A Systematic Survey of Automatic Multi-Agent System Optimization

Anonymous ACL submission

Abstract

Since the advent of large language models, multi agent systems have emerged as powerful architectures for executing complex reasoning tasks. However, relying on static manual design remains a critical impediment due to the rapidly expanding complexity of agent roles, communication topologies, and prompting strategies. To mitigate this, automated optimization techniques have recently emerged to dynamically construct and refine these architectures. In this paper, we present a comprehensive survey summarizing the current progress and methodologies within this critical domain. We introduce a unifying taxonomy that analyzes automated system construction across three fundamental dimensions. First, we examine the architectural design space, encompassing agent composition, structural topology, and textual prompt formulation. Second, we detail the temporal dynamics of optimization, spanning offline calibration, per query synthesis, and dynamic runtime adaptation. Third, we categorize the underlying algorithmic paradigms, tracing the progression from discrete combinatorial search toward continuous policy optimization and generative structural synthesis. Ultimately, by removing manual engineering bottlenecks, these frameworks establish a robust computational foundation for fully autonomous ecosystems. We hope this survey spurs further research guided by our proposed taxonomy.

1 Introduction

Large language models execute complex reasoning through multi agent systems (Li et al., 2023; Qian et al., 2025; Wu et al., 2024). Early frameworks rely on manual static pipelines with fixed roles and topologies (Qian et al., 2024; Chen et al., 2024). This rigidity limits adaptability, increases engineering overhead, creates token bottlenecks, and amplifies local prompt errors (Liu et al., 2023; Zhang et al., 2025d; Hu et al., 2025a; Verma et al., 2024; Zhou et al., 2024, 2026).

To resolve these limitations, recent research automates architectural optimization. Graph theoretic approaches construct sparse task adaptive topologies using discrete diffusion or Monte Carlo Tree Search (Zhuge et al., 2024; Jiang et al., 2025). Alternatively, policy based routing frames orchestration as a Markov Decision Process (Liu et al., 2023; Li et al., 2026c). These methods optimize next agent prediction (Wang et al., 2025a), eliminate redundant agents dynamically (Wang et al., 2025b), or execute holistic single step generation to bypass sequential evaluation errors (Ke et al., 2026).

To navigate combinatorial design spaces, frameworks apply evolutionary algorithms to generate agent populations (Yuan et al., 2025), recombine discrete modules (Shang et al., 2025), and evolve system prompts (Guo et al., 2024a; Fernando et al., 2024). Concurrently, generative paradigms synthesize architectures autoregressively from scratch (Li et al., 2026a). Systems achieve domain universality via Mixture of Experts (Li et al., 2026b) or enforce stability by generating workflows as finite state machines (Zhang et al., 2025e).

Automated design requires robust feedback protocols. Frameworks interleave topology and prompt optimization to address their inherent functional coupling (Zhou et al., 2025, 2026). To train orchestrators efficiently, systems apply offline reinforcement learning to specialized meta agents (Nie et al., 2025). Bootstrapping from curated experience libraries enables continuous offline refinement (Zhao et al., 2026).

The systematic review methodology utilized to identify and filter the literature analyzed in this survey is detailed in Appendix A.

While existing literature reviews single agent optimization (Du et al., 2026), tool integration (Hu et al., 2026), memory architectures (Wu et al., 2025), and prompt engineering (Ramnath et al., 2025), these studies focus on isolated components. Broader surveys examine manual multi agent col-

laboration (Tran et al., 2025) and general system frameworks (Guo et al., 2024b) but omits automatic optimization details. To our knowledge, this paper is the first to systematically categorize the automatic optimization of multi agent systems. We synthesize mechanisms across prompt evolution, topology search, dynamic routing, and generative construction, detailing the computational foundations of adaptive autonomous architectures.

2 Design Spaces for MAS Optimization

The architecture of a Multi-Agent System can be optimized across three deeply interconnected design spaces: the composition of individual agents, their interaction topology, and their governing prompts.

2.1 Agent Composition

Agent composition defines the functional nodes of the MAS, parameterized primarily by persona, tool access, and memory capacity (Hu et al., 2025b; Xu et al., 2026). To systematize these components, frameworks such as AgentSquare abstract agentic workflows into standardized, plug-and-play modules; specifically Planning, Reasoning, Tool Use, and Memory, establishing a formalized and bounded design space for automated search (Shang et al., 2025).

While manually crafted roles can yield strong performance in narrow application domains (Huang et al., 2023; Wang et al., 2024), they severely bottleneck system scalability and generalization (Zhuge et al., 2024). Consequently, contemporary frameworks automate role generation to construct bespoke teams per query. For instance, AgentVerse utilizes a recruiter meta-agent to dynamically synthesize and assemble task-specific experts on the fly (Chen et al., 2024). Alternatively, EvoAgent bypasses predefined role libraries entirely by treating composition as a biological evolutionary process; it applies crossover and mutation operators to spawn a diverse, collaborative multi-agent population from a single initialized agent (Yuan et al., 2025).

Beyond persona generation, the composition search space increasingly incorporates LLM heterogeneity to optimize the cost-capability Pareto front. Rather than deploying a monolithic, high-capacity model across all nodes, EvoFlow integrates diverse LLM backbones into its evolutionary search, allocating weaker, cost-effective models to simpler sub-tasks without degrading overarching cognitive capabilities (Zhang et al., 2025a). Formalizing this as a unified Multi-Agent System Routing (MASR) problem, MasRouter leverages a structured probabilistic cascade to sequentially assign interdependent roles while seamlessly directing task execution to the most appropriate LLM backbone from a diverse pool (Yue et al., 2025).

