GRAPH-OF-AGENTS: A GRAPH-BASED FRAMEWORK FOR MULTI-AGENT LLM COLLABORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

With an ever-growing zoo of LLMs and benchmarks, the need to orchestrate multiple models for improved task performance has never been more pressing. While frameworks like Mixture-of-Agents (MoA) attempt to coordinate LLMs, they often fall short in terms of (1) selecting relevant agents, (2) facilitating effective intra-agent communication, and (3) integrating responses efficiently. In this work, we propose Graph-of-Agents (GoA), a new graph-based framework for modeling multi-agent LLM communication. Our approach begins with node sampling, selecting only the most relevant agents by leveraging model cards that summarize each model's domain, task specialization, and other characteristics. Next, we construct edges between the selected agents by evaluating their responses against one another to determine relevance ordering. Directed message passing is then performed from highly relevant agents to less relevant ones to enhance their responses, followed by reverse message passing to refine the original responses of the more relevant agents. Finally, the updated responses are aggregated via graph-based pooling (e.g., max or mean pooling) to produce a single, unified answer. We evaluate GoA on diverse multi-domain benchmarks (MMLU, MMLU-Pro, GPQA) and domain-specific benchmarks (MATH, HumanEval, MedMCQA), with an agent pool of 6 LLMs spanning multiple domains. Surprisingly, GoA achieves superior performance using only 3 selected agents, outperforming recent multi-agent LLM baselines that utilize all 6 agents simultaneously. By adopting a graph structure, GoA offers both scalability and effectiveness through structured message passing—positioning it as a strong candidate for navigating the challenges of the ever-growing LLM zoo. The source code of GoA can be found at https://anonymous.4open.science/r/goa-F23C.

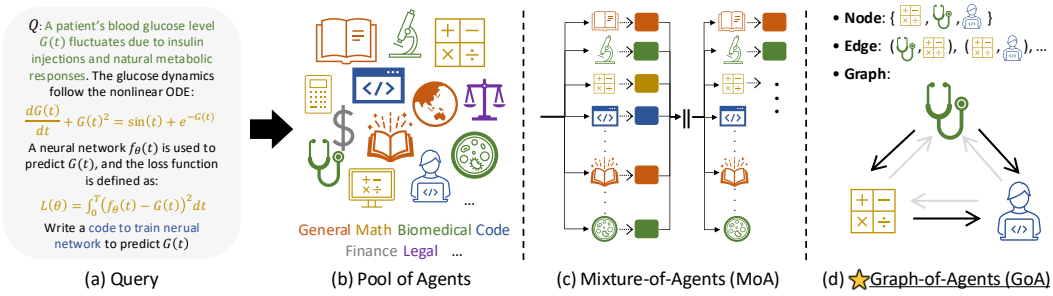
1 INTRODUCTION

Too many LLMs, too many benchmarks. As the ecosystem of Large Language Models (LLMs) (Wei et al., 2022a) rapidly diversifies, researchers are increasingly overwhelmed—not just by the sheer number of models available, but also by the complexity of evaluating and combining them effectively at test time to solve complex tasks. In this era of abundant LLMs, a central challenge emerges:

> ***(Q)*** *Given the diversity of available LLMs, how can we design an effective playground where agents interact synergistically—leveraging strengths, compensating for weaknesses, and improving decision-making through efficient collaboration?*

As an early attempt to address this challenge, Mixture-of-Agents (MoA) (Wang et al., 2024a; Li et al., 2025) has recently been introduced as a pioneering approach that explores how leveraging multiple LLMs can enhance overall model performance through the Mixture-of-Experts (MoE) (Shazeer et al., 2017) framework. As illustrated in Figure 1 (c), MoA aggregates responses from multiple LLM agents, appends them to the original query and feeds the enriched input to the next layer. This facilitates multi-agent synergy, enabling models to complement each other and refine predictions collaboratively. While MoA has shown that integrating multiple LLMs can be advantageous over a single model, it still faces several limitations that hinder its scalability and effectiveness:

Which agents? As shown in Figure 1 (b), following the scaling law of diverse LLM agents, selecting a relevant subset from a large pool of agents handling complex and diverse queries (Figure 1 (a)) is a crucial challenge. The current MoA approach lacks an effective agent selection mecha-



(a) Query (b) Pool of Agents (c) Mixture-of-Agents (MoA) (d) ☆Graph-of-Agents (GoA)

Figure 1: Current multi-agent LLM pipeline and our proposed approach.(a) Given a query that spans diverse domains (e.g., biomedical + math + code), (b) selecting and organizing agents from a large pool of LLMs to form an effective multi-agent system remains a significant challenge. (c) The current Mixture-of -Agents (MoA) approach integrates all available agents, aggregates their responses, and forwards the combined output to the next layer. However, it lacks generalizability to larger agent pools (e.g., 10 or 100) and suffers from heavy intra-layer communication overhead. (d) In contrast, our proposed Graph-of Agents (GoA) addresses this challenge through a graph-based structure. In GoA, only a subset of relevant agents is selected to form the graph's nodes, with intra-layer communication facilitated via directed message passing—flowing from more relevant agents to less relevant ones. By leveraging this graph structure, GoA achieves greater scalability and enables more efficient communication between agents, resulting in a more powerful and adaptive framework.

nism, instead forwarding queries to all available agents, leading to excessive computational costs. This not only risks multi-agent system explosion but also introduces noise from irrelevant agents, highlighting the need for a more efficient and practical agent selection strategy.

How do they communicate? Once agents are selected, facilitating effective communication among them becomes a pivotal challenge. MoA adopts a many-to-one aggregation scheme, but this approach has notable drawbacks: it requires collecting responses from all available agents and treating them as a single chunk, which fails to capture fine-grained interactions—such as one-to-one communication between individual agent pairs. Moreover, since different agents have varying strengths and relevance depending on the query, treating all agent messages equally can hinder consensus. Instead, adaptively weighting responses based on their relevance is crucial for improving decision-making.

How to integrate? Finally, when making the final decision, MoA aggregates responses by concatenating tokens from all agents. However, this approach incurs a huge computational cost, with a complexity of $\mathcal{O}(LNd)$, where L is the number of communication layers, N is the number of agents, and d is the token length per agent. Given that token usage is directly tied to cost, this method becomes prohibitively expensive. Moreover, not all agents contribute equally valuable responses—some domain-specialized agents produce significantly higher-quality outputs than others. A scalable integration mechanism that prioritizes more reliable agents while reducing the impact of less relevant ones is essential for cost-efficient and effective decision-making.

⋆ **Our Approach.** To address these challenges, we propose a *graph-based multi-agent framework*, Graph-of-Agents (GoA), as a novel remedy to enhance multi-agent communication. GoAfundamentally rethinks multi-agent collaboration by modeling agents as nodes and their relevance-based relationships as edges, enabling structured message passing. This graph-based design allows for the selective activation of only relevant agents (as a subgraph of entire pool) while capturing inter-agent interactions through message passing and finalizing decisions via graph-based pooling. GoA follows a structured process, as shown in Figure 1 (d):

❶ **Which agents?** → ***Node Sampling***: GoA begins by selecting relevant agents based on available metadata (*e.g.*, domain, task) from model cards. This information is provided to a meta-LLM, simply a general-domain LLM, which identifies the most relevant agents given the query.

❷ **How do they communicate?** → ***Edge Sampling & Message Passing***: Once nodes (agents) are selected, we obtain their initial responses and ask each agent to rank others, capturing the significance of each agent's output. Based on these rankings, we construct directed edges in two perspectives: (i) Source-to-Target: Higher-ranked agents (*i.e.*, highly influential nodes with more relevant responses) propagate their information to lower-ranked agents, allowing them to refine their responses based on more confident initial answers. (ii) Target-to-Source: After lower-ranked agents update their responses, the refined information is passed back to the higher-ranked agents, enabling them to further adjust their outputs based on the improved responses from their neighborhood agents.

