

# Topological Structure Learning Should Be A Research Priority for LLM-Based Multi-Agent Systems

Jiayi Yang<sup>1\*</sup>, Mengqi Zhang<sup>2\*</sup>, Yiqiao Jin<sup>3\*</sup>, Hao Chen<sup>4</sup>, Qingsong Wen<sup>5</sup>, Lu Lin<sup>1</sup>, Yi He<sup>2</sup>,  
Weijie Xu<sup>6</sup>, James Evans<sup>7</sup>, Jindong Wang<sup>2†</sup>

<sup>1</sup>The Pennsylvania State University <sup>2</sup>William & Mary <sup>3</sup>Georgia Institute of Technology

<sup>4</sup>AMD <sup>5</sup>Squirrel AI <sup>6</sup>Amazon <sup>7</sup>University of Chicago

## ABSTRACT

Large Language Model-based Multi-Agent Systems (MASs) have emerged as a powerful paradigm for tackling complex tasks through collaborative intelligence. Nevertheless, the question of how agents should be structurally organized for optimal cooperation remains largely unexplored. In this position paper, we aim to gently redirect the focus of the MAS research community toward this critical dimension: *develop topology-aware MASs for specific tasks*. Specifically, the system consists of three core components—agents, communication links, and communication patterns—that collectively shape its coordination performance and efficiency. To this end, we introduce a systematic, three-stage framework: agent selection, structure profiling, and topology synthesis. Each stage would trigger new research opportunities in areas such as language models, reinforcement learning, graph learning, and generative modeling; together, they could unleash the full potential of MASs in complicated real-world applications. Then, we discuss the potential challenges and opportunities in the evaluation of multiple systems. We hope our perspective and framework can offer critical new insights in the era of agentic AI.

## 1 Introduction

Large Language Model (LLM) agents [1–3] have demonstrated strong capabilities across diverse tasks, such as code generation [4–6], mathematical reasoning [7, 8], and social simulation [9, 10]. Building on the impressive capabilities of single-agents in general tasks, LLM-based Multi-Agent Systems (MASs) [11] have emerged as a powerful paradigm that coordinate multiple specialized agents to collaboratively solve more complex tasks. These systems assign distinct roles—such as planners, retrievers, and reasoners—to distinct agents, enabling human-like collaboration through a division of labor. Following the analogy of optimizing a human organization, an **agentic system** can be seen as a *dynamically evolving ensemble* that seeks a *moving equilibrium*: whenever new agents join the system or existing ones depart, the ensemble reconfigures itself to maximize its long-term utility subject to budget and latency constraints.

To enable inter-agent communication, coordination, and role specialization, MAS architectures must support agents that are not only individually competent but also capable of collaborating through well-orchestrated interaction patterns. Despite the promising progress made in the field, the question of how agents should be organized to support effective and efficient cooperation remains largely underexplored [12]. This reveals an urgent need to rethink the multi-agent design: **What topological structures of LLM-based MASs are best to accomplish an unseen, complex task?**

*Why topology matters?* In MASs, effective collaboration depends not only on the capabilities of individual agents but also on how they are organized and coordinated. Typically, as task complexity increases, larger groups of agents are required, leading to a significant rise in communication overhead. As revealed by recent studies, naïve scaling without adaptive structural design often results in redundant communication and suboptimal coordination [12–14].

\*These authors contributed equally to this research.

†Corresponding to: Jindong Wang <jwang80@wm.edu>

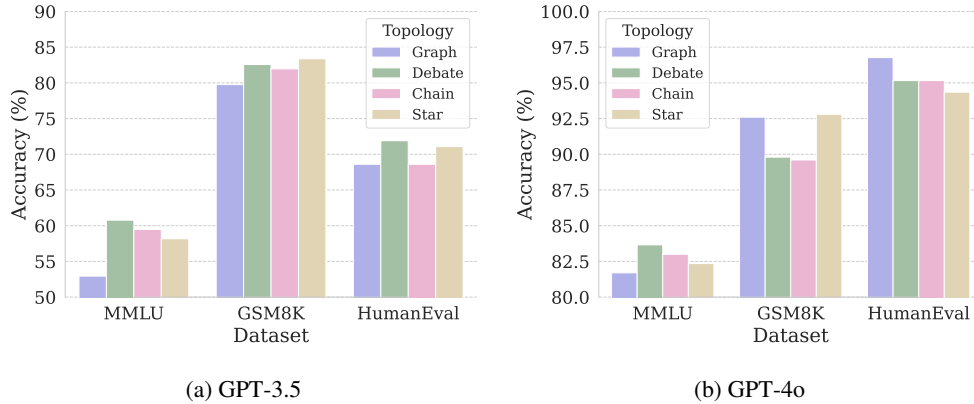


Figure 1: Comparison of topological structures across datasets. Results show that various tasks may require a specific topological structure.

Figure 1 shows that task performance varies by up to 10% if using different topologies. Furthermore, the choice of communication structure carries critical safety implications, whereby poorly designed topology may lead to misinformation propagation, biased reasoning, or the generation of harmful content by agents [15].

*What are the key challenges?* First, MASs are highly task-dependent. Reasonable topological structures often depend on the characteristics of a specific task; yet, these characteristics are often implicit, difficult to model and to generalize, thereby hindering the development of universal topology designs. Some prior works attempt to explore dynamic topologies, but remain in the early stages and still lack the ability to adapt topological structures based on queries or task demands [16, 17]. Second, the design space of topological structures is inherently combinatorial, involving both agent selection and inter-agent connectivity, which grows exponentially with increasing agents. This makes efficient structure optimization computationally prohibitive as the system scales. Structure pruning techniques like edge pruning [13] and agent pruning [14, 17] are often applied in a static and one-off manner, lacking the ability to model the dynamic interdependencies and interactions among agents. Third, unlike in single-agent settings, the effectiveness of a given topological structure in MASs cannot be evaluated in isolation; rather, its performance must be evaluated through complete system execution within a specific task context, which entails substantial runtime overhead.

*What should be learned and how?* We summarize three core components in MASs: agents, communication links, and the overall topology that determines how agents are connected and interact. The candidate subset of agents should be capable and complementary agents for a given task, while communication links should be optimized to eliminate redundant message passing and minimize communication overhead. The overall topology should align with the coordination pattern of tasks to enable efficient information flow. In this position paper, we advocate a structure-aware paradigm that treats topological design as a key element of LLM-based MASs. We propose a unified framework (Figure 2) that decomposes topological structure optimization into three stages: agent selection, structure profiling, and topology synthesis, each corresponding to a core aspect of the topological structure. From this perspective, we reveal critical limitations of existing approaches and outline future directions toward more adaptive and efficient multi-agent architectures.

*Potentially improved applications.* Current agent-based LLM applications can potentially benefit from a topology-aware structure. For instance, in customer support automation, dynamic agent topologies can reduce response latency, minimize redundant information routing, and improve relevance in multi-turn dialogues for support systems involving billing, troubleshooting, and multilingual hand-offs. For collaborative code generation, task-specific topologies can support better coordination among agents with distinct roles (planner, coder, tester), which adapt as tasks evolve. Moreover, in scientific discovery, research assistants powered by LLMs can benefit from topologies that optimize how domain-specific agents (*e.g.*, literature retrievers, experimental planners, hypothesis evaluators) interact, particularly when solving open-ended scientific queries. Given the dynamic and flexible nature of interaction topology, more applications will emerge and benefit.

The organization of this paper is as follows: we first introduce the basics of LLM-based MASs in Section 2. We then articulate the proposed research framework for topology optimization with open problems and research directions in Section 3. Subsequently, Section 4 presents challenges and opportunities in evaluation, followed by discussions on multiple systems in Section 5. Finally, Section 6 concludes the paper. Note that due to space limitations, more details of our proposed research directions are in the appendix.

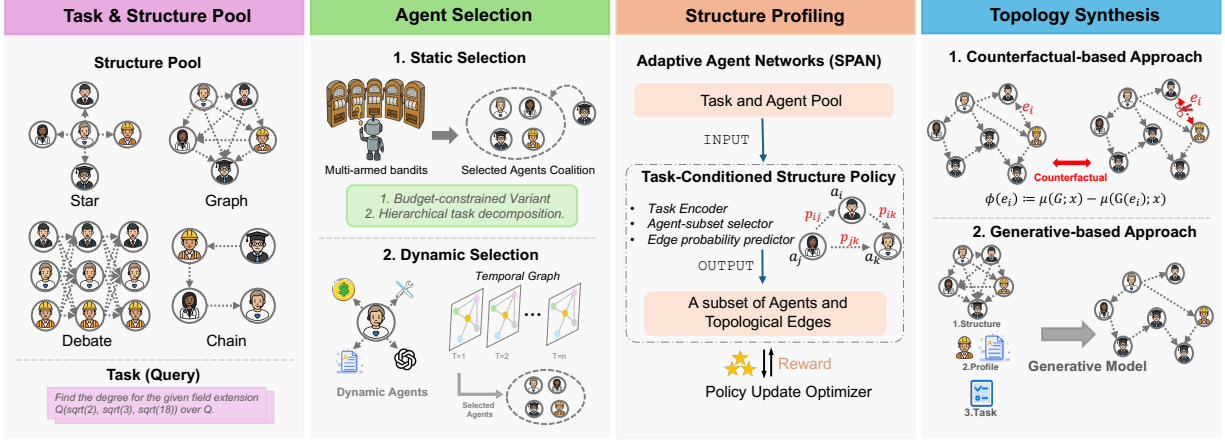


Figure 2: The proposed framework: agent selection, structure profiling, and topology synthesis.