2.2 Topology and Policy-Based Orchestration

When optimizing the macro-structure of a multi-agent system, recent literature bifurcates into two distinct paradigms: explicit graph-theoretic topology optimization and implicit policy-based dynamic routing.

2.2.1 Graph-Theoretic Topology

Explicit topological optimization formalizes the MAS as a network of agent nodes and communication edges, represented either as strict Directed Acyclic Graphs (DAGs) (Zhuge et al., 2024) or as executable code supporting complex control flows (Zhang et al., 2025d). To optimize these structures, methodologies generally branch into continuous relaxation and discrete generation. Continuous approaches treat topology search as a differentiable masking problem, applying policy gradients and low-rank constraints to mathematically prune non-contributory edges from an over-complete adjacency matrix (Zhang et al., 2025b). Conversely, discrete generative frameworks synthesize pathways directly: G-Designer refines predefined anchors using Variational Graph Auto-encoders (Zhang et al., 2025c), while Guided Topology Diffusion (GTD) employs iterative denoising to balance utility and sparsity (Jiang et al., 2025). Advancing beyond template pruning, autoregressive models construct task-conditioned topologies entirely from scratch (Li et al., 2026a), a paradigm that the OFA-MAS framework extends to zero-shot, cross-domain universality by dynamically routing the graph generation steps through a Mixture-of-Experts architecture (Li et al., 2026b).

2.2.2 Policy-Based Dynamic Routing

Conversely, dynamic routing paradigms removes rigid structural edges entirely, framing orchestration as a flexible decision process. Iterative routing frameworks formalize this as a Markov Decision Process (MDP), where a continuous policy or meta-agent evaluates evolving memory states to select the next agent or dynamically drop redundant nodes

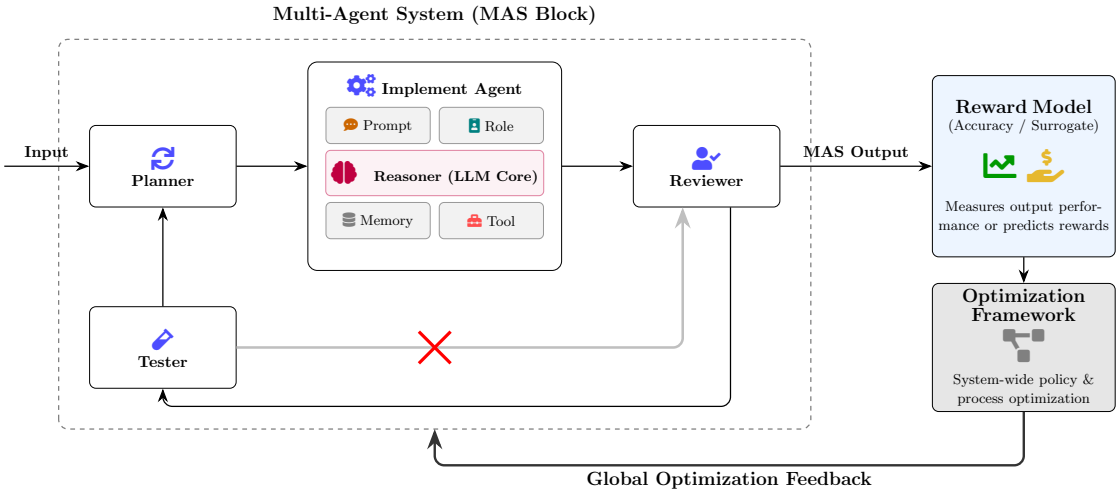

Figure 1: High level overview of Multi-agent system optimization pipeline

mid-task to conserve tokens (Liu et al., 2023; Li et al., 2026c; Wang et al., 2025b). The efficacy of this sequential orchestration is highly sensitive to the underlying model capacity; frameworks like Puppeteer demonstrate that orchestrating powerful models naturally favors early termination for token efficiency, whereas weaker models require sustained, cyclic reasoning to ensure reliable task completion (Dang et al., 2026). To maximize sequential flexibility, AnyMAC couples next-agent prediction with unrestricted next-context selection, bypassing traditional DAG constraints by retrieving history from any prior agent (Wang et al., 2025a). Moving beyond purely sequential step-by-step execution, probabilistic and holistic routing frameworks introduce broader architectural planning. MasRouter employs a variational latent variable model to probabilistically cascade roles and route execution to heterogeneous LLM backbones prior to execution (Yue et al., 2025). Alternatively, to overcome the global myopia inherent in sequential routing, MASOrchestra introduces holistic orchestration; by abstracting sub-agents as callable functions, it formulates routing as a single-step generation of the entire multi-agent trajectory, significantly improving scalability as individual agent complexity increases (Ke et al., 2026).

2.3 Prompt Optimization

At the micro-level, optimizing the textual instructions that govern individual agent behavior is essential for overarching system efficacy. Algorithms traversing this discrete language space generally fall into gradient-based or gradient-free paradigms. Methods such as TextGrad (Yuksekgonul et al., 2025) and DSPy (Khattab et al., 2024) formalize prompts as differentiable parameters, refining them via backpropagated textual gradients. Alternatively, evolutionary algorithms leverage LLMs to execute classical genetic operators (e.g., crossover and mutation) on prompt populations (Guo et al., 2024a). Advancing this gradient-free approach, Promptbreeder introduces a self-referential evolutionary paradigm that simultaneously mutates both the task instructions and the *mutation-prompts* governing the evolutionary process itself, continually adapting the system's cognitive heuristics without parameter updates (Fernando et al., 2024).

