approaches are in fact closely related. The feedback used to guide generation in the former can naturally serve as a verifier in the latter.

Feedback-based Strategy. A natural method is to adjust a model's behaviour based on the quality of its generated outputs. This process typically relies on feedback from a *verifier*, which provides either an exact or estimated score to guide the model. We categorise feedback into two types. *Outcome-level feedback* provides a single score or signal based on the final output, regardless of the number of reasoning steps taken. For tasks where ground-truth is easily accessible, the verifier can be implemented as an external tool to provide accurate feedback. For example, CodeT (Chen et al., 2023) and LEVER (Ni et al., 2023) leverage a compiler to execute the generated code and validate its correctness against test cases. Similarly, Baldur (First et al., 2023) utilises error messages from proof assistants to further repair incorrect proofs generated by LLMs. However, for most tasks, ground-truth is not always available at inference time. As a result, a more general approach is to train a model to serve as the verifier that assigns a score to each candidate response (Liu et al., 2024a, 2025b), allowing them to be ranked based on predicted quality. However, this form of feedback is relatively sparse, as it evaluates only the final output. In contrast, *step-level feedback* evaluates each intermediate step during the generation process, offering finer-grained supervision. Relying solely on outcome feedback often leads to the *unfaithful reasoning* problem (Turpin et al., 2023), where incorrect reasoning chains may still result in correct final answers. To address this, recent work (Wang et al., 2024d; Jiao et al., 2024; Setlur et al., 2025) increasingly focuses on training process reward models to detect and correct errors throughout the reasoning process, generally yielding better improvement than using outcome-level feedback.

Search-based Strategy. Complex reasoning tasks often admit multiple valid paths leading to the correct solution. Search-based approaches take advantage of this property by exploring several candidate reasoning trajectories in parallel, enabling the model to better navigate the solution space. With the help of critic models, various search strategies have been developed to guide the decoding process. For example, CoT-SC (Wang et al., 2023b) adopts a best-of-N approach: it generates multiple reasoning paths and selects the final answer based on the majority vote over outcomes. DBS (Zhu et al., 2024) proposes the use of beam search in combination with step-level feedback to refine intermediate reasoning steps, while CoRe (Zhu et al., 2023) and Tree-of-Thoughts (Yao et al., 2023a) explicitly model the reasoning process as a tree structure, using Monte Carlo Tree Search (MCST) for a balance between exploration and exploitation during searching. Forest-of-Thought (Bi et al., 2025) further generalises this idea by enabling multiple trees to make independent decisions and applying a sparse activation mechanism to filter and select outputs from the most relevant trees. Beyond tree-based methods, other approaches have also explored alternative structural formulations of reasoning. Graph-of-Thoughts (Besta et al., 2024) organises intermediate thoughts as nodes in a graph and applies graph-based operations to support flexible reasoning and information flow. Buffer-of-Thoughts (Yang et al., 2024c) introduces a dynamic memory buffer to store and instantiate meta-level thoughts during reasoning.

4.2 Prompt Optimisation

In single-agent systems, prompts play a critical role in defining the agent's goals, behaviour, and task-specific strategies. They typically contain instructions, illustrative demonstrations, and contextual information that guide the underlying LLM in generating appropriate outputs. However, it is well-known that LLMs are highly sensitive to prompts; even minor variations in phrasing, formatting, or word ordering can lead to significant changes in the LLMs' behaviour and output (Loya et al., 2023; Zhou et al., 2024b). This sensitivity makes it difficult to design robust and generalisable AI agent systems, motivating the development of prompt optimisation techniques to automatically search for high-quality prompts. Prompt optimisation methods can be categorised based on the strategies used to navigate the prompt space and identify high-quality prompts that enhance model performance. In this section, we review and summarise four representative categories: edit-based methods, generative methods, text gradient-based methods, and evolutionary methods.

4.2.1 Edit-Based Prompt Optimisation

Earlier attempts in prompt optimisation focus on *edit-based* approaches, which iteratively refine human-written prompts through predefined editing operations, such as token insertion, deletion or substitution (Prasad et al., 2023; Pan et al., 2024a; Lu et al., 2024c; Zhang et al., 2023b; Zhou et al., 2023a; Agarwal et al., 2024). These methods treat prompt optimisation as a local search problem over prompt space, aiming to gradually improve prompt quality while preserving the core semantics of the original instruction. For example, GRIPS (Prasad et al., 2023) splits instructions into phrases and applies phrase-level edit operations: delete, swap, paraphrase, and addition, to progressively improve prompt quality. Plum (Pan et al., 2024a) extends GRIPS by incorporating metaheuristic strategies such as simulated annealing, mutation, and crossover. TEMPERA (Zhang et al., 2023b) further frames the editing process as a reinforcement learning problem, training a policy model to perform different editing techniques to construct query-dependent prompts efficiently.

4.2.2 Generative Prompt Optimisation

In contrast to edit-based methods that apply local modifications to prompts, *generative* approaches leverage LLMs to iteratively generate entirely new prompts, conditioned on a base prompt and various optimisation signals. Compared to local edits, generative methods can explore a broader region of the prompt space and produce more diverse and semantically rich candidates.

The prompt generation process is typically driven by a variety of optimisation signals that guide the LLM towards producing improved prompts. These signals may include predefined rewriting rules (Xu et al., 2022; Zhou et al., 2024a), input-output examplars (Zhou et al., 2023c; Xu et al., 2024b), and dataset or program descriptions (Opsahl-Ong et al., 2024). Additional guidance can come from prior prompts along with their evaluation scores (Yang et al., 2024a), meta-prompts that specify task objectives and constraints (Ye et al., 2023; Hsieh et al., 2024; Wang et al., 2024i; Xiang et al., 2025), as well as signals that indicate the desired direction of change (Fernando et al., 2024; Guo et al., 2024b; Opsahl-Ong et al., 2024). Moreover, some methods also leverage success and failure examples to highlight effective or problematic prompt patterns (Wu et al., 2024b; Yao et al., 2024). For example, ORPO (Yang et al., 2024a) generates new instructions by prompting the LLM with previously generated candidates and their evaluation scores. StraGo (Wu et al., 2024b) leverages insights from both successful and failure cases to identify critical factors for obtaining high-quality prompts. The optimisation signals can be further integrated into advanced search strategies, such as Gibbs sampling (Xu et al., 2024b), Monte Carlo tree search (MCTS) (Wang et al., 2024i), Bayesian optimisation (Opsahl-Ong et al., 2024; Lin et al., 2024b; Hu et al., 2024; Schneider et al., 2025; Wan et al., 2025), and neural bandit-based methods (Lin et al., 2024b; Shi et al., 2024a; Lin et al., 2024a). These search strategies enable more efficient and scalable exploration of the prompt space. For instance, PromptAgent (Wang et al., 2024i) formulates prompt optimisation as a strategic planning problem and leverages MCTS to efficiently navigate the expert-level prompt space. MIPRO (Opsahl-Ong et al., 2024) employs Bayesian optimisation to efficiently search for the optimal combination of instruction candidates and few-shot demonstrations.

While most generative approaches use a frozen LLM to generate new prompts, recent work has explored the use of reinforcement learning to train a policy model for prompt generation (Deng et al., 2022; Sun et al., 2024a; Yao et al., 2024; Wang et al., 2025k). For example, Retroformer (Yao et al., 2024) trains a policy model to iteratively refine prompts by summarising the root cause of previous failed cases.

4.2.3 Text Gradient-Based Prompt Optimisation

In addition to editing and generating prompts directly, a more recent line of work explores the use of *text gradients* to guide prompt optimisation (Pryzant et al., 2023; Yuksekgonul et al., 2024; Wang et al., 2024g; Austin and Chartock, 2024; Yüksekgönül et al., 2025; Tang et al., 2025b; Zhang et al., 2025l). These methods draw inspiration from gradient-based learning in neural networks, but instead of computing numerical gradients over model parameters, they generate natural language feedback, which is referred to as "text gradient", that guides how a prompt should be updated to optimise a given objective. Once the text gradient is obtained, the prompt is updated according to the feedback. The key components within such approaches lie in how the text gradients are generated and how they are subsequently used to update the prompt. For example, ProTeGi (Pryzant et al., 2023) generates text gradients by criticising the current prompt. Subsequently, it edits the prompt in the opposite semantic direction of the gradient. Such "gradient descent" steps are guided by a beam search and bandit selection procedure to find optimal prompts efficiently. Similarly, TextGrad (Yuksekgonul et al., 2024; Yüksekgönül et al., 2025) generalises this idea to a broader framework for compound AI systems. It treats textual feedback as a form of "automatic differentiation" and uses LLM-generated suggestions to iteratively improve components such as prompts, code, or other symbolic variables. Recent work (Wu et al., 2025c) also explores the prompt optimisation in compound AI systems, where its goal is to automatically optimise the configuration across a heterogeneous set of components and parameters, e.g., model parameters, prompts, model selection choice, and hyperparameters.

4.2.4 Evolutionary Prompt Optimisation

In addition to the above optimisation techniques, *evolutionary algorithms* have also been explored as a flexible and effective approach for prompt optimisation (Guo et al., 2024b; Fernando et al., 2024). These approaches treat prompt optimisation as an evolutionary process, maintaining a population of candidate prompts that are iteratively refined through evolutionary operators such as mutation, crossover, and selection. For example, EvoPrompt (Guo et al., 2024b) leverages two widely used evolutionary algorithms: Genetic Algorithm (GA) and Differential Evolution (DE), to guide the optimisation process to find the high-performing prompts. It adapts the core evolutionary operations, namely mutation and crossover, to the prompt optimisation setting, where new candidate prompts are generated by combining segments from two parent prompts and introducing random alternation to specific elements. Similarly, PROMPTBREEDER (Fernando et al., 2024) also iteratively mutates a population of task-prompts to evolve these prompts. A key feature is its use of *mutation prompts*, which are instructions that specify how task-prompts should be modified during the mutation process. These mutation prompts can be either predefined or generated dynamically by the LLM itself, enabling a flexible and adaptive mechanism for guiding prompt evolution.

4.3 Memory Optimisation

Memory is essential for enabling agents to reason, adapt, and operate effectively over extended tasks. However, AI agents frequently face limitations arising from constrained context windows and forgetting, which can result in phenomena such as context drift and hallucination (Liu et al., 2024b; Zhang et al., 2024c,d). These limitations have driven increasing interest in memory optimisation to enable generalisable and consistent behaviours in dynamic environments. In this survey, we focus on inference-time memory strategies that enhance memory utilisation without modifying model parameters. In contrast to training-time techniques such as fine-tuning or knowledge editing (Cao et al., 2021; Mitchell et al., 2022), inference-time approaches dynamically decide what to retain, retrieve, and discard during the reasoning process.

We categorise existing methods into two optimisation objectives: *short-term memory*, which focuses on maintaining coherence within the active context, and *long-term memory*, which supports persistent retrieval across sessions. This optimisation-oriented perspective shifts the focus from static memory formats (e.g., internal vs. external) to dynamic memory control, with an emphasis on how memory is scheduled, updated, and reused to support decision-making. In the following subsections, we present representative methods within each category, emphasising their impact on reasoning fidelity and effectiveness in long-horizon settings.

4.3.1 Short-term Memory Optimisation

Short-term memory optimisation focuses on managing limited contextual information within the LLM's working memory (Liu et al., 2024b). This typically includes recent dialogue turns, intermediate reasoning traces, and task-relevant content from the immediate context. As the context expands, memory demands increase significantly, making it impractical to retain all information within a fixed context window. To address this, various techniques have been proposed to compress, summarise, or selectively retain key information (Zhang et al., 2024d; Wang et al., 2025d). Common strategies encompass summarisation, selective retention, sparse attention, and dynamic context filtering. For example, Wang et al. (2025d) proposes recursive summarisation to incrementally construct compact and comprehensive memory representations, enabling consistent responses throughout extended interactions. MemoChat (Lu et al., 2023) maintains dialogue-level memory derived from conversation history to support coherent and personalised interaction. COMEDY (Chen et al., 2025f) and ReadAgent (Lee et al., 2024d) further incorporate extracted or compressed memory traces into the generation process, allowing agents to maintain context over long conversations or documents. In addition to summarisation, other methods dynamically adjust the context or retrieve intermediate state traces to facilitate multi-hop reasoning. For example, MoT (Li and Qiu, 2023) and StructRAG (Li et al., 2025e) retrieve self-generated or structured memory to guide intermediate steps. MemoryBank (Zhong et al., 2024), inspired by the Ebbinghaus forgetting curve (Murre and Dros, 2015), hierarchically summarises events and updates memory based on recency and relevance. Reflexion (Shinn et al., 2023) enables agents to reflect on task feedback and store episodic insights, promoting self-improvement over time.

These methods significantly improve local coherence and context efficiency. However, short-term memory alone is insufficient for retaining knowledge across sessions or enabling generalisation over long horizons, highlighting the need for complementary long-term memory mechanisms.

4.3.2 Long-term Memory Optimisation

Long-term memory optimisation mitigates the limitations of short context windows by providing persistent and scalable storage that extends beyond the immediate input scope of the language model. It enables agents to retain and retrieve factual knowledge, task histories, user preferences, and interaction trajectories across sessions (Du et al., 2025), thereby supporting coherent reasoning and decision-making over time. A key objective in this area is to manage increasingly complex and expanding memory spaces while preserving a clear separation between memory storage and the reasoning process (Zhang et al., 2024d). External memory can be either unstructured or organised into structured formats such as tuples, databases, or knowledge graphs (Zeng et al., 2024b), and may span a wide range of sources and modalities.

A critical paradigm of long-term memory optimisation is Retrieval-Augmented Generation (RAG), which incorporates relevant external memory into the reasoning process via retrieval (Wang et al., 2023a; Efeoglu and Paschke, 2024; Gao et al., 2025c). For instance, EWE (Chen et al., 2025d) augments a language model with an explicit working memory that dynamically holds latent representations of retrieved passages, focusing on combining static memory entries at each decoding step. In contrast, A-MEM (Xu et al., 2025) constructs interconnected knowledge networks through dynamic indexing and linking, enabling agents to form evolving memory. Another prominent direction involves agentic retrieval, where agents autonomously determine when and what to retrieve, alongside trajectory-level memory, which utilises past interactions to inform future behaviour. Supporting techniques such as efficient indexing, memory pruning, and compression further enhance scalability (Zheng et al., 2023a; Alizadeh et al., 2024). For example, Wang et al. (2024e) propose a lightweight unlearning framework based on the RAG paradigm. By altering the external knowledge base used for retrieval, the system can simulate forgetting effects without modifying the underlying LLM. Similarly, Xu et al. (2025) introduce a self-evolving memory system that maintains long-term memory without relying on predefined operations. In addition to retrieval policies and memory control mechanisms, the structure and encoding of memory itself significantly affect system performance. Vector-based memory systems encode memory in dense latent spaces and support fast, dynamic access. For instance, MemGPT (Packer et al., 2023), NeuroCache (Safaya and Yuret, 2024), G-Memory (Zhang et al., 2025e), and AWESOME (Cao and Wang, 2024) enable consolidation and reuse across tasks. Mem0 (Chhikara et al., 2025) further introduces a production-ready memory-centric architecture for continuous extraction and retrieval. Other approaches draw inspiration from biological or symbolic systems to improve interpretability. HippoRAG (Gutierrez et al., 2024) implements hippocampus-inspired indexing via lightweight knowledge graphs. GraphReader (Li et al., 2024d) and Mem0^{g} (Chhikara et al., 2025) use graph-based structures to capture conversational dependencies and guide retrieval. In the symbolic domain, systems like ChatDB (Hu et al., 2023) issue SQL queries over structured databases, while Wang et al. (2024f) introduces a neurosymbolic framework that stores facts and rules in both natural and symbolic form, supporting precise reasoning and memory tracking.

