Hetero-Designer: Automated Design of Multi-Agent Systems with Heterogeneous LLMs

Zhiheng Zhang[1,2], Yuanzhe Zhang[3,†], Bohan Yu[1,5], Daojian Zeng[4], Kang Liu[1,2], Jun Zhao[1,2,†]
[1] The Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences
[3] National Science Library, Chinese Academy of Sciences [4] Hunan Normal University
[5] School of Advanced Interdisciplinary Sciences, University of Chinese Academy of Sciences
zhangzhiheng2024@ia.ac.cn, zhangyuanzhe@mail.las.ac.cn, yubohan2025@ia.ac.cn
zengdj@hunnu.edu.cn, kliu@nlpr.ia.ac.cn, jzhao@nlpr.ia.ac.cn

Abstract

LLM-based Multi-agent systems (MAS) have shown strong capabilities across a wide range of domains. Their success largely hinges on the collaboration topology design, which has emerged as a central research focus in the automated MAS design. However, existing approaches are fundamentally constrained by their reliance on homogeneous LLMs, which significantly limits overall system intelligence. In response to this limitation, we for the first time propose the concept of **Automated Design of Heterogeneous-LLMs-based MAS (ADHM)**. ADHM sheds light on a promising avenue for advancing collective intelligence, which focuses on the automated design of cost-effective MAS composed of diverse LLMs and roles to suit various queries. Toward this challenging goal, we propose **`Hetero-Designer`**, a novel pipeline that efficiently encodes intricate dependencies among queries, LLMs and roles through a novel Binary-Star Transformer and constructs Hetero-MAS in an autoregressive graph generation process. Extensive experiments demonstrate that **`Hetero-Designer`** is: (i) high-performing on various benchmarks, (ii) economical in reducing overhead, (iii) extensible to unseen LLMs and roles.

1 Introduction

Large Language Model (LLM)-based agents (Richards, 2023; Nakajima, 2023; Park et al., 2023), have demonstrated impressive capabilities across a wide range of tasks (Zhang et al., 2024; Li et al., 2025; Schmidgall et al., 2025). Building on the success of individual agent, recent research has increasingly focused on Multi-Agent Systems (MAS) (Du et al., 2024; Hong et al., 2024; Liang et al., 2024), which open up new avenues for advancing collective intelligence through synergistic interactions among agents with diverse expertise and specialized roles. Central to the advancement of MAS lies the design of **collaboration topology**, a structural graph that defines how agents with various roles are organized and how they exchange information (Zhang et al., 2025d; Yang et al., 2025; Li et al., 2026; Liu et al., 2026).

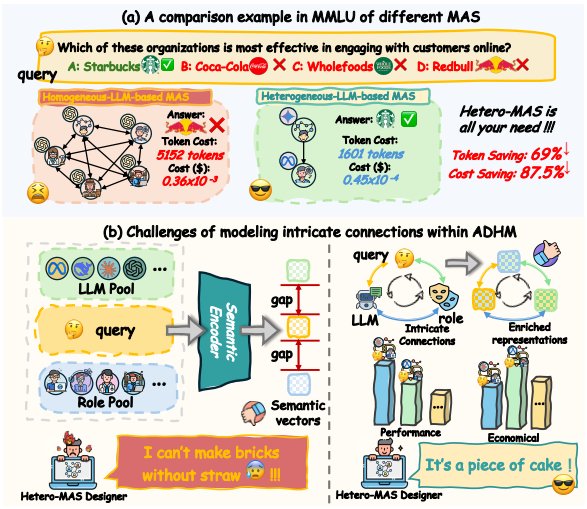


Figure 1: A comparison case and complex intrinsic connection within Hetero-MAS.

Early research on MAS topology design focused on manually crafted static topologies, such as chain (Wei et al., 2022; Hong et al., 2024), tree (Yao et al., 2023; Wu et al., 2024), complete graphs (Qian et al., 2025), and LLM-based network (Liu et al., 2024). Although such elaborately designed topologies can facilitate coordination in specific scenarios, Zhang et al. (2025e) highlighted that their inherent rigidity results in significant variations in MAS performance and cost across tasks. To improve flexibility and efficiency, contemporary studies have progressively explored **automated topology design** (Yang et al., 2025), which leverages data-driven graph learning to adaptively tailor MAS to specific domains (Zhang et al., 2025d; Wang et al., 2025b) or fine-grained queries (Zhang et al., 2025e; Wang et al., 2025a; Li et al., 2026).