❸ **How to integrate?** → ***Graph Pooling***: With refined responses, GoA applies max or mean pooling, akin to graph pooling, to adaptively aggregate the outputs of multiple agents. GoA is extensively evaluated on diverse multi-domain benchmarks (MMLU, MMLU-Pro, GPQA) and domain-specific benchmarks (MATH, HumanEval, MedMCQA), demonstrating its generalizability and adaptability across a wide range of tasks. These results highlight the effectiveness of viewing multi-agent collaboration through the lens of graph structures.

It is important to note that, unlike traditional multi-agent learning method that require model finetuning or additional training, our proposed GoA framework, as like MoA, operates purely through the prompt interface. This design ensures compatibility with black-box LLM APIs while maintaining high adaptability across diverse domains during test-time inference.

Our contributions are three-fold:

- We identify key challenges in current multi-agent LLM systems: selecting which agents to sample, facilitating effective communication, and integrating responses efficiently.
- We formulate multi-agent collaboration as a graph-based framework and introduce GoA, which incorporates node sampling, edge sampling, message passing, and graph pooling. This enables not only the construction of a scalable multi-agent LLM ecosystem, but also enhances inter-agent communication.
- We demonstrate the effectiveness of GoA across diverse benchmarks, including MMLU, MMLU-Pro, GPQA, MATH, HumanEval, and MedMCQA. Notably, using only 3 agents, GoA outperforms recent multi-agent baselines that rely on pools of 6 agents, highlighting how graph-based structures improve both the scalability and effectiveness of multi-agent collaboration.

2 Related Work

LLM Reasoning and Test-Time Inference. Test-time reasoning with LLMs has seen rapid advances through prompt engineering techniques such as Chain-of-Thought (CoT) (Wei et al., 2022b), Tree-of-Thought (Yao et al., 2023), and Graph-of-Thought (Besta et al., 2024; Yao et al., 2024), which enable models to decompose complex problems into structured sub-tasks. While most prior work focuses on improving a single LLM's reasoning via internal prompting strategies (Zhou et al., 2023; Xu et al., 2024; Feng et al., 2024), a parallel direction has emerged around collaborative reasoning at inference time using multiple LLMs (Du et al., 2023a; Chan et al., 2023). In these multi-agent setups, multiple LLMs interact in test-time without additional finetuning, often aiming to boost factual accuracy or reasoning diversity. However, these approaches typically rely on simplistic communication protocols such as symmetric debate or sequential refinement, lacking structured mechanisms for message routing or adaptive collaboration. Our work builds on this foundation by proposing a graph-based test-time collaboration framework that formalizes agent interactions through directional message passing, enabling richer and more scalable multi-agent reasoning.

LLM Ensembles and Multi-Agent LLM Collaboration. Another approach to leveraging multiple LLMs is ensemble-based inference, where outputs from several models are aggregated or selected (Fang et al., 2024; Yuan et al., 2023). Recent works introduce router mechanisms (Wang et al., 2023c; Hari & Thomson, 2023; Wang et al., 2024b; Yue et al., 2025) to reduce computational cost by selectively querying a subset of models. However, many of these ensemble strategies treat LLMs as interchangeable units without modeling their relationships. Meanwhile, graph-based structures have begun to emerge to coordinate multi-agent systems, notably in frameworks like MacNet (Qian et al., 2024) and GPTSwarm (Zhuge et al., 2024), which model agents as nodes in static DAGs. Also DyLAN (Liu et al., 2024b) performs dynamic agent activation using an Agent Importance Score computed via forward–backward peer-rating propagation and coordinates agents through a temporal feed-forward communication network. Yet, these approaches often rely on predefined topologies or assume a single agent role-playing multiple personas, limiting flexibility and expressiveness, and sometimes even requiring additional optimization.. In contrast, our proposed framework, Graph-of-Agents (GoA), constructs dynamic graphs based on task relevance, enabling one-to-one communication across a diverse pool of specialized LLMs. Through node sampling, directional edge construction, and graph pooling, GoA supports efficient and adaptive collaboration—addressing the underexplored challenge of coordinating heterogeneous agents for domain-diverse reasoning at test time.

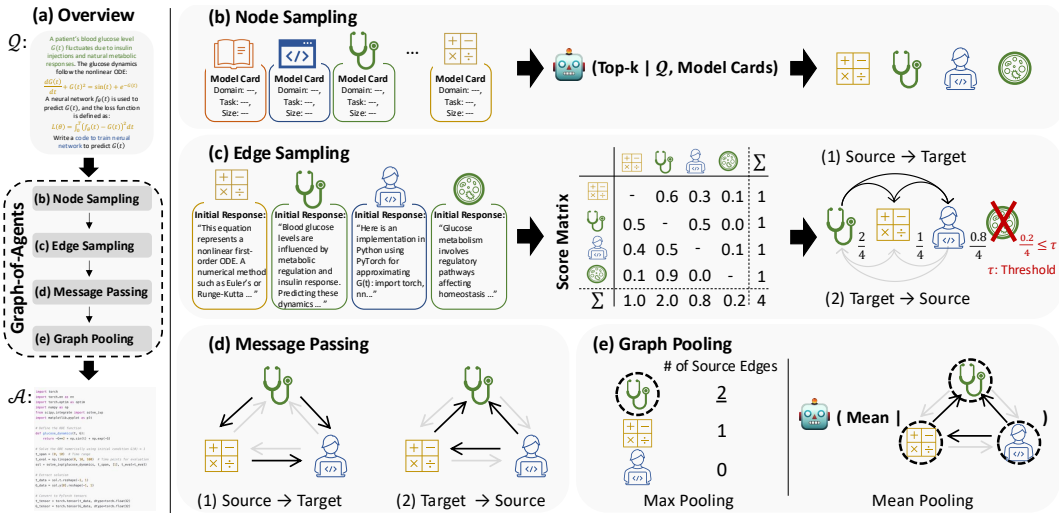


Figure 2: Overall pipeline of GoA. (a) **Overview**: Given a query ($\mathcal{Q}$) spanning diverse domains, GoA approaches multi-agent LLMs through the lens of a graph framework and produces an answer ($\mathcal{A}$). (b) **Node Sampling**: Each agent is mapped to a model card containing domain and task information. The Meta-LLM, a general-purpose LLM, takes $\mathcal{Q}$ and the model cards as input and selects the most relevant agents, forming an adaptive multi-agent framework. (c) **Edge Sampling**: After collecting initial responses from selected agents, each agent evaluates the relevance of others (excluding itself) to generate a normalized score matrix. Edges are established in Source-to-Target and Target-to-Source directions, while low-relevance nodes are pruned using a threshold τ=0.05. (d) **Message Passing**: GoA first passes messages from source to target nodes, allowing lower-ranked agents to refine responses, then reverses the flow to update source nodes. (e) **Graph Pooling**: With updated responses structured as a graph, GoA outputs the final prediction via max or mean pooling.

3 METHODOLOGY

3.1 PRELIMINARIES AND NOTATIONS

Multi-agent LLMs. We define a system of N LLM agents, where each agent $i \in 1, \ldots, N$ specializes in a particular domain or task. The agents collaboratively process a given query $\mathcal{Q}$ to generate an optimized response $\mathcal{A}$.

As a graph. We represent the multi-agent system as a directed graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = v_1, \ldots, v_N$ denotes the set of agent nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the directed edges. Once a subset of S relevant agents are selected from the multi-agent pool of size N, the adjacency matrix $\mathbf{A} \in \mathbb{R}^{S \times S}$ is constructed. The matrix is defined such that $\mathbf{A}_{ij} > 0$ if $(v_i, v_j) \in \mathcal{E}$ and $\mathbf{A}_{ij} = 0$ otherwise.

3.2 OUR APPORACH: GOA

In this section, we present GoA, a novel approach specifically designed to enhance multi-agent communication through a graph-based framework, as shown in Figure 2.

Overview. GoA begins with node sampling, where a meta-LLM (a general-purpose LLM) selects the most relevant agents from a given pool based on the query and a model card dictionary. Each model card summarizes information extracted from the Hugging Face README file, including the LLM's domain and specialized tasks (Sec. 3.2.1). After collecting initial responses from the selected agents, we rank each agent based on scores assigned by other agents. Agents are then sorted by these scores to define two node types: source nodes (highly relevant and influential) and target nodes (less influential). This establishes bidirectional relationships: source-to-target and target-to-source (Sec. 3.2.2). In the message-passing, we first propagate messages from source to target, allowing target nodes to refine their responses based on input from more relevant agents. We then reverse the flow—target to source—so that source nodes can further refine their outputs using the updated responses. This two-step process ensures that each agent's contribution is weighted by its relevance to the query (Sec. 3.2.3). Finally, we apply graph pooling: either max pooling based on the most connected source node or mean pooling guided by the meta-LLM to generate the final answer (Sec. 3.2.4).