Recent literature has shown that prompt and topology design spaces are mutually dependent. Rather than tuning prompts in isolation, frameworks like MASS interleave local, block-level prompt optimization with global topology search to manage combinatorial complexity (Zhou et al., 2025). Building on this deep coupling, the ResMAS framework introduces topology-aware prompt optimization. By explicitly refining each agent's system prompt based on its topological position and historical interactions, training on both corrective and misleading neighbor feedback ResMAS learns robust interaction behaviors that prevent cascading hallucinations across the network (Zhou et al., 2026). Extending this adaptability directly to execution time, frameworks like MetaGen perform intra-task prompt evolution, dynamically rewriting role instructions mid-run based on live execution logs to mitigate failure modes on the fly

(Wang et al., 2026a).

3 Temporal Dynamics of MAS Optimization

The optimization of a multi-agent system can be categorized by its temporal granularity, ranging from dataset-wide structural training to mid-execution adaptive routing.

3.1 Offline Optimization

Offline optimization calibrates multi agent architectures against domain specific datasets to establish static configurations for inference (Hu et al., 2025a; Zhang et al., 2025d). This paradigm ensures execution stability and eliminates runtime search latency. Alternatively, hybrid frameworks determine agent pools offline while delegating topological routing to runtime execution (Liu et al., 2023). To optimize models using historical execution trajectories, frameworks apply offline supervised learning to persistent experience libraries (Zhao et al., 2026) or utilize offline reinforcement learning to train specialized meta agents (Nie et al., 2025). To achieve zero shot generalization across unseen tasks, systems employ multi stage offline pre training to develop universal topology generators (Li et al., 2026b).

3.2 Per Query Optimization

To synthesize bespoke configurations immediately prior to execution, per query optimization treats architectural design as a direct mapping problem. Generative frameworks formulate system construction as a language modeling task, mapping queries directly to executable natural language programs or multi agent workflows in a single inference step (Ye et al., 2025; Li et al., 2024). Moving beyond textual generation, probabilistic routing frameworks utilize variational latent variable models and graph autoencoders to deduce optimal agent counts and tailor communication topologies based entirely on query complexity (Zhang et al., 2025c; Yue et al., 2025).

3.3 Runtime Adaptation

Runtime adaptation dynamically modifies orchestrations by reacting to intermediate execution states and evolving memory. To conserve computational resources, systems actively evaluate agent contributions to continuously prune redundant nodes and extraneous communication edges across successive conversation rounds (Liu et al., 2023; Wang et al., 2025b, 2026b). Furthermore, frameworks resolve logic errors by parsing live execution logs and verbal feedback to iteratively rewrite system prompts, swap communication pathways, and adjust group composition without parameter updates (Wang et al., 2026a; Chen et al., 2024). To handle runtime volatility systematically, orchestrators update their token generation policies using outcome rewards broadcasted directly from live sequential interaction rollouts (Li et al., 2026c).

4 Algorithmic Paradigms for MAS Optimization

4.1 Combinatorial and Evolutionary Search

4.1.1 Tree-Based Structural Search

To optimize workflow architectures via discrete exploration, frameworks utilize Monte Carlo Tree Search (Świechowski et al., 2023) over predefined operations. AFLOW (Zhang et al., 2025d) formalizes each tree node as a complete multi agent workflow. During the backpropagation phase, the system records structural modifications alongside their corresponding execution outcomes into an experience log. This log is propagated to the parent node, allowing the underlying language model optimizer to condition subsequent structural mutations directly on historical exploration data.

4.1.2 Evolutionary and Population Based Search

To automate architecture discovery, frameworks apply evolutionary operators across programmatic and structural spaces. Systems construct and mutate executable workflows either within unrestricted Python environments (Hu et al., 2025a) or by recombining standardized functional modules (Shang et al., 2025). To optimize macro structures and agent compositions, algorithms apply semantic crossover, hierarchical mutation, and tag based retrieval to breed specialized multi agent populations (Zhang et al., 2025a; Yuan et al., 2025). Shifting evolution to execution time, MetaGen dynamically rewrites role prompts and swaps communication edges utilizing live feedback, continuously updating component selection probabilities via historical rewards (Wang et al., 2026a).

Within discrete language spaces, language models act directly as evolutionary operators. While EvoPrompt performs semantic text crossover and mutation on parent instructions (Guo et al., 2024a), Promptbreeder introduces self referential evolution.

