LEARN TO COLLABORATE: AUTOREGRESSIVE DIVERSITY-AWARE GRAPH GENERATION FOR MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Model (LLM)-based multi-agent systems have demonstrated strong capabilities on complex reasoning and decision-making tasks. However, existing approaches typically suffer from temporal-agnostic node selection, homogeneous interaction patterns, or topology-agnostic LLM routing, which may compromise collaborative performance while increasing computational costs. In this work, we propose GraphMAC, an autoregressive diversity-aware graph generation method for multi-agent systems. GraphMAC formulates multi-agent collaboration consisting of three stages: temporal-aware node selection, diverse edge construction, and graph-aware LLM allocation. Specifically, we first recruit agents autoregressively using a GRU-based historical aggregation mechanism that captures temporal dependencies during role selection. We then construct heterogeneous interaction graphs with multiple self-loop and cross-node edge types, enabling diverse collaborative interaction patterns. Finally, we employ relational graph convolutional networks to perform topology-aware message passing and dynamically allocate heterogeneous LLM backbones from a graph-level perspective. To jointly optimize task performance, interaction diversity, and computational cost, we further introduce a reinforcement learning objective with structural diversity entropy. Extensive experiments on multiple datasets demonstrate that GraphMAC consistently outperforms existing methods, achieving up to 5.8% performance improvement while maintaining superior cost-performance trade-offs.

1 INTRODUCTION

Large Language Model (LLM)-based agents (Luo et al., 2025; Mohammadi et al., 2025; Tran et al., 2025; Qi et al., 2026) have achieved remarkable progress, finding widespread application across diverse domains such as code generation (Wang et al., 2026; Li et al., 2025), mathematical reasoning (Luo et al., 2026; Liu et al., 2026a), social simulation (Huang et al., 2026; Mou et al., 2026), and embodied intelligence (Obayashi et al., 2025; Wu et al., 2026). When tackling complex real-world scenarios, LLM-based Multi-Agent Systems (MAS) (Yang et al., 2026; Zhou et al., 2026; Hao et al., 2026) where multiple specialized agents collaborate in a coordinated manner have demonstrated substantially superior performance compared to single-agent systems. For example, in code generation, a programmer can produce an initial solution, a test engineer can validate it, and a critic can identify errors and suggest revisions, enabling the agents to collaboratively develop a correct and reliable program. With the growing availability of LLMs, models with larger parameter counts generally entail considerably higher inference costs. Paradoxically, small LLMs can match or even surpass the performance of their larger counterparts on certain tasks (Shen et al., 2024; Lepagnol et al., 2024). This observation motivates a natural question: *How can we construct an effective multi-agent system and route each agent to the most suitable LLM to achieve an optimal balance between collaboration performance and computational cost?*

To tackle this problem, Multi-Agent System Routing (MasRouter) (Yue et al., 2025) has been proposed (as shown in Figure 1(a)), which employs a cascaded controller network to determine the collaboration pattern, role assignment, and LLM routing. While MasRouter achieves superior performance over single-agent routing through effective multi-agent collaboration, it still suffers from several limitations:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

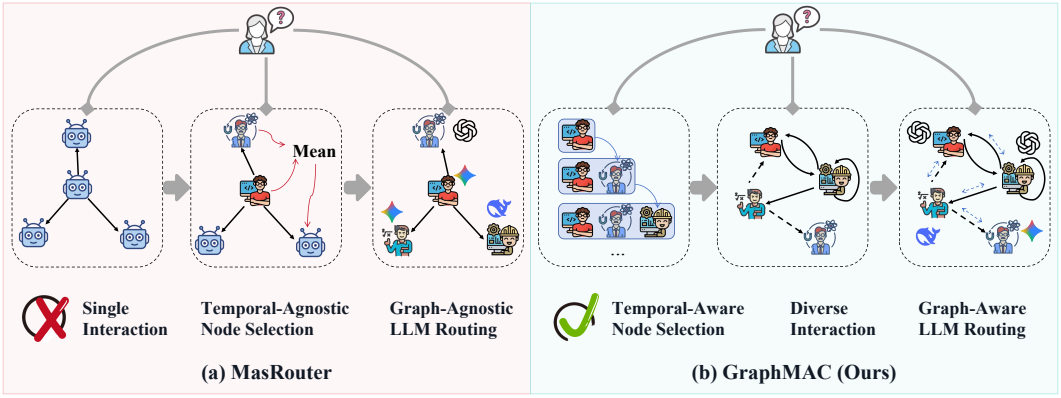


Figure 1: Comparison of MasRouter versus GraphMAC. GraphMAC models collaboration through temporal-aware node selection, diverse interactions, and graph-aware LLM routing.

Who to select? Selecting the most suitable roles from a candidate pool poses a significant challenge. Although MasRouter adopts an autoregressive approach to role selection and achieves reasonable performance, the historical role information is compressed via arithmetic mean, which inherently neglects the temporal dependencies in the role selection process. Since roles selected at different steps contribute equally to the mean-pooled historical representation, the selector cannot explicitly model temporal dependencies of previous selections. Consequently, the team construction process may become close to memoryless stepwise sampling.

How to interact? Modeling the interactions among agents is a major challenge, as it governs the flow and exchange of information. Prior work constructs interaction topologies through approaches such as edge pruning or autoencoders (Zhang et al., 2025a; Li et al., 2026b; Zhang et al., 2025b), whereas MasRouter restricts its selection to a single pattern drawn from a fixed, predefined collaboration pool. As a result, agent connections remain homogeneous and confined to a single interaction mode, severely limiting the structural expressiveness of the collaboration graph.

How to route? Assigning an appropriate LLM to each agent is critical, as it reflects the overall resource allocation of the system (Hu et al., 2024; Li et al., 2026a). Directly applying single-agent routing methods to multi-agent systems may lead to suboptimal adaptation. (Ong et al., 2025; Feng et al., 2025). MasRouter computes LLM selection probabilities for all agents using a single global context, without modeling the graph-structural dependencies among nodes. Consequently, the LLM routing for each agent is unable to perceive its topological position and neighborhood relationships within the collaboration graph.

To address these challenges, we propose an autoregressive diversity-aware graph generation method. As illustrated in Figure 1(b), inspired by human collaboration, GraphMAC first progressively recruits suitable agents through a temporal-aware autoregressive node selection process, then constructs diverse edge types to model the complex interaction patterns among agents, and finally allocates computational resources via a graph-aware LLM routing process. The process is expressed as: **(1) Who to select $\rightarrow$ Node Selection:** GraphMAC first derives query-aware role representations. A gated recurrent unit (GRU) is then employed to dynamically adjust the historical weights. Finally, appropriate roles are autoregressively selected from the role pool as nodes, conditioned on both the query and the historical vector. **(2) How to interact $\rightarrow$ Edge Construction**: Diverse types of self-loop edges and cross-node edges are utilized. The optimal edge type is selected by computing the similarity between node-pair features and edge features, and redundant edges are pruned based on the selection probabilities. **(3) How to route $\rightarrow$ Graph Allocation**: The proposed method employs a Relational Graph Convolutional Network (R-GCN) to perform message passing over the collaboration graph, followed by selecting the most suitable LLM for each node based on the query and node features. **(4) Policy Optimization**: The REINFORCE algorithm is adopted for policy optimization. The reward function incorporates utility, cost, and edge diversity. A diversity entropy is introduced to encourage diverse interaction patterns.