3.2.1 NODE SAMPLING

Given a query $\mathcal{Q}$, our goal is to select a subset of agents most relevant to the task. To achieve this, we leverage publicly available model cards from Hugging Face (Jain, 2022), which contain useful metadata such as the dataset the LLM was trained on, its specialized domain, model size. Using this information, we summarize each agent's model card into three key categories: (1) The domain of the LLM, (2) The specialized task, and (3) The model size and special features. Once the summarized model card is obtained, we prompt the Meta-LLM[1] to determine which agents are most likely to generate the most effective response given the query. Formally, the selected subset of agents, $\mathcal{V}_s \subseteq \mathcal{V}$, is obtained as:

$$\mathcal{V}_s = \text{Meta-LLM}(\text{Top-}k|\mathcal{Q}, \text{Model Cards}), \tag{1}$$

where Top-k selects the k most relevant agents based on their alignment with the query and model card information. For example, as illustrated in Figure 2 (b), if the query pertains to biomedical, mathematics, and code-related domains, the selected agents would primarily belong to these specialized areas to maximize multi-agent synergy. This approach effectively filters out unnecessary agents (e.g., law-related models), preventing agent explosion (i.e., involving an unmanageable number of agents) while maintaining relevance agents for handling the query, answering "**Which agents?**".

3.2.2 EDGE SAMPLING

Once the subset of selected agents $\mathcal{V}_s$ is obtained, we prompt each agent to generate its initial response to the query $\mathcal{Q}$, forming a response set $\mathcal{R} = \{v_1(\mathcal{Q}), \ldots, v_S(\mathcal{Q})\}$, where $S = |\mathcal{V}_s|$. To model inter-agent relevance, we construct a score matrix in which each agent scores the responses of all other agents (excluding its own to reduce self-bias) based on alignment with $\mathcal{Q}$. These scores are normalized such that each agent distributes a total score of 1.0 across the remaining $S-1$ agents. We then compute a relevance score $\mathcal{S}_j$ for each agent j by summing the scores it receives from others:

$$\mathcal{S}_j = \sum_{\substack{i=1 \\ i \neq j}}^{S} \text{Score}_{i \to j}. \tag{2}$$

The relevance scores $\mathcal{S}$ are then used to rank agents and determine their communication roles (e.g., source vs. target). To avoid including weak or noisy responders—particularly in cases where model cards may lack detailed information—we introduce a threshold hyperparameter τ. Agents with $\mathcal{S}_j < \tau$ are pruned from the communication graph, ensuring scalability and better task-fit in the constructed structure.

Using the remaining high-relevance agents, we define a weighted directed adjacency matrix $\mathbf{A} \in \mathbb{R}^{S \times S}$ to govern message passing. Each entry $\mathbf{A}_{ji}$ represents how much influence agent i exerts on agent j when passing messages. It is computed by normalizing the total relevance scores of all neighbors of agent j:

$$\mathbf{A}_{ji} = \frac{\mathcal{S}_i}{\sum_{k \in \mathcal{N}_j} \mathcal{S}_k}, \quad \text{where } \mathcal{N}_j = \{i \mid (i \to j) \in E\}. \tag{3}$$

This scoring formulation ensures that more relevant agents have proportionally greater influence, and it enables fine-grained 1-to-1 communication tailored to task-specific needs. Figure 2(c) illustrates how the score matrix and pruning mechanism dynamically shape the communication graph.

3.2.3 MESSAGE PASSING

Now, given the edge information and weighted adjacency matrix, we proceed with message passing, a key advantage of the graph structure. To incorporate the significance of each agent, GoA performs message passing in two steps: **Source-to-Target** followed by **Target-to-Source**.

[1]Any general-purpose LLM. In our experiments, we used Qwen2.5-7B-Instruct (Team, 2024) from 7–8B agent pool.

Source-to-Target. For highly influential nodes, it is crucial to maintain their initial strength rather than being influenced by less significant or noisy nodes. Conversely, less significant nodes benefit from receiving messages (i.e., more relevant responses to the query) from stronger nodes. Thus, we first propagate information from source nodes (higher-ranked agents) to target nodes (lower-ranked agents), allowing the latter to refine their responses based on more confident initial answers:

$$\mathcal{R}_j^{'} = v_j\left(\|_{j=i+1}^{S} \mathbf{A}_{ij}\mathcal{R}_i^{\text{sorted}}\right), \quad \text{where} \quad i < j \leq S, \tag{4}$$

where $\mathcal{R}_j^{'}$ represents the updated response for target node j after receiving messages from source node i. Here, $v_j(\cdot)$ denotes the forward pass of LLM j, $\|$ represents the concatenation of neighboring responses, and $\mathcal{R}_i^{\text{sorted}}$ represents the sorted responses, ranked from highly relevant to less relevant based on the relevance scores $\mathcal{S}$ obtained in Equation 2. With these updated responses, we now proceed with the Target-to-Source step.

Target-to-Source. Since the previous step only updates target nodes, we also allow source nodes to refine their responses based on the improved outputs of their neighbors ($\mathcal{R}_j^{'}$), rather than the initial responses from target nodes. This enables source nodes to incorporate the consensus of their neighboring agents, indirectly influenced by the original source node, leading to further refinement:

$$\mathcal{R}_i^{''} = v_i\left(\|_{j=i+1}^{S} \mathbf{A}_{ji}\mathcal{R}_j^{'}\right), \quad \text{where} \quad i < j \leq S, \tag{5}$$

where $\mathcal{R}_i^{''}$ represents the refined response for source nodes. With both source and target nodes refining their responses collaboratively through the graph structure, we effectively address the question of "**How do they communicate?**". We now move on to the final step: response integration.

3.2.4 GRAPH POOLING

To address the question of "**How to integrate?**" while minimizing the computational cost associated with token stacking, we formalize response integration through graph pooling, a common approach in graph-based tasks that requires aggregating node representations into a single graph representation. Motivated by this, we propose two pooling strategies: ❶ *Max-Pooling*, which relies on the most influential node (i.e., the agent with the highest number of incoming edges, indicating a higher relevance score). ❷ *Mean-Pooling*, which balances contributions by considering responses from all selected agents but on a reduced scale, unlike MoA, which involves all available agents. Formally, this can be expressed as:

$$\mathcal{A} = \begin{cases} \mathcal{R}_{\text{max-source}}^{''} & \text{if max-pooling} \\ \text{Meta-LLM}(\text{Average}|\mathcal{R}^{''}) & \text{if mean-pooling,} \end{cases} \tag{6}$$

where $\mathcal{R}_{\text{max-source}}^{''}$ denotes the refined response of the agent with the highest number of source edges (i.e., the most relevant agent). In summary, max-pooling prioritizes the response of the most significant agent, while mean-pooling incorporates responses from all selected agents, requiring an additional forward pass through the Meta-LLM. We introduce these two variants as GoA_{max} and GoA_{mean}, which will be analyzed in the experiment sections. The averaging is performed in a weighted manner using the relevance scores assigned during edge sampling.