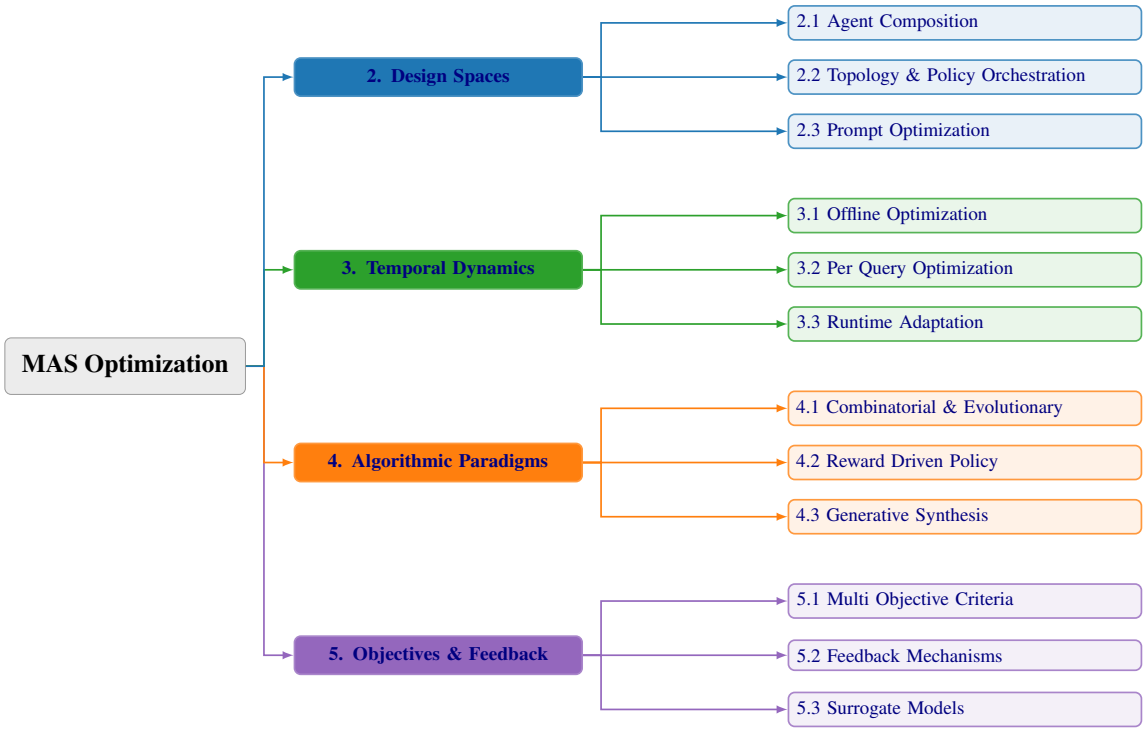

Figure 2: A hierarchical taxonomy of multi-agent system optimization components.

Utilizing a binary tournament, it simultaneously evolves task instructions and its own underlying mutation heuristics by deriving new prompt operations directly from successful intermediate reasoning steps (Fernando et al., 2024).

4.1.3 Multi Stage Interleaved Search

To manage the combinatorial expansion of joint optimization, MASS (Zhou et al., 2025) decomposes system design into a hierarchical pipeline. The framework executes three sequential stages comprising block level prompt initialization, pruned workflow topology search, and global prompt optimization. By conditioning subsequent structural searches on the refined textual components of preceding stages, the algorithm optimizes both language instructions and communication graphs dependently without performing simultaneous space exploration.

4.2 Reward Driven Policy Optimization

4.2.1 Differentiable Edge and Topology Optimization

Unlike discrete exploration, frameworks including GPTSwarm (Zhuge et al., 2024) and AgentPrune (Zhang et al., 2025b) formulate topology optimization as a continuous learning problem. By parameterizing the communication graph as a set of differentiable masks $S \in \mathbb{R}^{|V|\times|V|}$, these systems model the existence of each edge e_{ij} using a learned distribution $p_\theta(e_{ij})$. During optimization, the system samples a graph topology $\hat{G}$, evaluates its execution utility $\phi(\hat{G})$, and updates the edge probabilities via the REINFORCE algorithm:

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\hat{G}\sim p_\theta}\left[\phi(\hat{G})\nabla_\theta \log p_\theta(\hat{G})\right] \quad (1)$$

This mechanism relaxes the discrete adjacency matrix into continuous variables. To induce sparsity, AgentPrune further applies low rank constraints to these masks, mathematically driving the probabilities of non contributory communication edges toward zero without exhaustive combinatorial search.

4.2.2 On Policy Trajectory Optimization

To optimize sequential multi turn tool use decisions while mitigating the high gradient variance inherent in long horizon reinforcement learning, frameworks employ on policy proximal algorithms. AgentFlow (Li et al., 2026c) introduces Flow GRPO to optimize dynamic planning topologies. The technique formalizes execution as a sequential problem maximizing the expected outcome over a T turn trajectory τ:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$$

Rather than applying a single gradient update to the complete episode, the algorithm distributes the

trajectory level advantage back to individual turn levels $t \in \{1, \ldots, T\}$. By calculating group relative advantage estimation across multiple sampled trajectories, the method normalizes the policy gradients. This formulation enables the planner to update its token generation policy directly on policy, avoiding the need for complex differentiable environment simulators.

4.2.3 Holistic Single Step Orchestration

Sequential reinforcement learning models evaluate multi agent architectures one decision at a time. This iterative approach introduces mathematical difficulty when attempting to assign reward across a long execution sequence. To resolve this credit assignment problem, MAS Orchestra (Ke et al., 2026) formulates system orchestration as a singular function calling decision. Rather than building the network sequentially, the orchestrator generates the complete architecture including agent selection, data routing, and dependency mapping simultaneously. This holistic generation allows the model to compute the optimal structure from a global perspective, avoiding the error propagation complications inherent to multi step evaluation.

4.2.4 Probabilistic Modular Routing

Expanding policy gradients beyond fixed topologies or sequential agent dropping (Liu et al., 2023), MasRouter (Yue et al., 2025) optimizes the complete system routing pipeline through a cascaded controller network. The framework processes incoming queries using a variational latent variable model to predict the optimal collaboration structure and agent count. This structural prediction cascades into probabilistic role allocators and multinomial language model routers. The entire modular architecture is jointly trained via the REINFORCE algorithm to maximize an objective function that explicitly couples task utility with a parameterized cost penalty. By formulating routing as a differentiable probabilistic cascade rather than a discrete combinatorial search, the system constructs query aware architectures directly from learned continuous policies.