Recent work has also emphasised the importance of memory control mechanisms during inference (Zou et al., 2024; Chen et al., 2025d), which determine what, when, and how to store, update, or discard memory (Jin et al., 2025). For instance, MATTER (Lee et al., 2024b) dynamically selects relevant segments from multiple heterogeneous memory sources to support question answering, and AWM (Wang et al., 2024l) enables continuous memory updates in both online and offline settings. MyAgent (Hou et al., 2024) endows agents with memory-aware recall mechanisms for generation, addressing the temporal cognition limitations of LLMs. MemoryBank (Zhong et al., 2024) proposes a cognitively inspired update strategy, where periodic revisiting of past knowledge mitigates forgetting and enhances long-term retention. Reinforcement learning and prioritisation policies have also been employed to guide memory dynamics. For example, MEM1 (Zhou et al., 2025b) leverages reinforcement learning to maintain an evolving internal memory state, selectively consolidating new information while discarding irrelevant content. A-MEM (Xu et al., 2025) presents an agentic memory architecture that autonomously organises, updates, and prunes memory based on usage. MrSteve (Park et al., 2024) incorporates episodic "what-where-when" memory to hierarchically structure long-term knowledge, enabling goal-directed planning and task execution. These approaches allow agents to proactively manage memory and complement short-term mechanisms. Meanwhile, MIRIX (Wang and Chen, 2025) introduces an agent memory system with six specialised memory types in collaborative settings, enabling coordinated retrieval and achieving state-of-the-art performance in long-horizon tasks.

4.4 Tool Optimisation

Tools are critical components within agent systems, serving as interfaces that allow agents to perceive and interact with the real world. They enable access to external information sources, structured databases, computational resources, and APIs, thereby enhancing the agent's ability to solve complex, real-world problems (Patil et al., 2024; Yang et al., 2023; Guo et al., 2024d). As a result, tool use has become a core competence of AI agents, especially for tasks that require external knowledge and multi-step reasoning. However, simply exposing agents to tools is not sufficient. Effective tool use requires the agent to recognise when and how to invoke the right tools, interpret tool outputs, and integrate them into multi-step reasoning. Consequently, recent research has focused on tool optimisation, which aims to enhance the agent's ability to use tools intelligently and efficiently.

Existing research on tool optimisation largely falls into two complementary directions. The first, which has been more extensively explored, focuses on enhancing the agent's ability to interact with tools. This is achieved through different approaches, including training strategies, prompting techniques, and reasoning algorithms, that aim to improve the agent's ability to understand, select, and execute tools effectively. The second, which is more recent and still emerging, focuses on optimising the tools themselves by modifying existing tools or creating new ones that better align with the functional requirements of the target tasks.

4.4.1 Training-Based Tool Optimisation

Training-based tool optimisation aims to enhance an agent's ability to use tools by updating the underlying LLM's parameters through learning. The motivation behind this approach stems from the fact that LLMs are pretrained purely on text generation tasks, without any exposure to tool usage or interactive execution. Therefore, they lack an inherent understanding of how to invoke external tools and interpret tool outputs. Training-based methods aim to address this limitation by explicitly teaching the LLMs how to interact with tools, thereby embedding tool-use capabilities directly into the agent's internal policy.

Supervised Fine-Tuning for Tool Optimisation. Earlier efforts in this line of work rely on supervised fine-tuning (SFT), which trains the LLM on high-quality tool-use trajectories to explicitly demonstrate how tools should be invoked and integrated into task execution (Schick et al., 2023; Du et al., 2024; Liu et al., 2025d; Wang et al., 2025e). A central focus of these methods lies in the collection of high-quality tool-use trajectories, which typically consist of input queries, intermediate reasoning steps, tool invocations and final answers. These trajectories serve as explicit supervision signals for the agent, teaching it how to plan tool usage, execute calls, and incorporate results into its reasoning process. For example, approaches such as ToolLLM (Qin et al., 2024) and GPT4Tools (Yang et al., 2023) leverage more powerful LLMs to generate both instructions and corresponding tool-use trajectories. Inspired by the human learning process, STE (Wang et al., 2024a) introduces simulated trial-and-error interactions to collect tool-use examples, while TOOLEVO (Chen et al., 2025b) employs MCTS to enable more active exploration and collect higher-quality trajectories.

Moreover, recent work (Yao et al., 2025) indicates that even advanced LLMs face challenges with tool use in multi-turn interactions, especially when these interactions involve complex function calls, long-term dependencies, or requesting missing information. To generate high-quality training trajectories on multi-turn tool calling, Magnet (Yin et al., 2025) proposes to synthesise a sequence of queries and executable function calls from tools, and employs a graph to build a reliable multi-turn query. BUTTON (Chen et al., 2025e) generates synthetic compositional instruction tuning data via a two-stage process, where a bottom-up stage composes atomic tasks to construct the instructions and a top-down stage employs a multi-agent system to simulate the user, assistant, and tool to generate the trajectory data. To enable more realistic data generation, APIGen-MT (Prabhakar et al., 2025) proposes a two-phase framework that first generates tool call sequences and then transforms them into complete multi-turn interaction trajectories through simulated human-agent interplay.

Once the tool-use trajectories are collected, they are used to fine-tune the LLM through standard language modelling objectives, enabling the model to learn successful patterns of tool invocation and integration. In addition to this common paradigm, some studies have also explored more advanced training strategies to further enhance tool-use capabilities. For example, Confucius (Gao et al., 2024) introduces an easy-to-difficult curriculum learning paradigm that gradually exposes the model to increasingly complex tool-use scenarios. Gorilla (Patil et al., 2024) proposes integrating a document retriever into the training pipeline, allowing the agent to dynamically adapt to evolving toolsets by grounding tool usage in retrieved documentation.

Reinforcement Learning for Tool Optimisation. While supervised fine-tuning has proven effective for teaching agents to use tools, its performance is often constrained by the quality and coverage of the training data. Low-quality trajectories can lead to diminished performance gains. Moreover, fine-tuning on limited datasets may hinder generalisation, particularly when agents encounter unseen tools or task configurations at inference time. To address these limitations, recent research has turned to reinforcement learning (RL) as an alternative optimisation paradigm for tool use. By enabling agents to learn through interaction and feedback, RL facilitates the development of more adaptive and robust tool-use strategies. This approach has shown promising results in recent work such as ReTool (Feng et al., 2025a) and Nemotron-Research-Tool-N1 (Tool-N1) (Zhang et al., 2025m), both of which demonstrate how lightweight supervision in an interactive environment can lead to more generalisable tool-use capabilities. Building on these foundations, further studies have explored the design of more effective reward functions (Qian et al., 2025a), as well as the use of synthetic data generation and filtering to improve the stability and efficiency of RL-based training (Goldie et al., 2025).

4.4.2 Inference-Time Tool Optimisation

In addition to training-based approaches, another line of work focuses on enhancing tool-use capabilities during inference, without modifying LLM parameters. These methods typically operate by optimising tool-related contextual information within prompts or guiding the agent's decision-making process through structured reasoning at test time. There are two major directions within this paradigm: (1) *prompt-based methods*, which refine the representation of tool documentation or instructions to facilitate better understanding and utilisation of tools; (2) *reasoning-based methods*, which leverage test-time reasoning strategies, such as MCTS and other tree-based algorithms to enable more effective exploration and selection of tools during inference.

Prompt-Based Tool Optimisation. Tool-related information is typically provided to agents through tool documentation within prompts. These documents describe tool functionalities, potential usage, and invocation formats, helping the agent understand how to interact with external tools to solve complex tasks. Therefore, tool documentation within prompts serves as a crucial bridge between the agent and its available tools, directly influencing the quality of tool-use decisions. Recent efforts have focused on optimising how this documentation is presented, either by restructuring source documents or refining them through interactive feedback (Qu et al., 2025). For instance, EASYTOOL (Yuan et al., 2025b) transforms different tool documentation into unified, concise instructions, making them easier for LLMs to use. In contrast, approaches such as DRAFT (Qu et al., 2025) and PLAY2PROMPT (Fang et al., 2025) draw inspiration from human trial-and-error processes, introducing interactive frameworks that iteratively refine tool documentation based on feedback.

Beyond these methods, a more recent direction explores the joint optimisation of both tool documentation and the instructions provided to the LLM agent. For example, Wu et al. (2025a) propose an optimisation framework that simultaneously refines the agent's prompt instructions and the tool descriptions, collectively referred to as the *context*, to enhance their interaction. The optimised context has been shown to reduce computational overhead and improve tool-use efficiency, highlighting the importance of context design in effective inference-time tool optimisation.

Reasoning-Based Tool Optimisation. Test-time reasoning and planning techniques have demonstrated strong potential for improving tool-use capabilities in AI agents. Early work such as ToolLLM (Qin et al., 2024) has validated the effectiveness of the ReAct (Yao et al., 2023b) framework in tool-use scenarios, and further proposed a depth-first tree search algorithm that enables agents to quickly backtrack to the last successful state rather than restarting from scratch, which significantly improves efficiency. ToolChain (Zhuang et al., 2024) introduces a more efficient tree-based search algorithm by employing a cost function to estimate the future cost of a given branch. This allows agents to prune low-value paths early and avoid the inefficient rollouts commonly associated with traditional MCTS. Similarly, Tool-Planner (Liu et al., 2025e) clusters tools with similar functionalities into *toolkits* and leverages a tree-based planning method to quickly reselect and adjust tools from these toolkits.

4.4.3 Tool Functionality Optimisation

Beyond optimising the agent's behaviour, a complementary line of work focuses on modifying or generating tools themselves to better support task-specific reasoning and execution. Inspired by the human practice of continuously developing tools to meet task requirements, these approaches aim to extend the agent's action space by adapting the toolset to the task, rather than adapting the task to a fixed toolset (Wang et al., 2024k). For instance, CREATOR (Qian et al., 2023) and LATM (Cai et al., 2024) introduce frameworks that generate tool documentation and executable code for novel tasks. CRAFT (Yuan et al., 2024a) leverages reusable code snippets from prior tasks to create new tools for unseen scenarios. AgentOptimiser (Zhang et al., 2024b) treats tools and functions as learnable weights, allowing the agent to iteratively refine them using LLM-based updates. A more recent work, Alita (Qiu et al., 2025), extends tool creation into a Multi-Component Program (MCP) format, which enhances reusability and environment management.

5 Multi-Agent Optimisation

The multi-agent workflow defines how multiple agents collaborate to solve complex tasks through structured topologies and interaction patterns. The field has witnessed a fundamental shift: from manually designed agent architectures, where researchers explicitly specify collaboration patterns and communication protocols, to self-evolving systems that automatically discover effective collaboration strategies. This evolution reframes workflow design as a search problem over three interconnected spaces: the structural space of possible agent topologies, the semantic space of agent roles and instructions, and the capability space of LLM backbones. Recent approaches explore these spaces using a range of optimisation techniques, from evolutionary algorithms to reinforcement learning, each offering different trade-offs in balancing multiple optimisation targets (e.g., accuracy, efficiency, and safety).

This section traces the progression of multi-agent workflow optimisation across four key dimensions. Our starting point examines manually designed paradigms that establish foundational principles. From there, we consider prompt-level optimisation, which refines agent behaviours within fixed topologies. We subsequently address topology optimisation, which focuses on discovering the most effective architectures for multiple agents to accomplish a given task. We also discuss comprehensive approaches that simultaneously consider multiple optimisation spaces, jointly optimising prompts, topologies, and other system parameters in an integrated manner. Additionally, we investigate LLM-backbone optimisation, which enhances the fundamental reasoning and collaborative capabilities of the agents themselves through targeted training. Through this lens, we show how the field progressively expands its conception of what constitutes a searchable and optimisable parameter in multi-agent systems, from agent instructions and communication structures to the core competencies of the underlying models. Figure 6 provides an overview of multi-agent workflow optimisation across its core elements and key dimensions.

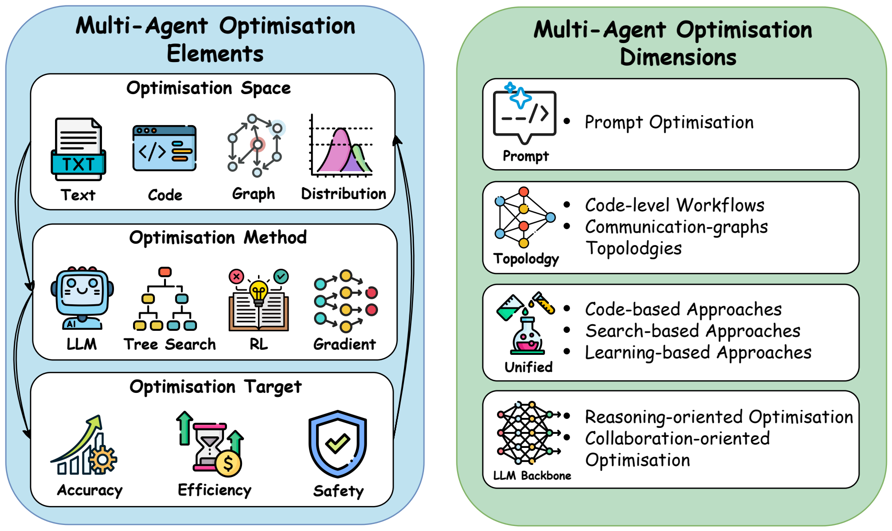


Figure 6 An overview of multi-agent systems optimisation approaches, with core optimisation elements (space, methods, and targets) on the left and optimisation dimensions (prompt, topology, unified, and LLM backbone) on the right.

5.1 Manually Designed Multi-Agent Systems

Manually designed workflows form the foundation of multi-agent collaboration research. These architectures encode researchers' insights about task decomposition, agent capabilities, and coordination mechanisms into explicit interaction patterns. By examining these handcrafted paradigms, we can understand the design principles that guide agent collaboration and the engineering considerations that shape system architecture.

Parallel Workflows. Parallel workflows employ concurrent execution followed by collective decision-making. The simplest form involves multiple independent agents generating solutions in parallel, followed by majority voting to select the final output. Empirical evidence shows that parallel generation with small LLMs can match or even outperform single large LLMs (Verga et al., 2024; Wang et al., 2025a). Multi-layer aggregation further reduces error bounds and improves robustness (Zhang et al., 2025d). Recent extensions incorporate dynamic task graphs and asynchronous threads to enable near-linear scaling and lower decision latency (Yu et al., 2025; Gu et al., 2025; Wang et al., 2025c). However, while computational throughput scales horizontally, the engineering costs of managing coordination and consistency grow exponentially.

Hierarchical Workflows. When subtasks exhibit strict contextual dependencies, hierarchical (Zhang et al., 2024c; Qian et al., 2024) workflows offer a structured alternative. These frameworks organise agents into multi-level top-down structures or sequential pipelines. The system decomposes tasks across layers, with each layer responsible for different subtasks. This design excels in complex goal-driven tasks such as deep research and code generation (Hong et al., 2024; Zhang et al., 2025n). However, its fixed topology limits adaptability, especially when facing dynamic goals or resource constraints.

Multi-Agent Debate. To balance accuracy with interpretability, researchers have developed the debate paradigm, where agents engage in adversarial-negotiation-arbitration cycles to discuss and correct reasoning errors. Early work explored symmetric debater mechanisms (Li et al., 2024g). More recent studies extend this framework by introducing role asymmetry, adjustable debate intensity, and persuasiveness-oriented strategies (Yin et al., 2023; Liang et al., 2024; Khan et al., 2024; Chang, 2024). In addition, confidence-gated debate strategies demonstrate that triggering multi-agent debates only when a single model exhibits low confidence can sharply reduce inference costs without hindering performance (Eo et al., 2025).

Despite the success of manually designed workflows and structured multi-agent paradigms, recent empirical studies reveal that single large LLMs with well-crafted prompts can match the performance of complex multi-agent discussion frameworks on multiple reasoning benchmarks (Pan et al., 2025a). This finding, coupled with the high implementation and maintenance costs of handcrafted multi-agent workflows (Li et al., 2024h; Zhang et al., 2025j), has driven the development of *self-evolving multi-agent systems* that can automatically learn, adapt, and restructure their workflows over time, rather than relying on fixed architectures and static coordination protocols.

5.2 Self-Evolving Multi-Agent System

The high engineering costs and limited adaptability of manually designed multi-agent workflows have motivated a shift towards automated, self-evolving systems. These systems can automatically design, evaluate, and refine agent workflows by adapting their prompts, topologies, and collaborative strategies based on performance feedback. Instead of relying on hard-coded configurations, they treat workflow optimisation as a search problem, where the system explores and optimises over a space of possible configurations. The search space spans multiple levels, from local prompts to global topology structures.