3.3 GOA GENERALIZES MOA

Lastly, we demonstrate that GoA generalizes the existing MoA framework. Specifically, as illustrated in Figure 1 (c), the message-passing and response-updating procedure of MoA (Wang et al., 2024a) can be formulated as:

$$\mathcal{R}_i^{'} = v_i(\|_{j=1}^{N} \mathcal{R}_j + \mathcal{Q}). \tag{7}$$

Comparing this to Equation 4, we establish the following:

Proposition 1 *Graph-of-Agents (GoA) reduces to MoA when the node sampling parameter k equals the total number of agents N, the adjacency matrix is fully connected with all edge weights set to 1, i.e., $\mathbf{A} \in \mathbb{R}^{N\times N} = 1$, and a self-loop is included with the initial query ($\mathcal{Q}$) at each layer, ultimately aggregated via mean pooling.*

Table 1: Benchmark performance across multi-domain and domain-specific benchmarks. *Single-agent baselines*: **General**: Qwen2.5-7B-Instruct (Team, 2024), **Code**: Qwen2.5-Coder-7B-Instruct (Hui et al., 2024), **Math**: Mathstral-7B-v0.1 (AI, 2024), **Biomedical**: Bio-Medical-Llama-3-8B (Con, 2024), **Finance**: finance-Llama3-8B (Cheng et al., 2024), **Legal**: Saul-7B-Instruct-v1 (Colombo et al., 2024). *Multi-agent baselines*: **Debate** (Du et al., 2023b), **Self-Consistency (SC)** (Wang et al., 2023a), **Refine** (Madaan et al., 2023), **ReConcile** (Chen et al., 2023), **MoA** (Wang et al., 2024a), and **Self-MoA** (Li et al., 2025). Our proposed framework, GoA **(Graph-of-Agents)**, despite using only *3 agents* (i.e., top-k=3), outperforms both multi-agent baselines with *6 agents* and single-agent models, demonstrating strong collaborative synergy and capability across both multi-domain and domain-specific tasks. All performance is measured using zero-shot CoT in test-time.

		Multi-Domain			Domain-Specific		
		MMLU	**MMLU-Pro**	**GPQA**	**MATH**	**Human Eval**	**MedMCQA**
Single-Agent	General	77.61	53.90	32.83	69.00	81.50	55.22
	Code	68.04	42.33	33.84	59.60	**85.37**	45.57
	Math	63.47	37.19	30.81	48.60	57.93	45.25
	Biomedical	46.60	27.90	25.25	18.80	20.73	47.00
	Finance	54.11	25.52	28.28	13.80	27.44	42.08
	Legal	55.86	27.57	30.30	12.40	36.59	41.50
Multi-Agent *(6 Agents)*	Debate	72.53	47.05	29.29	69.60	40.24	53.05
	SC	77.97	54.12	36.36	69.80	82.57	55.70
	Refine	77.40	54.71	38.92	71.60	80.49	54.94
	ReConcile	69.61	44.19	34.34	45.60	50.20	54.60
	MoA	75.71	53.33	32.83	65.80	76.22	54.94
	Self-MoA	78.14	54.19	33.84	68.20	79.27	55.56
Multi-Agent *(3 Agents)*	GoA_{Max}	**79.18**	**54.78**	39.98	69.83	84.67	**60.04**
	GoA_{Mean}	78.52	54.27	**40.54**	**73.12**	84.98	57.92

Thus, GoA serves as a more flexible and extensible generalization of multi-agent communication frameworks. By introducing structured message passing, adaptive agent selection, and weighted aggregation, GoA can scale to large agent pools while maintaining efficiency and robustness. As our work focuses on test-time inference, the prompts used in this study are provided in Appendix B.

4 Experiments

4.1 Implementation Details

Benchmarks. We evaluate performance on two multi-domain (MMLU (Hendrycks et al., 2020), MMLU-Pro (Wang et al., 2024c), GPQA (Rein et al., 2023)) and three domain-specific benchmarks (MATH (Lightman et al., 2023), HumanEval (Chen et al., 2021), MedMCQA (Pal et al., 2022)). For MMLU and MMLU-Pro, due to their large sizes, we used stratified sampling: 50 samples per category across 57 categories for MMLU, and 150 samples per category across 14 categories for MMLU-Pro. For the agent pool, we primarily leveraged six LLMs with 7–8B parameters, covering diverse domains such as General, Code, Math, Biomedical, Finance, and Legal. The specific LLMs are listed in Table 1, and details on the benchmarks and baselines are provided in Appendix C.

4.2 Main Results

Effectiveness. In Table 1, we present benchmark results across both multi-domain and domain-specific tasks, comparing single-agent baselines, multi-agent baselines with 6 agents, and our proposed GoA (3 agents). ❶ Among single-agent baselines, the general-purpose model achieves the highest average performance (61.15), but falls short of the best-performing multi-agent models. Specialized agents (e.g., Math, Biomedical, Finance) tend to perform well only in narrow domains and underperform in others, leading to lower overall averages. ❷ Multi-agent baselines with 6 agents—especially Refine (62.15) and Self-MoA (61.63)—demonstrate improved performance over all single-agent baselines, confirming the benefits of agent collaboration. However, their effectiveness varies across benchmarks, with no single method dominating all tasks. ❸ Our proposed GoA method outperforms all baselines across most metrics. GoA_{Max} achieves the highest aver-

age score, with top performance on MMLU (79.18), MMLU-Pro (54.78), and MedMCQA (60.04). GoA_{Mean} records the best scores on GPQA (40.54), MATH (73.12), and HumanEval (84.98), showing robust and consistent gains across both reasoning and domain-specific tasks. ❹ Notably, while other multi-agent methods rely on integrating all 6 agents, GoA achieves superior results using only 3, suggesting that selective and structured agent collaboration can be more effective than full integration. For detailed performance comparison plots for each category in MMLU and MMLU-Pro, please refer to Appendix D.

Efficiency. Another crucial perspective for test-time LLM collaboration is efficiency—that is, handling multiple LLMs in a scalable manner. Table 2 presents a comparison between MoA and GoA on the MMLU-Pro dataset in terms of accuracy, number of LLM calls, average token usage (input + output), and latency. Unlike MoA, which employs multiple rounds of proposers (i.e., all available agents) and a final aggregator, GoA reduces LLM usage and latency while improving accuracy. This improvement stems from its design philosophy—node and edge sampling followed by graph pooling—which selectively involves only relevant agents in the system, making it a more efficient and practical approach for multi-agent LLM collaboration. Appendix J provides additional results on GPQA and HumanEval benchmarks across more datasets.

Table 2: Efficiency analysis in MMLU-Pro.

	Acc.	Calls	Tokens (k)	Time (s)
MoA	53.33	19	56.05	240.26
GoA_{Max}	54.78	11	19.18	100.43
GoA_{Mean}	54.27	12	22.58	118.52

Scaling Up. To evaluate generalization to proprietary models, we tested GoA with `gpt-4o` on GPQA, MedMCQA (100 sampled), and HumanEval. As shown in Table 3, multi-agent setups consistently outperform the single-agent baseline. Notably, while DyLAN (Liu et al., 2024a) employs eight specialized agents (e.g., 'Python Assistant', 'Algorithm Developer'), our GoA with only three agents achieves higher performance, mirroring trends seen with open-source models. This advantage stems from our graph-based reasoning and relevance-aware message-passing mechanism (Section 4.3), which enables targeted and noise-resilient communication. These results highlight that well-designed communication strategies can be more effective than simply increasing the number of agents, underscoring both the effectiveness and scalability of GoA.

Table 3: Scaling up with `gpt-4o` model.

	GPQA	MedMCQA	HumanEval
`gpt-4o`	47.47	77.00	90.20
Debate (6 Agents)	53.03	80.00	85.98
SC (6 Agents)	54.27	81.00	92.07
Refine (6 Agents)	54.98	82.00	91.92
Reconcile (6 Agents)	53.03	75.00	91.46
MoA (6 Agents)	50.51	80.00	92.07
DyLAN (8 Agents)	**58.89**	81.00	92.07
GoA_{Max} (3 Agents)	55.05	82.00	93.29
GoA_{Max} (6 Agents)	56.57	**83.00**	**93.90**

4.3 Why Graph?

The key aspect of GoA is its graph-based framework for multi-agent LLM collaboration. In this section, we show how the **graph structure** and the **new message-passing** benefit LLM collaboration.

Graph-based Reasoning. Figure 3 presents a case study illustrating how a graph-based framework improves reasoning by comparing recent MoA with our proposed GoA when handling an anatomy-domain from the MMLU dataset. In MoA, all available agents are used, including those from unrelated domains such as math and code. These irrelevant agents introduce noise (e.g., 'Answer: 1'), which negatively affects the final prediction. In contrast, GoA avoids irrelevant agent usage by applying node sampling followed by edge sampling, thereby constructing a query-specific graph structure. This enables targeted message-passing among relevant agents, leading to more accurate discussions (e.g., 'Answer: 0') and final predictions. Overall, this comparison shows how a graph-based framework enhances reasoning in multi-agent scenarios by leveraging structured interactions and relevance-aware message passing.