4.3 Generative Synthesis of MAS Architectures

4.3.1 Language Model Driven Structural Synthesis

To bypass iterative search exploration, generative frameworks utilize language models to directly synthesize executable workflow specifications. Systems formulate architecture design as a text generation task, producing query adaptive workflows in a single inference step (Ye et al., 2025) or synthesizing natural language programs that undergo continuous refinement via execution feedback (Li et al., 2024). Expanding this generative scope beyond static topologies, architectures dynamically synthesize novel role identities and prompt templates during active inference (Wang et al., 2026a). Conversely, to enforce deterministic control over autonomous task execution, frameworks map architectures to formal structures, synthesizing the entire multi agent system as a finite state machine where agents operate as discrete states governed by conditional transitions (Zhang et al., 2025e).

4.3.2 Variational Graph Autoencoders

To construct topologies directly from task parameters rather than relying on discrete exploration, G Designer (Zhang et al., 2025c) utilizes a Variational Graph Autoencoder. The framework encodes task requirements alongside agent capabilities into a unified latent representation. By decoding the communication topology directly from this latent space, the system maps task specifications to structural connectivity in a single pass, effectively bypassing the sample inefficiency inherent to iterative search methodologies.

4.3.3 Discrete Topology Diffusion

Analogous to standard continuous diffusion models (Ho et al., 2020) that iteratively map data to noise and learn to reverse the sequence, Guided Topology Diffusion (Jiang et al., 2025) frames multi agent architecture generation as a discrete reverse denoising process. The algorithm progressively recovers a task adaptive communication structure from an initially noisy random graph. To optimize these non differentiable discrete structures against external execution rewards, the framework applies zeroth order gradient free optimization at each diffusion timestep. By sampling and evaluating multiple candidate graphs iteratively, the method steers the generation trajectory toward structural optima without requiring direct gradient backpropagation from the language model execution.

4.3.4 Autoregressive Topology Generation

Rather than modifying fixed adjacency matrices or refining predefined templates, autoregressive paradigms construct architectures incrementally.

ARG Designer (Li et al., 2026a) utilizes a hierarchical Gated Recurrent Unit architecture to sequentially generate agent nodes and communication edges conditioned on incoming queries. By building the topology entirely from scratch, this sequential process determines the precise agent count dynamically at inference time. Advancing this mechanism for cross domain application, OFA MAS (Li et al., 2026b) integrates a task aware graph state encoder with a Mixture of Experts architecture. During the incremental generation process, this module routes the prediction of new nodes and edges to specialized expert networks based on the encoded task context. This prevents catastrophic forgetting and enables a single unified model to synthesize topologies across entirely distinct problem domains.

5 Optimization Objectives and Feedback Mechanisms

5.1 Multi Objective Optimization Criteria

5.1.1 Execution Accuracy

Maximizing execution accuracy serves as the fundamental optimization objective across multi agent frameworks (Hu et al., 2025a; Zhuge et al., 2024; Dang et al., 2026; Yue et al., 2025). Systems evaluate empirical task utility to establish a primary reward signal, directing both discrete structural search algorithms and continuous policy gradient updates before applying secondary architectural penalties. (Hu et al., 2025a).

5.1.2 Token and Computational Efficiency

To minimize communication overhead, frameworks enforce structural sparsity by eliminating redundant agents and message passing pathways via learned masking, threshold removal, or generative decoding constraints (Zhang et al., 2025b; Wang et al., 2025b; Zhang et al., 2025c). Beyond macro structural pruning, optimization objectives explicitly penalize prompt token accumulation. By integrating computational constraints into composite reward functions, these systems mathematically filter irrelevant historical context and enforce compact reasoning trajectories (Wang et al., 2025a; Shah et al., 2026; Dang et al., 2026). Furthermore, when orchestrating heterogeneous networks, systems balance task utility against expected token expenditure to route simpler subtasks toward specialized smaller language models (Yue et al., 2025).

5.1.3 Adversarial Robustness and Resilience

To withstand targeted attacks and component failures, frameworks integrate architectural stability directly into their optimization objectives. G Designer (Zhang et al., 2025c) embeds explicit adversarial robustness constraints into its generative decoding pipeline, while Guided Topology Diffusion (Jiang et al., 2025) jointly optimizes for task utility and fault tolerance to maintain execution stability amid node failures.

Alternatively, systems achieve resilience implicitly via structural sparsity. AgentPrune (Zhang et al., 2025b) demonstrates that pruning non contributory communication pathways mathematically filters out systemic noise and adversarial inputs, shielding the broader network without requiring dedicated adversarial training loops.

5.2 Feedback Mechanisms

5.2.1 Trajectory Level Feedback

To enable continuous system refinement, frameworks evaluate intermediate and final execution states. AgentFlow (Li et al., 2026c) integrates an outcome driven feedback mechanism where a dedicated verifier module assesses intermediate results, appending this signal to the memory state to allow real time correction by the active planner. Operating offline, SIRIUS (Zhao et al., 2026) bootstraps trajectory level feedback to construct persistent experience libraries. By curating multi agent trajectories, retaining successful reasoning steps, and actively augmenting failed attempts, the framework provides a robust training signal for continuous offline optimization across both collaborative and competitive environments.