To effectively navigate the search space, various search algorithms have been introduced. These methods range from reinforcement learning, Monte Carlo Tree Search, and generative models that enable efficient exploration, to evolutionary operators that provide robust search capabilities. Moreover, the optimisation objectives have expanded from improving performance to balancing multi-dimensional goals, including task accuracy, computational efficiency, and safety. This evolution reveals that as search capabilities advance, the core challenge shifts from finding optimal solutions to defining what optimality means in dynamic multi-agent contexts.

5.2.1 Multi-Agent Prompt Optimisation

One promising direction for achieving such self-evolution is through prompt optimisation, where prompts define both agent roles and their corresponding task instructions. Recent approaches treat these prompt-encoded configurations as a formal search space for systematic refinement. In fact, prompt optimisation in multi-agent workflows often builds upon the single-agent techniques discussed in Section 4.2, but extends them to coordinate multiple agents and task dependencies. For example, DSPy (Singhvi et al., 2023) Assertions introduces runtime self-evolution, where the search space encompasses possible intermediate outputs from pipeline modules, using assertion-driven backtracking with explicit feedback to guide LLMs in self-correcting outputs that violate programmatic constraints. AutoAgents (Chen et al., 2024b) extends prompt optimisation from single-agent settings to entire multi-agent team configurations, optimising specialised agent roles and execution plans through structured dialogue between dedicated meta-agents.

5.2.2 Topology Optimisation

Topology optimisation represents a paradigm shift in multi-agent system design: rather than treating communication structure as a fixed constraint, it recognises topology itself as a powerful optimisation target. This insight emerged from a fundamental observation—even the best prompts cannot compensate for poor architectural choices. Viewed through a representation-centred lens, existing work clusters into two complementary families: program/code-level workflow topologies and communication-graph topologies; this classification foregrounds *what* is being optimised—the chosen representation of topology. This marks not just technical progress but a conceptual shift—the medium (topology) matters as much as the message (prompts).

Code-level workflows. Representing workflows as executable programs or typed code graphs makes agent coordination explicit and verifiable, enabling compositional reuse and automated checking. AutoFlow (Li et al., 2024h) sets the search space to natural-language programs (CoRE) and trains a generator LLM with reinforcement learning, supporting both fine-tuning and in-context use. Compared with AutoFlow, AFlow (Zhang et al., 2025j) replaces the NL program space with typed, reusable operators to form code graphs; Monte Carlo Tree Search with LLM-guided expansion and soft probabilistic selection provides a more structured, sample-efficient exploration of the vast design space than RL over CoRE. Pushing beyond these discrete search schemes, ScoreFlow (Wang et al., 2025j) lifts code representations into a continuous space and applies gradient-based optimisation with Score-DPO (a direct preference optimisation variant incorporating quantitative feedback) to improve the workflow generator. This addresses the exploration inefficiency inherent to RL/MCTS and enables task-level adaptive workflow generation. Orthogonal to search-based optimisation, MAS-GPT (Ye et al., 2025) uses supervised fine-tuning on a consistency-oriented corpus (inter- and intra-consistency) so that a single inference aims to produce a complete, executable MAS codebase, trading broad search coverage for one-shot efficiency and stronger dependence on data quality

Communication-graph topologies. Unlike code-level programs, this line treats the workflow as a multi-agent communication graph whose connections are the optimisation target (Liu et al., 2025f). GPTSwarm (Zhuge et al., 2024a) defines its search space as connections within a computational graph of agents. It relaxes this discrete space into continuous edge probabilities, also employing RL to learn optimal connection schemes. Building on GPTSwarm, DynaSwarm (Leong and Wu, 2025) extends the search space from a single optimised graph to a portfolio of graph structures with Actor–Critic (A2C) optimisation and a lightweight graph selector for per-instance topology selection, addressing the key observation that different queries require different graph structures for optimal performance. Rather than masking edges in a fixed space, G-Designer (Zhang et al., 2024a) employs a variational graph autoencoder to directly generate task-adaptive communication graphs, modulating structural complexity to balance quality and token cost. MermaidFlow (Zheng et al., 2025) represents topology as a typed, declarative graph with static verification and explores only semantically valid regions via safety-constrained evolutionary operators.

Beyond static graph synthesis, some approaches dynamically modulate the communication graph during execution. DyLAN (Liu et al., 2023b) treats the search space as active agents across layers with an early-stopping time axis; it prunes low-value agents via an LLM ranker and performs automated team optimisation with an Agent Importance Score using propagation–aggregation–selection. Captain Agent (Song et al., 2024) defines the search space as subtask-specific sets of agents and tools (retrieved, filtered, and, when needed, generated); nested group conversations and reflection iteratively refine team composition in situ rather than synthesising a fixed graph from scratch. Flow (Niu et al., 2025) contrasts with DyLAN's pruning and Captain Agent's team recomposition by dynamically adjusting the AOV graph structure: it selects an initial graph via parallelism/dependency metrics and then refines it online through workflow refinement and subtask reassignment, achieving modular concurrency with minimal coordination cost.

Orthogonal to graph synthesis, pruning methods optimise by removing redundant or risky communications while preserving essential collaboration. AgentPrune (Zhang et al., 2025g) treats the search space as a spatial-temporal communication graph where both intra-dialogue (spatial) and inter-dialogue (temporal) edges are pruning targets; it employs a trainable low-rank-guided graph mask to identify and eliminate redundant communications via one-shot pruning, optimizing for token economy. Building on this pruning paradigm, AGP (Adaptive Graph Pruning) (Li et al., 2025a) extends the search space to include both agent quantity (hard pruning) and communication edges (soft pruning). It employs a two-stage training strategy that jointly optimises these dimensions on a per-task basis, dynamically determining the optimal number of agents and their connections for task-specific topology generation. While the above methods prune for efficiency and adaptability, G-Safeguard (Wang et al., 2025f) applies pruning for security—it operates on communication edges as the search space, using a GNN to flag risky nodes and deterministic rules to cut outward edges under a model-driven threshold for defence against adversarial attacks. Relatedly, NetSafe (Yu et al., 2024a) summarises topological safety risks and proposes graph-based detection and intervention principles as a complementary safety lens.

5.2.3 Unified Optimisation

Unified optimisation emerges from a key insight: prompts and topology are not independent design choices but deeply interconnected aspects of agent systems (Zhou et al., 2025a). A well-crafted prompt cannot function effectively in a poor communication structure, while an elegant topology yields little benefit with poorly instructed agents. This interdependence has driven the field along three distinct technical paths: code-based unification, structured optimisation methods, and learning-driven architectures. Each approach tackles the joint optimisation challenge from a unique angle, revealing different trade-offs between efficiency and performance.

Code-based Approaches. The most direct approach to unified optimisation treats code as a universal representation for both prompts and topology. ADAS (Hu et al., 2025a) pioneered this approach through its Meta Agent Search framework, representing prompts, workflows, and tool use as Python code to enable iterative agent generation and evaluation. This code-centric view allows natural co-evolution, modifying agent logic inherently affects both instructional and structural aspects. FlowReasoner (Gao et al., 2025a) advanced the code-based paradigm by focusing on query-level adaptation, generating one MAS per query rather than per task. After distilling reasoning abilities from DeepSeek-R1, it employs GRPO with external execution feedback to enhance its meta-agent, optimising for performance and efficiency. Together, these methods demonstrate that code provides a flexible substrate for joint optimisation, though at different granularities of adaptation.

Search-based Approaches. Rather than relying on implicit co-evolution through code, another line of work develops explicit mechanisms for coordinating prompt and topology design. EvoAgent (Yuan et al., 2025a) defined search spaces as textual agent settings (roles, skills, prompts) and employed evolutionary algorithms with mutation, crossover, and selection operators to generate diverse agent populations. Compared with implicit code-based co-evolution, EvoAgent explicitly evolves configuration-level characteristics rather than synthesising programs. Relative to EvoAgent's text-centric configuration search, EvoFlow (Gao et al., 2025a) likewise adopts evolutionary search but over operator-node workflow graphs. It introduces predefined composite operators (e.g. CoT, debate) and uses an operator library with tag selection to constrain mutation/crossover and narrow the search space. EvoFlow further treats LLM selection as a decision variable to balance performance and cost; diversity-aware selection preserves population variety, and a multi-objective fitness drives cost–performance Pareto optimisation.

Complementary to evolutionary searches, MASS (Zhou et al., 2025a) proposes a three-stage, conditionally coupled optimisation framework: it first locally tunes each agent's prompts, then searches the workflow topology in a pruned space, and finally performs global prompt optimisation on the selected topology; the procedure alternates rather than fully decoupling, serving as a practical approximation to joint optimisation. Most recently, DebFlow (Su et al., 2025) represents search spaces as workflow graphs of operator nodes and employs multi-agent debate for optimisation. Guided by reflexion on execution failures, it avoids exhaustive search while pioneering debate mechanisms in automated agent design. These structured approaches trade some flexibility for more targeted optimisation strategies. Building on the operator node representation, MAS-ZERO (Ke et al., 2025) casts unified optimisation as a purely inference-time search, iteratively restructuring agent teams and task decompositions through solvability-guided refinement without any gradient updates or offline training.

Learning-based Approaches. The latest wave of research applies sophisticated learning paradigms to jointly optimise prompts and topology. MaAS (Zhang et al., 2025f) shifts from optimising single architectures to learning agentic supernets—probabilistic distributions over multi-agent systems. Its controller network samples query-specific architectures with Monte Carlo and textual gradient optimisation, achieving superior performance with dramatically reduced inference costs. ANN (Ma et al., 2025) conceptualises multi-agent collaboration as layered neural networks, where each layer forms specialised agent teams. It employs a two-phase optimisation process: forward task decomposition and backward textual gradient refinement. This approach jointly evolves agent roles, prompts, and inter-layer topologies, enabling post-training adaptation to novel tasks.

5.2.4 LLM Backbone Optimisation

The evolution of the LLM backbone behind agents is a critical aspect of multi-agent evolution, particularly how agents improve their *cooperative* or *reasoning* abilities through interaction.

Reasoning-oriented Optimisation. A prominent line of work focuses on enhancing the backbone LLM's reasoning capacity via multi-agent collaboration. For instance, *multi-agent finetuning* (Subramaniam et al., 2025) leverages high-quality cooperative trajectories sampled from multi-agent debates for supervised fine-tuning, enabling (1) role-specific specialisation of agents and (2) improved reasoning capabilities of the underlying backbone model. Similarly, Sirius (Zhao et al., 2025c) and MALT (Motwani et al., 2024) employ self-play to collect high-quality cooperative trajectories and train agents within their respective multi-agent collaboration frameworks. While both approaches leverage failed trajectories to some extent, they differ in methodology: Sirius relies solely on SFT and integrates incorrect trajectories via self-correction into the training dataset, whereas MALT adopts DPO, naturally utilising negative samples. These methods provide early evidence of the potential for self-improvement in multi-agent systems, though they are primarily applied in relatively simple settings (*e.g.*, multi-agent debate or "generator-verifier-answerer" system). Moving forward, MaPoRL (Park et al., 2025) introduces task-specific reward shaping to explicitly incentivise inter-agent communication and cooperation through reinforcement learning. MARFT (Liao et al., 2025) establishes a comprehensive bridge between conventional multi-agent reinforcement learning (MARL) and LLM-based multi-agent reinforcement tuning. Building on this, MARTI (Liao et al., 2025) proposes a more customizable framework for reinforced multi-agent fine-tuning, supporting flexible design of both agentic structures and reward functions. Empirical results show that LLM backbones exhibit considerable improvements in cooperative capabilities during their cooperative training.

Collaboration-oriented Optimisation. Beyond reasoning, a smaller body of work focuses on enhancing communication and collaboration abilities within multi-agent systems. The core assumption is that LLM agents are not inherently effective team players, and their collaborative communication skills require targeted training. An early example is COPPER (Bo et al., 2024), which employs PPO to train a shared reflector that generates high-quality, role-aware personalised reflections for multi-agent collaboration trajectories. OPTIMA (Chen et al., 2025h) more directly targets communication efficiency in multi-agent systems (measured by token usage and communication readability) and explores achieving an effectiveness-efficiency trade-off via SFT, DPO, and hybrid methods. It reports a 2.8× performance gain with less than 10% of the token cost on tasks demanding intensive information exchange, which vividly demonstrates the promising potential of scaling agents' collaborative capabilities. Further, MaPoRL (Park et al., 2025) argues that the prevalent paradigm of prompting out-of-the-box LLMs and relying solely on their innate collaborative abilities is questionable. Instead, it introduces carefully designed reinforcement learning signals within a multi-agent debate framework to explicitly elicit collaborative behaviours, encouraging agents to communicate more frequently and with higher quality.

6 Domain-Specific Optimisation

While previous sections have focused on agent optimisation and evolution techniques in general-domain settings, domain-specific agent systems introduce unique challenges that require tailored optimisation strategies. These domains, such as biomedicine, programming, and finance & legal research, are often characterised by specialised task structures, domain-specific knowledge bases, distinct data modalities, and operational constraints. Such factors can significantly influence how agents are designed, optimised, and evolved. In this section, we survey recent advances in domain-specific agent optimisation and evolution, highlighting effective techniques that have been developed to meet the unique demands of each domain.

6.1 Domain-Specific Optimisation in Biomedicine

In the biomedical domain, agent optimisation focuses on aligning agent behaviours with the procedural and operational requirements of real-world clinical settings. Recent studies have demonstrated the effectiveness of domain-specific agent design in two key application areas: medical diagnosis (Donner-Banzhoff, 2018; Almansoori et al., 2025; Zhuang et al., 2025) and molecular discovery (M. Bran et al., 2024; Inoue et al., 2025). In what follows, we examine representative agent optimisation strategies within these two domains.

6.1.1 Medical Diagnosis

Medical diagnosis requires determining a patient's condition based on clinical information such as symptoms, medical history, and diagnostic test results (Kononenko, 2001; Donner-Banzhoff, 2018). Recent research has increasingly explored the use of autonomous agents in this context, enabling systems to automatically conduct diagnostic dialogues, pose clarifying questions, and generate plausible diagnostic hypotheses (Li et al., 2024c; Chen et al., 2025i; Zuo et al., 2025; Ghezloo et al., 2025). These agents often operate under uncertain conditions, making decisions based on incomplete or ambiguous patient information (Chen et al., 2025i). The diagnostic process typically involves multi-turn interactions, during which agents elicit missing information through follow-up enquiries (Chen et al., 2025i). Moreover, to support robust clinical reasoning, agents often require integrating external knowledge bases or interacting with specialised medical tools for information retrieval and evidence-based reasoning (Feng et al., 2025b; Fallahpour et al., 2025).

Given these domain-specific requirements, recent studies have focused on developing agent architectures specifically optimised for medical diagnosis (Li et al., 2024a; Almansoori et al., 2025; Ghezloo et al., 2025; Wang et al., 2025l). One promising research direction focuses on multi-agent systems, which have shown strong potential for modelling the complexity and multi-step reasoning involved in medical diagnosis. These approaches can be broadly classified into two categories: *simulation-driven* and *collaborative designs*. Simulation-driven systems aim to reproduce real clinical settings by assigning specific roles to agents and enabling them to learn diagnostic strategies through interactions within a simulated medical environment. For instance, MedAgentSim (Almansoori et al., 2025) introduces a self-evolving simulation framework that integrates experience replay, chain-of-thought ensembling, and CLIP-based semantic memory to support diagnostic reasoning. PathFinder (Ghezloo et al., 2025) targets histopathological analysis by orchestrating multiple agents to emulate expert diagnostic workflows on gigapixel-scale medical images. In contrast, collaborative multi-agent systems focus on collective decision-making and collaboration among agents. For example, MDAgents (Kim et al., 2024) enables adaptive collaboration among multiple agents, where a moderator agent is responsible for integrating diverse suggestions and consulting external knowledge sources as needed. MDTeamGPT (Chen et al., 2025c) extends this paradigm to multidisciplinary consultation, supporting self-evolving, team-based diagnostic processes through reflective discussion mechanisms.