We summarize our contributions as follows:

- We identify three key limitations in existing multi-agent systems: temporal-agnostic node selection, single interaction patterns, and graph-agnostic LLM routing. These limitations hinder effective collaboration among agents.
- GraphMAC constructs multi-agent collaboration by autoregressively recruiting agents through temporal-aware node selection, modeling diverse interactions with heterogeneous edge types, and performing graph-aware LLM routing. This design enhances the structural expressiveness of collaboration graphs while jointly optimizing task utility, computational cost, and interaction diversity.
- Experiments on multiple datasets show that GraphMAC achieves superior performance–cost trade-offs, with case studies further visualizing collaboration graphs it generated.

2 Related Work

2.1 Multi-Agent Systems

LLM-based Multi-Agent Systems (MAS) (Yang et al., 2026; Zhou et al., 2026; Yun et al., 2026) aim to construct multi-agent networks by optimizing communication topologies and role configurations. Existing methods can be broadly categorized into two paradigms: static MAS and dynamic MAS. Static MAS rely on predefined fixed agent networks, such as chain, tree, fully-connected graph (Qian et al., 2025), and debate (Chan et al., 2024), which lack flexibility to adapt to varying queries. Dynamic MAS adaptively adjust roles and interactions based on environment feedback. For instance, GPTSwarm (Zhuge et al., 2024) dynamically optimizes graph-based multi-agent networks, AgentPrune (Zhang et al., 2025a) refines the topology through edge pruning, and AFlow (Zhang et al., 2025c) leverages Monte Carlo Tree Search to iteratively optimize both workflow topology and operator configurations. However, existing MAS approaches are constrained by homogeneous interaction and are predominantly driven by a single LLM. In contrast, our method achieves superior performance through diversified interaction design and automatic selection of heterogeneous LLMs.

2.2 LLM Routing

LLM routing is an effective approach to balancing task performance and computational cost (Hu et al., 2024; Li et al., 2026a). It can be broadly divided into two categories: single-agent LLM routing and multi-agent LLM routing. Single-agent routing methods range from encoder-based models such as BERT (Devlin et al., 2019) that make binary decisions (Huang et al., 2024; Ong et al., 2025; Chen et al., 2023) on whether to invoke a larger LLM, to more sophisticated routers that jointly consider the performance and cost trade-offs across multiple LLMs (Chen et al., 2024b). While these methods are effective in single-agent settings, they are unable to capture the collaborative interactions among multiple agents and therefore cannot be directly transferred to MAS. The representative multi-agent routing method MasRouter (Yue et al., 2025), achieves competitive performance through collaborative pattern selection, role selection, and LLM routing. However, its role selection process neglects the temporal dependencies among selected roles, the interaction topology is restricted to a single type, and the LLM routing stage does not explicitly account for graph dependencies. The proposed method addresses these limitations through temporal-aware role selection, diverse interaction, and graph-aware LLM routing, ensuring the effectiveness of MAS.

3 Preliminaries

A multi-agent system is formally defined as a directed heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$, where the node set $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ denotes the collection of agents and n is the number of agents. Each agent v_i is characterized by its own role definition and a selected LLM backbone ℓ_i. The edge set encodes the interaction modalities among agents, where each directed heterogeneous edge $e = (i, j, t) \in \mathcal{E}$ indicates that agent i applies an interaction of type $t \in \mathcal{T}$ to agent j.

The objective of multi-agent collaboration is to enable a set of agents to collaboratively process a given query and collectively produce the response. The collaborative graph generation process is modeled as $\mathbb{F}(\mathcal{G})$ and factorized into three sequential components $\mathbb{F}_\theta = \mathbb{F}_{\theta_v} \circ \mathbb{F}_{\theta_e} \circ \mathbb{F}_{\theta_l}$, which are jointly parameterized as $\theta = (\theta_v, \theta_e, \theta_l)$. Given a query q along with a role pool $\mathcal{R}$, an edge-type

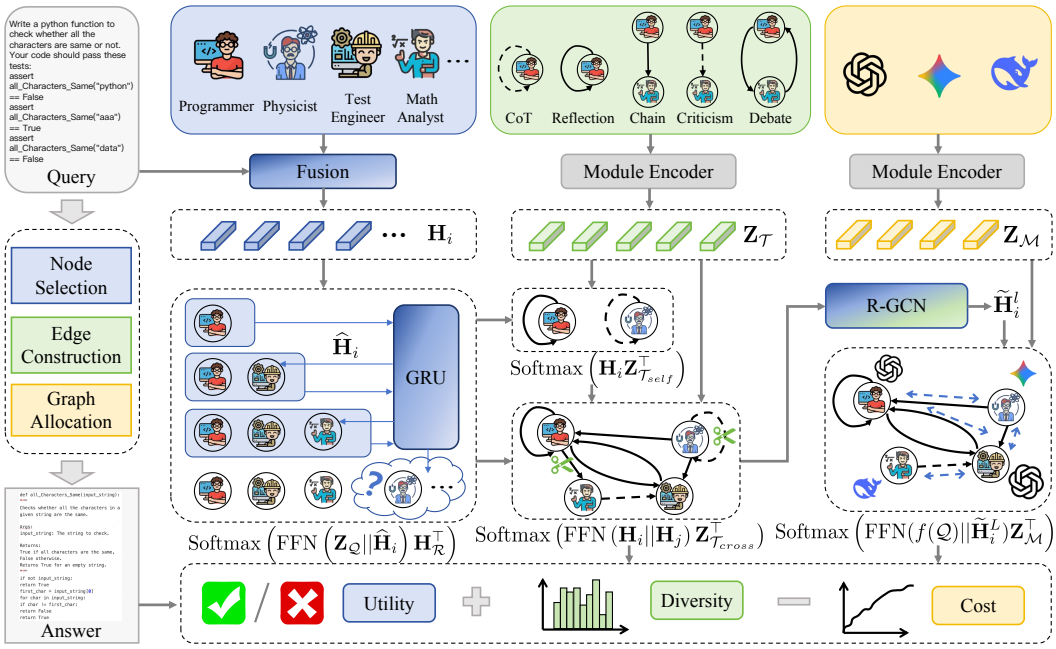


Figure 2: Overview of the proposed GraphMAC method. Appropriate roles are first selected leveraging query-aware role representations and the historical vector produced by the GRU model. Self-loop and cross-node edges are then sampled and pruned from diverse edges. Subsequently, the R-GCN model performs message passing and routes an appropriate LLM to each node. Finally, policy optimization is conducted based on answer correctness, edge diversity, and computational cost.