5.2.2 Textual Feedback

Language models utilize natural language evaluations as direct optimization signals. TextGrad (Yuksekgonul et al., 2025) computes textual gradients from language model evaluations, backpropagating natural language critiques to optimize specific system components. Similarly, GPTSwarm (Zhuge et al., 2024) leverages execution verdicts on sampled data to drive prompt evolution. Advancing evolutionary paradigms, Promptbreeder (Fernando et al., 2024) applies Lamarckian mutation operators by extracting successful intermediate reasoning traces and prompting the language model to reverse engineer new task prompts capable of eliciting those exact reasoning paths.

Bridging text and scalar signals, MetaGen (Wang et al., 2026a) employs a dual feedback architecture, utilizing textual execution logs to rewrite prompts dynamically while applying numerical cost rewards to update structural selection priors. Beyond prompt optimization, textual feedback regulates internal multi agent social dynamics. AgentVerse (Chen et al., 2024) utilizes explicit verbal critiques from peer agents to enforce behavioral conformity, rapidly realigning agents that develop detrimental traits during open ended execution.

5.3 Surrogate Models for Execution Approximation

Evaluating topologies via full system execution incurs severe computational overhead. To decouple structural search from active execution, frameworks utilize surrogate reward models. Guided Topology Diffusion (Jiang et al., 2025) trains a Graph Neural Network proxy to predict task utility and token cost, guiding generation trajectories without environment interaction. Alternatively, AgentSquare (Shang et al., 2025) employs the language model as an in context predictor. By processing task descriptions, module profiles, and historical data, it estimates candidate utility. This enables the search algorithm to prune unpromising architectures mathematically prior to formal evaluation.

6 Challenges and Future Directions

6.1 Data Scarcity for Offline Optimization

Offline structural evolution and policy fine tuning require extensive task specific execution trajectories. Currently, curated multi agent datasets remain severely limited across diverse problem domains, constraining empirical evaluation and reward driven training.

6.2 Universal Architecture Generation

Existing generative frameworks overfit to constrained problem spaces. Developing universal multi agent orchestrators capable of zero shot structural synthesis across open ended environments constitutes a primary requirement for robust deployment.

6.3 Multimodal Orchestration

Current topology optimization paradigms evaluate textual reasoning exclusively. Following advancements in unified foundational models (Team, 2026), expanding dynamic routing and architectural search to process interleaved multimodal inputs remains a critical unexplored domain.

6.4 Verification and Uncertainty Propagation

Generative architectures lack formal execution verification, requiring orchestrators to embed compliance constraints directly into the search space. Because reasoning errors cascade through dynamic pathways, quantifying and mitigating uncertainty propagation across interconnected agent nodes remains critical.

6.5 Human Guided Architectural Search

Fully autonomous structural synthesis risks divergence from human intent during complex reasoning tasks. Future optimization paradigms must integrate human guided co planning allowing deployment in sensitive and critical areas.

7 Conclusion

Automating multi agent architectures marks a critical evolution in artificial intelligence. The architectural design space encompasses agent composition, structural topology, and textual formulation. Frameworks increasingly optimize these variables jointly rather than in isolation. Optimization temporal dynamics span distinct execution phases. Systems utilize offline calibration for foundational stability. They employ per query synthesis for immediate task adaptation. Furthermore, runtime mechanisms dynamically prune and restructure topologies during active execution. Algorithmic paradigms are shifting fundamentally. Early methods relied heavily on discrete combinatorial search. Contemporary approaches favor continuous policy optimization and generative synthesis. These advanced techniques construct complex architectures without prohibitive inference overhead. Ultimately, automated synthesis removes the bottleneck of manual engineering. It establishes the foundation for reasoning ecosystems capable of autonomous structural evolution.

Limitations

While we attempted to cover all qualifying papers, it is possible that we may have unintentionally missed out on some relevant papers. Furthermore, mapping diverse and highly complex frameworks into a singular unifying taxonomy necessitates broad categorization. Because many recent

systems hybridize multiple techniques for instance, simultaneously applying reinforcement learning to structural routing and textual prompt evolution we categorize each framework based strictly on its most salient algorithmic contribution, which may obscure secondary operational characteristics. Finally, adhering to strict publication length constraints required significant structural tradeoffs. We prioritize synthesizing foundational mechanisms over providing an exhaustive empirical catalog, balancing necessary analytical depth with rigorous brevity.

References

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. 2026. Multi-agent collaboration via evolving orchestration. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Shangheng Du, Jiabao Zhao, Jinxin Shi, Zhentao Xie, Xin Jiang, Yanhong Bai, and Liang He. 2026. A survey on the optimization of large language model-based agents. *ACM Comput. Surv.*, 58(9).

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

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

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024b. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Jinchao Hu, Meizhi Zhong, Kehai Chen, Xuefeng Bai, and Min Zhang. 2026. Agentic tool use in large language models. *arXiv preprint arXiv:2604.00835*.

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

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, and 1 others. 2025b. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*.

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

Eric Hanchen Jiang, Guancheng Wan, Sophia Yin, Mengting Li, Yuchen Wu, Xiao Liang, Xinfeng Li, Yizhou Sun, Wei Wang, Kai-Wei Chang, and 1 others. 2025. Dynamic generation of multi-llm agents communication topologies with graph diffusion models. *arXiv preprint arXiv:2510.07799*.