Another line of work on agent optimisation for diagnosis focuses on tool integration and multimodal reasoning. For instance, MMedAgent (Li et al., 2024a) addresses the generalisability limitations of existing multimodal LLMs by dynamically incorporating specialised medical tools across different modalities. To improve clinical reliability, MedAgent-Pro (Wang et al., 2025l) introduces diagnostic planning guided by established clinical criteria and integrates multimodal evidence via task-specific tool agents. In contrast to fixed agent architectures, recent work has explored more flexible designs that adapt based on diagnostic performance. For example, Zhuang et al. (2025) proposes a graph-based agent framework where the reasoning process is continuously adjusted using feedback from diagnostic results.

These approaches highlight specialisation, multimodality, and interactive reasoning as key principles for developing agent-based systems in medical diagnosis.

6.1.2 Molecular Discovery and Symbolic Reasoning

Molecular discovery within biomedical domains demands precise symbolic reasoning over chemical structures, reaction pathways, and pharmacological constraints (Bilodeau et al., 2022; Makke and Chawla, 2024; M. Bran et al., 2024). To support molecular discovery, recent agent-based systems have introduced tailored techniques such as integrating chemical analysis tools, enhancing memory for knowledge retention, and enabling multi-agent collaboration (McNaughton et al., 2024; Inoue et al., 2025). One key approach is domain-specific tool integration, which allows agents to perform chemical reasoning through interaction with executable chemical operations. For instance, CACTUS (McNaughton et al., 2024) equips agents with cheminformatics tools such as RDKit (Landrum, 2013) to ensure the generation of chemically valid outputs. By grounding reasoning in domain-specific toolsets, CACTUS achieves significantly better performance than agents without tool integration. Similarly, LLM-RDF (M. Bran et al., 2024) automates chemical synthesis by coordinating specialised agents, each responsible for a specific task and equipped with corresponding tools for literature mining, synthesis planning, or reaction optimisation.

Another prominent line of research leverages memory-enabled reasoning (Hu et al., 2025c; Inoue et al., 2025), where agents learn from prior experience by recording how previous problems were solved. ChemAgent (Tang et al., 2025a) breaks down complex chemical tasks into smaller subtasks, which are stored within a structured memory module, enabling efficient retrieval and refinement. OSDA Agent (Hu et al., 2025c) extends this approach by introducing a self-reflective mechanism, where failed molecule proposals are abstracted into structured memory updates that inform and enhance future decision-making. In parallel, multi-agent coordination provides additional benefits. DrugAgent (Inoue et al., 2025) introduces a coordinator architecture that integrates evidence from machine learning-based predictors, biomedical knowledge graphs, and literature search agents. It employs Chain-of-Thought and ReAct (Yao et al., 2023b) frameworks to support interpretable, multi-source reasoning. LIDDIA (Averly et al., 2025) generalises this design by assigning modular roles, i.e., reasoner, executor, evaluator, and memory, which collectively emulate iterative workflows in medicinal chemistry and facilitate multi-objective molecule evaluation.

6.2 Domain-Specific Optimisation in Programming

In the programming domain, agent optimisation focuses on aligning agent behaviours with the procedural and operational requirements of established software engineering workflows. Recent studies have demonstrated the effectiveness of domain-specific agent design in two key application areas: code refinement (Rasheed et al., 2024; Tang et al., 2024; Pan et al., 2025b) and code debugging (Lee et al., 2024a; Puvvadi et al., 2025; Adnan et al., 2025). In what follows, we examine representative agent optimisation strategies within these two domains.

6.2.1 Code Refinement

Code refinement involves the iterative improvement of code quality, structure, and correctness while preserving its original functionality (Yang et al., 2024d; He et al., 2025; Islam et al., 2025). Recent studies have increasingly investigated agent-based systems that support domain-specific optimisation for this task, focusing on self-improvement, collaborative workflows, and integration with programming tools (Madaan et al., 2023; Tang et al., 2024; Rahman et al., 2025). These systems are designed to emulate human-in-the-loop refinement processes, enforce adherence to software engineering best practices, and ensure that code remains robust, readable, and maintainable throughout iterative development cycles. One critical optimisation strategy involves self-feedback mechanisms, where agents critique and revise their own outputs. For example, Self-Refine (Madaan et al., 2023) introduces a lightweight framework in which a language model generates natural language feedback on its own outputs and subsequently revises the code accordingly. Similarly, CodeCriticBench (Zhang et al., 2025a) presents a comprehensive benchmark designed to assess the self-critiquing and refinement capabilities of LLMs, where agents are evaluated on their ability to identify, explain, and revise code defects through structured natural language feedback. LLM-Surgeon (van der Ouderaa et al., 2023) proposes a systematic framework in which a language model diagnoses structural and semantic issues within its own code outputs and applies targeted edits based on learned repair patterns, thereby optimising code quality while preserving functionality. These approaches eliminate the need for task-specific retraining, providing consistent improvements in code quality.

Another line of research explores experience-driven learning, where agents improve their problem-solving capabilities by relying on memory-enabled reasoning, systematically recording and reusing solutions to previously encountered tasks (Wang et al., 2025g; Tang et al., 2024; Pan et al., 2025b). For example, AgentCoder (Huang et al., 2023a) and CodeAgent (Tang et al., 2024) simulate collaborative development workflows by assigning specialised roles to individual agents, such as coder, reviewer, and tester, which iteratively improve code through structured dialogue cycles. These systems support collective evaluation and revision, promoting role specialisation and deliberative decision-making. Additionally, tool-enhanced frameworks such as CodeCoR (Pan et al., 2025b) and OpenHands (Wang et al., 2025g) incorporate external tools and modular agent interactions to facilitate dynamic code pruning, patch generation, and context-aware refinement. VFlow (Wei et al., 2025b) reformulates the workflow optimisation problem of Verilog code generation task as a search task on a graph of LLM nodes with code-based representations, employing a Cooperative Evolution with Past Experience MCTS (CEPE-MCTS) algorithm. These developments highlight iterative feedback, modular design, and interactive reasoning as essential principles for building adaptive agent-based systems for code refinement.

6.2.2 Code Debugging

Code debugging presents intricate challenges that require precise fault localisation, execution-aware reasoning, and iterative correction. These capabilities are typically absent in general-purpose LLMs (Puvvadi et al., 2025; Mannadiar and Vangheluwe, 2010). To address these challenges, domain-specific optimisation focuses on aligning agent roles and workflows with the structured reasoning patterns and tool usage observed in human debugging practices. A key strategy involves leveraging runtime feedback to facilitate self-correction. For example, Self-Debugging (Chen et al., 2024c) and Self-Edit (Zhang et al., 2023a) exemplify this approach by incorporating execution traces into the debugging process. These agents operate through internal cycles of fault identification, natural language-based reasoning, and targeted code revision, enabling autonomous debugging without external supervision.

Recent research has explored modular agent architectures specifically designed to support the multi-stage structure of debugging workflows. For instance, PyCapsule (Adnan et al., 2025) introduces a separation of responsibilities between a programmer agent and an executor agent, thereby distinguishing code generation from semantic validation. More advanced systems, including Self-Collaboration (Dong et al., 2024) and RGD (Jin et al., 2024), employ collaborative pipelines in which agents are assigned specialised roles such as tester, reviewer, or feedback analyser, mirroring professional debugging practices. Additionally, FixAgent (Lee et al., 2024a) extends this paradigm through hierarchical agent activation, dynamically dispatching different agents based on bug complexity and required depth of analysis.

6.3 Domain-Specific Optimisation in Financial and Legal Research

In financial and legal domains, agent optimisation focuses on tailoring multi-agent architectures, reasoning strategies, and tool integration to the procedural and operational demands of domain-specific workflows (Sun et al., 2024b; He et al., 2024; Li et al., 2025b). Recent studies have demonstrated the effectiveness of such domain-specific designs in two key application areas: financial decision-making (Li et al., 2023c; Yu et al., 2024b; Wang et al., 2024j) and legal reasoning (Di Martino et al., 2023; Chen et al., 2025a), where modular design, collaborative interaction, and rule-grounded reasoning are essential for reliable performance. In what follows, we examine representative agent optimisation strategies within these two domains.

6.3.1 Financial Decision-Making

Financial decision-making requires agents to operate under uncertain and rapidly changing conditions, reason over volatile market dynamics, and integrate heterogeneous information sources such as numerical indicators, news sentiment, and expert knowledge (Li et al., 2023c; Sarin et al., 2024; Chudziak and Wawer, 2025). In response to these domain-specific demands, recent research has focused on developing multi-agent architectures tailored to the procedural and cognitive requirements of financial environments (Fatemi and Hu, 2024; Luo et al., 2025b). One critical strategy involves conceptual and collaborative agent design. For instance, FinCon (Yu et al., 2024b) proposes a synthesised multi-agent system built on LLMs, employing conceptual verbal reinforcement and domain-adaptive fine-tuning to enhance decision stability and policy alignment in dynamic markets. PEER (Wang et al., 2024j) extends this paradigm through a modular agent architecture comprising expert, retriever, and controller roles, which interact under a unified tuning mechanism to balance task specialisation with general adaptability. FinRobot (Yang et al., 2024b) further advances this line of work by integrating external tools for model-grounded reasoning, enabling agents to connect high-level strategies with executable financial models and real-time data streams.

Another line of work on agent optimisation for financial decision-making focuses on sentiment analysis and reporting (Xing, 2025; Tian et al., 2025; Raza et al., 2025). Heterogeneous LLM agent architectures (Xing, 2025) enhance robustness in financial reporting by combining specialised sentiment modules with rule-based validators to ensure compliance with domain-specific guidelines. Similarly, template-based reporting frameworks (Tian et al., 2025) decompose report generation into agent-driven retrieval, validation, and synthesis stages, enabling iterative refinement through real-world feedback. These approaches demonstrate the potential of self-evolving multi-agent systems to provide reliable, interpretable, and context-aware decision support in complex financial environments.

6.3.2 Legal Reasoning

Legal reasoning requires agents to interpret structured legal rules, analyse case-specific evidence, and produce outputs that are consistent with institutional regulations and judicial standards (Xu and Ju, 2023; Yuan et al., 2024c; Jiang and Yang, 2025). To address these domain-specific demands, recent research has explored multi-agent systems tailored to the procedural and interpretive requirements of legal settings (Di Martino et al., 2023; Hu and Shu, 2023; Chen et al., 2025a). One significant direction involves collaborative agent frameworks that simulate judicial processes and support structured argumentation. For instance, LawLuo (Sun et al., 2024b) introduces a co-run multiagent architecture in which legal agents are assigned specialised roles such as document drafting, legal argument generation, and compliance validation, all operating under the supervision of a central controller to ensure procedural consistency and legal correctness. Multi-Agent Justice Simulation (Di Martino et al., 2023) and AgentCourt (Chen et al., 2025a) extend this paradigm to model adversarial trial procedures, enabling agents to participate in role-based interactions that emulate real-world courtroom dynamics. In particular, AgentCourt incorporates self-evolving lawyer agents that refine their strategies through reflective self-play, leading to improved debate quality and enhanced procedural realism.

Another line of work focuses on structured legal reasoning and domain-grounded interpretability. LegalGPT (Shi et al., 2024b) integrates a legal chain-of-thought framework within a multi-agent system, guiding legal reasoning through interpretable and rule-aligned steps. Similarly, AgentsCourt (He et al., 2024) combines courtroom debate simulation with legal knowledge augmentation, enabling agents to perform judicial decision-making grounded in codified rules and case precedents. These approaches highlight the importance of rule grounding, modular role design, and collaborative reasoning in the development of robust, transparent, and legally reliable agent systems.

7 Evaluation

The rapid emergence of autonomous LLM-based agents has underscored the need for rigorous, multidimensional evaluation frameworks. As these agents are deployed across increasingly diverse tasks and environments, recent research has introduced a range of benchmarks and methodologies to assess not only task completion but also reasoning quality, generalisation ability, and compliance with safety and alignment standards. Evaluation is no longer viewed as a static endpoint but as a dynamic feedback mechanism: fine-grained performance signals are now used to guide agent optimisation, prompt refinement, and dataset augmentation, enabling self-evolving systems that continuously acquire new capabilities and address failure cases. Current evaluation paradigms encompass structured benchmark tasks with standardised metrics, safety- and alignment-oriented audits, and LLM-as-a-judge approaches that leverage large models as flexible, scalable evaluators.

7.1 Benchmark-based Evaluation

7.1.1 Tool and API-Driven Agents

Tool-augmented agents are evaluated based on their ability to invoke external APIs and functions to solve problems that exceed the scope of their intrinsic knowledge. Benchmarks such as ToolBench (Xu et al., 2023), API-Bank (Li et al., 2023b), MetaTool (Huang et al., 2023b), and ToolQA (Zhuang et al., 2023) define tasks that require tool usage and assess both the correctness and efficiency of API calls. Many of these evaluations employ simulated APIs or sandboxed environments, measuring task success alongside interaction efficiency. Early studies have shown that agents often overfit to specific tool schemas, exhibiting limited generalisation to previously unseen APIs. To address this limitation, recent benchmarks such as GTA (Wang et al., 2024b) and AppWorld (Trivedi et al., 2024) introduce more realistic, multi-step tasks that require planning and coordination across multiple tools, while placing greater emphasis on process-oriented evaluation metrics. This trend reflects a broader shift towards richer, reasoning-aware evaluations that assess not only final outcomes but also the quality of the decision-making process.

7.1.2 Web Navigation and Browsing Agents

Web agents are evaluated on their ability to interact with websites, extract information, and complete real-world online tasks. Benchmarks such as BrowseComp (Wei et al., 2025a), WebArena (Zhou et al., 2023b), VisualWebArena (Koh et al., 2024), WebCanvas (Pan et al., 2024b), WebWalker (Wu et al., 2025b), and AgentBench (Liu et al., 2023a) have progressively increased the realism and diversity of web-based evaluations, spanning simulated and live environments. These benchmarks test navigation skills, adaptability to interface changes, and the integration of textual and visual information. Recent work incorporates intermediate metrics (e.g., sub-goal completion) and robustness assessments, though reproducibility and generalisation remain challenging due to the dynamic nature of the web.

7.1.3 Multi-Agent Collaboration and Generalists

As agents become more general-purpose, new benchmarks target multi-agent coordination and cross-domain competence. MultiAgentBench (Zhu et al., 2025) and SwarmBench (Ruan et al., 2025) evaluate collaboration, competition, and decentralised coordination among LLM agents, assessing both task completion and the quality of communication and strategy. Generalist benchmarks such as GAIA (Mialon et al., 2023) and AgentBench (Liu et al., 2023a) test adaptability across diverse environments, from web navigation to coding and database queries. Recent work (Wang et al., 2025b) further explores the GAIA benchmark to analyse the efficiency–effectiveness trade-off in agentic systems, proposing Efficient Agents, a framework that achieves competitive performance with significantly reduced operational costs. These evaluations highlight challenges in aggregating metrics across heterogeneous tasks, risks of overfitting to narrow scenarios, and the need for unified, holistic leaderboards.

7.1.4 GUI and Multimodal Environment Agents

GUI and multimodal benchmarks challenge agents to operate in rich, interactive environments that combine textual and visual inputs. Mobile-Bench (Deng et al., 2024), AndroidWorld (Rawles et al., 2024), CRAB (Xu et al., 2024a), GUI-World (Chen et al., 2024a), and OSWorld (Xie et al., 2024) simulate realistic apps and operating systems, requiring complex action sequences. Tasks often combine natural language understanding, visual perception, and API invocation. Evaluations measure task success, state management, perception accuracy, and adaptability to GUI changes. However, the diversity of GUI environments makes standardisation and reproducibility difficult, and agents remain brittle when faced with interface variability.

7.1.5 Domain-Specific Task Agents

Domain-focused benchmarks in coding (SWE-bench (Jimenez et al., 2024)), data science (DataSciBench (Zhang et al., 2025c), MLGym (Nathani et al., 2025)), enterprise productivity (WorkBench (Styles et al., 2024)), and scientific research (OpenAGI (Ge et al., 2023), SUPER (Bogin et al., 2024)) assess specialised competencies that integrate planning, tool use, and adherence to domain norms. SWE-bench, for example, evaluates code-editing agents on real GitHub repositories, while AgentClinic (Schmidgall et al., 2024) and MMedAgent (Li et al., 2024a) test multimodal reasoning in clinical settings. Evaluation criteria have expanded from binary success measures to encompass metrics such as test pass rates, policy adherence, and conformity to domain-specific constraints. Despite these advances, inconsistencies in metric definitions and persistent gaps in generalisation remain significant challenges.