Despite these advances, most automated frame-

† Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 27553–27574

works rely on a homogeneous LLM (*e.g.*, GPT-4o) to drive all agents. This "*one-LLM-fits-all*" paradigm is steadily exposing fundamental bottlenecks in the overall intelligence of the system: recent studies reveal that homogeneous models often result in redundant topology design (Zhang et al., 2025d; Wang et al., 2025b; Zhang et al., 2026a), constrained capabilities in reasoning diversity and critical thinking (Zhang et al., 2025f), and a suboptimal balance between performance and cost (Yue et al., 2025). Empirical evidence, as shown in Figure 1(a), further underscores these limitations: a lightweight heterogeneous MAS can actually outperform a complex homogeneous system while incurring substantially lower token costs, enabled by complementary capabilities among fewer agents. Consequently, we for the first time introduce the concept of **Automated Design of Heterogeneous-LLMs-based MAS (ADHM)**, which sheds light on a promising avenue for advancing collective intelligence. ADHM focuses on the automated design of cost-effective MAS composed of diverse LLMs and roles to suit various queries.

Although several studies like LLM routing (Chen et al., 2024; Ong et al., 2025; Yue et al., 2025), and LLM & role assignment on predefined topologies (Ye et al., 2025; Hegazy, 2025) have made progress, these methods exhibit limitations for ADHM in: ***(i) Predefined Topology.*** They still rely on manually predefined topologies, which lack the structural flexibility to autonomously adapt to the specific nuances of each query; ***(ii) Inadequate Modeling of Implicit Correlations.*** As shown in Figure 1(b), designing cost-effective topologies requires precise modeling of the coupling agent roles and LLM capabilities under diverse queries. However, existing methods, following homogeneous MAS design (Zhang et al., 2025e; Li et al., 2026; Wang et al., 2025a), rely excessively on static semantic embeddings extracted by text encoder. Such static representations fail to capture the nuanced interrelations among heterogeneous LLMs, specialized roles and queries. Moreover, this information insufficiency further propagates through the topological construction process, resulting in suboptimal Hetero-MAS orchestration.

In response to these limitations, we propose **`Hetero-Designer`**, a novel pipeline that explicitly models and utilizes these complex interrelations to enable automated design of *flexible* and *cost-effective* Hetero-MAS. Technically, **`Hetero-Designer`** first encodes queries, LLMs and roles into semantic vectors. Subsequently, we design a Binary-Star Transformer (BSFormer) to synergize static features into enriched, dependency-aware representations by contextualizing the latent interplay among queries, LLMs, and roles. Specifically, BSFormer consists of two symmetric star-shaped modules: LLM-former and Role-former, where the query serves as the central relay node and LLMs (or roles) act as satellite nodes. Within each module, multi-head attention extracts fine-grained representations; between modules, cross-attention facilitates bidirectional information coupling. Finally, **`Hetero-Designer`** formulates MAS construction as a flexible autoregressive graph generation problem, leveraging the efficient representations produced by BSFormer and a GRU-based graph encoder to alternately predict nodes (LLMs and roles) and edges, thereby step-by-step synthesizing the complete Hetero-MAS topology.

Our contributions can be summarized as follows:

- ***Problem Definition***: We *for the first time* define Automated Design of Heterogeneous-LLM-based MAS (ADHM) and highlight that its challenge lies in modeling complex intrinsic connections among queries, LLMs and roles, which are essential for cost-effective Hetero-MAS topology design.
- ***Practical Solution***: We introduce a novel pipeline, **`Hetero-Designer`**, which leverages a Binary-Star Transformer to model the intricate dependencies and flexibly constructs MAS through an autoregressive graph generation process.
- ***Experimental Validation***: Extensive experiments across diverse benchmarks demonstrate that our method outperforms state-of-the-art baselines in both accuracy and cost-saving, demonstrating superior collective intelligence.

2 Formalization

2.1 Hetero-Multi-Agent-System

We model a MAS as a directed acyclic graph (DAG) $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, v_2, ..., v_{N_g}\}$ represents the set of nodes and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the flow of information among them. Each Node $v_i \in \mathcal{V}$ represents an agent instance, formalized as: $v_i = \{m_i, r_i, \text{State}_i, \text{Tool}_i\}$, where $m_i \in \mathbb{M} = \{\mathtt{m}_1, \mathtt{m}_2, ..., \mathtt{m}_{N_m}\}$ is the LLM powering v_i, while prior research employed homogeneous LLM to drive; $r_i \in \mathbb{R} = \{\mathtt{r}_1, \mathtt{r}_2, .., \mathtt{r}_{N_r}\}$