Relevance-aware Message-Passing. We now delve into GoA's relevance-aware message-passing framework under a stricter fixed-node setting, targeting the HumanEval benchmark with three code-specific models. As shown in Table 4, the baseline models exhibit varying

Table 4: Effectiveness of Message-Passing.

	HumanEval
Qwen2.5-Coder-7B-Instruct	85.37
Seed-Coder-8B-Instruct	80.49
deepseek-coder-7b-instruct-v1.5	73.17
MoA - MoE-based (3 Agents)	85.37
Debate - Debate-based (3 Agents)	71.95
Reconcile - Confidence-based (3 Agents)	80.61
GoA_{Max} - Graph-based (3 Agents)	**85.98**

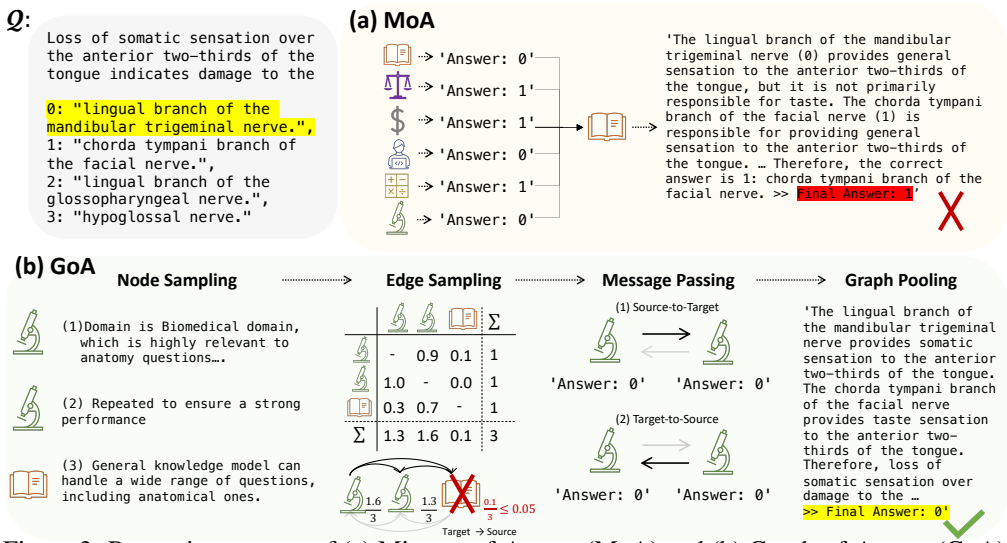


Figure 3: Reasoning process of (a) Mixture-of-Agents (MoA) and (b) Graph-of-Agents (GoA).

performance—Qwen2.5-Coder-7B-Instruct performs best, while the others fall behind. In the multi-agent setting, we compare our approach against existing frameworks: MoA (MoE-based) Wang et al. (2024a), Debate Du et al. (2023b), and Reconcile (confidence-based) Chen et al. (2023), with our GoA (graph-based) achieving the highest performance. This gain stems from its tailored message-passing mechanism, where edges are formed based on node relevance to the given question: information flows from highly relevant nodes (e.g., Qwen2.5-Coder-7B-Instruct) to less relevant ones (e.g., deepseek-coder-7b-instruct-v1.5), which update their internal state, followed by a feedback phase that produces an aggregated response. These results demonstrate that graph-based approaches—especially when combined with relevance-aware message passing—offer a promising paradigm for LLM collaboration.

4.4 Ablation Study

Table 5: Ablation study on modules, top-k, and τ.

	MMLU-Pro	GPQA
GoA (Top-k=3, τ=0.05)	**54.78**	**39.98**
Reverse Message Passing	52.18	34.93
w/o Target-to-Source	53.66	38.03
w/o Source-to-Target	52.21	36.12
w/o Scoring ($\mathbf{A}_{ij} = 1$)	52.91	37.34
Top-k=2	53.54	36.75
Top-k=5	54.65	39.13
τ=0.1	53.12	38.43
τ=0.2	52.78	37.12

Table 5 presents an ablation analysis on MMLU-Pro and GPQA. The top row shows the original GoA setting with the best performance. ❶ Interestingly, reversing the message-passing direction causes the largest performance drop (–2.60 on MMLU-Pro and –5.05 on GPQA), even worse than removing only one direction (Source-to-Target or Target-to-Source). This matches our expectation: when less-influential nodes become artificially dominant, they disrupt the intended flow and inject noise back into the original source nodes. This underscores a key design insight—preserving the correct bidirectional flow, (1) Source-to-Target and (2) Target-to-Source, is essential for GoA's effectiveness.. Removing Target-to-Source message passing leads to a noticeable drop (–1.12 MMLU-Pro, –1.95 GPQA), showing that feedback from target nodes is crucial. ❷ Removing Source-to-Target causes even larger degradation (–2.57, –3.86), highlighting the importance of initial information flow. Together, these results confirm the complementary role of bidirectional message passing. ❸ Disabling edge scoring ($A_{ij} = 1$) consistently reduces performance, validating the benefit of relevance-based weighting. ❹ Varying the number of agents shows $k = 2$ limits diversity while $k = 5$ introduces slight degradation (likely requiring tailored τ), with $k = 3$ remaining scalable and performant. ❺ Adjusting the edge threshold τ shows that overly sparse graphs ($\tau = 0.1, 0.2$) harm performance, while $\tau = 0.05$ balances focus and connectivity. Overall, these results demonstrate that bidirectional message passing, adaptive edge scoring, and selective agent activation all contribute to GoA's effectiveness.

5 Conclusion

In this study, we introduce GoA, a graph-based framework that re-designs multi-agent communication to address key challenges: selecting relevant agents, enabling effective communication, and integrating responses efficiently. Extensive experiments demonstrate consistent gains across diverse benchmarks. We believe GoA can advance the development of large-scale collective intelligence in the emerging multi-agent LLM era.

Ethics Statement

We propose a graph-based multi-agent LLM collaboration framework, GoA. However, the LLMs used in GoA may still exhibit inherent biases and undesirable traits from pretraining. Consequently, its outputs carry similar risks of misuse as other test-time methods.

Reproducibility Statement

We provide our code at the anonymous link: https://anonymous.4open.science/r/goa-F23C. All experiments were conducted on an A6000 GPU. The datasets used are publicly available.

References

Bio-medical: A high-performance biomedical language model. https://huggingface.co/ContactDoctor/Bio-Medical-Llama-3-8B, 2024.

Mistral AI. MathΣtral. https://mistral.ai/news/mathstral/, July 2024.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2374-3468. doi: 10.1609/aaai.v38i16.29720.

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, August 2023.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *arxiv preprint*, 2309.07864, 2023. URL https://arxiv.org/abs/2309.13007.

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.

Daixuan Cheng, Yuxian Gu, Shaohan Huang, Junyu Bi, Minlie Huang, and Furu Wei. Instruction pre-training: Language models are supervised multitask learners. *arXiv preprint arXiv:2406.14491*, 2024.

Pierre Colombo, Telmo Pessoa Pires, Malik Boudiaf, Dominic Culver, Rui Melo, Caio Corro, Andre F. T. Martins, Fabrizio Esposito, Vera Lúcia Raposo, Sofia Morgado, and Michael Desa. Saullm-7b: A pioneering large language model for law, 2024.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *CoRR*, abs/2305.14325, 2023a. doi: 10.48550/arXiv.2305.14325. URL https://doi.org/10.48550/arXiv.2305.14325.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023b.

Jiangnan Fang, Cheng-Tse Liu, Jieun Kim, Yash Bhedaru, Ethan Liu, Nikhil Singh, Nedim Lipka, Puneet Mathur, Nesreen K. Ahmed, Franck Dernoncourt, Ryan A. Rossi, and Hanieh Deilamsalehy. Multi-LLM Text Summarization, December 2024.