Zixuan Ke, Yifei Ming, Austin Xu, Ryan Chin, Xuan-Phi Nguyen, Prathyusha Jwalapuram, Jiayu Wang, Semih Yavuz, Caiming Xiong, and Shafiq Joty. 2026. Mas-orchestra: Understanding and improving multi-agent reasoning through holistic orchestration and controlled benchmarks. *arXiv preprint arXiv:2601.14652*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful Haq, Ashutosh Sharma, Thomas Joshi, Hanna Moazam, Heather Miller, and 1 others. 2024. Dspy: compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations*, volume 2024, pages 54928–54958.

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

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. 2026a. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 23142–23150.

Shiyuan Li, Yixin Liu, Yu Zheng, Mei Li, Quoc Viet Hung Nguyen, and Shirui Pan. 2026b. Ofamas: One-for-all multi-agent system topology design based on mixture-of-experts graph generative models. In *Proceedings of the ACM Web Conference 2026*, pages 1333–1344.

Zelong Li, Shuyuan Xu, Kai Mei, Wenyue Hua, Balaji Rama, Om Raheja, Hao Wang, He Zhu, and

Yongfeng Zhang. 2024. Autoflow: Automated workflow generation for large language model agents. *arXiv preprint arXiv:2407.12821*.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. 2026c. In-the-flow agentic system optimization for effective planning and tool use. In *The Fourteenth International Conference on Learning Representations*.

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

Fan Nie, Lan Feng, Haotian Ye, Weixin Liang, Pan Lu, Huaxiu Yao, Alexandre Alahi, and James Zou. 2025. Weak-for-strong: Training weak meta-agent to harness strong executors. In *Second Conference on Language Modeling*.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, and 1 others. 2024. 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.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations*.

Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, and 1 others. 2025. A systematic survey of automatic prompt optimization techniques. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33066–33098.

Deep Shah, Sanket Badhe, Nehal Kathrotia, and Priyanka Tiwari. 2026. Crop: Token-efficient reasoning in large language models via regularized prompt optimization. *arXiv preprint arXiv:2604.14214*.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. 2025. Agentsquare: Automatic LLM agent search in modular design space. In *The Thirteenth International Conference on Learning Representations*.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. 2023. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562.

Qwen Team. 2026. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*.

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

Mudit Verma, Siddhant Bhambri, and Subbarao Kambhampati. 2024. On the brittle foundations of react prompting for agentic large language models. *arXiv preprint arXiv:2405.13966*.

Song Wang, Zhen Tan, Zihan Chen, Shuang Zhou, Tianlong Chen, and Jundong Li. 2025a. Anymac: Cascading flexible multi-agent collaboration via next-agent prediction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11566–11578.

Yimeng Wang, Jiaxing Zhao, Hongbin Xie, Hexing Ma, Yuzhen Lei, Shuangxue Liu, Xuan Song, Zichen Zhang, and Haoran Zhang. 2026a. Metagen: Self-evolving roles and topologies for multi-agent llm reasoning. *arXiv preprint arXiv:2601.19290*.

Yutong Wang, Siyuan Xiong, Xuebo Liu, Wenkang Zhou, Liang Ding, Miao Zhang, and Min Zhang. 2026b. Agentdropoutv2: Optimizing information flow in multi-agent systems via test-time rectify-or-reject pruning. *arXiv preprint arXiv:2602.23258*.

Zhefan Wang, Yuanqing Yu, Wendi Zheng, Weizhi Ma, and Min Zhang. 2024. Macrec: A multi-agent collaboration framework for recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2760–2764.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. 2025b. AgentDropout: Dynamic agent elimination for token-efficient and high-performance LLM-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24013–24035, Vienna, Austria. Association for Computational Linguistics.

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

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. 2025. From human memory to ai memory: A survey on memory mechanisms in the era of llms. *arXiv preprint arXiv:2504.15965*.

Haoyuan Xu, Chang Li, Xinyan Ma, Xianhao Ou, Zihan Zhang, Tao He, Xiangyu Liu, Zixiang Wang, Jiafeng Liang, Zheng Chu, and 1 others. 2026. The evolution of tool use in llm agents: From single-tool call to multi-tool orchestration. *arXiv preprint arXiv:2603.22862*.

Rui Ye, Shuo Tang, Rui Ge, Yaxin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. 2025. MAS-GPT: Training LLMs to build LLM-based multi-agent systems. In *Forty-second International Conference on Machine Learning*.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. 2025. 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 (Volume 1: Long Papers)*, pages 6192–6217.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. MasRouter: Learning to route LLMs for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15549–15572, Vienna, Austria. Association for Computational Linguistics.

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

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

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Yu, and Tianlong Chen. 2025b. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *International Conference on Learning Representations*, volume 2025, pages 75389–75428.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2025c. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Forty-second International Conference on Machine Learning*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, and 1 others. 2025d. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pages 34040–34077.

Yaolun Zhang, Xiaogeng Liu, and Chaowei Xiao. 2025e. Metaagent: Automatically constructing multi-agent systems based on finite state machines. In *Forty-second International Conference on Machine Learning*.

Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Zou. 2026. Sirius: Self-improving multi-agent systems via bootstrapped reasoning. *Advances in Neural Information Processing Systems*, 38:124475–124504.

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

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

Zhilun Zhou, Zihan Liu, Jiahe Liu, Qingyu Shao, Yihan Wang, Kun Shao, Depeng Jin, and Fengli Xu. 2026. Resmas: Resilience optimization in llm-based multi-agent systems. *arXiv preprint arXiv:2601.04694*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*.