## 2 LLM-based Multi-Agent Systems

While LLM-based single-agent systems have obtained notable success on a wide range of tasks, complicated problems can exceed the capabilities of a single model. This motivates the development of MASs, where multiple LLM-based agents collaborate through natural language communication to constructively divide roles, specialize functions, and jointly solve tasks. To support such collaboration, we define an LLM-based MAS as a set of autonomous agents instantiated from LLMs, denoted as  $\hat{\mathcal{A}} = \{a_1, \dots, a_i\} \subseteq \mathcal{A}$ . Each agent  $a_i \in \mathcal{A}$  is assigned a specific role, such as planner, coder, or verifier, and interacts with other agents to collectively solve tasks. Appendix A shows our notations.

In practice, MASs adopt various structural topologies, such as chains, stars, trees, or densely connected graphs, each encoding different reasoning workflows and communication costs. For instance, chain structures enable step-by-step refinement (e.g., MetaGPT [18]), while tree structures support hierarchical planning (e.g., AutoGen [1]). Star topologies simplify control via centralized agents (e.g., CAMEL [2]), and fully connected graphs maximize information sharing (e.g., GPTswarm [19]). Each structure presents distinct trade-offs in coordination efficiency, scalability, and reasoning capability, and is inherently better suited to different tasks depending on its specific communication patterns and collaborative requirements. Formally, we model the collaboration structure as a directed graph  $\mathcal{G} = (\mathcal{A}, \mathcal{E})$ , where  $\mathcal{E} \subseteq \mathcal{A} \times \mathcal{A}$  represents the communication links. At each round  $t$ , agent  $a_i$  receives messages from its in-neighbors and generates an updated response  $R_i^{(t)} = a_i(P_{\text{sys}}, P_{\text{user}}^{(t)})$ , where  $P_{\text{sys}}, P_{\text{user}}$  denote the system prompt and user prompt at round  $t$  respectively.  $P_{\text{user}}^{(t)} = P_{\text{user}}^{(0)} \cup \mathcal{O}_i^{(t-1)}$ , where  $\mathcal{O}_i^{(t-1)}$  denotes the aggregated messages from neighbors of agent  $a_i$ .

## 3 Holistic Research Framework

In this section, we propose a framework to learn the optimal topological structure for MAS. Our framework consists of three sequential stages: agent selection, structure profiling, and topology synthesis, as illustrated in Figure 2. Each stage has the potential to trigger new open problems and research directions. Specifically, ❶ Given a new task, a pool of candidate agents with diverse capabilities is instantiated and made available. The first step is to select a subset of agents  $\hat{\mathcal{A}} \subseteq \mathcal{A}$  best suited to collaborate on the task, based on factors such as skill specialization, role diversity, and past performance. This determines “who” participates in the communication topology. ❷ For each task, the system identifies a *macro structure* as a communication pattern, which captures abstract interaction forms, such as chains, stars, or trees. These macro structures reflect coarse-grained collaboration strategies and serve as an initial prototype for the following topology synthesis. ❸ Based on steps ❶ and ❷, the final stage synthesizes a concrete and directed communication graph, which we reference as the *micro structure*. It specifies the exact flow of information among the selected agents under the given macro structure, and determines which agents exchange messages and in what direction.

**End-to-end Optimization.** While we argue that the entire problem should be decomposed into three stages due to its challenging nature, it does not mean that end-to-end optimization is impossible. These three stages are tightly coupled and can be optimized jointly, adopting an “Expectation-Maximization” paradigm [20]. For instance, the agent selection algorithms might not be optimal, which can then be refined by the following procedures. On the other hand, the feedback from structure profiling could also have some impact on the *micro-level* structure of the selected agents. The flexibility of an end-to-end algorithm enables the system to dynamically refine all components in a closed-loop manner, adapting to changing task requirements and coordination dynamics.

Due to the high cost and complex applications for MASSs, however, it is easier to start with a “divide-and-conquer” manner to decompose it into the above three stages. In the following, we articulate how to learn each stage separately. The decomposition allows us to provide deeper insights into each aspect, while the overall framework remains flexible enough to support joint optimization across stages.

### 3.1 Agent Selection

Modern AI platforms (e.g., TogetherAI and HuggingFace) offer thousands of LLMs that can participate in MAS, varying in modality, capability, latency, parameter size, and cost. We summarize two major problems in selecting agents to ensemble an effective and efficient system.

**Open problem 1.** *Selecting proper agents is a cost-utility optimization problem that must balance exploration (testing alternatives) and exploitation (using the best-known option).*

Users must decide *which* agents to choose as candidates: high-performance but expensive SaaS LLMs versus smaller local checkpoints, and *how many* queries to assign to each agent. Under-exploration risks missing superior models, while over-exploration and exhaustive trials are extremely expensive.

**Open problem 2.** *Existing agent selection schemes are fundamentally limited by three major challenges: Single-Agent Focus, Structural Myopia, and Lack of Scalability.*

Users must balance *when* generalists and specialists fit into a particular task pipeline.

**Position.** We advocate a *structure-aware, lifecycle-aware* approach to agent selection that optimizes for multi-objective reward (task success, cost, latency, safety), which should remain adaptive to catalog churn and evolving task distributions.

We propose two viable solutions, *Static Selection* and *Dynamic Selection*, depending on the specific scenarios. When task distribution, communication structure, evaluation metrics, costs, and the LLM catalog remain unchanged over the deployment horizon, e.g., nightly batch summarization with a locked-in set of SaaS APIs, where the utility of an agent can be seen as stationary, the problem reduces to estimating and selecting an optimal subset based on the interaction structure, and thus *Static Selection* is preferred. Formally, assuming that each agent in the pool is associated with a cost  $C(a_i) \in \mathbb{R}_{\geq 0}$  (e.g., API credits, GPU runtime), for a downstream task  $x$  under macro structure  $M$ , we can define a utility function  $\mathcal{U} : (x, M, \hat{\mathcal{A}}) \mapsto [0, 1]$  to evaluate the collective performance of a subset  $\hat{\mathcal{A}} \subseteq \mathcal{A}$  on task  $x$ . This utility can reflect metrics such as answer accuracy, semantic relevance, or user engagement (e.g., click-through rate). The goal of static selection to find  $k$  agents to balance performance-cost trade-offs:

$$\mathcal{A}^* = \operatorname{argmax}_{\hat{\mathcal{A}} \subseteq \mathcal{A}, |\hat{\mathcal{A}}|=k} \left[ \mathbb{E}_{x \sim \mathcal{D}} [\mathcal{U}(x, M, \hat{\mathcal{A}})] - \sum_{a_i \in \hat{\mathcal{A}}} \gamma(a_i) C(a_i) \right], \quad (1)$$

where  $\gamma_i \in \mathbb{R}_{\geq 0}$  is a per-agent cost multiplier that emphasizes economy ( $\gamma_i \uparrow$ ) or accuracy ( $\gamma_i \downarrow$ ).

**Research direction 1.** *Incorporate topology-aware perspectives to frame agent selection as a structured reasoning problem, enabling efficient routing, selective activation, and dynamic composition.*

**Research direction 2.** *Integrate topology-aware perspectives to transform selection into a structured reasoning problem, which enables more efficient routing, selective activation, and dynamic composition.*

**Research direction 3.** *The multiplier  $\gamma$  in Equation 1 can be treated as a stochastic, context-dependent signal, enabling adaptive selection policies that dynamically balance performance and economy as conditions drift.*

Multi-armed bandits (MAB) and their combinatorial variants offer a principled framework to efficiently explore promising agents and exploit high-utility subsets. Unlike reinforcement-learning (RL) training loops that continuously update a policy’s parameters, MAB treats each agent as an *immutable* stochastic arm, where one can query the arm (i.e., a model), observe a reward, and pay its cost, but cannot directly alter its internal parameters. We provide a specific implementation in Appendix C.1.