indicates the agent's role; State_i captures its memory and interaction history; Tool_i is a set of plugins available to v_i (*e.g.,* code compilers, search engine). An edge $e_{ji} = (v_j, v_i) \in \mathcal{E}$ indicates that the output of agent v_j is propagated as an input to agent v_i. The set of direct predecessors of agent v_i is denoted by $\mathcal{N}_{in}(v_i) = \{v_j | (v_j, v_i) \in \mathcal{E}\}$. $\mathcal{G}$ executes tasks following a topological ordering derived from $\mathcal{E}$. Each agent v_i generates a response o_i conditioned on its role-specific prompt, the global query $\mathcal{Q}$, and the output set $\{o_j \mid v_j \in \mathcal{N}_{in}(v_i)\}$ from its predecessor agents. Finally, the final response a of $\mathcal{G}$ is produced by an aggregation function $\text{Aggregate}\{o_i | v_i \in \mathcal{V}\}$ (*e.g.,* a summarizer agent).

2.2 Definition of ADHM

Based on the MAS defined above, we formalize Automated Design of Heterogeneous-LLMs-based MAS (ADHM): Given a query $\mathcal{Q}$, a set of heterogeneous LLMs $\mathbb{M}$, and a set of candidate roles $\mathbb{R}$, ADHM aims to learn a mapping function $f : (\mathcal{Q}, \mathbb{M}, \mathbb{R}) \to \mathcal{G}$. The objective of f is to facilitate automated MAS design, achieving an ideal trade-off between task performance and costs:

$$\max_{P(\mathcal{G}|\mathcal{Q})} \mathbb{E}_{\substack{(\mathcal{Q},\hat{a})\sim\mathcal{D}, \\ \mathcal{G}\in\mathbb{G}\sim P(\mathcal{G}|\mathcal{Q})}} \big[\underbrace{U(\mathcal{G};\mathcal{Q},\hat{a})}_{\textit{Performance}} - \lambda \cdot \underbrace{C(\mathcal{G};\mathcal{Q})}_{\textit{Cost}}\big], \tag{1}$$

where $(\mathcal{Q}, \hat{a})$ denotes the training instance in dataset $\mathcal{D}$, $\hat{a}$ is the ground-truth answer. $U(\cdot)$ measures the performance of the MAS, while $C(\cdot)$ quantifies the corresponding API token cost. The probability of DAG $\mathcal{G}$ is specifically defined as a joint distribution:

$$P(\mathcal{G}|\mathcal{Q},\mathbb{M},\mathbb{R}) = \prod_{i=1}^{|\mathcal{V}|} \underbrace{P(v_i|\mathcal{G}_{<i},\mathcal{Q},\mathbb{M},\mathbb{R})}_{\textit{node probability}} \times \prod_{j=1}^{i-1} \underbrace{P(e_{ji}|v_i,\mathcal{G}_{<i},\mathcal{Q})}_{\textit{edge probability}} \tag{2}$$

where $\mathcal{G}_{<i}$ denotes the subgraph induced by the previous $i-1$ nodes. The first term represents the joint probability of assigning the role r_i and LLM m_i to node v_i. Under the ADHM paradigm, the topology, agent roles, and LLM selection are jointly designed. This departs from existing techniques such as pruning homogeneous graphs (Zhang et al., 2025d; Wang et al., 2025b) or performing routing over predefined topologies (Yue et al., 2025).

2.3 Autoregressive Graph Generation

To make MAS topology generation process more tractable, we reformulate the problem as an autoregressive graph generation task (Zhang et al., 2019), where the model sequentially predicts node attributes v_i and their connectivity e_{ji} to preceding nodes. By decomposing global design into local decision-making steps, the model effectively navigates the vast search space (You et al., 2018; Wang et al., 2025a; Li et al., 2026). To enable adaptive termination, we introduce a terminal node v_{end}. The generation process is terminated when either a predefined maximum agent count γ is reached or the current MAS architecture is considered sufficient (the terminal signal v_{end} is predicted).

3 Hetero-Designer

Figure 2 illustrates the overall framework of **`Hetero-Designer`**. It begins by (i) encoding query, LLMs, roles, and the v_{end} into semantic embeddings. (ii) These embeddings are subsequently fed into a Binary-Star Transformer to capture the complex inter-dependencies. (iii) Ultimately, these enriched representations facilitate the autoregressive generation of the MAS. In addition, we train the model by cold-start supervision and reinforcement learning to enable superior Hetero-MAS Design.