A Survey Methodology

To ensure a reproducible review of automated multi agent system optimization, we conducted a systematic literature search targeting publications between 2023 and 2026.

A.1 Search Strategy and Keywords

We queried major academic databases (IEEE Xplore, ACM Digital Library, Scopus, Google Scholar) using a structured boolean string combining the core technology, architecture, and optimization mechanisms:

> (*"Large Language Model"* OR *"LLM"*) AND (*"Multi-Agent System"* OR *"MAS"* OR *"Agentic Workflow"*) AND (*"Automated Search"* OR *"Topology Optimization"* OR *"Dynamic Routing"* OR *"Prompt Evolution"* OR *"Generative Architecture"*).

We iteratively refined keywords based on initial returns, integrating specific algorithmic terms including *"Monte Carlo Tree Search"* and *"Variational Graph Autoencoder"*.

A.2 Inclusion and Exclusion Criteria

Papers were included if they explicitly proposed a method for automatically generating, routing, or evolving the structural topology, agent composition, or textual prompts of an LLM based multi agent system. We excluded: (1) manual, static multi agent pipelines lacking automated optimization; (2) research focused exclusively on internal foundational model weight updates; (3) research focused on single agent optimization; and (4) applications utilizing existing MAS frameworks without contributing novel automated design mechanisms. Due to the rapid progression of this research domain, peer-reviewed literature was supplemented with non-peer-reviewed pre-prints from open-access repositories to capture contemporary architectural methodologies.

B MAS Optimization Taxonomy

Framework	Optimization Scope	Temporal Phase	Algorithmic Paradigm
Combinatorial & Evolutionary Search			
ADAS (Hu et al., 2025a)	Agent, Topo, Prompt	Offline	Foundation Model Guided Search
AFLOW (Zhang et al., 2025d)	Agent, Topo, Prompt	Offline	Monte Carlo Tree Search
AgentSquare (Shang et al., 2025)	Agent, Topo, Prompt	Per Query	Modular Evolutionary Recombination
EvoAgent (Yuan et al., 2025)	Agent, Topo, Prompt	Per Query	Agent Population Evolution
EvoFlow (Zhang et al., 2025a)	Agent, Topo, Prompt	Offline	Tag Based Evolutionary Search
EvoPrompt (Guo et al., 2024a)	Prompt	Per Query	Semantic Evolutionary Crossover
MASS (Zhou et al., 2025)	Topo, Prompt	Offline	Hierarchical Interleaved Search
Promptbreeder (Fernando et al., 2024)	Prompt	Offline	Self Referential Lamarckian Evolution
Reward Driven Policy Optimization			
AgentDropout (Wang et al., 2025b)	Topo	Runtime	Dynamic Node Pruning via Thresholds
AgentFlow (Li et al., 2026c)	Topo	Runtime	Proximal Policy Optimization
AgentPrune (Zhang et al., 2025b)	Topo	Per Query	Policy Gradient with Low Rank Masking
AnyMAC (Wang et al., 2025a)	Topo	Runtime	RL with Context Sparsity Penalty
DyLAN (Liu et al., 2023)	Topo	Runtime	Heuristic Based Node Pruning
GPTSwarm (Zhuge et al., 2024)	Topo, Prompt	Offline	Continuous Policy Gradient
MAS Orchestra (Ke et al., 2026)	Agent, Topo, Prompt	Per Query	Holistic Single Step RL Orchestration
MasRouter (Yue et al., 2025)	Topo	Per Query	Variational RL Cascade Routing
Puppeteer (Dang et al., 2026)	Topo	Runtime	RL with Step Wise Cost Penalty
ResMAS (Zhou et al., 2026)	Topo, Prompt	Per Query	Topology Aware Policy Gradient
Generative Architecture Synthesis			
ARG Designer (Li et al., 2026a)	Agent, Topo, Prompt	Per Query	Autoregressive GRU Generation
AutoFlow (Li et al., 2024)	Agent, Topo, Prompt	Per Query	Program Synthesis via RL Fine Tuning
G Designer (Zhang et al., 2025c)	Topo	Per Query	Variational Graph Autoencoder
GTD (Jiang et al., 2025)	Topo	Per Query	Discrete Reverse Denoising Diffusion
MAS GPT (Ye et al., 2025)	Agent, Topo, Prompt	Per Query	Generative Language Modeling
MetaAgent (Zhang et al., 2025e)	Topo	Per Query	Finite State Machine Generation
MetaGen (Wang et al., 2026a)	Agent, Topo, Prompt	Runtime	Dynamic LLM Driven Synthesis
OFA MAS (Li et al., 2026b)	Topo	Per Query	Mixture of Experts Graph Generation
Gradient Based & Trajectory Feedback			
AgentVerse (Chen et al., 2024)	Agent, Topo, Prompt	Runtime	Explicit Verbal Critique
DSPy (Khattab et al., 2024)	Prompt	Offline	Backpropagated Textual Gradients
SIRIUS (Zhao et al., 2026)	Prompt	Offline	Supervised Tuning on Trajectories
TextGrad (Yuksekgonul et al., 2025)	Prompt	Offline	LLM Computed Textual Gradients

Table 1: Taxonomy of Automated Multi Agent System Optimization Frameworks. Optimization Scope specifies whether the framework targets Agent Composition (Agent), Communication Topology (Topo), or Textual Prompts (Prompt). Temporal Phase maps evaluation periods to Offline tuning, Per Query synthesis, or dynamic Runtime adaptation.