pool $\mathcal{T}$, and an LLM pool $\mathcal{M}$, this process can be expressed as:

$$P(\mathcal{G} \mid \mathcal{Q}, \mathcal{R}, \mathcal{T}, \mathcal{M}) = \underbrace{\prod_{i=1}^{n} P(v_i \mid v_{0:i-1},\, \mathcal{Q},\, \mathcal{R})}_{\text{Node Selection}} \cdot \underbrace{\prod_{i,j=1}^{n\times n} P(e_{ij} \mid v_i, v_j,\, \mathcal{Q},\, \mathcal{T})}_{\text{Edge Construction}} \cdot \underbrace{\prod_{i=1}^{n} P(l_i \mid \mathcal{V}, \mathcal{E}, \mathcal{Q},\, \mathcal{M})}_{\text{Graph Allocation}}. \tag{1}$$

Once the multi-agent system is formed, each agent produces an output. The final output is obtained by aggregating the outputs of all agents: $\mathcal{A} = \text{Aggregate}(\{\mathcal{O}_i\}_{i=1,2,\cdots,n})$.

4 Methodology

The proposed GraphMAC consists of four stages: node selection (Section 4.1), edge construction (Section 4.2), graph allocation (Section 4.3), and policy optimization (Section 4.4). The overview of GraphMAC is depicted in Figure 2. The pseudo-code can be found in Appendix A.2.

4.1 Node Selection

The node selection process involves autoregressively selecting an appropriate subset of agents for a given query. At each step i, the node selector assigns a role to the next agent. Query-aware role representations are constructed to model the complex relationships between the query and the agents, formulated as:

$$\mathbf{H}_i = \text{Fusion}\left(f(\mathcal{R}_i), \mathbf{Z}_\mathcal{Q}\right), \quad \mathbf{Z}_\mathcal{Q} \sim \mathcal{N}\left(\mu_\theta(\mathcal{Q}), \text{diag}\left(\sigma_\theta^2(\mathcal{Q})\right)\right), \tag{2}$$

where the role representations $\mathbf{H}_i$ of each agent are dynamically adjusted with respect to the query via fusion, yielding query-aware role representations. $\mathbf{Z}_\mathcal{Q}$ is sampled from a Gaussian distribution conditioned on the query to capture the latent high-dimensional semantics of the query. f denotes a text encoder (*e.g.*, Sentence-BERT (Reimers & Gurevych, 2019), MiniLM (Wang et al., 2020)).

We compute the prediction probability of each role in the role pool by measuring the similarity between the accumulated historical information combined with the query and the remaining role representations, defined as:

$$P(v_i \mid v_{0:i-1},\, \mathcal{Q},\, \mathcal{R}) = \mathrm{Softmax}\left(\mathrm{FFN}\left(\mathbf{Z}_{\mathcal{Q}} || \widehat{\mathbf{H}}_i\right) \mathbf{H}_{\mathcal{R}}^{\top}\right), \quad \widehat{\mathbf{H}}_i = \mathrm{GRU}\left(\mathbf{H}_{i-1}, \widehat{\mathbf{H}}_{i-1}\right), \quad (3)$$

where the historical feature $\widehat{\mathbf{H}}$ is derived from a gated recurrent unit (GRU) model and $\mathbf{H}_{\mathcal{R}}$ denotes the representation matrix of all candidate roles in the role pool. The gating mechanism of GRU enables the model to dynamically adjust historical weights, enabling each role selection step to make more complementary decisions with full awareness of preceding selections. We leverage the latent features of the query, passed through a sigmoid activation function, to estimate the complexity of the query. The resulting value is then multiplied by the maximum number of agents to determine the number of agents n to be selected for the multi-agent system. The agent subset is subsequently obtained through n rounds of selection.

4.2 Edge Construction

Diverse collaboration patterns can significantly enhance the capabilities of multi-agent systems. Inspired by prior work, we model both the self-interaction of individual agents and the diverse interaction relationships among agents, rather than restricting to a single, fixed interaction paradigm. The construction process of self-loop edges and cross-node edges is formulated as follows:

$$P(e_{ij} \mid v_i, v_j,\, \mathcal{Q},\, \mathcal{T}) = \mathrm{Softmax}\left(\mathrm{FFN}\left(\mathbf{H}_i || \mathbf{H}_j\right) \mathbf{Z}_{\mathcal{T}_{cross}}^{\top}\right), \quad \mathbf{Z}_{\mathcal{T}_{cross}} = \mathrm{FFN}\left(f(\mathcal{T}_{cross})\right), \quad (4)$$

$$P(e_{ii} \mid v_i, \mathcal{Q},\, \mathcal{T}) = \mathrm{Softmax}\left(\mathbf{H}_i \mathbf{Z}_{\mathcal{T}_{self}}^{\top}\right), \quad \mathbf{Z}_{\mathcal{T}_{self}} = \mathrm{FFN}\left(f(\mathcal{T}_{self})\right), \quad (5)$$

where $\mathbf{Z}_{\mathcal{T}}$ denotes the embedding of each edge type, obtained by processing the edge type description through a text encoder followed by a feed-forward network (FFN). Cross-agent edges comprise three types—(1) Chain (Qian et al., 2025): the classical downstream edge where agent v_j handles the query q conditioned on the output of agent v_i, propagating information in a single feed-forward pass. (2) Criticism (Huang & Chang, 2023): a supervisory edge where agent v_j critically evaluates the intermediate output of v_i, and v_i re-handles the query q with reference to the critique of v_j, simulating an evaluation-and-revision loop. (3) Debate (Chan et al., 2024): a dialectical edge modeling a two-round argumentative exchange in which v_j challenges the proposition of v_i and vice versa, after which both agents finalize the query q under the accumulated debate context—while self-loop edge types include two variants—(1) Chain-of-Thought (CoT) (Wei et al., 2022): the agent decomposes and reasons through the query q step by step within a single call, producing an explicit intermediate reasoning trace before answering. (2) Reflection (Shinn et al., 2023): the agent first drafts an initial answer to q, then self-critiques the draft and revises it in a subsequent pass, emulating an internal review-and-refine process. The appropriate edge type is sampled from the edge type pool $\mathcal{T}$.

Unlike fully-connected edges, we leverage the edge probabilities to prune redundant edges, retaining only semantically meaningful connections while substantially reducing computational cost (Wang et al., 2025; Li et al., 2024). This is formulated as follows:

$$\mathcal{E}'_{cross} = \mathrm{TopK}\left(\left\{P\left(e_{ij}\right)\right\}_{\forall i,j}, \delta\right), \quad \mathcal{E}'_{self} = \mathrm{TopK}\left(\left\{P\left(e_{ii}\right)\right\}_{\forall i}, \delta\right), \quad (6)$$

where δ denotes the retention ratio of edges.