3.1 Input Representation

`Hetero-Designer` first encodes all textual conditioning information into dense vector representations. The textual descriptions of LLMs and roles are provided in Appendix E. Specifically, the textual instructions of query $\mathcal{Q}$, is mapped into a fixed-dimensional vector $\mathbf{q} \in R^{d_i}$ by a pre-trained sentence encoder. This vector is then processed through a feed-forward network (FFN) with Layer Normalization (LN):

$$\mathbf{q} = \text{FFN}(\text{LN}(\text{SentenceEncoder}(\mathcal{Q}))) \tag{3}$$

Similarly, each available LLM $m_i \in \mathbb{M}$, role $r_j \in \mathbb{R}$, and the end token v_{end} are projected into respective embeddings $\mathbf{m}_i$, $\mathbf{r}_j$, and $\mathbf{e}$. The representations are aggregated into embedding matrices $\mathbf{M} \in R^{N_m \times d_i}$ and $\mathbf{R} \in R^{N_r \times d_i}$, which serve as the foundational feature sets for the MAS Design.

3.2 Binary-Star Transformer

ADHM's challenge lies in modeling the intricate correlations among queries, LLMs, and roles.

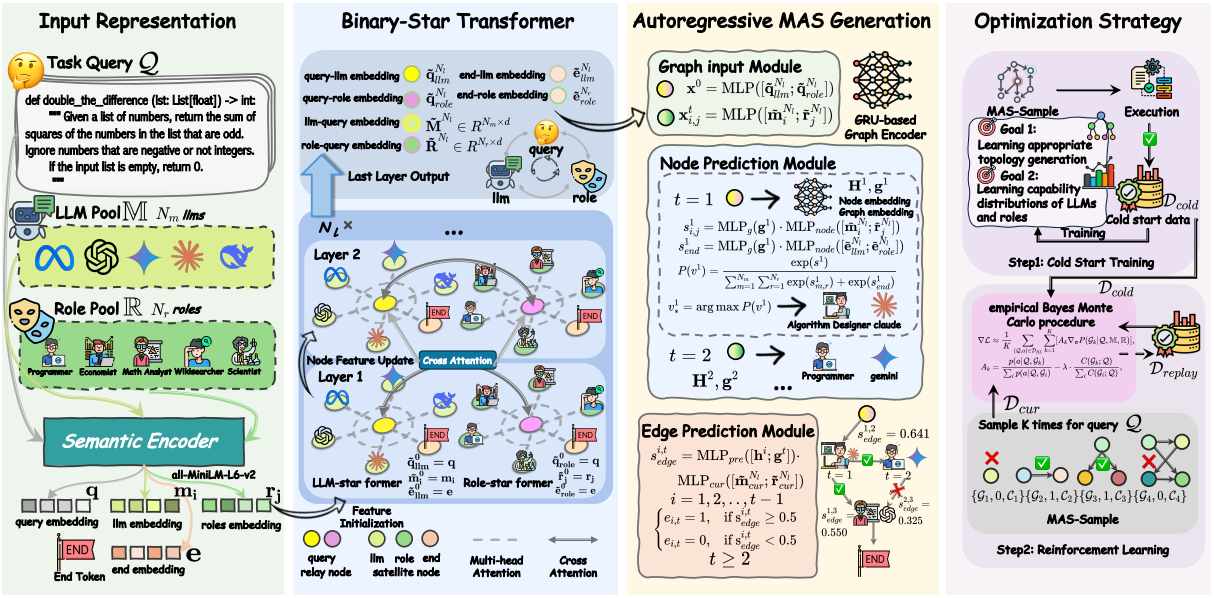


Figure 2: The overall framework of our proposed **Hetero-Designer**.

Static semantic vectors ($\mathbf{q}, \mathbf{M}, \mathbf{R}$) alone are insufficient to capture the complex dependencies essential for sophisticated MAS design. To this end, we propose a novel Binary-Star Transformer (BSFormer), which extends the architectural principles of the Star-Transformer (Guo et al., 2019).

Specifically, BSFormer comprises two symmetric modules: the LLM-Former and the Role-Former. Each module adopts a star-shaped topology with N_l layers consisting of a central relay node and $n+1$ satellite nodes. The relay nodes represent the central query-related information, and the satellite nodes represent the candidate set of either LLMs or agent roles, as well as the end token. Relay nodes in both modules are linked via cross-attention, enabling bidirectional information exchange between LLMs and roles. This bridges the semantic gap between model capabilities and functional roles, producing contextualized representations informed by global system dependencies.

Update of BSFormer. In the l-th layer, we denote the hidden states of the relay nodes as $\tilde{\mathbf{q}}^l_{llm}$ and $\tilde{\mathbf{q}}^l_{role}$, respectively. Correspondingly, the sets of satellite node representations are defined as $\tilde{\mathbf{S}}^l_{llm} = [\tilde{\mathbf{M}}^l; \tilde{\mathbf{e}}^l_{llm}]$ and $\tilde{\mathbf{S}}^l_{role} = [\tilde{\mathbf{R}}^l; \tilde{\mathbf{e}}^l_{role}]$. The input features are initialized using semantic embeddings, specifically $\mathbf{q}$, $[\mathbf{M}; \mathbf{e}]$, and $[\mathbf{R}; \mathbf{e}]$.