7.2 LLM-based Evaluation

7.2.1 LLM-as-a-Judge

The LLM-as-a-Judge paradigm refers to employing large language models to assess the quality of outputs generated by AI systems, such as text, code, or conversational responses, via structured prompts (Arabzadeh et al., 2024; Li et al., 2024b; Qian et al., 2025b). This approach has attracted attention as a scalable and cost-effective alternative to conventional evaluation methods, including human judgment and automatic metrics (e.g., BLEU, ROUGE), which often fail to capture semantic depth or coherence (Arabzadeh et al., 2024). LLM judges typically operate in two modes: *pointwise* evaluation (Ruan et al., 2024), where outputs are scored directly against criteria such as factuality and helpfulness, and *pairwise* comparison, where two outputs are compared and the preferred one is selected with justification (Li et al., 2024b; Zhao et al., 2025b).

Recent studies demonstrate that LLM-based evaluations can correlate with human judgments, in some cases reaching parity with inter-annotator agreement levels (Arabzadeh et al., 2024). However, these methods are sensitive to prompt design and susceptible to biases introduced by subtle instructional variations (Arabzadeh et al., 2024; Zhao et al., 2025b). Furthermore, single-step, output-focused evaluations may overlook the reasoning depth in multi-step processes (Zhuge et al., 2024b; Wang et al., 2025h). To address these limitations, enhancements have been proposed, including multi-agent deliberation frameworks such as CollabEval (Qian et al., 2025b) and structured meta-evaluation benchmarks to calibrate and improve the reliability of LLM judges (Li et al., 2024b; Zhao et al., 2025b).

7.2.2 Agent-as-a-Judge

The Agent-as-a-Judge framework extends LLM-based evaluation by employing full-fledged agentic systems capable of multi-step reasoning, state management, and tool use to critique other AI agents (Zhuge et al., 2024b; Zhao et al., 2025b; Qian et al., 2025b). Different from traditional LLM judges, which focus solely on final outputs, agent judges evaluate the entire reasoning trajectory, capturing decision-making processes and intermediate actions (Zhuge et al., 2024b). For example, Zhuge et al. (2024b) applied an agent judge to the DevAI benchmark for code-generation agents. The framework incorporated specialised modules to analyse intermediate artefacts, construct reasoning graphs, and validate hierarchical requirements, resulting in evaluations that aligned more closely with human expert judgments than traditional LLM-based approaches. Agent judges also delivered substantial efficiency gains, reducing evaluation time and cost relative to manual review (Zhuge et al., 2024b; Zhao et al., 2025b).

Nevertheless, implementing the Agent-as-a-Judge methodology introduces additional complexity and raises challenges for generalisation to domains other than code generation. Current research seeks to improve adaptability and simplify deployment across a broader range of AI tasks (Zhao et al., 2025b; Qian et al., 2025b).

7.3 Safety, Alignment, and Robustness in Lifelong Self-Evolving Agents

In the context of the Three Laws of Self-Evolving AI Agents, **Endure**, the maintenance of safety and stability during any modification, forms the primary constraint on all other forms of adaptation. For lifelong, self-evolving agentic systems, safety is not a one-off certification but an ongoing requirement: every evolution step, from prompt updates to topology changes, must be assessed for unintended or malicious behaviours. This necessitates evaluation protocols that are *continuous*, *granular*, and *scalable*, ensuring that agents can remain aligned while adapting over extended lifetimes.

Recent work has introduced diverse evaluation paradigms. Risk-focused benchmarks such as AgentHarm (Andriushchenko et al., 2025) measure an agent's propensity to comply with explicitly malicious multi-step requests—requiring coherent tool use to execute harmful objectives such as fraud or cybercrime, revealing that even leading LLMs can be coaxed into complex unsafe behaviours with minimal prompting. Domain-specific probes such as RedCode (Guo et al., 2024a) (code security) and MobileSafetyBench (Lee et al., 2024c) (mobile control) stress-test agents in realistic, sandboxed environments. Behavioural probes like MACHIAVELLI (Pan et al., 2023) explore whether agents develop unethical, power-seeking strategies under reward optimisation, highlighting the interplay between **Endure** and **Excel**, safe adaptation must not degrade core task competence.

Meta-evaluation approaches, e.g., Agent-as-a-Judge (Zhuge et al., 2024b), AgentEval (Arabzadeh et al., 2024), and R-Judge (Yuan et al., 2024b) – position LLM agents themselves as evaluators or safety monitors, offering scalable oversight but also exposing the limitations of current "risk awareness." These studies underline the multi-dimensional nature of safety, where accuracy alone is insufficient; over-reliance on correctness metrics can conceal epistemic risks and systemic biases (Li et al., 2025f). Legal alignment tests such as SafeLawBench (Cao et al., 2025) further show that even state-of-the-art models struggle to satisfy established legal principles, reflecting the difficulty of codifying alignment in domains with open-textured norms.

Despite these advances, most current evaluations are *snapshot-based*, assessing agents at a single point in time. For MASE systems, safety evaluation must itself become dynamic – continuously monitoring, diagnosing, and correcting behaviours as the system evolves. Developing longitudinal, evolution-aware benchmarks that track safety, alignment, and robustness across the full lifecycle of an agent ecosystem remains an open and urgent challenge.

8 Challenges and Future Directions

Despite rapid advances, the evolution and optimisation of AI agents still face fundamental obstacles. These challenges are closely tied to the Three Laws of Self-Evolving AI Agents and need to be addressed to realise the vision of lifelong agentic systems. We group the key open problems accordingly.

8.1 Challenges

8.1.1 Endure – Safety Adaptation

(1) **Safety, Regulation, and Alignment.** Most optimisation pipelines prioritise task metrics over safety constraints, neglecting risks such as unintended behaviours, privacy breaches, and misaligned objectives. The dynamic nature of evolving agents undermines existing legal frameworks (e.g., EU AI Act, GDPR), which assume static models and fixed decision logic. This calls for new evolution-aware audit mechanisms, adaptive licences, provable-safety sandboxes, and legal protocols capable of tracking and constraining an agent's self-directed evolutionary path.

(2) **Reward Modelling and Optimisation Instability.** Learned reward models for intermediate reasoning steps often suffer from dataset scarcity, noisy supervision, and feedback inconsistency, leading to unstable or

divergent agent behaviours. Stability is central to safety: even small perturbations in inputs or update rules can undermine the trustworthiness of an evolving workflow.

8.1.2 Excel – Performance Preservation

(1) **Evaluation in Scientific and Domain-Specific Scenarios.** In domains like biomedicine or law, reliable ground truth is often absent or disputed, complicating the construction of trustworthy feedback signals for optimisation.

(2) **Balancing Efficiency and Effectiveness in MAS Optimisation.** Large-scale multi-agent optimisation improves task performance but incurs significant computational cost, latency, and instability. Designing methods that explicitly trade off effectiveness against efficiency remains unresolved.

(3) **Transferability of Optimised Prompts and Topologies.** Optimised prompts or agent topologies are often brittle, showing poor generalisation across LLM backbones with differing reasoning abilities. This undermines scalability and reusability in production settings.

8.1.3 Evolve – Autonomous Optimisation

(1) **Optimisation in Multimodal and Spatial Environments.** Most optimisation algorithms are text-only, yet real-world agents must process multimodal inputs and reason in spatially grounded or continuous environments. This demands internal world models and perceptual–temporal reasoning.

(2) **Tool Use and Creation.** Current methods typically assume a fixed toolset, overlooking the autonomous discovery, adaptation, and co-evolution of tools alongside agents.

8.2 Future Directions

Looking forward, many of these limitations point to promising research avenues. We highlight several directions and link them to their role in the **MOP→MOA→MAO→MASE** paradigm shift.

(1) **Simulated Environments for Fully Autonomous Self-Evolution (MASE).** Develop open-ended, interactive simulation platforms where agents can iteratively interact, receive feedback, and refine prompts, memory, tools, and workflows via closed-loop optimisation.

(2) **Advancing Tool Use and Creation (MAO→MASE).** Move beyond static tool invocation toward agents that adaptively select, compose, or create tools. Incorporate reinforcement learning and feedback-driven strategies, paired with robust evaluation pipelines.

(3) **Real-World Evaluation and Benchmarking (Cross-stage).** Create benchmarks and protocols that reflect real-world complexity, support interaction-based and longitudinal assessment, and align with long-term improvement signals.

(4) **Effectiveness–Efficiency Trade-offs in MAS Optimisation (MAO).** Design optimisation algorithms that jointly model performance and resource constraints, enabling MAS deployment under strict latency, cost, or energy budgets.

(5) **Domain-Aware Evolution for Scientific and Specialised Applications (MASE).** Tailor evolution methods to domain-specific constraints in science, medicine, law, or education, integrating heterogeneous knowledge sources, bespoke evaluation criteria, and regulatory compliance.

Outlook. Addressing these challenges will require optimisation pipelines that are not only high-performing and domain-adaptive, but also safe, regulation-aware, and self-sustaining. Embedding these solutions within the **MOP→MOA→MAO→MASE** trajectory, and grounding them in the THREE LAWS OF SELF-EVOLVING AI AGENTS, offers a coherent roadmap toward truly *lifelong, autonomous agentic systems* – systems that can **endure**, **excel**, and **evolve** across the full span of their operational lifetimes.

9 Conclusions

In this survey, we have presented a comprehensive overview of the emerging paradigm of self-evolving AI agents, which bridge the static capabilities of foundation models with the continuous adaptability required by lifelong agentic systems. We situated this evolution within a unified four-stage trajectory, from Model Offline Pretraining (MOP) and Model Online Adaptation (MOA), through Multi-Agent Orchestration (MAO), and ultimately to Multi-Agent Self-Evolving (MASE), highlighting the progressive shift from static, human-configured models to dynamic, autonomous ecosystems.

To formalise this transition, we introduced a conceptual framework that abstracts the feedback loop underlying agent evolution, with four key components: Inputs, Agent System, Objectives, and Optimisers, that together determine how agents improve through continual interaction with their environment. Building on this, we systematically reviewed optimisation techniques across agent components, domain-specific strategies, and evaluation methodologies critical for building adaptive and resilient agentic systems.

We also proposed the Three Laws of Self-Evolving AI Agents, Endure (safety adaptation), Excel (performance preservation), and Evolve (autonomous evolution), as guiding principles to ensure that lifelong self-improvement remains safe, effective, and aligned. These laws are not mere principles but practical design constraints, ensuring that the path toward autonomy remains aligned with safety, performance, and adaptability. They serve as the guardrails for the MASE paradigm, guiding research from narrow, single-shot optimisation toward continuous, open-ended self-improvement.

Looking forward, the ability to endure, excel, and evolve will be decisive for agents operating in dynamic, real-world environments, whether in scientific discovery, software engineering, or human–AI collaboration. Achieving this will demand breakthroughs in scalable optimisation algorithms, lifelong evaluation protocols, safe coordination in heterogeneous agent environments, and mechanisms for adapting to unforeseen domains.

We hope this survey serves as both a reference point and a call to action to build an ecosystem of self-evolving AI agents that do not simply execute tasks, but live, learn, and last. By aligning technical innovation with principled self-evolution, we can pave the way toward truly autonomous, resilient, and trustworthy lifelong agentic systems.

Acknowledgements

We would like to thank Shuyu Guo for his valuable contributions to the early-stage exploration and literature review on agent optimisation.

References

Muntasir Adnan, Zhiwei Xu, and Carlos CN Kuhn. Large language model guided self-debugging code generation. *arXiv preprint arXiv:2502.02928*, 2025.

Eshaan Agarwal, Joykirat Singh, Vivek Dani, Raghav Magazine, Tanuja Ganu, and Akshay Nambi. Promptwizard: Task-aware prompt optimization framework. *arXiv preprint arXiv:2405.18369*, 2024.

Keivan Alizadeh, Seyed-Iman Mirzadeh, Dmitry Belenko, S. Khatamifard, Minsik Cho, Carlo C. del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. LLM in a flash: Efficient large language model inference with limited memory. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12562–12584, 2024.

Mohammad Almansoori, Komal Kumar, and Hisham Cholakkal. Self-evolving multi-agent simulations for realistic clinical interactions. *arXiv preprint arXiv:2503.22678*, 2025.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, J. Zico Kolter, Matt Fredrikson, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

Negar Arabzadeh, Julia Kiseleva, Qingyun Wu, Chi Wang, Ahmed Awadallah, Victor Dibia, Adam Fourney, and Charles Clarke. Towards better human-agent alignment: Assessing task utility in llm-powered applications. *arXiv preprint arXiv:2402.09015*, 2024.

Derek Austin and Elliott Chartock. GRAD-SUM: Leveraging gradient summarization for optimal prompt engineering. *arXiv preprint arXiv:2407.12865*, 2024.

Reza Averly, Frazier N Baker, and Xia Ning. LIDDIA: Language-based intelligent drug discovery agent. *arXiv preprint arXiv:2502.13959*, 2025.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, pages 17682–17690, 2024.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing LLM reasoning. In *Forty-second International Conference on Machine Learning*, 2025.

Camille Bilodeau, Wengong Jin, Tommi Jaakkola, Regina Barzilay, and Klavs F Jensen. Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 12(5): e1608, 2022.

Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. Reflective multi-agent collaboration based on large language models. *Advances in Neural Information Processing Systems*, 37: 138595–138631, 2024.

Ben Bogin, Kejuan Yang, Shashank Gupta, Kyle Richardson, Erin Bransom, Peter Clark, Ashish Sabharwal, and Tushar Khot. SUPER: evaluating agents on setting up and executing tasks from research repositories. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12622–12645, 2024.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *The Twelfth International Conference on Learning Representations*, 2024.

CAMEL-AI. Workforce — camel-ai documentation. https://docs.camel-ai.org/key_modules/workforce, 2025. Accessed: 2025-08-09.

Chuxue Cao, Han Zhu, Jiaming Ji, Qichao Sun, Zhenghao Zhu, Yinyu Wu, Josef Dai, Yaodong Yang, Sirui Han, and Yike Guo. SafeLawBench: Towards safe alignment of large language models. In *Findings of the Association for Computational Linguistics*, pages 14015–14048, 2025.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, 2021.

Shuyang Cao and Lu Wang. AWESOME: GPU memory-constrained long document summarization using memory mechanism and global salient content. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5925–5941, 2024.

Edward Y Chang. Socrasynth: Multi-llm reasoning with conditional statistics. *arXiv preprint arXiv:2402.06634*, 2024.

GaoWei Chang and Agent Network Protocol Contributors. Agent Network Protocol (ANP). https://github.com/agent-network-protocol/AgentNetworkProtocol. MIT License, accessed 2025-07-31.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. In *The Eleventh International Conference on Learning Representations*, 2023.

Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Huichi Zhou, Qihui Zhang, Zhigang He, Yilin Bai, Chujie Gao, Liuyi Chen, et al. GUI-world: A video benchmark and dataset for multimodal GUI-oriented understanding. In *The Thirteenth International Conference on Learning Representations*, 2024a.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje Karlsson, Jie Fu, and Yemin Shi. AutoAgents: A framework for automatic agent generation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 22–30, 2024b.

Guhong Chen, Liyang Fan, Zihan Gong, Nan Xie, Zixuan Li, Ziqiang Liu, Chengming Li, Qiang Qu, Hamid Alinejad-Rokny, Shiwen Ni, and Min Yang. AgentCourt: Simulating court with adversarial evolvable lawyer agents. In *Findings of the Association for Computational Linguistics*, pages 5850–5865. Association for Computational Linguistics, 2025a.

Guoxin Chen, Zhong Zhang, Xin Cong, Fangda Guo, Yesai Wu, Yankai Lin, Wenzheng Feng, and Yasheng Wang. Learning evolving tools for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Kai Chen, Xinfeng Li, Tianpei Yang, Hewei Wang, Wei Dong, and Yang Gao. MDTeamGPT: A self-evolving llm-based multi-agent framework for multi-disciplinary team medical consultation. *arXiv preprint arXiv:2503.13856*, 2025c.

Mingda Chen, Yang Li, Karthik Padthe, Rulin Shao, Alicia Yi Sun, Luke Zettlemoyer, Gargi Ghosh, and Wen-tau Yih. Improving factuality with explicit working memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11199–11213, 2025d.