4.3 Graph Allocation

Assigning appropriate LLMs to each agent from a graph-level perspective is crucial, as it reflects the overall resource allocation and contributes to reducing cost. We employ a simple relational graph convolutional network (Schlichtkrull et al., 2018; Wang et al., 2024) to achieve hierarchical message passing between nodes, formulated as follows:

$$\widetilde{\mathbf{H}}_i^{l+1} = \mathrm{ReLU}\left(\sum_{t \in \mathcal{T}_{self}} \widetilde{\mathbf{H}}_i^{l} \mathbf{W}_{self,t}^{l} + \sum_{t \in \mathcal{T}_{cross}} \sum_{j \in \widetilde{\mathcal{N}}_i^t} \frac{1}{|\widetilde{\mathcal{N}}_i^t|} \widetilde{\mathbf{H}}_j^{l} \mathbf{W}_t^{l}\right), \quad (7)$$

where $\widetilde{\mathbf{H}}_i^l$ denotes the node representation after message passing and $\widetilde{\mathbf{H}}_i^0 = \mathbf{H}_i$, which aggregates information from neighboring nodes. $\widetilde{\mathcal{N}}_i^t$ denotes the set of neighbors of node i, and $\mathbf{W}_{self,t}^l$ and $\mathbf{W}_t^l$ denote the weight matrices. During the graph allocation process, the probability assigned to each LLM for each node is computed as:

$$P(\ell_i \mid \mathcal{V}, \mathcal{E}, \mathcal{Q}, \mathcal{M}) = \mathrm{Softmax}\left(\mathrm{FFN}(f(\mathcal{Q})||\widetilde{\mathbf{H}}_i^L)\mathbf{Z}_{\mathcal{M}}^\top\right), \quad \mathbf{Z}_{\mathcal{M}} = \mathrm{FFN}\left(f(\mathcal{M})\right). \tag{8}$$

Rather than relying solely on aggregated graph representations, we directly compute the relevance between the agent features combined with the query features and the vectorized LLM descriptions, enabling more accurate LLM routing.

4.4 Policy Optimization

On benchmark $\mathcal{D}$, given a set of query-answer pairs and a graph $\mathcal{G}$ retrieved from the $\mathbb{F}_\theta$ based on the query, the optimal strategy for multi-agent routing is to balance task performance, computational cost, and structural diversity. The objective function is defined as:

$$\mathcal{P}(\theta) = \mathbb{E}_{(\mathcal{Q},\mathcal{A})\sim\mathcal{D},\mathcal{G}\sim\mathbb{F}_\theta}[\underbrace{U(\mathcal{G};\mathcal{Q},\mathcal{A})}_{\text{Utility}} + \lambda_d \cdot \underbrace{D(\mathcal{G};\mathcal{Q})}_{\text{Diversity}} - \lambda_c \cdot \underbrace{C(\mathcal{G};\mathcal{Q})}_{\text{Cost}}], \tag{9}$$

where the utility term measures task performance, the cost term quantifies the expected overhead (*e.g.*, LLM calls, API cost, token cost), and the diversity term measures the variety of edge types employed in the collaboration graph. λ_c and λ_d are trade-off parameters controlling the relative importance of cost and diversity, respectively.

Accordingly, the reward function R_i is represented as follows:

$$R(\mathcal{G})_i = 1\left[\mathcal{A}_i \text{ is correct}\right] + \lambda_d \cdot \mathrm{div}_i - \lambda_c \cdot \mathrm{cost}_i, \tag{10}$$

$$\mathrm{div}_i = \frac{1}{|\mathcal{T}_{\text{self}}| + |\mathcal{T}_{\text{cross}}|} \sum_{t\in\mathcal{T}_{\text{self}}\cup\mathcal{T}_{\text{cross}}} 1[\exists(u,v) : \mathcal{E}(u,v) = t], \tag{11}$$

where $1[\cdot]$ is an indicator function that equals 1 when the generated answer is correct and 0 otherwise. The cost represents the expense incurred by token generation during LLM inference. The diversity term div_i measures the normalized number of distinct edge types utilized in the collaboration graph. A greater diversity of edge types yields a higher reward, indicating richer and more varied interaction patterns among agents.

We optimize the objective using the REINFORCE policy gradient algorithm (Williams, 1992; Konda & Tsitsiklis, 1999). To improve training stability, we subtract a baseline $\overline{R}$ implemented as an exponential moving average of historical rewards, which effectively reduces variance (Chen et al., 2024a). Furthermore, a structural diversity entropy regularization term is incorporated into the reward signal to discourage mode collapse and encourage the policy to explore a broader range of interaction patterns across edge types (Ma et al., 2026; Liu et al., 2026b; Sun et al., 2026). In graph generation $\mathbb{F}_\theta$ parameterized by θ, the policy gradient is obtained by maximizing the objective $\max \mathcal{P}(\theta)$:

$$\nabla_\theta \mathcal{P}(\theta) = \mathbb{E}_{(\mathcal{Q},\mathcal{A})\sim\mathcal{D},\mathcal{G}\sim\mathbb{F}_\theta}\left[\nabla_\theta \log \mathbb{F}_\theta(\mathcal{G}) \cdot (R(\mathcal{G}) - \overline{R}) + \eta\nabla_\theta\left(-\sum_{t\in\mathcal{T}_{\text{self}}\cup\mathcal{T}_{\text{cross}}} d_t \log d_t\right)\right], \tag{12}$$

where η is a trade-off parameter controlling the weight of structural diversity entropy and d_t denotes the expected selection probability of edge type t under the current policy.

5 Experiments

5.1 Experimental Setup

Datasets and Metrics We evaluate our approach on five widely adopted datasets: MMLU (Hendrycks et al., 2020) for knowledge comprehension and reasoning, GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) for mathematical problem-solving, and HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for code generation. We report accuracy for MMLU, GSM8K, and MATH, and adopt pass@1 as the metric for HumanEval and MBPP. All datasets are partitioned into training and test sets at a 4:1 ratio. For MATH, we apply sampling to select 519 problems across varying difficulty levels, ensuring balanced representation of problem complexity.

Table 1: Performance comparison of GraphMAC with existing methods across five datasets.

Method	LLM Pool	MAS	Routing	MATH	MBPP	GSM8K	HumanEval	MMLU
Vanilla (GPT)	✗	✗	✗	66.09	70.20	77.97	72.05	69.28
Vanilla (GPT+)	✗	✗	✗	67.77	82.00	94.60	89.84	86.61
Vanilla (Gemini)	✗	✗	✗	74.61	77.20	87.31	92.19	78.77
Vanilla (DeepSeek)	✗	✗	✗	84.96	74.80	89.49	83.59	88.10
CoT	✗	✗	✗	67.24	69.60	93.68	86.69	78.43
ComplexCoT	✗	✗	✗	67.05	75.80	93.43	87.58	81.05
SC (CoT)	✗	✗	✗	66.28	73.00	93.32	87.58	81.05
SC (ComplexCoT)	✗	✗	✗	66.86	75.80	93.94	88.19	82.35
Chain	✗	✓	✗	64.72	75.40	94.40	85.63	82.01
Tree	✗	✓	✗	65.11	75.60	93.89	87.50	82.98
Complete Graph	✗	✓	✗	67.63	75.20	94.66	85.00	83.06
LLM-Debate	✗	✓	✗	64.68	73.60	94.66	84.38	81.04
GPTSwarm	✗	✓	✗	68.85	75.40	94.66	86.28	82.80
AgentPrune	✗	✓	✗	68.45	75.40	94.89	86.80	83.02
AFlow	✗	✓	✗	73.35	82.20	92.30	90.06	83.10
PromptLLM	✓	✗	✓	73.03	73.60	93.92	86.33	78.43
RouteLLM	✓	✗	✓	71.29	72.60	93.42	83.85	81.04
FrugalGPT	✓	✗	✓	67.05	74.40	90.76	87.31	76.24
RouterDC	✓	✗	✓	73.46	75.20	93.68	87.75	82.01
MasRouter	✓	✓	✓	88.48	83.00	94.79	95.31	89.58
GraphMAC	✓	✓	✓	**89.84**	**88.80**	**95.55**	**96.09**	**90.48**