Instead, in evolving environments, agent capabilities, task requirements, and agent interactions (e.g., critique-revise chains) change over time, which favors **Dynamic Selection**. The *marginal value* of an agent is contingent on 1) its own characteristics (capability, latency, cost), 2) the temporal context (current quotas, tool availability), and 3) which other agents are currently active. Specifically, the overarching objective for dynamic selection is to learn a set function  $\mathcal{U}(\hat{\mathcal{A}}, x) \in \mathbb{R}$  that estimates the **utility** of invoking a subset of agents  $\hat{\mathcal{A}} \subseteq \mathcal{A}$  on an incoming task  $x$ , given the total budget  $B$ :

$$\max_{\hat{\mathcal{A}} \subseteq \mathcal{A}} \mathcal{U}(\hat{\mathcal{A}}, x), \quad C(\hat{\mathcal{A}}) \leq B. \quad (2)$$

**Research direction 4.** *Develop comprehensive task banks that provide fine-grained coverage of task types, difficulty levels, and required agent capabilities, enabling the learning of rich, transferable graph representations for effective agent selection.*

**Research direction 5.** *Developing principled initialization strategies for node embeddings to enable effective graph representation learning and designing readout mechanisms to select agent subsets based on the learned embeddings.*

In Appendix C.2, we propose several options for the construction of task banks and fundamental features for the initialization of the embedding. This dynamic and inter-dependent setting can be naturally modeled by temporal graph neural networks (TGNNs) [21, 22]. Each agent is a node, and each time-varying edge denotes an interaction (e.g., routing a sub-query or forwarding a partial answer) that arrives with a timestamp. TGNNs maintain a latent state for every node that is updated only when events involving that node occur, making them ideal for sparse, asynchronous communication patterns that dominate large-scale MAS.

We model agent interactions as a temporal graph  $\mathcal{G}$ , represented by a sequence of time-indexed graph snapshots  $\mathcal{G} = \{\mathcal{G}_t \mid t = 1, \dots, \mathcal{T}\}$ . At training time, a task  $x$  is sampled from  $\mathcal{X}$ . The agent selection algorithm in response selects a candidate subset  $\hat{\mathcal{A}}$  (i.e., agents that) for  $x$  and constructs  $\mathcal{G}$ . At each time  $t$ , an active snapshot  $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$  specifies the *participating* agents (i.e., nodes)  $\mathcal{V}_t = \{v_1, v_2, \dots, v_{n^t}\} = \hat{\mathcal{A}}^t \subseteq \hat{\mathcal{A}}$ , along with their communication pathways  $\mathcal{E}_t$ , where an edge  $e_{ij}^t \in \mathcal{E}_t$  indicates an interaction between agent  $i$  and  $j$ . Each edge induces a message  $\mathbf{m}_{ij}^t$ , which is aggregated by node  $v_i$  from its neighborhood  $\mathcal{N}_i$  to update its embedding  $\mathbf{v}_i^t$ :

$$\mathbf{m}_{ij}^t = \text{Message}(\mathbf{v}_i^{t-1}, \mathbf{v}_j^{t-1}, \Delta t, \mathbf{e}_{ij}^t), \quad (3)$$

$$\mathbf{z}_i^t = \text{Aggregate}(\{\mathbf{m}_{ij}^t, j \in \mathcal{N}_i\}), \quad \mathbf{v}_i^t = \text{Update}(\mathbf{z}_i^t, \mathbf{v}_i^{t-1}), \quad (4)$$

where any node embedding  $\mathbf{v}_i^t$  is updated only if agent  $i$  participates in an interaction (edge) event, ensuring computational efficiency under sparse and asynchronous communication patterns.

At the final timestamp  $\mathcal{T}$ , a permutation-invariant read-out module maps the node embeddings  $\mathbf{v}_i^{\mathcal{T}}$  of  $\hat{\mathcal{A}}$  and the task embedding  $\mathbf{x}$  to a utility estimate:

$$\mathcal{U}(\hat{\mathcal{A}}, x) = \text{Readout}\left(\sum_{a_i \in \hat{\mathcal{A}}} [\mathbf{v}_i^{\mathcal{T}} \parallel \mathbf{x}]\right), \quad (5)$$

where  $\parallel$  is concatenation. The subset  $\hat{\mathcal{A}}$  is scored via automatic or LLM-based evaluation as a function of its utility. Both the TGNN and read-out are trained jointly using a pairwise ranking loss that promotes higher scores for subsets yielding greater utility. The model learns to predict both the subset utility and the contribution of each node’s response.

### 3.2 Structure Profiling

Structure profiling seeks to determine the *macro-level* pattern of how agents should be connected to effectively coordinate and solve tasks. However, fixed structure, such as chain, star, or fully connected graph, often fails to generalize across diverse tasks. The optimal *macro-level* structure pattern is typically task-specific and implicit, depending on factors like reasoning complexity, agent roles, and task demands. For instance, sequential workflows (e.g., multi-step reasoning) align well with chains, while decentralized exploration may favor tree structures. One pressing open problem is to identify the optimal *macro-level* communication pattern among agents in MASs that maximizes task performance while minimizing unnecessary communication overhead.

**Position.** We argue that *adaptive topology policies* that generate a bespoke graph for each query  $x$  are now essential.

This section studies how selected agents are connected, communicate, and divide tasks with each other to maximize task reward while controlling communication cost. Formally, a structure is a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  over the agent pool  $\mathcal{A}$ , where an edge  $e_{ij} \in \mathcal{E}$  indicates agent  $a_i$ ’s output is forwarded as input context to agent  $a_j$ , namely  $a_i \rightarrow a_j$ . Designing an optimal agent interaction structure is non-trivial. Manually crafting a fixed communication or workflow pattern, such as a chain-of-thought style relay between agents, may work for some tasks but fails to generalize across diverse problems [23]. The appropriate structure is often task-dependent, e.g., a simple question might only require a single agent, whereas a complex query might benefit from a divide-and-conquer approach with multiple specialists. This dependency creates a need for multi-agent structure profiling, i.e., determining the best interaction architecture for a given task.

We define our objective by the utility function  $\mathcal{U}$  defined in Section 3.1, which measures the quality of result (and possibly efficiency) when using structure  $\mathcal{G}$  on task  $x$ . This could be, for instance, the task reward or accuracy achieved by the agents minus a penalty for communication cost (number of messages or tokens exchanged). The goal of structure profiling is to find the optimal agent network  $\mathcal{G}$  for each task. Formally, for a given query  $x$  we seek:

$$\mathcal{G}^*(x) = \arg \max_{\mathcal{G} \in \mathcal{S}} \mathcal{U}(\mathcal{G}; x). \quad (6)$$

**Open problem 3.** *Automatically profiling and selecting an effective structure for agents rather than relying on ad-hoc human design is thus a critical open problem.*

Pre-defined structures give useful inductive biases but lack flexibility. Meta-agent planners such as AOP [24] and ReMA [25] add task awareness, yet still rely on manually crafted rule sets. Graph-generation approaches, such as G-Designer [26] and CommFormer [27]), demonstrate that fully learned topologies can beat fixed ones, but they target narrow domains or small agent pools. We will propose SPAN, a task-conditioned and scalable network.

**Research direction 6.** *Model the macro-structure determination problem as a probabilistic graph.*

Existing work shows that the same agents succeed or fail relying on their connection structure. A fully-connected graph outperforms a chain on open-ended research questions [28], whereas a lightweight DAG scales better than regular patterns when hundreds of agents cooperate [12]. Fixed, hand-designed patterns break down as tasks diversify [23]. Therefore, we propose our policy SPAN (Structure Profiling Adaptive Network). The SPAN involves a task encoder, an agent-subset selector, and an edge probability predictor. An encoder will map the raw task description, *e.g.*, the user prompt plus optional metadata to a continuous latent vector. It may be instantiated as a frozen foundation model or a shallow adapter network on top of an LLM prompt embedding. Its sole role is to provide a compact task representation that conditions every subsequent decision. Given task representation and possible learned embeddings from Section 3.1 for candidate agents, the selector may output the Bernoulli probabilities to filter an agent subset by a deterministic threshold. Finally, the key idea is a probabilistic adjacency matrix for edge probability prediction:

$$P(x) \in [0, 1]^{N \times N}, \quad P_{ij}(x) = \Pr(a_i \rightarrow a_j \mid x). \quad (7)$$

Given a new task, the policy scores every possible communication link. Thresholding or sampling  $P(x)$  instantiates a discrete directed graph, which may be sparse or irregular, always tailored to the task. The probabilities in  $P(x)$  factorize decisions, keeping search linear in the number of potential edges rather than exponential in whole graphs. This abstraction (1) unifies prior work that searches end-to-end workflows [29], evolves graphs [23], or routes queries dynamically [16]; and (2) exposes continuous parameters that can be optimized by gradient or RL methods. More SPAN details and related work in Appendix D.