Mingyang Chen, Haoze Sun, Tianpeng Li, Fan Yang, Hao Liang, KeerLu, Bin CUI, Wentao Zhang, Zenan Zhou, and Weipeng Chen. Facilitating multi-turn function calling for LLMs via compositional instruction tuning. In *The Thirteenth International Conference on Learning Representations*, 2025e.

Nuo Chen, Hongguang Li, Jianhui Chang, Juhua Huang, Baoyuan Wang, and Jia Li. Compress to impress: Unleashing the potential of compressive memory in real-world long-term conversations. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 755–773, 2025f.

Weize Chen, Ziming You, Ran Li, Yitong Guan, Chen Qian, Chenyang Zhao, Cheng Yang, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Internet of agents: Weaving a web of heterogeneous agents for collaborative intelligence. In *The Thirteenth International Conference on Learning Representations*, 2025g.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for llm-based multi-agent system. In *Findings of the Association for Computational Linguistics*, pages 11534–11557. Association for Computational Linguistics, 2025h.

Xi Chen, Huahui Yi, Mingke You, WeiZhi Liu, Li Wang, Hairui Li, Xue Zhang, Yingman Guo, Lei Fan, Gang Chen, et al. Enhancing diagnostic capability with multi-agents conversational large language models. *NPJ digital medicine*, 8(1):159, 2025i.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to Self-Debug. In *The Twelfth International Conference on Learning Representations*, 2024c.

Yuyang Cheng, Yumiao Xu, Chaojia Yu, and Yong Zhao. HAWK: A hierarchical workflow framework for multi-agent collaboration. *arXiv preprint arXiv:2507.04067*, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Jarosław A Chudziak and Michał Wawer. ElliottAgents: a natural language-driven multi-agent system for stock market analysis and prediction. *arXiv preprint arXiv:2507.03435*, 2025.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, 2022.

Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, Rui Yan, and Shuo Shang. Mobile-Bench: An evaluation benchmark for llm-based mobile agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8813–8831, 2024.

Beniamino Di Martino, Antonio Esposito, and Luigi Colucci Cante. Multi agents simulation of justice trials to support control management and reduction of civil trials duration. *Journal of Ambient Intelligence and Humanized Computing*, 14(4):3645–3657, 2023.

Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. Self-collaboration code generation via chatgpt. *ACM Transactions on Software Engineering and Methodology*, 33(7):1–38, 2024.

Norbert Donner-Banzhoff. Solving the diagnostic challenge: a patient-centered approach. *The Annals of Family Medicine*, 16(4):353–358, 2018.

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z Pan. Rethinking memory in ai: Taxonomy, operations, topics, and future directions. *arXiv preprint arXiv:2505.00675*, 2025.

Yu Du, Fangyun Wei, and Hongyang Zhang. AnyTool: Self-reflective, hierarchical agents for large-scale API calls. In *Forty-first International Conference on Machine Learning*, 2024.

Sefika Efeoglu and Adrian Paschke. Retrieval-augmented generation-based relation extraction. *arXiv preprint arXiv:2404.13397*, 2024.

Sugyeong Eo, Hyeonseok Moon, Evelyn Hayoon Zi, Chanjun Park, and Heuiseok Lim. Debate only when necessary: Adaptive multiagent collaboration for efficient llm reasoning. *arXiv preprint arXiv:2504.05047*, 2025.

Adibvafa Fallahpour, Jun Ma, Alif Munim, Hongwei Lyu, and Bo Wang. Medrax: Medical reasoning agent for chest x-ray. *arXiv preprint arXiv:2502.02673*, 2025.

Wei Fang, Yang Zhang, Kaizhi Qian, James Glass, and Yada Zhu. Play2prompt: Zero-shot tool instruction optimization for llm agents via tool play. *arXiv preprint arXiv:2503.14432*, 2025.

Sorouralsadat Fatemi and Yuheng Hu. Finvision: A multi-agent framework for stock market prediction. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 582–590, 2024.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025a.

Jinghao Feng, Qiaoyu Zheng, Chaoyi Wu, Ziheng Zhao, Ya Zhang, Yanfeng Wang, and Weidi Xie. M^{3}builder: A multi-agent system for automated machine learning in medical imaging. *arXiv preprint arXiv:2502.20301*, 2025b.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Forty-first International Conference on Machine Learning*, 2024.

Emily First, Markus N Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-proof generation and repair with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1229–1241, 2023.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

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

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025b.

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 18030–18038, 2024.

Yunfan Gao, Yun Xiong, Yijie Zhong, Yuxi Bi, Ming Xue, and Haofen Wang. Synergizing RAG and reasoning: A systematic review. *arXiv preprint arXiv:2504.15909*, 2025c.

Yingqiang Ge, Wenyue Hua, Kai Mei, Juntao Tan, Shuyuan Xu, Zelong Li, Yongfeng Zhang, et al. OpenAGI: When llm meets domain experts. *Advances in Neural Information Processing Systems*, 36:5539–5568, 2023.

Caleb Geren, Amanda Board, Gaby G. Dagher, Tim Andersen, and Jun Zhuang. Blockchain for large language model security and safety: A holisticsurvey. *SIGKDD Explorations*, 26(2):1–20, 2024.

Fatemeh Ghezloo, Mehmet Saygin Seyfioglu, Rustin Soraki, Wisdom O Ikezogwo, Beibin Li, Tejoram Vivekanandan, Joann G Elmore, Ranjay Krishna, and Linda Shapiro. Pathfinder: A multi-modal multi-agent system for medical diagnostic decision-making applied to histopathology. *arXiv preprint arXiv:2502.08916*, 2025.

Anna Goldie, Azalia Mirhoseini, Hao Zhou, Irene Cai, and Christopher D.Manning Manning. Synthetic data generation & multi-step RL for reasoning & tool use. *arXiv preprint arXiv:2504.04736*, 2025.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zhouhong Gu, Xiaoxuan Zhu, Yin Cai, Hao Shen, Xingzhou Chen, Qingyi Wang, Jialin Li, Xiaoran Shi, Haoran Guo, Wenxuan Huang, et al. AgentGroupChat-V2: Divide-and-conquer is what llm-based multi-agent system need. *arXiv preprint arXiv:2506.15451*, 2025.

Chengquan Guo, Xun Liu, Chulin Xie, Andy Zhou, Yi Zeng, Zinan Lin, Dawn Song, and Bo Li. Redcode: Risky code execution and generation benchmark for code agents. *Advances in Neural Information Processing Systems*, 37: 106190–106236, 2024a.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. EvoPrompt: Connecting llms with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2024b.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8048–8057, 2024c.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics*, pages 11143–11156, 2024d.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, 2024.

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

Junda He, Christoph Treude, and David Lo. LLM-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–30, 2025.

Zhitao He, Pengfei Cao, Chenhao Wang, Zhuoran Jin, Yubo Chen, Jiexin Xu, Huaijun Li, Xiaojian Jiang, Kang Liu, and Jun Zhao. AgentsCourt: Building judicial decision-making agents with court debate simulation and legal knowledge augmentation. *arXiv preprint arXiv:2403.02959*, 2024.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for A multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.

Yuki Hou, Haruki Tamoto, and Homei Miyashita. “my agent understands me better”: Integrating dynamic human-like memory recall and consolidation in llm-based agents. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI ’24, page 1–7. ACM, May 2024.

Zhipeng Hou, Junyi Tang, and Yipeng Wang. Halo: Hierarchical autonomous logic-oriented orchestration for multi-agent llm systems. *arXiv preprint arXiv:2505.13516*, 2025.

Cho-Jui Hsieh, Si Si, Felix X. Yu, and Inderjit S. Dhillon. Automatic engineering of long prompts. In *Findings of the Association for Computational Linguistics*, pages 10672–10685, 2024.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. ChatDB: Augmenting llms with databases as their symbolic memory. *arXiv preprint arXiv:2306.03901*, 2023.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*, 2022.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Wenyang Hu, Yao Shu, Zongmin Yu, Zhaoxuan Wu, Xiaoqiang Lin, Zhongxiang Dai, See-Kiong Ng, and Bryan Kian Hsiang Low. Localized zeroth-order prompt optimization. *Advances in Neural Information Processing Systems*, 37:86309–86345, 2024.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, et al. OS agents: A survey on mllm-based agents for computer, phone and browser use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7436–7465, 2025b.

Zhaolin Hu, Yixiao Zhou, Zhongan Wang, Xin Li, Weimin Yang, Hehe Fan, and Yi Yang. Osda agent: Leveraging large language models for de novo design of organic structure directing agents. In *The Thirteenth International Conference on Learning Representations*, 2025c.

Zhiting Hu and Tianmin Shu. Language models, agent models, and world models: The law for machine reasoning and planning. *arXiv preprint arXiv:2312.05230*, 2023.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025.

Dong Huang, Jie M Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. AgentCoder: Multi-agent-based code generation with iterative testing and optimisation. *arXiv preprint arXiv:2312.13010*, 2023a.

Jen-tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Michael R Lyu, and Maarten Sap. On the resilience of llm-based multi-agent collaboration with faulty agents. *arXiv preprint arXiv:2408.00989*, 2024a.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024b.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, et al. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*, 2023b.

Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. O1 replication journey–part 2: Surpassing o1-preview through simple distillation, big progress or bitter lesson? *arXiv preprint arXiv:2411.16489*, 2024c.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Yoshitaka Inoue, Tianci Song, Xinling Wang, Augustin Luna, and Tianfan Fu. Drugagent: Multi-agent large language model-based reasoning for drug-target interaction prediction. In *ICLR 2025 Workshop on Machine Learning for Genomics Explorations*, 2025.

Md. Ashraful Islam, Mohammed Eunus Ali, and Md. Rizwan Parvez. MapCoder: Multi-agent code generation for competitive problem solving. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 4912–4944, 2024.

Md. Ashraful Islam, Mohammed Eunus Ali, and Md. Rizwan Parvez. CodeSim: Multi-agent code generation and problem solving through simulation-driven planning and debugging. In *Findings of the Association for Computational Linguistics: NAACL*, 2025.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *CoRR*, 2024.

Cong Jiang and Xiaolei Yang. Agentsbench: A multi-agent llm simulation framework for legal judgment prediction. *Systems*, 13(8):641, 2025.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024.

Fangkai Jiao, Chengwei Qin, Zhengyuan Liu, Nancy F. Chen, and Shafiq Joty. Learning planning-based reasoning by trajectories collection and process reward synthesizing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 334–350, 2024.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024.

Haolin Jin, Zechao Sun, and Huaming Chen. RGD: Multi-LLM based agent debugger via refinement and generation guidance. In *2024 IEEE International Conference on Agents (ICA)*, pages 136–141. IEEE, 2024.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. Disentangling memory and reasoning ability in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1681–1701, 2025.

Zixuan Ke, Austin Xu, Yifei Ming, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. MAS-ZERO: Designing multi-agent systems with zero supervision. *arXiv preprint arXiv:2505.14996*, 2025.

Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R. Bowman, Tim Rocktäschel, and Ethan Perez. Debating with more persuasive llms leads to more truthful answers. In *Forty-first International Conference on Machine Learning*, 2024.

Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik S Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae W Park. MDAgents: An adaptive collaboration of llms for medical decision-making. *Advances in Neural Information Processing Systems*, 37:79410–79452, 2024.

Ronny Ko, Jiseong Jeong, Shuyuan Zheng, Chuan Xiao, Tae-Wan Kim, Makoto Onizuka, and Won-Yong Shin. Seven security challenges that must be solved in cross-domain multi-agent llm systems. *arXiv preprint arXiv:2505.23847*, 2025.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905, 2024.

Dezhang Kong, Shi Lin, Zhenhua Xu, Zhebo Wang, Minghao Li, Yufeng Li, Yilun Zhang, Hujin Peng, Zeyang Sha, Yuyuan Li, Changting Lin, Xun Wang, Xuan Liu, Ningyu Zhang, Chaochao Chen, Muhammad Khurram Khan, and Meng Han. A survey of llm-driven ai agent communication: Protocols, security risks, and defense countermeasures, 2025.

Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in Medicine*, 23(1):89–109, 2001. ISSN 0933-3657. doi: https://doi.org/10.1016/S0933-3657(01)00077-X.

Naveen Krishnan. Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications. *arXiv preprint arXiv:2504.21030*, 2025.

Bespoke Labs. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation, 2025. Accessed: 2025-01-22.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. AutoWebGLM: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5295–5306. ACM, 2024a.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-DPO: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*, 2024b.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Greg Landrum. Rdkit documentation. *Release*, 1(1-79):4, 2013.

Cheryl Lee, Chunqiu Steven Xia, Longji Yang, Jen-tse Huang, Zhouruixin Zhu, Lingming Zhang, and Michael R Lyu. A unified debugging approach via llm-based multi-agent synergy. *arXiv preprint arXiv:2404.17153*, 2024a.

Dongkyu Lee, Chandana Satya Prakash, Jack FitzGerald, and Jens Lehmann. MATTER: memory-augmented transformer using heterogeneous knowledge sources. In *Findings of the Association for Computational Linguistics*, pages 16110–16121, 2024b.

Juyong Lee, Dongyoon Hahm, June Suk Choi, W Bradley Knox, and Kimin Lee. MobileSafetyBench: Evaluating safety of autonomous agents in mobile device control. *arXiv preprint arXiv:2410.17520*, 2024c.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John F. Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In *Forty-first International Conference on Machine Learning*, 2024d.

Hui Yi Leong and Yuqing Wu. DynaSwarm: Dynamically graph structure selection for llm-based multi-agent system. *arXiv preprint arXiv:2507.23261*, 2025.

Binxu Li, Tiankai Yan, Yuanting Pan, Jie Luo, Ruiyang Ji, Jiayuan Ding, Zhe Xu, Shilong Liu, Haoyu Dong, Zihao Lin, et al. MMedAgent: Learning to use medical tools with multi-modal agent. *arXiv preprint arXiv:2407.02483*, 2024a.

Boyi Li, Zhonghan Zhao, Der-Horng Lee, and Gaoang Wang. Adaptive graph pruning for multi-agent communication. *arXiv preprint arXiv:2506.02951*, 2025a.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for ”mind” exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. LLMs-as-Judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*, 2024b.

Junkai Li, Yunghwei Lai, Weitao Li, Jingyi Ren, Meng Zhang, Xinhui Kang, Siyu Wang, Peng Li, Ya-Qin Zhang, Weizhi Ma, et al. Agent hospital: A simulacrum of hospital with evolvable medical agents. *arXiv preprint arXiv:2405.02957*, 2024c.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, 2023b.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, Wenbo Su, and Bo Zheng. GraphReader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 12758–12786, 2024d.

Xiangyu Li, Yawen Zeng, Xiaofen Xing, Jin Xu, and Xiangmin Xu. HedgeAgents: A balanced-aware multi-agent financial trading system. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 296–305, 2025b.

Xiaonan Li and Xipeng Qiu. MoT: Memory-of-thought enables chatgpt to self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6354–6374, 2023.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. WebThinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*, 2025c.

Xin Sky Li, Qizhi Chu, Yubin Chen, Yang Liu, Yaoqi Liu, Zekai Yu, Weize Chen, Chen Qian, Chuan Shi, and Cheng Yang. GraphTeam: Facilitating large language model-based graph analysis via multi-agent collaboration. *arXiv preprint arXiv:2410.18032*, 2024e.

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024f.

Yang Li, Yangyang Yu, Haohang Li, Zhi Chen, and Khaldoun Khashanah. TradingGPT: Multi-agent system with layered memory and distinct characters for enhanced financial trading performance. *arXiv preprint arXiv:2309.03736*, 2023c.

Yaoru Li, Shunyu Liu, Tongya Zheng, and Mingli Song. Parallelized planning-acting for efficient LLM-based multi-agent systems. *arXiv preprint arXiv:2503.03505*, 2025d.

Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. Improving multi-agent debate with sparse communication topology. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 7281–7294. Association for Computational Linguistics, 2024g.

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

Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. StructRAG: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. In *The Thirteenth International Conference on Learning Representations*, 2025e.

Zihao Li, Weiwei Yi, and Jiahong Chen. Accuracy paradox in large language models: Regulating hallucination risks in generative ai. *arXiv preprint*, 2025f.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agen debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, 2024.

Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. MARFT: Multi-agent reinforcement fine-tuning. *arXiv preprint arXiv:2504.16129*, 2025.

Xiaoqiang Lin, Zhongxiang Dai, Arun Verma, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. Prompt optimization with human feedback. *arXiv preprint arXiv:2405.17346*, 2024a.

Xiaoqiang Lin, Zhaoxuan Wu, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. Use your INSTINCT: instruction optimization for llms using neural bandits coupled with transformers. In *Forty-first International Conference on Machine Learning*, 2024b.

Yi-Cheng Lin, Kang-Chieh Chen, Zhe-Yan Li, Tzu-Heng Wu, Tzu-Hsuan Wu, Kuan-Yu Chen, Hung-yi Lee, and Yun-Nung Chen. Creativity in llm-based multi-agent systems: A survey. *arXiv preprint arXiv:2505.21116*, 2025.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, et al. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*, 2025a.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024a.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, et al. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. *arXiv preprint arXiv:2507.01352*, 2025b.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024b.

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

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. ToolACE: Winning the points of LLM function calling. In *The Thirteenth International Conference on Learning Representations*, 2025d.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023a.

Yanming Liu, Xinyue Peng, Jiannan Cao, Shi Bo, Yuwei Zhang, Xuhong Zhang, Sheng Cheng, Xun Wang, Jianwei Yin, and Tianyu Du. Tool-Planner: Task planning with clusters across multiple tools. In *The Thirteenth International Conference on Learning Representations*, 2025e.

Yixin Liu, Guibin Zhang, Kun Wang, Shiyuan Li, and Shirui Pan. Graph-augmented large language model agents: Current progress and future prospects. *arXiv preprint arXiv:2507.21407*, 2025f.

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

Google LLC and A2A Project Contributors. Agent2Agent (A2A) Protocol. https://github.com/a2aproject/A2A. Apache License 2.0, accessed 2025-07-31.

Manikanta Loya, Divya Sinha, and Richard Futrell. Exploring the sensitivity of llms' decision-making capabilities: Insights from prompt variations and hyperparameters. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 3711–3716, 2023.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024a.

Junru Lu, Siyu An, Mingbao Lin, Gabriele Pergola, Yulan He, Di Yin, Xing Sun, and Yunsheng Wu. MemoChat: Tuning LLMs to use memos for consistent long-range open-domain conversation. *arXiv preprint arXiv:2308.08239*, 2023.

Siyuan Lu, Jiaqi Shao, Bing Luo, and Tao Lin. MorphAgent: Empowering agents through self-evolving profiles and decentralized collaboration. *arXiv preprint arXiv:2410.15048*, 2024b.

Yao Lu, Jiayi Wang, Raphael Tang, Sebastian Riedel, and Pontus Stenetorp. Strings from the library of babel: Random sampling as a strong baseline for prompt optimisation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2221–2231, 2024c.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, et al. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*, 2025a.

Yichen Luo, Yebo Feng, Jiahua Xu, Paolo Tasca, and Yang Liu. LLM-powered multi-agent system for automated crypto portfolio management. *arXiv preprint arXiv:2501.00826*, 2025b.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535, 2024.

Xiaowen Ma, Chenyang Lin, Yao Zhang, Volker Tresp, and Yunpu Ma. Agentic neural networks: Self-evolving multi-agent systems via textual backpropagation. *arXiv preprint arXiv:2506.09046*, 2025.

Yubo Ma, Zhibin Gou, Junheng Hao, Ruochen Xu, Shuohang Wang, Liangming Pan, Yujiu Yang, Yixin Cao, and Aixin Sun. SciAgent: Tool-augmented language models for scientific reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing 2024*, pages 15701–15736, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-Refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, pages 46534–46594, 2023.

Nour Makke and Sanjay Chawla. Interpretable scientific discovery with symbolic regression: a review. *Artificial Intelligence Review*, 57(1):2, 2024.

Raphael Mannadiar and Hans Vangheluwe. Debugging in domain-specific modelling. In *International Conference on Software Language Engineering*, pages 276–285. Springer, 2010.

Samuele Marro and Agora Protocol Contributors. Agora Protocol (AGORA). https://agoraprotocol.org/. MIT License, accessed 2025-07-31.

Andrew D McNaughton, Gautham Krishna Sankar Ramalaxmi, Agustin Kruel, Carter R Knutson, Rohith A Varikoti, and Neeraj Kumar. Cactus: Chemistry agent connecting tool usage to science. *ACS omega*, 9(46):46563–46573, 2024.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, et al. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413*, 2024.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *The Tenth International Conference on Learning Representations*, 2022.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. RET-LLM: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*, 2023.

Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Ivan Laptev, Philip HS Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. MALT: Improving reasoning with multi-agent llm training. *arXiv preprint arXiv:2412.01928*, 2024.

Jaap MJ Murre and Joeri Dros. Replication and analysis of ebbinghaus' forgetting curve. *PloS one*, 10(7):e0120644, 2015.

Magnus Müller and Gregor Žunič. Browser use: Enable AI to control your browser, 2024. https://github.com/browser-use/browser-use.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, et al. MLGym: A new framework and benchmark for advancing ai research agents. *arXiv preprint arXiv:2502.14499*, 2025.

Ansong Ni, Srini Iyer, Dragomir Radev, Veselin Stoyanov, Wen-tau Yih, Sida Wang, and Xi Victoria Lin. LEVER: Learning to verify language-to-code generation with execution. In *International Conference on Machine Learning*, pages 26106–26128. PMLR, 2023.

Ansong Ni, Miltiadis Allamanis, Arman Cohan, Yinlin Deng, Kensen Shi, Charles Sutton, and Pengcheng Yin. NExT: Teaching large language models to reason about code execution. In *Forty-first International Conference on Machine Learning*, 2024.

Boye Niu, Yiliao Song, Kai Lian, Yifan Shen, Yu Yao, Kun Zhang, and Tongliang Liu. Flow: Modularized agentic workflow automation. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9340–9366, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Charles Packer, Vivian Fang, Shishir G Patil, Kevin Lin, Sarah Wooders, and Joseph E Gonzalez. MemGPT: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Alexander Pan, Jun Shern Chan, Andy Zou, Nathaniel Li, Steven Basart, Thomas Woodside, Hanlin Zhang, Scott Emmons, and Dan Hendrycks. Do the rewards justify the means? measuring trade-offs between rewards and ethical behavior in the machiavelli benchmark. In *International conference on machine learning*, pages 26837–26867. PMLR, 2023.

Melissa Z Pan, Mert Cemri, Lakshya A Agrawal, Shuyi Yang, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Kannan Ramchandran, Dan Klein, Joseph E. Gonzalez, Matei Zaharia, and Ion Stoica. Why do Multi-Agent systems fail? In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025a.

Rui Pan, Shuo Xing, Shizhe Diao, Wenhe Sun, Xiang Liu, Kashun Shum, Jipeng Zhang, Renjie Pi, and Tong Zhang. Plum: Prompt learning using metaheuristics. In *Findings of the Association for Computational Linguistics*, pages 2177–2197, 2024a.

Ruwei Pan, Hongyu Zhang, and Chao Liu. CodeCoR: An llm-based self-reflective multi-agent framework for code generation. *arXiv preprint arXiv:2501.07811*, 2025b.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, et al. WebCanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*, 2024b.

Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman E. Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. MAPoRL: Multi-agent post-co-training for collaborative large language models with reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30215–30248, 2025.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

Junyeong Park, Junmo Cho, and Sungjin Ahn. MrSteve: Instruction-following agents in minecraft with what-where-when memory. *arXiv preprint arXiv:2411.06736*, 2024.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive APIs. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

Anthropic PBC and Model Context Protocol Contributors. Model Context Protocol (MCP). https://modelcontextprotocol.io/overview. MIT License, accessed 2025-07-31.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 487–503, 2021.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, et al. APIGen-MT: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *arXiv preprint arXiv:2504.03601*, 2025.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. GRIPS: Gradient-free, edit-based instruction search for prompting large language models. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3827–3846, 2023.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, 2023.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent Q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*, 2024.

Meghana Puvvadi, Sai Kumar Arava, Adarsh Santoria, Sesha Sai Prasanna Chennupati, and Harsha Vardhan Puvvadi. Coding agents: A comprehensive survey of automated bug fixing systems and benchmarks. In *2025 IEEE 14th International Conference on Communication Systems and Network Technologies (CSNT)*, pages 680–686. IEEE, 2025.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, 2024.

Cheng Qian, Chi Han, Yi R Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. CREATOR: Tool creation for disentangling abstract and concrete reasoning of large language models. In *2023 Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6922–6939. Association for Computational Linguistics (ACL), 2023.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025a.

Yiyue Qian, Shinan Zhang, Yun Zhou, Haibo Ding, Diego Socolinsky, and Yi Zhang. Enhancing LLM-as-a-Judge via multi-agent collaboration. *amazon.science*, 2025b.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. In *Advances in Neural Information Processing Systems*, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*, 2024.

Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. From exploration to mastery: Enabling LLMs to master tools via self-driven interactions. In *The Thirteenth International Conference on Learning Representations*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Asif Rahman, Veljko Cvetkovic, Kathleen Reece, Aidan Walters, Yasir Hassan, Aneesh Tummeti, Bryan Torres, Denise Cooney, Margaret Ellis, and Dimitrios S Nikolopoulos. MARCO: Multi-agent code optimization with real-time knowledge integration for high-performance computing. *arXiv preprint arXiv:2505.03906*, 2025.

Zeeshan Rasheed, Malik Abdul Sami, Kai-Kristian Kemell, Muhammad Waseem, Mika Saari, Kari Systä, and Pekka Abrahamsson. CodePori: Large-scale system for autonomous software development using multi-agent technology. *arXiv preprint arXiv:2402.01411*, 2024.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. AndroidWorld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

Shaina Raza, Ranjan Sapkota, Manoj Karkee, and Christos Emmanouilidis. TRiSM for agentic ai: A review of trust, risk, and security management in llm-based agentic multi-agent systems. *arXiv preprint arXiv:2506.04133*, 2025.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. "smolagents": a smol library to build great agentic systems. https://github.com/huggingface/smolagents, 2025.

Kai Ruan, Xuan Wang, Jixiang Hong, Peng Wang, Yang Liu, and Hao Sun. Liveideabench: Evaluating llms' divergent thinking for scientific idea generation with minimal context. *arXiv preprint arXiv:2412.17596*, 2024.

Kai Ruan, Mowen Huang, Ji-Rong Wen, and Hao Sun. Benchmarking LLMs' swarm intelligence. *arXiv preprint arXiv:2505.04364*, 2025.

Ali Safaya and Deniz Yuret. Neurocache: Efficient vector retrieval for long-range language modeling. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 870–883, 2024.

Saket Sarin, Sunil K Singh, Sudhakar Kumar, Shivam Goyal, Brij Bhooshan Gupta, Wadee Alhalabi, and Varsha Arya. Unleashing the power of multi-agent reinforcement learning for algorithmic trading in the digital financial frontier and enterprise information systems. *Computers, Materials & Continua*, 80(2), 2024.

Anjana Sarkar and Soumyendu Sarkar. Survey of LLM agent communication with mcp: A software design pattern centric review. *arXiv preprint arXiv:2506.05364*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. ToolFormer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Samuel Schmidgall, Rojin Ziaei, Carl Harris, Eduardo Reis, Jeffrey Jopling, and Michael Moor. AgentClinic: a multimodal agent benchmark to evaluate ai in simulated clinical environments. *arXiv preprint arXiv:2405.07960*, 2024.

Lennart Schneider, Martin Wistuba, Aaron Klein, Jacek Golebiowski, Giovanni Zappella, and Felice Antonio Merra. Hyperband-based bayesian optimization for black-box prompt selection. In *Forty-second International Conference on Machine Learning*, 2025.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Chengshuai Shi, Kun Yang, Jing Yang, and Cong Shen. Best arm identification for prompt learning under a limited budget. *arXiv preprint arXiv:2402.09723*, 2024a.

Juanming Shi, Qinglang Guo, Yong Liao, and Shenglin Liang. Legalgpt: Legal chain of thought for the legal large language model multi-agent framework. In *International Conference on Intelligent Computing*, pages 25–37. Springer, 2024b.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations*, 2021.

Arnav Singhvi, Manish Shetty, Shangyin Tan, Christopher Potts, Koushik Sen, Matei Zaharia, and Omar Khattab. Dspy assertions: Computational constraints for self-refining language model pipelines. *arXiv preprint arXiv:2312.13382*, 2023.

Linxin Song, Jiale Liu, Jieyu Zhang, Shaokun Zhang, Ao Luo, Shijian Wang, Qingyun Wu, and Chi Wang. Adaptive in-conversation team building for language model agents. *arXiv preprint arXiv:2405.19425*, 2024.

Dilara Soylu, Christopher Potts, and Omar Khattab. Fine-Tuning and Prompt Optimization: Two great steps that work better together. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10696–10710, 2024.

Olly Styles, Sam Miller, Patricio Cerda-Mardini, Tanaya Guha, Victor Sanchez, and Bertie Vidgen. Workbench: a benchmark dataset for agents in a realistic workplace setting. *arXiv preprint arXiv:2405.00823*, 2024.

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

Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*, 2025.

Emilio Sulis, Stefano Mariani, and Sara Montagna. A survey on agents applications in healthcare: Opportunities, challenges and trends. *Computer Methods and Programs in Biomedicine*, 236:107525, 2023.

Hao Sun, Alihan Hüyük, and Mihaela van der Schaar. Query-Dependent prompt evaluation and optimization with offline inverse RL. In *The Twelfth International Conference on Learning Representations*, 2024a.

Jingyun Sun, Chengxiao Dai, Zhongze Luo, Yangbo Chang, and Yang Li. LawLuo: A Chinese law firm co-run by LLM agents. *arXiv preprint arXiv:2407.16252*, 2024b.

Yashar Talebirad and Amirhossein Nadiri. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*, 2023.

Xiangru Tang, Tianyu Hu, Muyang Ye, Yanjun Shao, Xunjian Yin, Siru Ouyang, Wangchunshu Zhou, Pan Lu, Zhuosheng Zhang, Yilun Zhao, et al. ChemAgent: Self-updating memories in large language models improves chemical reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Xinyu Tang, Xiaolei Wang, Wayne Xin Zhao, Siyuan Lu, Yaliang Li, and Ji-Rong Wen. Unleashing the potential of large language models as prompt optimizers: Analogical analysis with gradient-based model optimizers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 25264–25272, 2025b.

Xunzhu Tang, Kisub Kim, Yewei Song, Cedric Lothritz, Bei Li, Saad Ezzini, Haoye Tian, Jacques Klein, and Tegawendé F Bissyandé. CodeAgent: Autonomous communicative agents for code review. *arXiv preprint arXiv:2402.02172*, 2024.

Yong-En Tian, Yu-Chien Tang, Kuang-Da Wang, An-Zi Yen, and Wen-Chih Peng. Template-based financial report generation in agentic and decomposed information retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2706–2710. ACM, 2025.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. DART-math: Difficulty-aware rejection tuning for mathematical problem-solving. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. Multi-Agent collaboration mechanisms: A survey of LLMs. *arXiv preprint arXiv:2501.06322*, 2025.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. *arXiv preprint arXiv:2407.18901*, 2024.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Tycho FA van der Ouderaa, Markus Nagel, Mart Van Baalen, Yuki M Asano, and Tijmen Blankevoort. The LLM surgeon. *arXiv preprint arXiv:2312.17244*, 2023.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*, 2024.

Xingchen Wan, Han Zhou, Ruoxi Sun, and Sercan Ö. Arik. From few to many: Self-improving many-shot reasoners through iterative optimization and generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

Boshi Wang, Hao Fang, Jason Eisner, Benjamin Van Durme, and Yu Su. LLMs in the imaginarium: Tool learning through simulated trial and error. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10583–10604, 2024a.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2023a.

Jize Wang, Ma Zerun, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. GTA: a benchmark for general tool agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024b.

Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345, 2024c.

Ningning Wang, Xavier Hu, Pai Liu, He Zhu, Yue Hou, Heyuan Huang, Shengyu Zhang, Jian Yang, Jiaheng Liu, Ge Zhang, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. Efficient agents: Building effective agents while reducing cost, 2025b. https://arxiv.org/abs/2508.02694.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024d.