Based on multi-head attention (Vaswani et al., 2017), BSFormer updates satellite nodes and relay nodes sequentially. Each satellite node $\tilde{\mathbf{s}}^l_i$ is updated by integrating contextual information from the relay node $\tilde{\mathbf{q}}^{l-1}$, its own previous state $\tilde{\mathbf{s}}^{l-1}_i$, and its initial state $\tilde{\mathbf{s}}^0_i$:

$$\tilde{\mathbf{s}}^l_i = \text{MHAttn}(\tilde{\mathbf{s}}^{l-1}_i, \mathbf{C}^l_i), \mathbf{C}^l_i = [\tilde{\mathbf{s}}^{l-1}_i; \tilde{\mathbf{s}}^0_i; \tilde{\mathbf{q}}^{l-1}] \quad (4)$$

where $\mathbf{C}^l_i$ denotes the context of the i-th satellite node. The relay nodes $\tilde{\mathbf{q}}^l_{llm}$ and $\tilde{\mathbf{q}}^l_{role}$ are updated via their respective satellites and prior states, followed by information exchange via cross-attention:

$$\begin{aligned}
\tilde{\mathbf{q}}^l_{llm} &= \text{MHAttn}(\tilde{\mathbf{q}}^{l-1}_{llm}, [\tilde{\mathbf{q}}^{l-1}_{llm}; \tilde{\mathbf{S}}^l_{llm}]), \\
\tilde{\mathbf{q}}^l_{role} &= \text{MHAttn}(\tilde{\mathbf{q}}^{l-1}_{role}, [\tilde{\mathbf{q}}^{l-1}_{role}; \tilde{\mathbf{S}}^l_{role}]), \\
\tilde{\mathbf{q}}^l_{llm} &= \text{CrossAttn}(\tilde{\mathbf{q}}^l_{llm}, [\tilde{\mathbf{q}}^l_{llm}; \tilde{\mathbf{q}}^0; \tilde{\mathbf{q}}^l_{role}]), \\
\tilde{\mathbf{q}}^l_{role} &= \text{CrossAttn}(\tilde{\mathbf{q}}^l_{role}, [\tilde{\mathbf{q}}^l_{role}; \tilde{\mathbf{q}}^0; \tilde{\mathbf{q}}^l_{llm}]).
\end{aligned} \quad (5)$$

Afterwards, the updated nodes are regularized through a layer normalization. The enriched representations from last layer N_l of BSFormer serve as the foundational input for the autoregressive MAS generation process, and empirical results further validate their effectiveness in achieving cost-efficient MAS design.

3.3 Autoregressive MAS Generation

After modeling the complex correlations, we designed a GRU-based network (Chung et al., 2014) to organize the information into a structured topology. Specifically, at each time step t, the model first encodes the current graph to obtain the node embeddings $\mathbf{H}^t \in R^{(t+1)\times d_g}$ and the graph embedding $\mathbf{g}^t \in R^{d_g}$. Subsequently, we utilize the updated representations $\mathbf{H}^t$ and $\mathbf{g}^t$ to alternately perform node and edge prediction, thereby constructing the MAS step by step in an autoregressive manner.