**Open problem 4.** *How can a society of agent self-edit its topology while a dialogue is still in flight, without creating instability or runaway costs?*

While we can select a suitable macro-structure, the agents involved in is actually dynamically changing, *i.e.*, the macro-structure could also be changing due to newly added or deleted agent nodes. In this case, there should be new research problems for macro-structure adaptation.

**Research direction 7.** *It is possible to design local edge policies that treat each message as evidence for edge build or break, and meta agents monitor utility gradients and trigger Markov Chain Monte Carlo (MCMC) moves, *i.e.*, add, drop, swap, subject to acyclicity constraints.*

In addition, it is hard to compare structural performance and innovations without a public benchmark. So, it is valuable to curate an open suite that pairs each task with a groundtruth or at least oracle-verified near-optimal topology, plus cost models for token, latency, and energy usage, as well as provide diagnostic leader boards that break out performance by macro-structure class.

### 3.3 Topological Structure Synthesis

While agent selection and structure profiling identify the macro-level pattern of interactions, synthesizing a micro-level communication topology is a pressing problem to further enhance the communication efficiency and task performance. Specifically, agent selection determines *who* participates in the collaboration, and structure profiling identifies the *macro-level pattern* of interaction. Effectively executing the task requires synthesizing a fine-grained communication topology that governs the actual flow of information between agents. This *micro-level* structure plays a critical role in determining not only which agents should communicate but also the edge directions, weights, and potential hierarchy (*e.g.*, tree depth or breadth). These decisions directly influence how efficiently information propagates throughout the system, ultimately shaping task performance [14], communication cost [14, 13], and the robustness of the multi-agent collaboration [15].

**Open problem 5.** *Fully connecting every pair of agents leads to redundancy and high communication costs: How do we prune unnecessary links while preserving task performance?*

Recent efforts have explored topological structure optimization in MASs through topology pruning-based methods, which aim to reduce redundancy and improve efficiency by selectively removing communication links from the topological structure. AgentPrune [13] learns a sparse spatial-temporal communication graph by optimizing differentiable

edge masks with policy gradients and performs one-shot pruning to remove redundant message-passing edges while preserving overall task performance. Similarly, G-Designer [26] adopts a reinforcement learning-based framework to learn task-specific sparse communication graphs by selecting critical communication links. AgentDropout [14], in contrast, adaptively drops low-utility intra- and inter-round edges to compress communication graphs and reduce token usage during inference. GPTSwarm [19] regards agents as nodes in a computational graph and optimizes inter-agent communication links, enabling end-to-end learning of sparse collaboration topologies tailored to task requirements.

**Open problem 6.** *Different agent roles should occupy specific positions within the topological structure: How do we design the micro-structure while considering the specific role of each agent to maximize performance?*

While many existing approaches focus on the communication connectivity among agents, they often ignore the problem of the different, specialized roles of agents that should occupy distinct and meaningful positions within the topology. For example, a manager should be in the central position for overall coordination, while a verifier may serve best at the end of the pipeline.

**Position.** We advocate that we should further optimize the *macro-level* topological patterns in Sec. 3.2 and provide a *micro-level* topological structure in MASs.

**Formulation.** The objective of topology synthesis is to synthesize an optimized directed graph  $\mathcal{G} = (\hat{\mathcal{A}}, \mathcal{E})$  that concretely realizes the macro-structure  $M$  over the selected agents  $\hat{\mathcal{A}}$ . Let  $\mathcal{G}_M(\hat{\mathcal{A}})$  denote the space of valid topologies consistent with macro-structure  $M$ . For each candidate graph  $\mathcal{G} \in \mathcal{G}_M(\hat{\mathcal{A}})$ , we define  $\mathcal{R}(\mathcal{T}, \mathcal{G})$  as the expected task reward and  $C(\mathcal{G})$  as the total communication cost. We then solve:

$$\mathcal{G}^* = \arg \max_{\mathcal{G} \in \mathcal{G}_M(\hat{\mathcal{A}})} \mathcal{R}(\mathcal{T}, \mathcal{G}) - \lambda \cdot C(\mathcal{G}). \quad (8)$$

**Research direction 8.** *Measuring the fine-grained, task-specific influence of individual communication links is essential for optimizing micro-level topological structure of MASs.*

A central challenge in optimizing *micro-level* topologies within MASs lies in understanding the specific contribution of each communication link to task performance. Unlike coarse-grained structural assumptions, such as fully connected or fixed topologies, fine-grained assessment allows the system to tailor communication pathways to the demands of each task. One promising approach is counterfactual reasoning, where the utility of a given link is estimated by observing the performance drop caused by its removal. Identifying the essential links or redundant communication links forms pruning strategies that retain only high-utility connections, thereby reducing communication overhead without compromising performance. To further mitigate the high computational cost of evaluating the effects of each communication link, approximation techniques need to be used for scalability. For instance, sampling-based methods estimate the marginal effects of a link across multiple sampled subgraphs, while gradient-based methods calculate importance by measuring the sensitivity of task utility to changes in link strength. Finally, communication links often lead to combined causal effects, and their value may not be independent. Group-level perturbation methods evaluate the collective contribution of link clusters, capturing synergistic effects that individual-level estimation may miss. Appendix E.1 provides further details on potential implementations of this research direction.

**Research direction 9.** *Exploring generative models to synthesize task-specific communication topologies offers a flexible pathway to automate and adapt multi-agent coordination structures.*

One promising direction is to leverage pre-trained language models to generate topologies directly from task descriptions. Given a prompt with task descriptions, agent-role assignments, and interaction patterns, the model could output the *micro-level* topological structure design by natural language conditioned on intent. This enables rapid adaptation to novel tasks and removes the expensive costs for measuring the effects of each communication link. In addition, Graph generative models, such as graph diffusion [30], conditioned on agent properties, such as roles or functional descriptions, can generate communication topologies that reflect specialization and interdependence. This perspective treats topology construction as a learned mapping from agent features to interaction graphs, enabling compositional and role-aware generation.

## 4 Challenges and Opportunities in Evaluation

Topological structure learning not only triggers new research directions in methodology and algorithms, but also opens opportunities in evaluation [31, 32]. Current evaluation protocols and systems do not sufficiently satisfy the scenarios in this case. Here, we propose several research directions.

**Developing Topology-Aware Benchmarks.** There have been plenty of efforts in LLM and agent evaluation [31, 32], primarily focusing on general language understanding, coding, and reasoning tasks. However, more should be done to

design topology-aware benchmarks to evaluate MASs in extreme, diverse, and comprehensive scenarios. For instance, how to design a benchmark more suitable for agent selection? How to evaluate the performance if agent numbers are scaling? What data should be collected as the golden standard for validation, given the complex nature of multi-agent systems. Finally, how do we analyze the error propagation in different stages of the topology optimization? We see many open problems and challenges in benchmark design.

**Investigating Real Efficacy.** While LLMs embedded in multi-agent systems often achieve impressive scores on established benchmarks [33, 34], these metrics may not accurately represent their true capabilities due to potential overfitting and data contamination issues, where models exploit static benchmarks for training rather than demonstrating genuine understanding. It has become critical to determine whether LLMs’ results stem from authentic understanding or mere memorization of training data. Recent research has highlighted that LLM agents can be easily misled by prompts containing incorrect information [35], making them vulnerable to hallucinations and misinformation.

**Scaling to Real-world Scenarios.** Most benchmark environments test only small-to-medium agent populations, far removed from the scale of real-world systems such as autonomous vehicle fleets or distributed sensor networks. Dynamically Scaling evaluations to hundreds or thousands of agents introduces novel challenges—bandwidth constraints, delayed credit assignment, and intensified non-stationarity—that current setups often ignore. There is a growing need for large-scale, standardized, and dynamic evaluation platforms [36–38] with reproducible configurations, versioned simulators, and public baselines to facilitate rigorous and comparable research at scale.

**Bridging the Simulation-to-Reality Gap.** While high-fidelity simulators offer controlled testbeds, they often overlook critical aspects of real-world deployments, such as asynchronous communications, sensor drift, and unmodeled link failures. Evaluation should integrate hardware-in-the-loop testing, field experiments, or shadow deployments to assess performance degradation under operational constraints. Such hybrid evaluation strategies are essential to surface failures that emerge only outside the laboratory and to drive algorithms toward greater robustness in unstructured environments.

**Performance Attribution.** Understanding why a particular topology performs well is non-trivial. Structural attribution—quantifying the contribution of individual agents or edges—is often confounded by downstream interactions [39]. While counterfactual utility estimation (removing or perturbing agents/edges) is promising, it introduces high computational overhead. Future work should explore group-level attribution techniques that scale with agent count.

## 5 Discussion: Research in Multiple Systems

While the major content of this paper explores research opportunities for a single system (i.e., a centralized multi-agent architecture), we now discuss emerging research challenges and opportunities in the context of multiple systems—that is, collaborative multi-agent systems distributed across organizations, platforms, or regions. As real-world deployments increasingly involve interconnected services and agent platforms, designing cross-system topological intelligence becomes essential.