Baselines Our baselines are categorized into five groups: (1) single-agent methods, comprising CoT (Wei et al., 2022), ComplexCoT (Fu et al., 2022), and Self-Consistency (SC) (Wang et al., 2023); (2) fixed multi-agent topologies, including Chain, Tree, Complete Graph (Qian et al., 2025), LLM-Debate (Chan et al., 2024); (3) multi-agent systems, consisting of GPTSwarm (Zhuge et al., 2024), AgentPrune (Zhang et al., 2025a), and AFlow (Zhang et al., 2025c); (4) single LLM routers, represented by PromptLLM (Huang et al., 2024), RouteLLM (Ong et al., 2025), FrugalGPT (Chen et al., 2023), and RouterDC (Chen et al., 2024b); and (5) the multi-agent system routing method MasRouter (Yue et al., 2025).

LLM Backbones We consider LLMs spanning a wide range of sizes and capacities, including GPT-4o-mini (GPT) (OpenAI, 2024a), GPT-4o-2024-11-20 (GPT+) (OpenAI, 2024b), Gemini-2.5-flash-lite (Gemini) (Google DeepMind, 2025), and DeepSeek-v4-flash-tencent (DeepSeek) (DeepSeek-AI, 2026). Those methods based on a single LLM all use GPT-4o-mini. Gemini-3.1-pro-preview (Gemini+) (Google DeepMind, 2026) is introduced to assess the inductive capabilities of GraphMAC.

Implementation Details Following the setup of MasRouter (Yue et al., 2025), we adopt a role pool of 26 roles covering domains such as code, commonsense, and math. The edge type pool consists of two self-loop edges (CoT and reflection) and three cross-node edges (chain, criticism, and debate), and the LLM pool contains four backbone models mentioned before. The learning rate is set to 0.01. Specific tuning includes the retention ratio of edges $\delta \in \{0, 0.25, 0.5, 0.75, 1\}$, diversity parameters $\lambda_d \in \{0, 0.25, 0.5, 0.75, 1\}$, cost parameters $\lambda_c \in \{0, 5, 10, 15, 20\}$ or $\{100, 200, 300, 400, 500\}$, and diversity entropy parameters $\eta \in \{0, 0.001, 0.01, 0.1, 1\}$.

5.2 Performance and Cost Comparison

Performance comparison demonstrates the effectiveness of GraphMAC in facilitating multi-agent collaboration. Table 1 suggests that GraphMAC consistently achieves stronger performance than existing methods across all five benchmarks. In particular, GraphMAC outperforms the recent multi-agent routing baseline MasRouter on every dataset, indicating the benefits of jointly modeling temporal-aware node selection, diverse interactions, and graph-aware LLM allocation. The largest

improvement is observed on MBPP, where GraphMAC achieves an accuracy gain of 5.80% over MasRouter, while also reaching 96.09% on HumanEval. These results demonstrate that the proposed collaboration graphs are especially effective for code generation and complex reasoning tasks.

Cost comparison demonstrates that GraphMAC improves task performance while maintaining high inference efficiency. As shown in Figure 3, GraphMAC achieves the largest cost reduction on MATH, reducing the average cost by approximately 43% (from 10.93 to 6.20) compared with MasRouter. It also reduces inference costs on most of the remaining datasets, while simultaneously delivering consistent performance improvements. Although GraphMAC incurs a slightly higher cost on GSM8K, the overall results reveal a favorable performance-cost trade-off. This suggests that the proposed edge pruning and graph-aware LLM routing mechanisms can effectively allocate computational resources to the most relevant agents without sacrificing task accuracy.

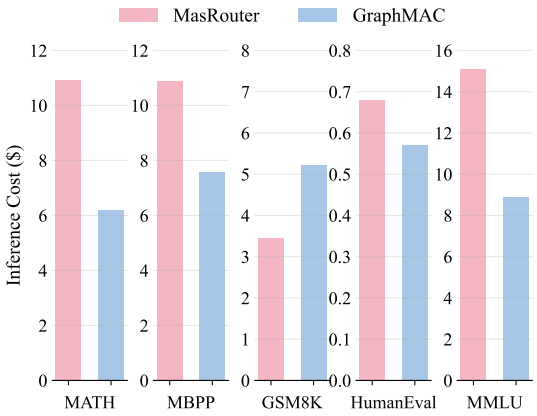


Figure 3: Inference cost comparison between MasRouter and GraphMAC across five datasets.

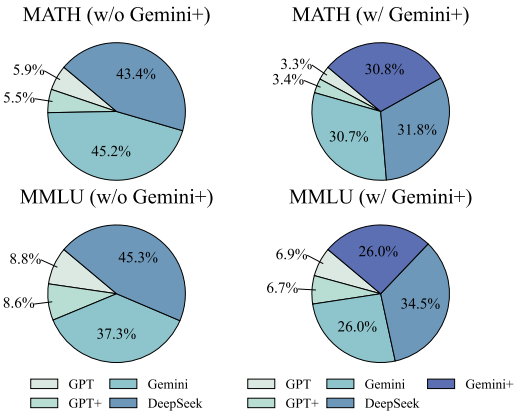


Figure 4: The selected LLM distribution of GraphMAC on MATH and MMLU datasets.

5.3 Ablation Study

Table 2 evaluates the contribution of each component in GraphMAC. Removing node selection, edge construction, or graph allocation consistently degrades performance on most datasets. In particular, removing graph allocation causes a notable accuracy drop on MATH, while removing edge construction substantially increases inference cost on MATH and MMLU. The full model achieves the best overall performance with competitive inference cost, demonstrating the effectiveness of all three components.

Table 2: Ablation comparison on five datasets. w/o denotes replacing the corresponding component with a random strategy.