Method	LLM	Mul.	Hetero.	Flex.	MMLU	GSM8K	MATH	HumanEval	MBPP	Avg.
Vanilla	gpt-4o-mini	✗	✗	✗	77.81	93.17	66.09	85.71	72.20	79.00
	claude-3.5-haiku	✗	✗	✗	67.97	92.16	65.89	86.33	73.40	77.15
	gemini-1.5-flash	✗	✗	✗	80.04	92.67	74.39	82.61	73.00	80.54
	llama-3.1-70b	✗	✗	✗	79.08	92.68	60.31	80.75	68.20	76.20
CoT	gpt-4o-mini	✗	✗	✗	78.43	93.68	67.24	86.69	69.60	79.13
	gemini-1.5-flash	✗	✗	✗	81.35	92.92	74.34	81.37	73.00	80.60
SC(CoT)	gpt-4o-mini	✗	✗	✗	81.05	93.32	66.28	87.58	73.00	80.25
	gemini-1.5-flash	✗	✗	✗	81.66	93.43	74.37	80.75	72.00	80.44
Chain	gpt-4o-mini	✓	✗	✗	82.01	94.40	64.72	85.63	75.40	80.43
	gemini-1.5-flash	✓	✗	✗	83.01	93.13	72.10	82.50	73.20	80.79
Tree	gpt-4o-mini	✓	✗	✗	82.98	93.89	65.11	87.50	75.60	81.02
	gemini-1.5-flash	✓	✗	✗	81.74	94.91	71.36	77.50	73.60	79.82
Complete Graph	gpt-4o-mini	✓	✗	✗	83.06	94.66	67.63	85.00	75.20	81.11
	gemini-1.5-flash	✓	✗	✗	81.35	94.40	68.60	83.75	74.20	80.46
LLM-Debate	gpt-4o-mini	✓	✗	✗	81.04	94.66	64.68	84.38	73.60	79.67
	gemini-1.5-flash	✓	✗	✗	80.40	93.98	72.45	79.38	73.40	79.92
Agentprune	gpt-4o-mini	✓	✗	✔	83.02	94.89	68.45	86.80	75.40	81.71
	gemini-1.5-flash	✓	✗	✔	83.10	93.88	73.54	82.55	75.80	81.77
AFlow	gpt-4o-mini	✓	✗	✔	83.10	92.30	73.35	90.06	82.20	84.20
	gemini-1.5-flash	✓	✗	✔	82.35	94.91	72.70	85.69	76.00	82.33
MaAS	gpt-4o-mini	✓	✗	✓	83.01	92.30	74.45	<u>92.85</u>	82.17	84.96
	gemini-1.5-flash	✓	✗	✓	83.42	93.36	75.18	90.55	82.69	85.04
FrugalGPT	LLM pool	✗	✓	✗	76.24	90.76	67.05	87.31	74.40	79.15
RouterDC	LLM Pool	✗	✓	✗	82.01	93.68	73.46	87.75	75.20	82.42
MasRouter	LLM Pool	✓	✓	✔	<u>84.25</u>	<u>95.45</u>	<u>75.42</u>	90.62	<u>84.00</u>	<u>85.93</u>
Hetero-Designer	LLM Pool	✓	✓	✓	**87.60**	**95.92**	**79.52**	**93.80**	**84.60**	**88.29**

Table 1: Performance comparison against baseline methods. The best results are shown in **bold**, and the second-best results is <u>underlined</u>. "Mul." and "Hetero." indicate support for multi-agent and heterogeneous LLMs settings. "Flex." denotes flexible topology design. ✗, ✔and ✓signifies no/partial/full support in these aspects.

DAG Graph Encoder. The encoder input $\mathbf{x}_{i,j}^t$ is synthesized by aggregating information from LLM and role of v^t via $\text{MLP}([\tilde{\mathbf{m}}_i^{N_l}; \tilde{\mathbf{r}}_j^{N_l}])$. At $t = 0$, we employ $\text{MLP}([\tilde{\mathbf{q}}_{llm}^{N_l}; \tilde{\mathbf{q}}_{role}^{N_l}])$ as the initial graph input $\mathbf{x}^0$, thereby enabling query-adaptive MAS design. Additionally, to enhance topological awareness of DAG, we concatenate positional encodings to each input feature $\hat{\mathbf{x}}^t = [\mathbf{x}^t; \mathbf{PE}(v^t)]$.

Following the DAG modeling approach established by Zhang et al. (2019), node hidden states are updated according to their topological ordering:

$$\begin{aligned} \mathbf{h}^t &= \mathcal{U}(\hat{\mathbf{x}}^t, \mathbf{h}_{in}^t) \\ \mathbf{h}_{in}^t &= \mathcal{A}(\{\mathbf{h}^k : v^k \in \mathcal{N}_{in}(v^t)\}) \end{aligned} \tag{6}$$

where $\mathcal{U}$ and $\mathcal{A}$ denote the update and aggregation functions. Specifically, we employ a GRU for node updates, and $\mathcal{A}$ is a gated sum:

$$\mathbf{h}_{in}^t = \sum_{v^k \in \mathcal{N}_{in}(v^t)} \varphi(\mathbf{h}^k) \odot \sigma(\mathbf{h}^k) \tag{7}$$

where φ is a mapping network and σ is a gating network. Once all node hidden states are sequentially updated, a readout function ϕ aggregates these representations into a global graph embedding $\mathbf{g}^t$.