**Privacy-Preserving Training and Deployment.** The growing demand for privacy in LLM systems makes it infeasible to directly share raw data across users, institutions, or geographic boundaries. For example, a personal assistant might interface with separate systems—weather, music, healthcare, finance—each bound by different data governance rules. In such settings, how can we coordinate multiple agent systems while preserving data locality? Future research could explore privacy-aware topological learning, where only encoded summaries or message embeddings are shared. Concepts from federated learning, differential privacy, and secure multi-party computation could be integrated to enable collaboration without information leakage.

**Efficient Distributed Optimization.** In cross-regional or cross-organizational MAS deployments, agents reside in physically distributed infrastructures. This raises a critical need for latency-aware, topology-optimized communication across system boundaries. How can we dynamically learn which systems should communicate, when, and through which intermediate agents? Opportunities include learning task-conditional, sparse communication bridges, developing hierarchical routing strategies, and combining graph-based scheduling with asynchronous distributed optimization. Moreover, bandwidth-aware structure synthesis and token-efficient inter-system protocols could emerge as a new class of research problems.

**Heterogeneous Knowledge Integration.** Different systems may specialize in distinct modalities or domains: a hospital system may rely on radiology reports and lab results; a travel planner on temporal and geospatial data; a classroom agent on curriculum structures and student models. These systems exhibit heterogeneous knowledge structures, tool APIs, and communication preferences. The challenge is to design meta-topologies that allow dynamic integration of such capabilities. Future work may explore cross-system role mapping, interface standardization for agent schemas, and structure-aware knowledge alignment techniques. Topology learning in this setting could also facilitate zero-shot system-to-system generalization.



**System-Level Alignment and Conflict Resolution.** As multiple agent systems interact, they may exhibit divergent objectives, conflicting reasoning paths, or inconsistent internal states. How should topologies be adapted to resolve disagreement across systems? New paradigms in multi-agent negotiation, hierarchical arbitration, and meta-level consensus mechanisms (e.g., judge agents, governance substructures) may need to be integrated. Structure learning in this space will not only optimize for task performance but also for alignment, fairness, and resolution efficiency.

**Trust, Robustness, and Verification Across Systems.** Multi-system collaborations introduce unique challenges in trust calibration and fault tolerance. If one agent cluster becomes compromised—e.g., via adversarial prompts, faulty tool outputs, or outdated models—how should other systems respond structurally? Research is needed on topological anomaly detection, trust-aware edge pruning, and robust structure adaptation in the presence of partial failures or adversarial agents. Verifiable topologies and explainable structure profiles could become essential for accountability and safety in high-stakes deployments (e.g., finance, healthcare, governance).

## 6 Conclusion

This position paper advocates for topological structure learning as a critical and underexplored direction in LLM-based MASs. We have outlined a conceptual decomposition of the topology design problem into agent selection, structure profiling, and topology synthesis, each posing distinct algorithmic and theoretical challenges. To this end, we attempt to propose a systemic framework to solve these three sub-problem. We hope this paper will catalyze future research into topological structure learning and lay the foundation for more intelligent and adaptive MASs.

## References

- [1] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv:2308.08155*, 2023.
- [2] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *NeurIPS*, 36:51991–52008, 2023.
- [3] Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, et al. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv:2504.01990*, 2025.
- [4] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *ICLR*, 2024.
- [5] Md Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. Mapcoder: Multi-agent code generation for competitive problem solving. In *ACL*, 2024.
- [6] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.
- [7] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv:2309.17452*, 2023.
- [8] Wenbei Xie, Donglin Liu, Haoran Yan, Wenjie Wu, and Zongyang Liu. Mathlearner: A large language model agent framework for learning to solve mathematical problems. *arXiv:2408.01779*, 2024.
- [9] Yizhe Huang, Xingbo Wang, Hao Liu, Fanqi Kong, Aoyang Qin, Min Tang, Xiaoxi Wang, Song-Chun Zhu, Mingjie Bi, Siyuan Qi, and Xue Feng. Adasociety: An adaptive environment with social structures for multi-agent decision-making. In *NeurIPS Datasets and Benchmarks Track*, 2024.
- [10] Qinlin Zhao, Jindong Wang, Yixuan Zhang, Yiqiao Jin, Kaijie Zhu, Hao Chen, and Xing Xie. CompeteAI: Understanding the competition dynamics of large language model-based agents. In *ICML*, 2024.
- [11] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. In *IJCAI*, 2024.
- [12] Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration. In *ICLR*, 2025.
- [13] Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. *arXiv:2410.02506*, 2024.

- [14] Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. Agent-dropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration. *arXiv:2503.18891*, 2025.
- [15] Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Qingsong Wen, Kun Wang, and Yang Wang. Netsafe: Exploring the topological safety of multi-agent networks. *arXiv:2410.15686*, 2024.
- [16] Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. Masrouter: Learning to route llms for multi-agent systems. *arXiv:2502.11133*, 2025.
- [17] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration. In *COLM*, 2024.
- [18] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv:2308.00352*, 3(4):6, 2023.
- [19] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: Language agents as optimizable graphs. In *ICML*, 2024.
- [20] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.
- [21] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv:2006.10637*, 2020.
- [22] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD*, pages 1269–1278, 2019.
- [23] Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. *arXiv:2502.04180*, 2025.
- [24] Ao Li, Yuexiang Xie, Songze Li, Fuguee Tsung, Bolin Ding, and Yaliang Li. Agent-oriented planning in multi-agent systems. *arXiv:2410.02189*, 2024.
- [25] Ziyu Wan, Yunxiang Li, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. Rema: Learning to meta-think for llms with multi-agent reinforcement learning. *arXiv:2503.09501*, 2025.
- [26] Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv:2410.11782*, 2024.
- [27] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. In *ICLR*, 2024.
- [28] Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. Multiagentbench: Evaluating the collaboration and competition of llm agents, 2025.
- [29] 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. *arXiv:2410.10762*, 2024.
- [30] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *NeurIPS*, 32, 2019.
- [31] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.
- [32] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv:2308.03688*, 2023.
- [33] Shanshan Han, Qifan Zhang, Yuhang Yao, Weizhao Jin, Zhaozhuo Xu, and Chaoyang He. Llm multi-agent systems: Challenges and open problems. *arXiv:2402.03578*, 2024.
- [34] Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, et al. Multiagentbench: Evaluating the collaboration and competition of llm agents. *arXiv:2503.01935*, 2025.
- [35] Yusu Qian, Haotian Zhang, Yinfei Yang, and Zhe Gan. How easy is it to fool your multimodal llms? an empirical analysis on deceptive prompts. *arXiv:2402.13220*, 2024.

- [36] Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. Dyval: Dynamic evaluation of large language models for reasoning tasks. In *ICLR*, 2024.
- [37] Kaijie Zhu, Jindong Wang, Qinlin Zhao, Ruochen Xu, and Xing Xie. Dyval 2: Dynamic evaluation of large language models by meta probing agents. In *ICML*, 2024.
- [38] Shudong Liu, Yiqiao Jin, Cheng Li, Derek F Wong, Qingsong Wen, Lichao Sun, Haipeng Chen, Xing Xie, and Jindong Wang. Culturevlm: Characterizing and improving cultural understanding of vision-language models for over 100 countries. *arXiv:2501.01282*, 2025.
- [39] Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Global counterfactual explainer for graph neural networks. In *WSDM*, pages 141–149, 2023.
- [40] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.
- [41] Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Matei Zaharia, James Zou, and Ion Stoica. Optimizing model selection for compound ai systems. *arXiv:2502.14815*, 2025.
- [42] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv:2406.04692*, 2024.
- [43] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318, 2002.
- [44] Yiqiao Jin, Qinlin Zhao, Yiyang Wang, Hao Chen, Kaijie Zhu, Yijia Xiao, and Jindong Wang. Agentreview: Exploring peer review dynamics with llm agents. In *EMNLP*, 2024.
- [45] Chuanneng Sun, Songjun Huang, and Dario Pompili. Llm-based multi-agent reinforcement learning: Current and future directions. *arXiv:2405.11106*, 2024.
- [46] Yiqiao Jin, Mohit Chandra, Gaurav Verma, Yibo Hu, Munmun De Choudhury, and Srijan Kumar. Better to ask in english: Cross-lingual evaluation of large language models for healthcare queries. In *Web Conference*, pages 2627–2638, 2024.
- [47] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *AISTATS*, pages 208–214, 2011.
- [48] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *ICML*, pages 11492–11502. PMLR, 2020.
- [49] Noga Alon, Nicolo Cesa-Bianchi, Ofer Dekel, and Tomer Koren. Online learning with feedback graphs: Beyond bandits. In *COLT*, pages 23–35. PMLR, 2015.
- [50] Nicolò Cesa-Bianchi, Tommaso R Cesari, and Riccardo Della Vecchia. Cooperative online learning with feedback graphs. *TMLR*, 2021.
- [51] Chloé Rouyer, Dirk van der Hoeven, Nicolò Cesa-Bianchi, and Yevgeny Seldin. A near-optimal best-of-both-worlds algorithm for online learning with feedback graphs. *NeurIPS*, 35:35035–35048, 2022.
- [52] Wenkui Ding, Tao Qin, Xu-Dong Zhang, and Tie-Yan Liu. Multi-armed bandit with budget constraint and variable costs. In *AAAI*, volume 27, pages 232–238, 2013.
- [53] Yingce Xia, Haifang Li, Tao Qin, Nenghai Yu, and Tie-Yan Liu. Thompson sampling for budgeted multi-armed bandits. In *AAAI*, pages 3960–3966, 2015.
- [54] Minzhi Li, William Barr Held, Michael J Ryan, Kunat Pipatanakul, Potsawee Manakul, Hao Zhu, and Diyi Yang. Mind the gap! static and interactive evaluations of large audio models. *arXiv:2502.15919*, 2025.
- [55] Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.
- [56] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *NeurIPS*, 36:46595–46623, 2023.