Dataset	MATH		MBPP		GSM8K		HumanEval		MMLU	
Metric	Acc	Cost	Acc	Cost	Acc	Cost	Acc	Cost	Acc	Cost
w/o Node Selection	87.89	5.09	83.40	2.41	93.37	1.75	92.19	0.71	89.58	9.03
w/o Edge Construction	87.89	14.84	82.40	2.18	95.08	7.15	93.75	0.98	87.70	26.90
w/o Graph Allocation	80.66	4.36	85.60	7.60	94.89	3.95	95.31	0.88	87.80	8.19
Full	89.84	6.20	88.80	7.56	95.55	5.23	96.09	0.57	90.48	8.88

5.4 Inductive Ability Analysis

To evaluate GraphMAC's ability to generalize to unseen LLMs, we introduce the stronger Gemini-3.1-pro-preview (Gemini+) backbone during inference without retraining the routing policy. As shown in Figure 4, Gemini+ is selected for 30.8% of agent nodes on MATH and 26.0% on MMLU, while the remaining agents continue to use a diverse combination of existing backbones. With Gemini+ available, GraphMAC improves MATH accuracy from 89.84% to 92.00% and MMLU

accuracy from 90.48% to 90.87%. These results suggest that GraphMAC can adapt its routing decisions when the LLM pool is expanded. Rather than following a fixed model-specific allocation, it selectively integrates the newly available backbone according to the query and graph context, indicating promising generalization to unseen LLMs.

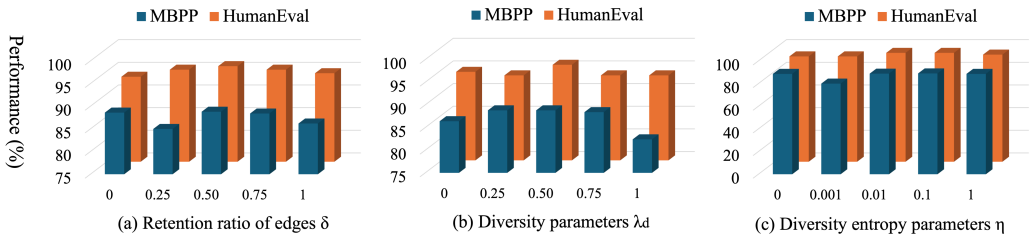


Figure 5: Hyperparameter analysis of GraphMAC on MBPP and HumanEval datasets.

5.5 Hyperparameter Analysis

We examine how the edge-retention ratio δ, diversity coefficient λ_d, and diversity entropy coefficient η affect GraphMAC in Figure 5. Overly sparse graphs restrict information exchange, whereas retaining a moderate fraction of candidate edges better balances collaboration quality and inference cost. Increasing λ_d generally promotes more heterogeneous interaction patterns and improves task performance, although excessive diversity regularization may introduce redundant communication. The diversity entropy coefficient η further encourages exploration of diverse edge types and generally maintains or improves performance, with a clearer benefit on MBPP. Overall, GraphMAC benefits from sufficient structural diversity while avoiding unnecessary communication overhead.

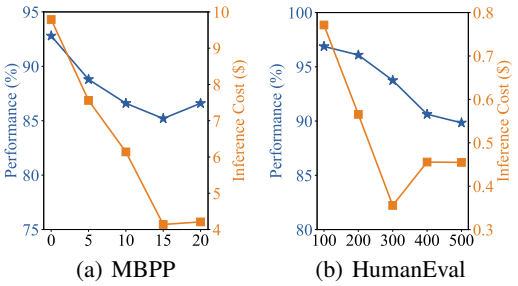


Figure 6: Cost parameter analysis of GraphMAC on MBPP and HumanEval datasets.

In Figure 6, we further study the effect of the cost parameter λ_c on MBPP and HumanEval. As λ_c increases, GraphMAC increasingly favors smaller or less expensive LLMs and sparser collaboration graphs, resulting in a consistent reduction in inference cost. This efficiency gain is accompanied by a mild decline in task performance when the cost penalty becomes overly dominant. The performance–cost trade-off is relatively stable across a broad range of λ_c, indicating that GraphMAC can flexibly adapt to different computational budgets. In our experiments, we select λ_c from the intermediate range to preserve strong performance while avoiding excessive inference expenditure.

5.6 Case Study

Case studies across five datasets in Appendix A.3 visualize multi-agent systems generated by our method and reveal its collaboration patterns.

6 Conclusion

We presented GraphMAC, an autoregressive diversity-aware graph generation method for effective multi-agent collaboration. GraphMAC integrates temporal-aware node selection, heterogeneous edge construction, and graph-aware LLM allocation to jointly optimize task performance, interaction diversity, and computational cost. Experiments on five datasets show that GraphMAC consistently outperforms existing methods, achieving up to a 5.8% improvement over MasRouter while maintaining favorable performance–cost trade-offs. In future work, we will explore more scalable graph generation, richer interaction types, and adaptive collaboration in dynamic environments.

REFERENCES

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

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate. In *International conference on learning representations*, volume 2024, pp. 9079–9093, 2024.

Jinchi Chen, Jie Feng, Weiguo Gao, and Ke Wei. Decentralized natural policy gradient with variance reduction for collaborative multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(172):1–49, 2024a.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

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

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems*, 37:66305–66328, 2024b.

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

DeepSeek-AI. Deepseek-v4-flash-tencent. `OFFICIAL_MODEL_CARD_OR_API_URL`, 2026. Accessed: 2026-09-23.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186, 2019.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. In *International Conference on Learning Representations*, volume 2025, pp. 26186–26203, 2025.

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

Google DeepMind. Gemini 2.5 flash-lite. `https://ai.google.dev/gemini-api/docs/models/gemini-2.5-flash-lite`, 2025. Accessed: 2026-09-23.

Google DeepMind. Gemini 3.1 pro preview. `OFFICIAL_MODEL_CARD_OR_API_URL`, 2026. Accessed: 2026-09-23.

Zhezheng Hao, Tianfu Wang, Huanshuo Dong, Ziyan Liu, Hong Wang, Xiankun Lin, Qiang Lin, Can Wang, Hande Dong, and Jiawei Chen. Evolve as a team: Collaborative self-evolution for llm-based multi-agent systems. *arXiv preprint arXiv:2605.29790*, 2026.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*, 2024.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In *Findings of the association for computational linguistics: ACL 2023*, pp. 1049–1065, 2023.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder task needs more experts: Dynamic routing in moe models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 12883–12895, 2024.

Renhong Huang, Ning Tang, Jiarong Xu, Yuxuan Cao, Qingqian Tu, Sheng Guo, Bo Zheng, Huiyuan Liu, and Yang Yang. Policysim: An llm-based agent social simulation sandbox for proactive policy optimization. In *Proceedings of the ACM Web Conference*, pp. 4781–4792, 2026.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12:1–7, 1999.

Pierre Lepagnol, Thomas Gerald, Sahar Ghannay, Christophe Servan, and Sophie Rosset. Small language models are good too: An empirical study of zero-shot classification. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pp. 14923–14936, 2024.

Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, et al. Llmrouterbench: A massive benchmark and unified framework for llm routing. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37733–37754, 2026a.

Jierui Li, Hung Le, Yingbo Zhou, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Codetree: Agent-guided tree search for code generation with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3711–3726, 2025.