Li Feng, Ryan Yen, Yuzhe You, Mingming Fan, Jian Zhao, and Zhicong Lu. CoPrompt: Supporting Prompt Sharing and Referring in Collaborative Natural Language Programming. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, pp. 1–21, New York, NY, USA, May 2024. Association for Computing Machinery. ISBN 9798400703300. doi: 10.1145/3613904.3642212.

Surya Narayanan Hari and Matt Thomson. Tryage: Real-time, intelligent Routing of User Prompts to Large Language Models, August 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*, October 2020.

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

Shashank Mohan Jain. Hugging face. In *Introduction to transformers for NLP: With the hugging face library and models to solve problems*, pp. 51–67. Springer, 2022.

Wenzhe Li, Yong Lin, Mengzhou Xia, and Chi Jin. Rethinking mixture-of-agents: Is mixing different large language models beneficial?, 2025. URL https://arxiv.org/abs/2502.00674.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration, 2024a. URL https://arxiv.org/abs/2310.02170.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A Dynamic LLM-Powered Agent Network for Task-Oriented Agent Collaboration. In *First Conference on Language Modeling*, August 2024b.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pp. 248–260. PMLR, 2022.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large-language-model-based multi-agent collaboration. *arXiv preprint arXiv:2406.07155*, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024a.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-Agents Enhances Large Language Model Capabilities, June 2024b.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023a. URL https://arxiv.org/abs/2203.11171.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yiding Wang, Kai Chen, Haisheng Tan, and Kun Guo. Tabi: An Efficient Multi-Level Inference System for Large Language Models. In *Proceedings of the Eighteenth European Conference on Computer Systems*, EuroSys '23, pp. 233–248, New York, NY, USA, May 2023c. Association for Computing Machinery. ISBN 978-1-4503-9487-1. doi: 10.1145/3552326.3587438.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2024c.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022a. URL https://arxiv.org/abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, 35:24824–24837, December 2022b.

Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, Jian-Guang Lou, and Shuai Ma. Re-Reading Improves Reasoning in Large Language Models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15549–15575, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.871.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *Advances in Neural Information Processing Systems*, 36:11809–11822, December 2023.

Yao Yao, Zuchao Li, and Hai Zhao. GoT: Effective Graph-of-Thought Reasoning in Language Models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2901–2921, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.183.

Xingdi Yuan, Tong Wang, Yen-Hsiang Wang, Emery Fine, Rania Abdelghani, Hélène Sauzéon, and Pierre-Yves Oudeyer. Selecting Better Samples from Pre-trained LLMs: A Case Study on Question Generation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 12952–12965, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.820.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. Masrouter: Learning to route llms for multi-agent systems, 2025. URL https://arxiv.org/abs/2502.11133.

Denny Zhou, Nathanael Scharli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. LEAST-TO-MOST PROMPTING ENABLES COMPLEX REASONING IN LARGE LANGUAGE MODELS. 2023.

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*, 2024.

APPENDIX

A THE USE OF LARGE LANGUAGE MODELS (LLMS)

We use ChatGPT[2] exclusively for grammar checking and text refinement. Its role was limited to polishing author-written text and did not involve research ideation.

B PROMPTS

As our proposed GoA framework operates during test-time inference, the detailed prompts for each stage are provided below:

- **Model Card Extraction (Example: Qwen2.5-Coder-7B):**

Model Card Extraction - System Prompt

You are an expert in analyzing and summarizing AI model documentation.

[2]https://chatgpt.com/

Model Card Extraction - User Prompt

You are given the README file of a language model:

```
language:
- en
base_model:
- Qwen/Qwen2.5-Coder-7B
pipeline_tag: text-generation
library_name: transformers
tags:
- code
- codeqwen
- chat
- qwen
- qwen-coder
---

# Introduction
Qwen2.5-Coder is the latest series of Code-Specific ...
- code generation, code reasoning and code fixing.
- ...
```

Please extract and summarize the model's key characteristics clearly and concisely in the following structured format:

1. **Domain**: The primary domain or application area the model is designed for (e.g., general-purpose, biomedical, finance, coding, math, etc.).
2. **Task Specialization**: Describe the task types the model is designed for or excels at. Be as specific as possible, including the domain context of each task (e.g., biomedical question answering, clinical decision support, financial sentiment classification, code generation). Do not include performance metrics, benchmark names, or evaluation results.
3. **Parameter Size**: The number of parameters in the model (approximate if not explicitly stated).
4. **Special Features**: Any distinguishing aspects such as fine-tuning datasets (if applicable).

Your summary will later be used to compare multiple models for selection purposes. Return your answer in bullet-point format, using the exact field names shown above. Keep it concise but specific enough for model comparison.

Answer:

- **Node Sampling**:

Node Sampling - System Prompt

You are an AI model selection expert. Your task is to select the most relevant AI models to answer a given question based on their domain, specialized capabilities, and overall performance.

Selection Criteria:

- **Domain Match (Primary Factor)**: Prefer models that are trained in the relevant domain for the question.
- **Task Specialization**: If the question requires a specific skill (e.g., reasoning, code generation, biomedical text processing), prioritize models that explicitly specialize in that area.
- **Generalist Models Consideration**: If a generalist model is known to perform well in the given domain or task, include at least one such model in the selection.
- **Size vs. Performance Balance**: Do NOT rely solely on model size when selecting between generalist and specialized models. If a generalist model is significantly larger (e.g., 13B vs. 7B), prefer the larger model. Otherwise, choose based on task performance and known effectiveness.

If a model is highly relevant, you may select it multiple times. If a question does not specify a domain, general models should be preferred.

Node Sampling - Use Prompt

Given the question: `messages[0]` and the following model descriptions: `model_descriptions`, select the top `top_k` models that best fit the question.

Selection Rules:

- Return a comma-separated list of indices (e.g., `"0,1,1,3"`).
- The list must contain exactly `top_k` values.
- You may repeat an index if the model is highly relevant.
- Only use numbers in the range [0, `max_index`]. Do not include explanations.

Example Selections:

- If the question is about biomedical research: `"0,5,5"` (favoring biomedical models, but keeping a generalist model if available).
- If the question is a reasoning-based general question: `"0,0,1,2"` (favoring generalist models with some specialized).

```
example_dict = {
    1: "0",
    2: "0,3",
    3: "0,1,5",
    4: "0,0,4,5",
    5: "0,1,2,3,5",
    6: "0,0,2,3,4,5"
}
```

Answer Format:
Strictly follow the format below. Do not add any explanations or extra text. Example: `example_dict[max_index+1]`

```
Answer:
```

- **Edge Sampling**:

Edge Sampling - System Prompt

You are an expert at evaluating AI model responses. You must rank and score the responses **relatively**, assigning a numerical score to each such that the total sum of all scores is exactly 1.0.

Edge Sampling - User Prompt

Given the question: `messages[0]`, and other models' responses: `other_responses`

Evaluate the following responses from models: `other_models`
Assign a score to each response based on:

- **Correctness** (most important)
- **Coherence**
- **Relevance**

Distribute a total of **1.0** point across all responses. Better responses should receive higher scores. Do not provide explanations.

Response Format:
Please assign scores in the same order as the responses shown above, from top to bottom. The order of models is: `other_models`

Make sure:

- The list has exactly len(`other_models`) scores
- The sum of the scores is exactly 1.0
- Do not include any extra text

Example Dictionary:

```
example_dict = {
    1: [1.0],
    2: [0.7, 0.3],
    3: [0.1, 0.1, 0.8],
    4: [0.3, 0.4, 0.1, 0.2],
    5: [0.2, 0.4, 0.1, 0.3, 0.0]
}

example_str = ", ".join(f"'
{other_models[i]}': {example_dict[len(other_models)][i]}"
for i in range(len(other_models)))

Example: example_str



Answer:
```

- **Message-Passing (1) *Source-to-Target***:

Source-to-Target - System Prompt

You are refining your response by incorporating insights from other models. Each model's contribution is weighted by its relevance score. Use their ideas carefully to improve your answer, while keeping the strengths and clarity of your original response.

Source-to-Target - User Prompt

Question: `messages[0]`

Your initial response: `initial_responses[target]`

The following responses were provided by other models. They are considered more relevant and may offer valuable insights to improve your answer.