Node Prediction. We perform joint prediction for both LLM and role. This manner captures the inherent coupling of specific LLM-role pairs. Concretely, we use an extensible metric learning-based module for node generation, the score $s_{i,j}^t$ of a LLM-role pair $(m_i \& r_j)$ is acquired by a dot product operation: $\text{MLP}_g(\mathbf{g}^t) \cdot \text{MLP}_{node}([\tilde{\mathbf{m}}_i^{N_l}; \tilde{\mathbf{r}}_j^{N_l}])$. Similarly, $s_{end}^t = \text{MLP}_g(\mathbf{g}^t) \cdot \text{MLP}_{node}([\tilde{\mathbf{e}}_{llm}^{N_l}; \tilde{\mathbf{e}}_{role}^{N_l}])$. Finally, we can obtain the predicted probability:

$$P(v^t | \mathcal{G}_{<t}, \mathcal{Q}, \mathbb{M}, \mathbb{R}) = \text{softmax}(\{s_{i,j}^t\} \cup \{s_{end}^t\}) \tag{8}$$

Edge Prediction. Once agent node v^t is chosen, the edge generator determines its incoming connections from existing agents $v^1, ..., v^{t-1}$ through:

$$s_{edge}^{i,t} = \text{MLP}_{pre}([\mathbf{h}^i; \mathbf{g}^t]) \cdot \text{MLP}_{cur}([\tilde{\mathbf{m}}_{cur}^{N_l}; \tilde{\mathbf{r}}_{cur}^{N_l}]) \tag{9}$$

We can then obtain the probability of edge $e_{i,t}$ by:

$$P(e_{i,t} = 1 | v^t, \mathcal{G}_{<t}, \mathcal{Q}) = \text{Sigmoid}(s_{edge}^{i,t}) \tag{10}$$

3.4 Optimization

Cold Start. We collect a subset $\mathcal{D}_{\text{cold}}$ containing successful instances for supervised training. This

stage aims to (i) train the model to generate valid collaboration topologies, and (ii) characterize the initial cost and capability distributions of LLMs and roles. This cold-start training equips the model with a foundational capacity for topology generation, enabling the next RL phase to focus on exploring and optimizing high-value strategic designs. Specifically, for each query, we randomly sample multiple candidate topologies and retain only those instances that successfully complete the task.

Reinforcement Learning. Subsequently, we conduct reinforcement learning on $\mathcal{D} \setminus \mathcal{D}_{\text{cold}}$, optimizing the network via the objective in Eq.1 to discover superior strategies. To account for varying query complexities and resource costs, we perform K independent samplings per query to facilitate robust exploration. We employ an empirical Bayes Monte Carlo procedure (Carlin and Louis, 2000; Zhang et al., 2025c) to estimate the gradient loss:

$$\nabla \mathcal{L} \approx \frac{1}{K} \sum_{(\mathcal{Q},a)\in\mathcal{D}_{RL}} \sum_{k=1}^{K} \left[A_k \nabla_{\boldsymbol{\pi}} P(\mathcal{G}_k|\mathcal{Q},\mathbb{M},\mathbb{R}) \right],$$
$$A_k = \frac{p(a|\mathcal{Q},\mathcal{G}_k)}{\sum_i p(a|\mathcal{Q},\mathcal{G}_i)} - \lambda \cdot \frac{C(\mathcal{G}_k;\mathcal{Q})}{\sum_i C(\mathcal{G}_i;\mathcal{Q})}, \tag{11}$$

where A_k denotes the cost-aware importance weights of the MAS, $p(a|\mathcal{Q},\mathcal{G}_k) \in \{0,1\}$ is the task success indicator for $\mathcal{G}_k$. In addition to the current samples $\mathcal{D}_{\text{cur}}$, we sample several strategies from historical trajectories $\mathcal{D}_{\text{replay}}$ and $\mathcal{D}_{\text{cold}}$. The resulting training pool $\mathcal{D}_{\text{RL}}$ facilitates efficient experience replay and robust exploration.

Extended technical details on model architectures, training and inference specifics and the algorithmic workflow, are provided in Appendix B.

4 Experiments

4.1 Experimental Setup

Benchmarks & Metric. Following Yue et al. (2025), we evaluated model on three categories of datasets: ■ **General Reasoning**: MMLU (Hendrycks et al., 2021a); ■ **Mathematical Reasoning**: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b); ■ **Code**: HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021). We use Accuracy to evaluate MMLU, GSM8K, and MATH, and Pass@1 to evaluate MBPP and HumanEval.

Baselines. We compare our method with (i) single-agent approaches, including **COT** (Wei et al., 2022), **Self-Consistency** (Wang et al., 2023); (ii) fixed topologies including **Chain**, **Tree**, and **Complete Graph** (Qian et al., 2025), **LLM-Debate** (Du et al., 2025); (iii) dynamic MAS like **AgentPrune** (Zhang et al., 2025d), **AFlow** (Zhang et al., 2025g) and **MaAS** (Zhang et al., 2025c); (iv) LLM routers, **FrugalGPT**(Chen et al., 2023), **RouterDC** (Chen et al., 2024) and **MasRouter** (Yue et al., 2025).