## A Notations

Table 1 shows the notations used throughout the paper.

Table 1: Notation used in this paper.

Symbol	Meaning
$\mathcal{A} = \{a_i\}$	Set of candidate agents
$\mathcal{A}^* \subseteq \mathcal{A}$	Optimal set of agents
$\hat{\mathcal{A}}$	Selected agents, $ \hat{\mathcal{A}}  = k$
$T = \{x\}$	Specific task instance
$M$	Macro-structure / coordination pattern
$C(a_i)$	Cost of agent $a_i$
$\gamma$	Cost-reward trade-off coefficient ( $\gamma \geq 0$ )
$\mathcal{R}(x, M, \hat{\mathcal{A}})$	Expected reward under $M$ with $\hat{\mathcal{A}}$
$\mathcal{U}(x, M, \hat{\mathcal{A}})$	$\mathcal{R} - \gamma \sum C(a_i)$ utility
$\mu(\hat{\mathcal{A}})$	True expected utility of $\hat{\mathcal{A}}$
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$	Temporal graph
Message( $\cdot$ )	Message function
$R_i$	response from agents
$P_{sys}, P_{user}$	system prompt, user prompt
$\mathcal{O}_i$	aggregated messages from neighbor agents

## B Algorithm

Algorithm 1 presents the strategy of greedy selection under budget  $B$  for agent selection.

---

### Algorithm 1 Greedy Selection Under Budget $B$ .

---

**Require:** Ground set  $\mathcal{A}$ , submodular  $f(\cdot)$ , cost  $C(\cdot)$ , budget  $B$

```

1:  $\hat{\mathcal{A}} \leftarrow \emptyset, ; b \leftarrow 0$  ▷  $b$  tracks accumulated cost
2: while  $\exists a_i \in \mathcal{A} \setminus \hat{\mathcal{A}}$  s.t.  $b + C(a_i) \leq B$  do
3:    $\hat{a} \leftarrow \operatorname{argmax}_{a_i \in \mathcal{A} \setminus \hat{\mathcal{A}}} \frac{f(\hat{\mathcal{A}} \cup \{a_i\}) - f(\hat{\mathcal{A}})}{C(a_i)}$ 
4:   if  $b + C(a_i) \leq B$  and  $f(\hat{\mathcal{A}} \cup \{a_i\}) > f(\hat{\mathcal{A}})$  then
5:      $\hat{\mathcal{A}} \leftarrow \hat{\mathcal{A}} \cup \{a_i\}; \quad b \leftarrow b + C(a_i)$ 
6:   else
7:     break ▷ No affordable beneficial addition remains
8:   end if
9: end while
10: return  $\hat{\mathcal{A}}$  ▷  $(1 - 1/e)/2$  approximation [40]

```

---

## C Details of Agent Selection

Following the formulation in Section 3.1, we can apply variations of MAB (Appendix C.2), and design intermediary rewards to overcome the sparsity in reward signals (Appendix C.2).

### C.1 Static Agent Selection

Once cost-constrained exploration stabilizes empirical rewards  $\hat{\mu}_i$ , the problem reduces to choosing a subset  $\hat{\mathcal{A}} \subseteq \mathcal{A}$  that maximizes an ensemble utility  $\mathcal{U}(\mathcal{A}, x)$  subject to the same budget. An exhaustive search over all  $2^{|\mathcal{A}|}$  subsets is generally intractable [41], but we can exploit the structure of  $\mathcal{U}$ . *Diminishing-returns*. In ensembling [42], the marginal benefit of adding one additional agent can diminish due to overlapping knowledge or capabilities. Consequently,  $\mathcal{U}(x, M, \hat{\mathcal{A}})$  is approximately submodular in  $\hat{\mathcal{A}}$ —the benefit of adding an agent decreases as the committee grows. When this holds, the classic greedy algorithm—adding agents one at a time based on their marginal gain—achieves a  $(1 - 1/e)$  approximation to the optimal selection.

**Budget-constrained Variant.** When the committee size is flexible but a hard budget  $B$  exists, the classical ratio-greedy knapsack algorithm can be applied (Appendix Algorithm 1). At each step, the algorithm selects the agent with the highest *benefit-to-cost* ratio  $(\hat{\mu}_i - \gamma(a_i)C(a_i))/C(a_i)$ , and continues until the total cost reaches the budget  $B$ . Finally, this approach naturally extends to *hierarchical task decomposition*. When a complex task can be divided into smaller sub-tasks (e.g., planning, solving, verification), we run separate budgeted bandits and greedy selections for each sub-task. We then compose the selected agents into a full workflow, enabling fine-grained agent specialization while preserving global budget efficiency.

## C.2 Dynamic Agent Selection

**Node Embedding Curation.** We propose constructing node embeddings  $\mathbf{v}_i^0$  from four feature groups: 1) *Capability*: context length, training corpus size, parameter counts, embedding dimensions; 2) *Cost*: price per 1000 tokens, hardware costs; 3) *Performance history*: success rate, latency, recent tasks metrics such as F1/BLUE scores [43]; and 4) *Persona attributes*: communication styles, creativity, safety stance.

**Task Bank.** Tasks  $\mathcal{X}$  are organized in a *coarse-to-fine* faceted taxonomy: 1) *skill*: reasoning, code generation, tool use; 2) *modality*: text, code, image, audio, and video; 3) *interaction pattern*: single/multi-turn conversations or planning. Each task receives multiple labels across these axes, with associated seed prompts and evaluation rubrics.

**Systematic Persona Synthesis.** Current approaches to agent selection often rely on handcrafted prompt templates to define personas [44]. For example, *creative* agents favor broader exploration, while *conservative* agents prefer safe, high-confidence regions [45]. Agents might prefer clarity (*executive*), critical thinking (*Socratic*), or story-telling (*narrative*).

However, LLMs can deviate from their assigned roles, even when provided with explicit instructions and behavioral criteria. This leads to behavioral homogeneity and communication redundancy across agents.

To overcome this, future work could explore two complementary directions for persona synthesis:

- **Parametric Persona Curation.** Rather than *discrete*, natural-language-based templates, personas can be embedded in a continuous, interpretable latent space—parameterized along axes such as exploration-exploitation, verbosity, or risk tolerance. Trait-conditioned adapters or prompt generators can then enable flexible and fine-grained control by smooth interpolation between styles.
- **Data-Driven Persona Discovery.** Alternatively, we can mine diverse agent behaviors by prompting LLMs with varied instructions and clustering resulting dialogue patterns. Embedding these interactions reveals the range of emergent personas, which can then be selectively instantiated.

In both approaches, automated validation—through competency, bias, and efficiency checks—ensures that newly synthesized agents are both safe and effective before deployment [46].

**Smooth Learning and Cold Start.** Deploying an under-initialized agent-selection algorithm in a new environment can be risky, incurring prohibitive cost, latency, or safety violations. Naive exploration can lead to excessive cost, latency, or even safety violations before effective learning. To ensure *smooth learning*, one promising research direction is to design effective initialization strategies that provide low-regret starting points and enable safe, efficient adaptation from the outset.

**Online Identification via Cost-aware Bandits.** Each candidate agent  $a_i$  is modeled as a stochastic bandit arm with cost  $C(a_i)$  and reward from a utility scoring function. Our objective is to identify a subset of agents that maximizes utility per unit cost across a stream of tasks. Variants based on existing algorithms, such as contextual bandits [47] or Neural UCB [48], guide exploration using query features (e.g., domain tag, length, etc.), prioritizing under-explored yet promising agents. Each round, the controller predicts the expected reward, applies an optimism bonus, and selects the top- $k$  arms. While feedback-graph bandits [49–51] considered structural interactions among arms, they mostly focus on leveraging this topology to improve sample efficiency by assuming fixed and given graph structures rather than optimizing them. Under a spend budget  $B$ , budget-aware variants based on *budgeted-UCB* [52] and *budgeted Thompson Sampling* [53] manage agent selection by treating each pull as consuming cost  $C(a_i)$  until budget  $B$  is reached.