`descriptions`

While reviewing these responses, carefully consider the accuracy and reliability of the information they contain, as some parts may be incorrect or influenced by bias. Update your response by integrating any useful information from these answers, while preserving your own original strengths.

- **Message-Passing (2) *Target-to-Source***:

Target-to-Source - System Prompt

You are finalizing your response after seeing how other models refined their answers based on your initial response. Use their improvements to further refine your answer and make it as complete and accurate as possible.

Target-to-Source - User Prompt

Question: `messages[0]`

Your initial response: `initial_responses[source]`

The following models updated their responses based on your original answer. Their updates incorporate insights from you while maintaining their own strengths:

`descriptions`

Now, review how your answer influenced others. While reviewing these responses, carefully consider the accuracy and reliability of the information they contain, as some parts may be incorrect or influenced by bias. Based on these updated responses, write your final, upgraded response, incorporating any new ideas or refinements you find valuable.

- **Graph-Pooling**:

Graph Pooling - System Prompt

You are synthesizing multiple model responses into one. Prioritize relevance but consider all inputs. Your final answer should be accurate, coherent, and well-structured.

Graph Pooling - User Prompt

Given the question: `messages[0]`,

the following responses have been generated by different models:

`input_responses`

Please synthesize a final, well-reasoned answer that is cohesive, accurate, and best addresses the question by effectively integrating these responses. While reviewing these responses, carefully consider the accuracy and reliability of the information they contain, as some parts may be incorrect or influenced by bias.

C Details on Experimental Settings

Benchmarks. We evaluate our framework on the following benchmarks:

- **MMLU** (Hendrycks et al., 2020): A comprehensive multi-domain benchmark with 57 diverse subjects (which can be grouped into 4 broader categories), designed to test general

knowledge and reasoning across humanities, STEM, and other domains. In this study, we employed stratified sampling with 50 samples per subject to ensure balanced representation.

- **MMLU-Pro** (Wang et al., 2024c): A professional-level extension of MMLU comprising 14 categories focused on advanced reasoning tasks in professional fields. In this study, we employed stratified sampling with 150 samples per category to ensure balanced representation.
- **GPQA** (Rein et al., 2023): A challenging benchmark targeting graduate-level question answering with an emphasis on deep understanding and domain-specific reasoning.
- **MATH** (Lightman et al., 2023): A benchmark comprising high school and competition-level mathematics problems, designed to test symbolic reasoning and multi-step problem solving. We used a subest of 500 problems.
- **HumanEval** (Chen et al., 2021): A code generation benchmark consisting of programming problems that evaluate functional correctness of generated code.
- **MedMCQA** (Pal et al., 2022): A multiple-choice medical question answering benchmark based on real medical entrance exams, assessing biomedical and clinical knowledge.

Multi-Agent Baselines. We compare our method against the following multi-agent LLM baselines:

- **Debate** (Du et al., 2023b): A multi-agent collaboration approach where agents engage in structured debates to converge on a final answer.
- **Self-Consistency (SC)** (Wang et al., 2023b): A single agent generates multiple reasoning paths, and the final answer is chosen by majority voting across these diverse outputs.
- **Refine** (Madaan et al., 2023): A refinement-based collaboration framework where agents iteratively improve one another's responses.
- **ReConcile** (Chen et al., 2023): A confidence-based collaboration strategy where agent responses are weighted and reconciled based on self-assessed confidence.
- **MoA** (Wang et al., 2024a): A Mixture-of-Agents method where different experts contribute to decision-making in a proposer-and-aggregator fashion.
- **Self-MoA** (Li et al., 2025): A variant of MoA composed of multiple instances of a single model, where the single model is the top-performing model for each task.

D Detailed Comparison on each Category

Figure 4 and Figure 5 show the fine-grained performance across each category in the MMLU and MMLU-Pro benchmarks, respectively. We compare domain-specific agents and a strong baseline, Self-MoA (utilizing 6 agents), with our proposed method, GoA (utilizing 3 agents). We observe that in most cases, the general-domain agent (G) performs strongly across diverse domains. However, it lacks generalizability across all categories and, like Self-MoA (Li et al., 2025), underperforms compared to GoA. This is due to GoA's ability to adapt to diverse domains through relevant node selection (e.g., the law agent in law-specific question (Figure 5)) and a tailored message-passing framework that effectively captures domain-relevant information in response to each question.

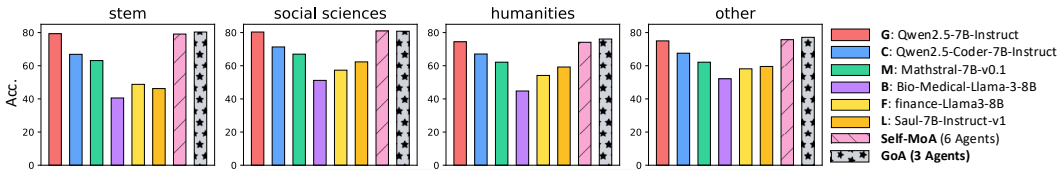


Figure 4: MMLU benchmark results across 4 categories. The last two color bars represent Self-MoA and GoA, respectively. Self-MoA integrates 6 agents, while GoA utilizes 3 agents.

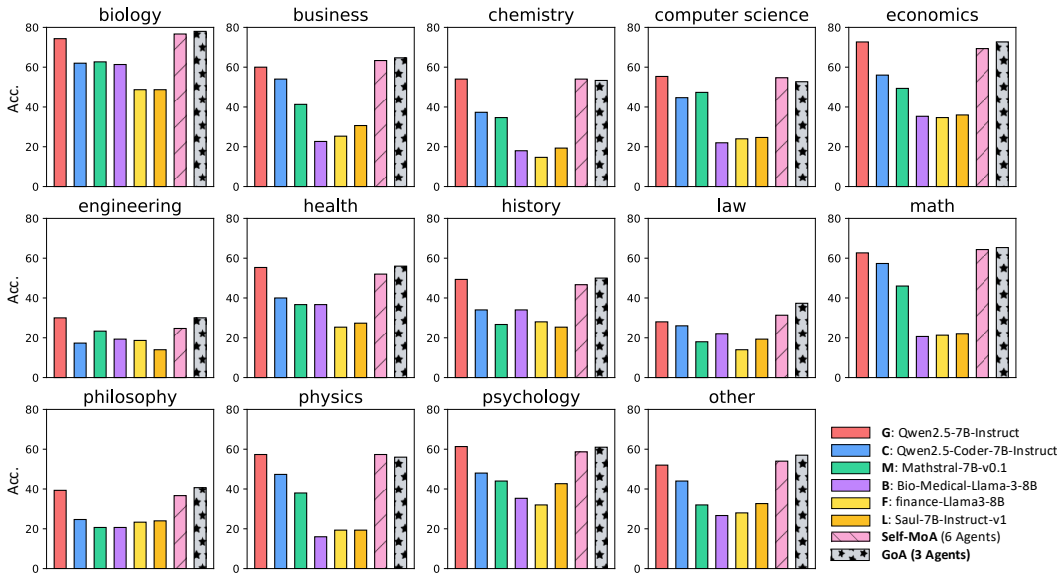


Figure 5: MMLU-Pro benchmark results across 14 categories. The last two color bars represent Self-MoA and GoA, respectively. Self-MoA integrates 6 agents, while GoA utilizes 3 agents.

E Limitation and Future Work

While GoA demonstrates strong performance in multi-agent collaboration, several limitations remain. First, the current node sampling mechanism primarily relies on publicly available model card information to select relevant agents. However, model card descriptions may be incomplete, inconsistent, or lack crucial details about a model's strengths and weaknesses in specific domains. As a result, the selection process may not always optimally align with the query requirements. Future work could explore more adaptive selection mechanisms, such as meta-learning-based optimization or reinforcement learning techniques, to dynamically refine agent selection based on historical performance and contextual relevance.

Moreover, GoA primarily focuses on text-based reasoning tasks, leveraging structured communication among language models. However, many real-world applications require multimodal reasoning, where agents process and integrate textual, visual, and numerical information simultaneously. Extending GoA to multimodal domains by incorporating vision-language models, structured numerical reasoning, or even cross-modal knowledge fusion would be a promising direction.