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

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 2024*, pp. 7281–7294, 2024.

Fan Liu, Zherui Yang, Cancheng Liu, Tianrui Song, Xiaofeng Gao, and Hao Liu. Mm-agent: Llm as agents for real-world mathematical modeling problem. *Advances in Neural Information Processing Systems*, 38:20881–20934, 2026a.

Shiyu Liu, Yongjing Yin, Jianhao Yan, Yunbo Tang, Qinggang Zhang, Bei Li, Xin Chen, Jingang Wang, Xunliang Cai, and Jinsong Su. Bapo: Boundary-aware policy optimization for reliable agentic search. In *Findings of the Association for Computational Linguistics*, pp. 36408–36426, 2026b.

Haipeng Luo, Huawen Feng, Qingfeng Sun, Can Xu, Kai Zheng, Yufei Wang, Tao Yang, Winston Hu, and Yansong Tang. Agentmath: Empowering mathematical reasoning for large language models via tool-augmented agent. In *International Conference on Learning Representations*, volume 2026, pp. 56253–56305, 2026.

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*, 2025.

Shichao Ma, Zhiyuan Ma, Ming Yang, Xiaofan Li, Xing Wu, Jintao Du, Yu Cheng, Weiqiang Wang, Qiliang Liu, Zhengyang Zhou, et al. Tspo: Breaking the double homogenization dilemma in multi-turn search policy optimization. In *Findings of the Association for Computational Linguistics*, pp. 28607–28623, 2026.

Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 6129–6139, 2025.

Xinyi Mou, Xuanwen Ding, Qi He, Liang Wang, Jingcong Liang, Xinnong Zhang, Libo Sun, Jiayu Lin, Jie Zhou, Huang Xuanjing, et al. From individual to society: A survey on social simulation driven by large language model-based agents. *ACM Computing Surveys*, 58(11):1–41, 2026.

Nana Obayashi, Arsen Abdulali, Fumiya Iida, and Josie Hughes. Embodied intelligence paradigm for human-robot communication. *Science Robotics*, 10(105):eads8528, 2025.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025.

OpenAI. Gpt-4o mini: Advancing cost-efficient intelligence. `https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/`, 2024a.

OpenAI. Gpt-4o system card. `https://openai.com/index/gpt-4o-system-card/`, 2024b.

Shihao Qi, Jie Ma, Rui Xing, Wei Guo, Xiao Huang, Zhitao Gao, Jianhao Deng, Jun Liu, Lingling Zhang, Bifan Wei, et al. Beyond individual intelligence: Surveying collaboration, failure attribution, and self-evolution in llm-based multi-agent systems. *arXiv preprint arXiv:2605.14892*, 2026.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, et al. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*, volume 2025, pp. 41488–41505, 2025.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing*, pp. 3982–3992, 2019.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607, 2018.

Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. Small llms are weak tool learners: A multi-llm agent. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 16658–16680, 2024.

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.

Rui Sun, Jie Ding, Chenghua Gong, Tianjun Gu, Yihang Jiang, Juyuan Zhang, Liming Pan, and Linyuan Lü. Topodim: One-shot topology generation of diverse interaction modes for multi-agent systems. In *Findings of the Association for Computational Linguistics*, pp. 4252–4269, 2026.

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.

Fengyi Wang, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. Llm-enhanced cascaded multi-level learning on temporal heterogeneous graphs. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 512–521, 2024.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788, 2020.

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*, pp. 1–15, 2023.

Zhengren Wang, Chufan Wang, Yongan Yu, Sizhe Wang, Feiyu Xiong, Wentao Zhang, et al. Maintaincoder: Maintainable code generation under dynamic requirements. *Advances in Neural Information Processing Systems*, 38:15399–15436, 2026.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. 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*, pp. 24013–24035, 2025.

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

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Shaokai Wu, Yanbiao Ji, Qiuchang Li, Zhiyi Zhang, Qichen He, Wenyuan Xie, Guodong Zhang, Bayram Bayramli, Yue Ding, and Hongtao Lu. Dejavu: Towards experience feedback learning for embodied intelligence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 29578–29587, 2026.

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. *Advances in Neural Information Processing Systems*, 38:107309–107336, 2026.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. Masrouter: Learning to route llms for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 15549–15572, 2025.

Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Y Zou, Guohao Li, and Tianlong Chen. Graph-of-agents: A graph-based framework for multi-agent llm collaboration. In *International Conference on Learning Representations*, volume 2026, pp. 19745–19760, 2026.

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

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. In *Forty-second International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025b.

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

Zhilun Zhou, Zihan Liu, Jiahe Liu, Qingyu Shao, Yihan Wang, Kun Shao, Depeng Jin, and Fengli Xu. Resmas: Resilience optimization in llm-based multi-agent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35176–35184, 2026.

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*, volume 235, pp. 62743–62767, 2024.

A APPENDIX

A.1 LLM USAGE

Large Language Models (LLMs) were used to aid in the writing and polishing of the manuscript. Specifically, the LLMs supported sentence rephrasing, grammar correction, readability enhancement, and improvement of the overall clarity and flow of the text.

The LLMs were not involved in the conceptualization of the study, research methodology, experimental design, data analysis, or interpretation of the results. All scientific ideas, research decisions, analyses, and conclusions were developed and conducted by the authors.

The authors take full responsibility for the content of the manuscript, including all text that was generated or edited with the assistance of LLMs. All LLM-assisted text was carefully reviewed and verified by the authors. The authors also confirm that the use of LLMs did not involve plagiarism, fabrication, falsification, or any other form of research misconduct.

A.2 ALGORITHM

We present the overall algorithm workflow of GraphMAC in Algorithm 1.

Algorithm 1 Training of GraphMAC

Input: Benchmark dataset $\mathcal{D}$, role pool $\mathcal{R}$, edge type pool $\mathcal{T}$, LLM pool $\mathcal{M}$.

for each query $\mathcal{Q} \in \mathcal{D}$ **do**
  for each epoch **do**
    /* Node Selection */
    Sample latent query representation and construct query-aware role representations as Equation 2
    for $i = 1$ to n **do**
      Update historical state and compute role selection probability as Equation 3
      Select role v_i from role pool $\mathcal{R}$
    end for
    /* Edge Construction */
    for each node pair (v_i, v_j) **do**
      Compute cross-edge probability as Equation 4
      Compute self-edge probability as Equation 5
    end for
    Retain top-δ edges using Equation 6
    /* Graph Allocation */
    Perform relational graph convolution as Equation 7:
    for each selected node v_i **do**
      Compute LLM allocation probability as Equation 8
      Assign LLM ℓ_i from LLM pool $\mathcal{M}$
    end for
    /* Policy Optimization */
    Execute generated communication graph $\mathcal{G}$ on query $\mathcal{Q}$
    Compute reward as Equation 10
    Update parameters using REINFORCE as Equation 12
  end for
end for

A.3 CASE STUDY

Tables 3-7 illustrate the query-adaptive multi-agent collaboration graphs generated by GraphMAC across different difficulty levels on five datasets.