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

Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. Recursively summarizing enables long-term dialogue memory in large language models. *Neurocomputing*, 639:130193, 2025d.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. ToolGen: Unified tool retrieval and calling via generation. In *The Thirteenth International Conference on Learning Representations*, 2025e.

Shang Wang, Tianqing Zhu, Dayong Ye, and Wanlei Zhou. When machine unlearning meets retrieval-augmented generation (RAG): Keep secret or forget knowledge? *arXiv preprint arXiv:2410.15267*, 2024e.

Shilong Wang, Guibin Zhang, Miao Yu, Guancheng Wan, Fanci Meng, Chongye Guo, Kun Wang, and Yang Wang. G-Safeguard: A topology-guided security lens and treatment on llm-based multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7261–7276, 2025f.

Siyuan Wang, Zhongyu Wei, Yejin Choi, and Xiang Ren. Symbolic working memory enhances language models for complex rule application. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17583–17604, 2024f.

Wenyi Wang, Hisham A Alyahya, Dylan R Ashley, Oleg Serikov, Dmitrii Khizbullin, Francesco Faccio, and Jürgen Schmidhuber. How to correctly do semantic backpropagation on language-based agentic systems. *arXiv preprint arXiv:2412.03624*, 2024g.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024h.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, and et al. OpenHands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025g.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. PromptAgent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*, 2024i.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b.

Yingxu Wang, Shiqi Fan, Mengzhu Wang, and Siwei Liu. Dynamically adaptive reasoning via LLM-guided mcts for efficient and context-aware KGQA. *arXiv preprint arXiv:2508.00719*, 2025h.

Yingxu Wang, Siwei Liu, Jinyuan Fang, and Zaiqiao Meng. EvoAgentX: An automated framework for evolving agentic workflows. *arXiv preprint arXiv:2507.03616*, 2025i.

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

Yiying Wang, Xiaojing Li, Binzhu Wang, Yueyang Zhou, Yingru Lin, Han Ji, Hong Chen, Jinshi Zhang, Fei Yu, Zewei Zhao, et al. PEER: Expertizing domain-specific tasks with a multi-agent framework and tuning methods. *arXiv preprint arXiv:2407.06985*, 2024j.

Yu Wang and Xi Chen. MIRIX: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957*, 2025.

Zhiruo Wang, Graham Neubig, and Daniel Fried. TroVE: Inducing verifiable and efficient toolboxes for solving programmatic tasks. In *Forty-first International Conference on Machine Learning*, 2024k.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. RAGEN: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025k.

Ziyue Wang, Junde Wu, Chang Han Low, and Yueming Jin. MedAgent-Pro: Towards multi-modal evidence-based medical diagnosis via reasoning agentic workflow. *arXiv preprint arXiv:2503.18968*, 2025l.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2024l.

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

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025a.

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

Bin Wu, Edgar Meij, and Emine Yilmaz. A joint optimization framework for enhancing efficiency of tool utilization in LLM agents. In *Findings of the Association for Computational Linguistics, ACL*, pages 22361–22373, 2025a.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. WebWalker: Benchmarking llms in web traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10290–10305, 2025b.

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

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

Yurong Wu, Yan Gao, Bin Zhu, Zineng Zhou, Xiaodi Sun, Sheng Yang, Jian-Guang Lou, Zhiming Ding, and Linjun Yang. StraGo: Harnessing strategic guidance for prompt optimization. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 10043–10061, 2024b.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1819–1862, 2024c.

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

Jinyu Xiang, Jiayi Zhang, Zhaoyang Yu, Fengwei Teng, Jinhao Tu, Xinbing Liang, Sirui Hong, Chenglin Wu, and Yuyu Luo. Self-supervised prompt optimization. *arXiv preprint arXiv:2502.06855*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*, 2024.

Huajian Xin, Z.Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Haowei Zhang, Qihao Zhu, Dejian Yang, Zhibin Gou, Z.F. Wu, Fuli Luo, and Chong Ruan. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. In *The Thirteenth International Conference on Learning Representations*, 2025.

Frank Xing. Designing heterogeneous LLM agents for financial sentiment analysis. *ACM Transactions on Management Information Systems*, 16(1):1–24, 2025.

Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. GPS: genetic prompt search for efficient few-shot learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8162–8171, 2022.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*, 2023.

Tianqi Xu, Linyao Chen, Dai-Jie Wu, Yanjun Chen, Zecheng Zhang, Xiang Yao, Zhiqiang Xie, Yongchao Chen, Shilong Liu, Bochen Qian, et al. Crab: Cross-environment agent benchmark for multimodal language model agents. *arXiv preprint arXiv:2407.01511*, 2024a.

Tianwen Xu and Fengkui Ju. Multi-agent logic for reasoning about duties and powers in private law. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, pages 361–370, 2023.

Weijia Xu, Andrzej Banburski, and Nebojsa Jojic. Reprompting: Automated chain-of-thought prompt inference through gibbs sampling. In *Forty-first International Conference on Machine Learning*, 2024b.

Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. A-MEM: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024a.

Hongyang Yang, Boyu Zhang, Neng Wang, Cheng Guo, Xiaoli Zhang, Likun Lin, Junlin Wang, Tianyu Zhou, Mao Guan, Runjia Zhang, et al. FinRobot: an open-source ai agent platform for financial applications using large language models. *arXiv preprint arXiv:2405.14767*, 2024b.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin CUI. Buffer of thoughts: Thought-augmented reasoning with large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024c.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. GPT4Tools: Teaching large language model to use tools via self instruction. In *Advances in Neural Information Processing Systems*, 2023.

Weiqing Yang, Hanbin Wang, Zhenghao Liu, Xinze Li, Yukun Yan, Shuo Wang, Yu Gu, Minghe Yu, Zhiyuan Liu, and Ge Yu. Enhancing the code debugging ability of llms via communicative agent based data refinement. *CoRR*, 2024d.

Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. AgentNet: Decentralized evolutionary coordination for llm-based multi-agent systems. *arXiv preprint arXiv:2504.00587*, 2025b.

Yingxuan Yang, Huacan Chai, Yuanyi Song, Siyuan Qi, Muning Wen, Ning Li, Junwei Liao, Haoyi Hu, Jianghao Lin, Gaowei Chang, et al. A survey of AI agent protocols. *arXiv preprint arXiv:2504.16736*, 2025c.

Yingxuan Yang, Qiuying Peng, Jun Wang, Ying Wen, and Weinan Zhang. Unlocking the potential of decentralized llm-based MAS: privacy preservation and monetization in collective intelligence. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, pages 2896–2900, 2025d.

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

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

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. τ-bench: A benchmark for Tool-Agent-User interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh R. N., Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. Retroformer: Retrospective large language agents with policy gradient optimization. In *The Twelfth International Conference on Learning Representations*, 2024.

Qinyuan Ye, Maxamed Axmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt engineer. *arXiv preprint arXiv:2311.05661*, 2023.

Rui Ye, Shuo Tang, Rui Ge, Yaxin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. MAS-GPT: Training LLMs to build LLM-based multi-agent systems. *arXiv preprint arXiv:2503.03686*, 2025.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. Survey on evaluation of LLM-based agents. *arXiv preprint arXiv:2503.16416*, 2025.

Fan Yin, Zifeng Wang, I Hsu, Jun Yan, Ke Jiang, Yanfei Chen, Jindong Gu, Long T Le, Kai-Wei Chang, Chen-Yu Lee, et al. Magnet: Multi-turn tool-use data synthesis and distillation via graph translation. *arXiv preprint arXiv:2503.07826*, 2025.

Shuo Yin, Weihao You, Zhilong Ji, Guoqiang Zhong, and Jinfeng Bai. MuMath-Code: Combining tool-use large language models with multi-perspective data augmentation for mathematical reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4770–4785, 2024.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. Exchange-of-Thought: Enhancing large language model capabilities through cross-model communication. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15135–15153, 2023.

Junwei Yu, Yepeng Ding, and Hiroyuki Sato. DynTaskMAS: A dynamic task graph-driven framework for asynchronous and parallel LLM-based multi-agent systems. *arXiv preprint arXiv:2503.07675*, 2025.

Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Qingsong Wen, Kun Wang, and Yang Wang. NetSafe: Exploring the topological safety of multi-agent networks. *arXiv preprint arXiv:2410.15686*, 2024a.

Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan Suchow, Zhenyu Cui, Rong Liu, et al. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. *Advances in Neural Information Processing Systems*, 37:137010–137045, 2024b.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi Fung, Hao Peng, and Heng Ji. CRAFT: Customizing LLMs by creating and retrieving from specialized toolsets. In *The Twelfth International Conference on Learning Representations*, 2024a.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. EvoAgent: Towards automatic multi-agent generation via evolutionary algorithms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 6192–6217, 2025a.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. EASYTOOL: enhancing llm-based agents with concise tool instruction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 951–972, 2025b.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, et al. R-judge: Benchmarking safety risk awareness for llm agents. *arXiv preprint arXiv:2401.10019*, 2024b.

Weikang Yuan, Junjie Cao, Zhuoren Jiang, Yangyang Kang, Jun Lin, Kaisong Song, Pengwei Yan, Changlong Sun, Xiaozhong Liu, et al. Can large language models grasp legal theories? enhance legal reasoning with insights from multi-agent collaboration. *arXiv preprint arXiv:2410.02507*, 2024c.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*, 2024d.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Mert Yüksekgönül, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488, 2022.

Ruihong Zeng, Jinyuan Fang, Siwei Liu, and Zaiqiao Meng. On the structural memory of llm agents. *arXiv preprint arXiv:2412.15266*, 2024a.

Ruihong Zeng, Jinyuan Fang, Siwei Liu, and Zaiqiao Meng. On the structural memory of llm agents. *arXiv preprint arXiv:2412.15266*, 2024b.

Alexander Zhang, Marcus Dong, Jiaheng Liu, Wei Zhang, Yejie Wang, Jian Yang, Ge Zhang, Tianyu Liu, Zhongyuan Peng, Yingshui Tan, et al. CodeCriticBench: A holistic code critique benchmark for large language models. *arXiv preprint arXiv:2502.16614*, 2025a.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. AppAgent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 70:1–70:20. ACM, 2025b.

Dan Zhang, Sining Zhoubian, Min Cai, Fengzu Li, Lekang Yang, Wei Wang, Tianjiao Dong, Ziniu Hu, Jie Tang, and Yisong Yue. DataSciBench: An llm agent benchmark for data science. *arXiv preprint arXiv:2502.13897*, 2025c.

Enhao Zhang, Erkang Zhu, Gagan Bansal, Adam Fourney, Hussein Mozannar, and Jack Gerrits. Optimizing sequential multi-step tasks with parallel llm agents. *arXiv preprint arXiv:2507.08944*, 2025d.

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

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-Memory: Tracing hierarchical memory for multi-agent systems. *arXiv preprint arXiv:2506.07398*, 2025e.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, LEI BAI, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Forty-second International Conference on Machine Learning*, 2025f.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *The Thirteenth International Conference on Learning Representations*, 2025g.

Hangfan Zhang, Zhiyao Cui, Xinrun Wang, Qiaosheng Zhang, Zhen Wang, Dinghao Wu, and Shuyue Hu. If multi-agent debate is the answer, what is the question. *arXiv preprint arXiv:2502.08788*, 2025h.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025i.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025j.

Jun Zhang, Yuwei Yan, Junbo Yan, Zhiheng Zheng, Jinghua Piao, Depeng Jin, and Yong Li. A parallelized framework for simulating large-scale LLM agents with realistic environments and interactions. In Georg Rehm and Yunyao Li, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 1339–1349, Vienna, Austria, July 2025k. Association for Computational Linguistics. ISBN 979-8-89176-288-6. doi: 10.18653/v1/2025.acl-industry.94.

Kechi Zhang, Zhuo Li, Jia Li, Ge Li, and Zhi Jin. Self-Edit: Fault-aware code editor for code generation. *arXiv preprint arXiv:2305.04087*, 2023a.

Peiyan Zhang, Haibo Jin, Leyang Hu, Xinnuo Li, Liying Kang, Man Luo, Yangqiu Song, and Haohan Wang. Revolve: Optimizing AI systems by tracking response evolution in textual optimization. In *Forty-second International Conference on Machine Learning*, 2025l.

Shaokun Zhang, Jieyu Zhang, Jiale Liu, Linxin Song, Chi Wang, Ranjay Krishna, and Qingyun Wu. Offline training of language model agents with functions as learnable weights. In *Forty-first International Conference on Machine Learning*, 2024b.

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. Nemotron-research-tool-n1: Tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*, 2025m.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. TEMPERA: test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023b.

Wentao Zhang, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving. *arXiv preprint arXiv:2506.12508*, 2025n.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237, 2024c.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. *ACM Transactions on Information Systems*, 2024d.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 19632–19642, 2024.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute Zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025a.

Ruochen Zhao, Wenxuan Zhang, Yew Ken Chia, Weiwen Xu, Deli Zhao, and Lidong Bing. Auto-Arena: Automating LLM evaluations with agent peer battles and committee discussions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4440–4463, 2025b.

Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Zou. SiriuS: Self-improving multi-agent systems via bootstrapped reasoning. *arXiv preprint arXiv:2502.04780*, 2025c.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

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

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*, 2023a.

Sipeng Zheng, Jiazheng Liu, Yicheng Feng, and Zongqing Lu. Steve-Eye: Equipping llm-based embodied agents with visual perception in open worlds. In *The Twelfth International Conference on Learning Representations*, 2024.

Zhiling Zheng, Oufan Zhang, Ha L Nguyen, Nakul Rampal, Ali H Alawadhi, Zichao Rong, Teresa Head-Gordon, Christian Borgs, Jennifer T Chayes, and Omar M Yaghi. Chatgpt research group for optimizing the crystallinity of mofs and cofs. *ACS Central Science*, 9(11):2161–2170, 2023b.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 19724–19731, 2024.

Han Zhou, Xingchen Wan, Ivan Vulic, and Anna Korhonen. Survival of the most influential prompts: Efficient black-box prompt search via clustering and pruning. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 13064–13077, 2023a.

Han Zhou, Xingchen Wan, Yinhong Liu, Nigel Collier, Ivan Vulić, and Anna Korhonen. Fairer preferences elicit improved human-aligned large language model judgments. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1241–1252, Miami, Florida, USA, November 2024a. Association for Computational Linguistics.

Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine A Heller, and Subhrajit Roy. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. In *The Twelfth International Conference on Learning Representations*, 2024b.

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

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023b.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023c.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025b.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, et al. MultiAgentBench: Evaluating the collaboration and competition of llm agents. *arXiv preprint arXiv:2503.01935*, 2025.

Tinghui Zhu, Kai Zhang, Jian Xie, and Yu Su. Deductive beam search: Decoding deducible rationale for chain-of-thought reasoning. In *First Conference on Language Modeling*, 2024.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. Solving math word problems via cooperative reasoning induced language models. In *Proceedings of the 61st Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics (ACL), 2023.

Yangyang Zhuang, Wenjia Jiang, Jiayu Zhang, Ze Yang, Joey Tianyi Zhou, and Chi Zhang. Learning to be a doctor: Searching for effective medical agent architectures. *arXiv preprint arXiv:2504.11301*, 2025.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. ToolQA: A dataset for LLM question answering with external tools. In *Advances in Neural Information Processing Systems*, 2023.

Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A Rossi, Somdeb Sarkhel, and Chao Zhang. Toolchain*: Efficient action space navigation in large language models with a* search. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, et al. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*, 2024b.

Huhai Zou, Rongzhen Li, Tianhao Sun, Fei Wang, Tao Li, and Kai Liu. Cooperative scheduling and hierarchical memory model for multi-agent systems. In *2024 IEEE International Symposium on Product Compliance Engineering - Asia (ISPCE-ASIA)*, pages 1–6, 2024. doi: 10.1109/ISPCE-ASIA64773.2024.10756271.

Kaiwen Zuo, Yirui Jiang, Fan Mo, and Pietro Lio. KG4Diagnosis: A hierarchical multi-agent LLM framework with knowledge graph enhancement for medical diagnosis. In *AAAI Bridge Program on AI for Medicine and Healthcare*, pages 195–204. PMLR, 2025.