**Intermediary Rewards via Automatic Scorers.** Human or gold-standard labels offer accurate utility assessments but are costly and sparse. Instead, automatic scorers can be used to provide instantaneous, low-cost reward estimates. Scorer models are architecturally agnostic and can take various forms: a *smaller, instruction-tuned* specialist model [54, 55], a rule-based evaluator (e.g. exact string match, BLEU/ROUGE, BERTScore, toxicity-detectors), or LLM-as-a-Judge [56]. Critically, the scorer does not need to surpass production agents in size or capability—it merely serves as a lightweight, cost-effective proxy for reward signals. As LLM-based automatic feedback might not always be accurate, we can

develop superior mechanisms for providing corrective feedback or rewards to shape agent behaviors in real time without blocking the training flow.

### Motivation for Temporal GNN for Agent Selection.

- *Time-varying Connections*: Communication patterns between agents can change rapidly (e.g., requests for help, forwarding solutions, ephemeral collaborations). An agent may broadcast to many peers during peak load but fall back to a few trusted collaborators when quotas tighten. TGNNs explicitly encode such time-varying edges.
- *Sequential Dependencies*: The impact of a message depends on *when* it arrives: an expensive agent triggered late in the interaction window may violate a latency SLA even if its answer is accurate. TGNNs condition updates on inter-event time  $\Delta t$ .
- *Path Dependencies*: Agents accumulate *context* over rounds of communication. TGNNs enable long-range modeling by preserving this information through their node memories.

## D Details of Structure Profiling

### D.1 Existing Studies

Automated design of agentic systems is a nascent paradigm aiming to replace static, hand-crafted agent organizations with learned ones [55]. AFlow [29] reformulates an agent workflow as a programmatic graph and uses Monte Carlo tree search to iteratively refine this workflow. Similarly, MaAS [23] can sample a query-dependent agent configuration from this supernet at runtime. MasRouter [16] uses a cascaded controller network to dynamically decide how to route a user’s query through a bunch of agents and determine the collaboration mode. G-Designer [26] leverages a graph neural network to generate a communication topology for the agents based on the task at hand. Similarly, CommFormer [27] uses a continuous relaxation of the adjacency matrix to allow gradient-based search of the optimal inter-agent communication links. To guide the formation of an effective communication structure, AOP [24] emphasizes structural principles like solvability, non-redundancy and completeness, where meta agent decomposes and assigns agents. The reinforced meta-thinking agents framework ReMA [25] has a high-level agent that plans and monitors the reasoning strategy, as well as a low-level agent that executes the reasoning steps. The existing work motivates developing a general method to determine optimal multi-agent structures adaptively.

### D.2 SPAN

**Formulation.** We formalize structure profiling as an optimization problem. Let  $\mathcal{A} = \{a_1, a_2, \dots, a_N\}$  be a set of available LLM agents (with possibly diverse expertise or roles), and let  $x$  denote an input problem or query. A structure  $\mathcal{G}$  describes the organization of these agents for solving  $x$ , named SPAN (Structure Profiling Adaptive Network). Similar to the temporal graph defined in Section C.2, we can represent  $\mathcal{G}$  as a *directed* graph (or network)  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V} \subseteq \mathcal{A}$  is the subset of agents utilized and  $\mathcal{E}$  is the set of directed communication links between agents. For example, an edge  $e_{ij} \in \mathcal{E}$  indicates agent  $a_i$ ’s output is forwarded as input context to agent  $a_j$ , namely  $a_i \rightarrow a_j$ . This graph can encode various topologies (chain, star, fully connected, etc.) and implicit schedules of agent interactions (a directed acyclic graph imposes a partial order of information flow). Given a particular structure  $\mathcal{G}$  and task  $x$ , the agents execute a collaborative reasoning process (e.g. passing messages along  $\mathcal{E}$  and performing computations at each node  $a_i \in \mathcal{V}$ ) to produce a final result  $y = f_\theta(\mathcal{G}, x)$ .

As the utility function  $\mathcal{U}(\mathcal{G}; x)$  defined in Section 3.1, it measures the quality of the result (and possibly the efficiency) when using structure  $\mathcal{G}$  on task  $x$ . This could be, for instance, the task reward or accuracy achieved by the agents minus a penalty for communication cost (number of messages or tokens exchanged). The goal of structure profiling is to find the optimal agent network  $\mathcal{G}$  for each task. Formally, for a given query  $x$  we seek:

$$\mathcal{G}^*(x) = \arg \max_{\mathcal{G} \in \mathcal{S}} \mathcal{U}(\mathcal{G}; x), \quad (9)$$

Let  $\mathcal{S}$  be the space of all feasible structures (all possible directed graphs over the agent set  $\mathcal{A}$  that respect any required constraints, such as acyclicity or communication budget limits). In words,  $\mathcal{G}^*(x)$  maximizes the chosen utility, achieving the best trade-off between solution quality and costs for that particular problem. In practice, directly searching  $\mathcal{S}$  for each new query is intractable, where  $\mathcal{S}$  grows combinatorially with the number of agents and possible connections. Instead, our aim is to learn a structure profiling policy that can rapidly select or construct a high-performing  $\mathcal{G}$  for any given task, without brute-force search.

**Task-Conditioned Structure Policy.** To realize this, we propose a policy function  $\Pi$  that maps a task specification to a structured multi-agent configuration. Concretely,  $\Pi$  takes as input some representation of the query  $x$  (for example, an

encoding of the user’s request or a brief summary of the problem) and outputs a definition of the graph  $\mathcal{G}$  (which agents to use and how to connect them). We denote this  $\mathcal{G}_\Pi(x) = \Pi(x)$ . The policy can be understood as a planner or graph constructor specialized for assembling LLM agents. In our approach,  $\Pi$  is implemented as a parametric model that is trained to approximate the optimal mapping  $x \mapsto \mathcal{G}^*(x)$ .

A convenient formulation is to model  $\Pi$  as a function that outputs the probability of forming a communication link between each pair of agents, conditioned on the task. Specifically, for any two agents  $a_i$  and  $a_j$ ,  $\Pi$  produces a probability  $p_{ij}(x)$  indicating the likelihood that  $a_i$  should transmit information to  $a_j$  given task input  $x$ . Collectively, these probabilities form a task-dependent probabilistic adjacency matrix  $P(x) \in [0, 1]^{N \times N}$ , with the  $(i, j)$ -th entry of  $P(x)$  being  $p_{ij}(x)$ . A discrete graph  $\mathcal{G}$  can then be instantiated by sampling each edge  $e_{ij}$  independently with probability  $p_{ij}(x)$ , or selected deterministically, e.g., including  $e_{ij}$  only if  $p_{ij}(x)$  exceeds a predefined threshold. In this way,  $\Pi$  defines a distribution over topological structures, tailored to each task.

We incorporate domain knowledge and constraints into  $\Pi$  to reduce the search space. For instance, if we know certain agents are relevant only for specific domains or sub-tasks,  $\Pi$ ’s architecture can be designed to consider those aspects of  $x$ . One might use a two-step decision: first choose a subset of agents  $\mathcal{V} \subseteq \mathcal{A}$  to be utilized, then decide the directed connections  $\mathcal{E}$  among them. This could be achieved by outputting additional binary variables  $q_i(x)$  indicating whether to activate agent  $a_i$  for task  $x$ . The resulting structure policy could be factorized as  $\Pi(x) = (\mathcal{V}(x), \mathcal{E}(x))$ . For example,  $\Pi$  might select agents with the required skills for  $x$  (using  $q_i$  predictions), and then configure a communication graph  $\mathcal{E}$  among those agents to best facilitate information flow for the task. We note that many structure profiling strategies can be encoded by an appropriate  $\Pi$ : a trivial  $\Pi$  that always returns a fixed  $\mathcal{G}_0$  corresponds to a static topology baseline, whereas an ideal  $\Pi$  would tailor the graph in a highly task-specific way (e.g. forming teams of experts and a coordinator for a complex analytical question, but choosing a single-agent or linear chain for a simple narrative generation task).

The  $\Pi$  might be realized by a graph neural network that, given an embedding of the task (and possibly initial agent state representations), predicts edge scores for each pair of agents [26]. Another approach is to use the LLM agents themselves in a meta-capacity, i.e., one agent (or the system prompt) could be designated to plan the collaboration (writing pseudocode or instructions that effectively set  $\mathcal{E}$  and  $\mathcal{V}$ ). While the implementation can vary, the core idea is that  $\Pi$  provides a mapping from task features to multi-agent structure in a principled, learnable way.