Table 3: Case study on MATH dataset.

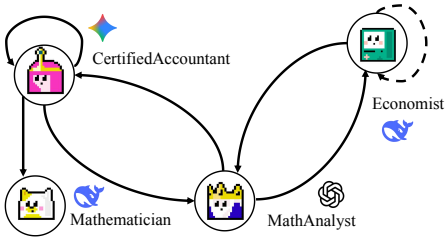
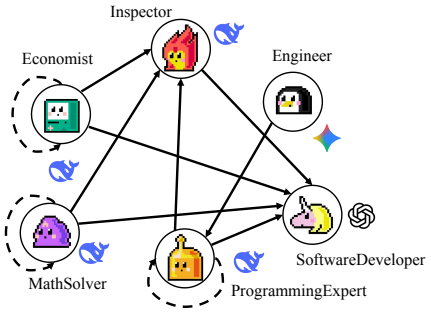
Query	Workflow
What is $\left(\frac{6}{7}\right)^2 \cdot \left(\frac{1}{2}\right)^2$?	
Find the smallest positive integer solution to $\tan 19x^\circ = \frac{\cos 96^\circ + \sin 96^\circ}{\cos 96^\circ - \sin 96^\circ}$.	

Table 4: Case study on MBPP dataset.

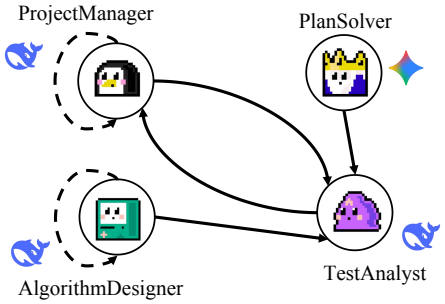
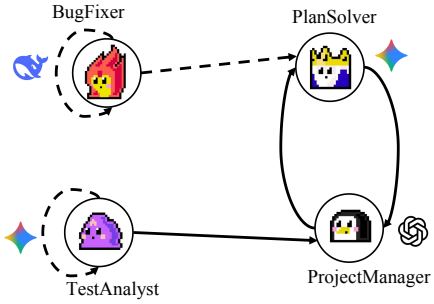
Query	Workflow
Write a python function to find the product of non-repeated elements in a given array. Your code should pass these tests: assert find_Product([1,1,2,3],4) == 6 assert find_Product([1,2,3,1,1],5) == 6 assert find_Product([1,1,4,5,6],5) == 120	
Write a function that matches a word at the end of a string, with optional punctuation. Your code should pass these tests: assert text_match_word(“python.”)==(‘Found a match!’) assert text_match_word(“python.”)==(‘Found a match!’) assert text_match_word(“ lang .”)==(‘Not matched!’)	

Table 5: Case study on GSM8K dataset.

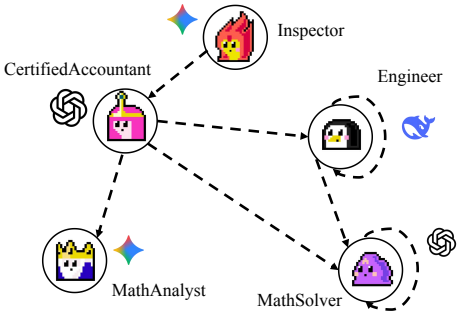
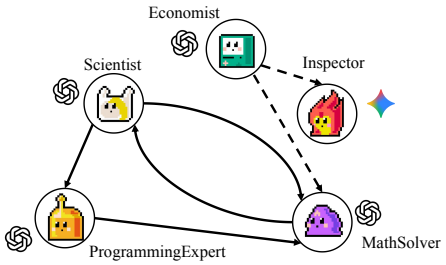
Query	Workflow
Bernie has 4 dogs. They each need a certain amount of exercise per day. The first needs to walk 1 mile. The second needs to walk 4 miles. The third needs to walk 3 miles. On average, they need to walk 3 miles per day. How many miles does the last dog need?	
Finley took part in a 100-meter race. She started off in first, but then fell back 5 spots. She then moved ahead 2 spots, before falling behind 3. Lastly, she jumped ahead 1 spot to finish the race. What place did she finish in?	

Table 6: Case study on HumanEval dataset.

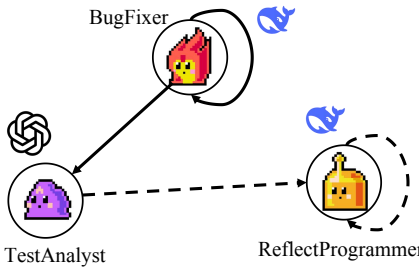
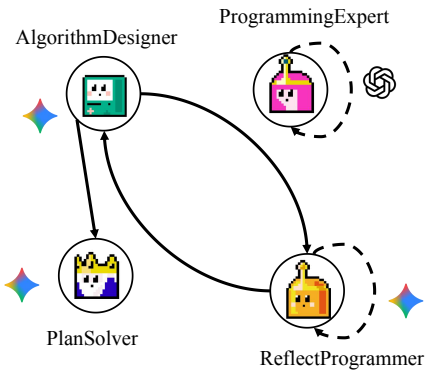
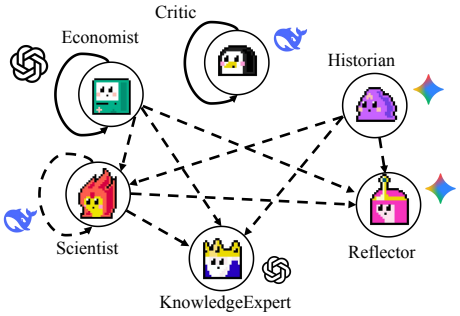
Query	Workflow
def words_in_sentence(sentence: str) -> str: """ You are given a string representing a sentence, the sentence contains some words separated by a space, and you have to return a string that contains the words from the original sentence, whose lengths are prime numbers, the order of the words in the new string should be the same as the original one. Example 1: »> words_in_sentence('This is a test') 'is' Example 2: »> words_in_sentence('lets go for swimming') 'go for' Constraints: * 1 <= len(sentence) <= 100 * sentence contains only letters """	
def string_xor(a: str, b: str) -> str: """ Input are two strings a and b consisting only of 1s and 0s. Perform binary XOR on these inputs and return result also as a string. »> string_xor('010', '110') '100' """	

Table 7: Case study on MMLU dataset.

Query	**Workflow**
Which of the following is NOT an anxiety disorder? Option A: panic disorder Option B: hypochondriasis Option C: agoraphobia Option D: post-traumatic stress disorder	
Which of the following factors should an external auditor obtain updated information about when assessing an internal auditor's competence? Option A: The reporting status of the internal auditor within the organization. Option B: The educational level and professional experiences of the internal auditor. Option C: Whether policies prohibit the internal auditor from auditing areas where relatives are employed. Option D: Whether the board of directors, audit committee, or owner-manager oversees employment decisions related to the internal auditor.	