Lastly, while the current benchmarks used in our study (e.g., MMLU, GPQA, MedMCQA) are not interactive environments, our framing is consistent with the Mixture-of-Agents (MoA) line of work—both conceptually (multiple LLMs collaborating as an agent system, illustrated in Figure 1 of the main paper) and in evaluation setup, which similarly relies on question–answering tasks (e.g., AlpacaEval 2.0, MT-Bench, FLASK) to study test-time agent collaboration. That said, in future work, we plan to explore more interactive, environment-grounded agentic benchmarks, such as WebAgent and GUI navigation. Incorporating GoA into such settings—where agents interact with tools, execute actions, or access the web—would require adapting the agent pool and communication protocols, and is indeed a promising direction for future work. In particular, GoA's graph-based construction could help integrate heterogeneous tool responses into a coherent final answer, somewhat analogous to GPTSwarm's application of Self-Consistency.

F Extension to Larger Agent Pool

In addition to the 6 agents covering (general, code, math, biomedical, finance, legal) domains, we have added 4 additional agents from general (`Meta-Llama-3.1-8B-Instruct`), code (`Seed-Coder-8B-Instruct`), math (`deepseek-math-7b-instruct`), and biomedical

(JSL-MedLlama-3-8B-v2.0) domains, resulting in an agent pool of 10 diverse LLMs. Given this extended agent pool in the GPQA benchmark, the performance is shown below:

Agent Pool={G,C,M,B,F,L}+{G2,C2,M2,B2}	GPQA
Refine (10 agents)	35.86
MoA (10 agents)	31.82
Self-MoA (10 agents)	35.86
ReConcile (10 agents)	37.89
GoA_max (3 agents)	38.23
GoA_mean (3 agents)	39.39

Interestingly, even after increasing the number of agents from 6 to 10, the baseline methods still fail to outperform GoA_max and GoA_mean, despite GoA using only 3 agents. This suggests that in a heterogeneous agent-pool setting, naively scaling up the total number of agents does not necessarily yield performance gains; instead, selectively leveraging only the most relevant agents can more effectively and efficiently achieve superior reasoning performance.

G Robustness of Node Sampling

To evaluate the robustness of the node-sampling module—which is based on model card information from open-source sources—we conducted an experiment where we initially fix the sampled nodes to the relevant code agent (Qwen2.5-Coder-7B-Instruct) for solving the HumanEval benchmark, and then gradually replace the relevant agent (Code) with an irrelevant, noisy agent (Bio). This setting simulates scenarios where node sampling mistakenly selects suboptimal agents due to misleading or incomplete model card information. The agents used in this experiment are:

- Code: `Qwen2.5-Coder-7B-Instruct`
- Bio: `ContactDoctor/Bio-Medical-Llama-3-8B`

The results are as follows:

HumanEval Agents	MoA	GoA
Code, Code, Code	84.93	**86.59**
Code, Code, Bio	82.54 (Δ=-2.39)	**84.15** (Δ=-2.44)
Code, Bio, Bio	79.88 (Δ=-5.05)	**82.93** (Δ=-3.66)

Here, we quantify Δ as the performance gap between the fully relevant agent pool (top row) and the noisy settings (each subsequent row). We observe that when more noisy agents are introduced, MoA suffers larger performance degradation. This is because MoA concatenates the responses from all participating agents uniformly, without weighting or filtering, causing noise from irrelevant (Bio) agents to propagate through subsequent layers and accumulate.

GoA, on the other hand, passes messages in a relevance-aware manner (most-relevant $\rightarrow$ least-relevant) and introduces a threshold that filters out noisy agents from actively participating in the graph. This prevents irrelevant signals from contaminating the message passing process, resulting in significantly smaller performance drops than MoA and consistent outperformance in all settings.

H Comparison with MacNet

We additionally ran experiments on the MacNet (Qian et al., 2024) baseline, another network-based multi-agent collaboration framework utilizing predefined topologies such as chain and tree, to evaluate how existing graph-based multi-agent methods compare to our proposed approach, GoA. The results are shown below:

Interestingly, across diverse fixed topologies—chain, star, tree, and random—we observe that MacNet consistently underperforms GoA (and in some cases even underperforms MoA and Self-MoA).

Agent Pool={G,C,M,B,F,L}	MMLU-Pro	GPQA
Refine	54.71	38.92
ReConcile	44.19	34.34
MoA	53.33	32.83
Self-MoA	54.19	33.84
MacNet – Chain	50.67	36.36
MacNet – Star	50.52	35.35
MacNet – Tree	50.52	35.35
MacNet – Random	50.71	32.83
GoA_Max	**54.78**	**39.98**
GoA_Mean	54.27	**40.55**

This indicates that in multi-agent graph construction, incorporating task relevance and leveraging relevance-weighted message passing (such as GoA's bidirectional propagation) is crucial for boosting reasoning performance, rather than relying on static graph structures.

I Discussion on Adding More Agents

Referring to Table 1 (top-k = 3) and GoA's ablation study in Table 5 (top-k = 2,5), we have already varied the number of agents. We additionally include the $k = 6$ case below:

Agent Pool={G,C,M,B,F,L}	MMLU-Pro	GPQA
top-$k = 2$	53.54	36.75
top-$k = 3$ (GoA)	**54.78**	**39.98**
top-$k = 5$	54.65	39.13
top-$k = 6$	54.27	38.45

As shown above—and as the reviewer suggested—adding more agents does not necessarily lead to performance gains, especially in a heterogeneous agent-pool spanning diverse domains. Our goal is to create an environment where agents interact synergistically, yet as seen in Table 1, domain-specific agents alone are often individually weak (e.g., B, F, L agents score 27.90, 25.52, and 27.57 on MMLU-Pro), likely due to domain mismatch, limited coverage, or insufficient instruction-tuning. Therefore, selecting a subset of relevant agents and emphasizing more task-relevant, influential agents becomes crucial—an ability that existing approaches lack, and that GoA uniquely enables through relevance-guided sampling and graph-based communication. Consequently, top-$k = 3$ yields better performance than naively increasing the number of agents, since including weaker or domain-irrelevant agents introduces noise (given their inherently weak single-agent performance) rather than useful signal. This aligns with DyLAN (Liu et al., 2024b), which similarly observes that a smaller set of well-chosen agents can outperform a larger unfiltered group.

However, we would also like to highlight Table 3 in the main paper, where we evaluated a more controlled scaling scenario—namely, a homogeneous agent setting—by fixing the agent pool to a single LLM (gpt-4o) and simply varying the number of agent copies. The results are shown below:

Agent Pool={GPT-4o}	GPQA	MedMCQA	HumanEval
`gpt-4o`	47.47	77.00	90.20
GoA (3 Agents)	55.05	82.00	93.29
GoA (6 Agents)	**56.57**	**83.00**	**93.90**

In this controlled setting where all agents have uniform capability, GoA with 6 agents consistently outperforms GoA with 3 agents across all benchmarks. This demonstrates that when agent quality is consistent (single-LLM replication), increasing agent count can indeed improve performance.

J MORE RESULTS ON EFFICIENCY ANALYSIS

We further conducted efficiency analysis on two additional benchmarks—GPQA (Table 6) and HumanEval (Table 7) verify that our improvements are not dataset-specific but hold consistently across reasoning and program synthesis settings. This suggests that selective collaboration among relevant agents—realized through node/edge sampling and graph pooling—not only preserves core reasoning quality but also enables scalable multi-agent pipeline.

Table 6: Efficiency comparison on GPQA.

	Acc.	**Calls**	**Tokens (k)**	**Time (s)**
MoA	32.83	19	56.87	245.65
GoA_{max}	39.98	11	17.32	88.15
GoA_{mean}	40.54	12	20.59	105.72

Table 7: Efficiency comparison on HumanEval.

	Acc.	**Calls**	**Tokens (k)**	**Time (s)**
MoA	76.22	19	29.44	122.52
GoA_{max}	84.67	11	10.21	59.04
GoA_{mean}	84.98	12	12.32	70.01