**Optimization Strategy.** The parameters of  $\Pi$  are trained to maximize the expected utility of the chosen structures. We define an objective across the distribution of tasks (or a training set of tasks  $\mathcal{X}$ ):

$$J(\Pi) = \mathbb{E}_{x \sim \mathcal{D}} [\mathcal{U}(\mathcal{G}_\Pi(x); x)],$$

where  $\mathcal{D}$  is the task distribution and  $\mathcal{G}_\Pi(x)$  is the structure output by policy  $\Pi$ . Optimizing  $J(\Pi)$  aligns the policy to prefer structures that yield high task performance. In practice, this optimization can be approached in different ways depending on how  $\Pi$  is implemented. If  $\Pi$  is differentiable (e.g., using continuous relaxations for graph decisions [27]), one can apply gradient-based methods to directly increase  $\mathcal{U}$ . More generally, since  $\mathcal{G}$  selection may be discrete, we can employ reinforcement learning that treat the selection of a structure for a given task as an action, receive the utility  $\mathcal{U}$  as a reward, and update  $\Pi$  to reinforce choices that lead to higher rewards. The policy gradient or evolutionary search can be used to improve  $\Pi$  over many sampled tasks. Notably, some recent work has demonstrated the viability of search-based optimization for agent structures (e.g., AFlow’s MCTS optimization of workflows [29]). Our method can leverage similar search procedures:  $\Pi$  can initially generate a candidate structure for  $x$ , which is then refined by local adjustments (adding/removing an edge, activating a new agent, etc.) if those adjustments empirically improve  $\mathcal{U}(\mathcal{G}; x)$ . Such iterative refinement can be guided by heuristics or learned value functions that predict how a structure change would impact future reward.

Importantly, we impose any known constraints and invariance during optimization. For instance, to avoid cycles in communication (which could cause infinite loops or repeated content), we restrict  $\mathcal{E}$  to DAGs by design or add a large penalty for cyclic graphs. We may also regularize  $\Pi$  to prefer simpler structures (fewer edges) unless additional communication is clearly beneficial, thus preventing gratuitous use of fully connected messaging. The outcome of training is a policy  $\Pi^*$  that profiles the task structure, i.e., given a new problem,  $\Pi^*$  can swiftly instantiate a near-optimal multi-agent network for solving it. Formally, we hope to approximate  $\mathcal{G}_\Pi(x) \approx \mathcal{G}^*(x)$  for all  $x$  in the domain of interest, without an exhaustive search at query time.

By framing structure profiling as a learning problem, SPAN adapts to the patterns in tasks it has seen. For example, it might learn that analytical math problems warrant a pair of thinker with checker agents, whereas creative writing prompts are best handled by a single agent to maintain coherence. The policy improves its ability to configure the right topology, division of labor, and information routing, which yields a general and extensible solution, that is, as new agent capabilities or tools are added to the pool  $\mathcal{A}$ , the policy  $\Pi$  can learn to incorporate those into structures when beneficial (e.g. including a code-generation agent in the loop only for programming-related queries). The final output of SPAN is

therefore not a single static agent system, but a profiling mechanism that flexibly assembles ad hoc agent teams with optimized interaction structures for each task.

### D.3 Future Discussion

The framework above demonstrates that exploring adaptive structures that evolve during problem solving is a promising approach. That is, instead of fixing  $\mathcal{G}$  from the start, agents could dynamically rewire their communication links based on intermediate results. Another direction is to incorporate heterogeneous objectives and game-theoretic considerations, e.g., in competitive or mixed cooperative-adversarial settings, the optimal structure might involve forming coalitions or introducing mediator agents. Our task can be extended to design structures that are robust against adversarial agents or that balance fairness and information sharing in a team. Furthermore, consider to integrate human insights and interpretability into structure profiling, i.e., the learned graphs and role assignments could be constrained to mirror effective human organizational strategies, such as hierarchical delegation and review committees, so that the resulting agent society is interpretable and trustworthy. Finally, as foundation models diversify, an exciting frontier is profiling structures across modality-diverse agents, e.g., LLMs alongside vision or robotics agents. In summary, structure profiling for LLM-based agents is a rich field, and future work can build on this foundation by developing more adaptive, robust, and generalizable mechanisms for architecting multi-agent intelligence.

## E Details of Topological Structure Synthesis

### E.1 Details for Task-Aligned Utility Estimation for Topology Optimization

**Counterfactual Topology Intervention.** To quantify the topological structure importance of each agent  $a_i \in \hat{\mathcal{A}}$  or communication edge  $e_i \in E$  within a synthesized topological structure  $G = (\hat{\mathcal{A}}, E) \in \mathcal{G}_M(\hat{\mathcal{A}})$ , we adopt a counterfactual reasoning approach. Specifically, we define the task-aligned contribution of each component, which can be either an agent  $a_i$  or a communication edge  $e$ , as the drop in utility resulting from its removal. Let  $\mathcal{U}(x, M, \hat{\mathcal{A}})$  denote the utility associated with executing task  $x \in T$  under macro-structure  $M$  and agent subset  $\hat{\mathcal{A}}$ . Formally, we define the contribution of each agent  $a_i \in \hat{\mathcal{A}}$  as

$$\phi(a_i) := \mathcal{U}(x, M, \hat{\mathcal{A}}) - \mathcal{U}(x, M, \hat{\mathcal{A}} \setminus \{a_i\}), \quad (10)$$

and the contribution of each edge  $e_i \in E$  as

$$\phi(e_i) := \mathcal{U}(G; x) - \mathcal{U}(G \setminus \{e\}; x). \quad (11)$$

Here,  $\mathcal{U}(G; x)$  denotes the utility under a given micro-topology  $G$ , and  $G \setminus \{e\}$  or  $\hat{\mathcal{A}} \setminus \{a_i\}$  respectively represent topologies with the edge or agent removed. Given these contribution scores, we formalize the topology refinement as a constrained selection problem. The goal is to identify a subgraph  $G' = (\hat{\mathcal{A}}', E') \subseteq G$  that retains only components with sufficiently high task-aligned utility, while maximizing the overall system performance. Let  $\delta_v$  and  $\delta_e$  denote contribution thresholds for nodes and edges, respectively. Then, the resulting substructure is obtained by solving:

$$G^* = \arg \max_{\hat{\mathcal{A}}', E'} \mathcal{U}(G'; x), \text{ subject to } \phi(a_i) \geq \delta_v, \forall a_i \in \hat{\mathcal{A}}'; \phi(e) \geq \delta_e, \forall e \in E'. \quad (12)$$

**Efficient Estimation Approaches.** To mitigate the computational burden of explicit counterfactual interventions, two efficiency-oriented approximations are explored that estimate structural importance while preserving alignment with task utility. The first, *Marginal Contribution Sampling*, approximates the counterfactual utility drop by averaging over multiple stochastic subgraph configurations. Instead of removing one agent or edge at a time, it estimates the marginal gain of including a component by evaluating its incremental effect across sampled subsets  $S_j \subseteq \hat{\mathcal{A}}$ , as formalized by  $\phi(a_i) \approx \frac{1}{m} \sum_{j=1}^m [\mathcal{U}(S_j \cup \{a_i\}) - \mathcal{U}(S_j)]$ . In contrast, *Gradient-Informed Attribution* leverages differentiability in the utility function to infer structural relevance via sensitivity analysis. By relaxing discrete structures into continuous soft masks (e.g., edge weights  $w_{e_i}$  or agent activation factors  $\alpha_{a_i}$ ), it computes importance scores via backpropagation as  $\phi(e_i) \approx \left| \frac{\partial \mathcal{U}}{\partial w_{e_i}} \right|$  or  $\phi(a_i) \approx \left| \frac{\partial \mathcal{U}}{\partial \alpha_{a_i}} \right|$ .

**Group-level Structural Effects.** While the aforementioned approaches estimate structural importance at the level of individual agents or edges, they implicitly assume that each component contributes independently to task performance. However, the utility of an agent is tightly coupled with others in many cases, such as collaborative planning. To capture such interdependencies, we consider *Group-level Perturbation*, which generalizes counterfactual intervention from individual units to agent clusters. Concretely, we evaluate the collective utility drop incurred by removing a group of agents  $\mathcal{S} \subseteq \hat{\mathcal{A}}$ , defining their joint contribution as:

$$\phi(\mathcal{S}) := \mathcal{U}(\hat{\mathcal{A}}) - \mathcal{U}(\hat{\mathcal{A}} \setminus \mathcal{S}). \quad (13)$$



## **E.2 Future Discussion**

Rather than relying on known training datasets, which may only generalize to specific task domains, for topological structure optimization. Future work should explore task-oriented dynamic topology synthesis that can adapt during execution to unseen test instances or evolving task requirements. In addition, training-free approaches should also be explored in the future to construct communication structures on prior knowledge, such as agent roles or prompt-level heuristics.