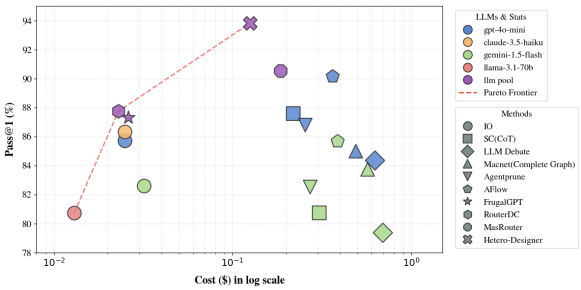


Figure 3: Comparison of performance and inference cost on Humaneval. Different shapes of the scatter points represent various types of baselines, and different colors of the points indicate different LLM backbones.

LLM Backbones. For fair comparison, we adopt the same LLM backbones with varying sizes and capacities as Yue et al. (2025), including: `gpt-4o-mini`, `claude-3.5-haiku`, `gemini-1.5-Flash` and `llama-3.1-70b`. Furthermore, we selected a series of cutting-edge models along a chronological timeline to validate the inductive capabilities of **`Hetero-Designer`**, including: `DeepSeek-V3` (2024.12), `gpt-5-mini` (2025.8), `claude-4.5-Haiku` (2025.10), and `gemini-3-Flash` (2025.12)

Implementation Details. We set the learning rate $lr_{cold} = 1e-4$ for cold start and $lr_{RL} = 1e-5$ for reinforcement learning. The temperature $\tau = 1$. To mitigate the structural imbalance, we employ a weight $\alpha = 0.7$ to scale the log-probability, assigning α to nodes and $1-\alpha$ to edges. The cost penalty $\lambda \in \{5e-2, 1e-1\}$. Both our model and the MAS baselines employ a maximum agent count of $\gamma = 6$. We use `all-MiniLM-L6-v2` (Wang et al., 2020) as the semantic encoder. The semantic and graph embedding dim set to $d_i = 128$ and $d_g = 256$, respectively. The layer num $N_l = 4$. Following Yue et al. (2025), our role pool features 26 distinct roles, ranging from programmers with compilers to researchers leveraging Wikipedia for information retrieval (details in Appendix E).

4.2 Main Results

In this section, we evaluate **`Hetero-Designer`** against multiple baselines across five benchmarks.

Dataset	GSM8K		MATH	
Metric	Accuracy (%)	Cost ($)	Accuracy (%)	Cost ($)
Hetero-Designer	95.92	0.68	79.52	0.73
***w/o* q**	94.98	1.10	64.34	0.68
***w/o* LLM Pool**	93.27	0.48	65.78	0.59
***w/o* BSFormer**	94.31	0.57	70.12	0.49
***w/o* Cold Start**	93.65	1.04	72.77	0.76
***w/o* $C(\cdot)$**	95.73	1.21	80.96	0.89

Table 2: Ablation results of **Hetero-Designer** on GSM8K and MATH.

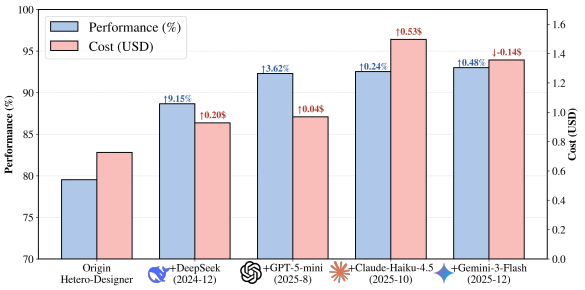


Figure 4: Evolution of performance and cost on MATH benchmark as models are added over time.

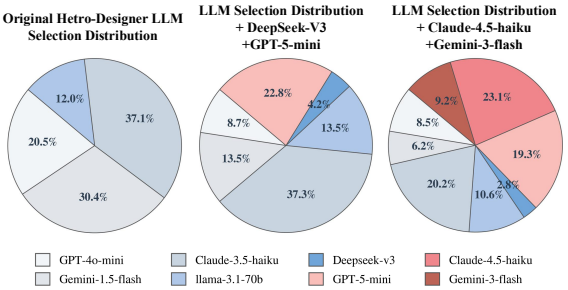


Figure 5: The selected LLM distribution of **Hetero-Designer** on MATH benchmark.

We verify our methods is:

High-performing. The experimental results in Table 1 demonstrates that **Hetero-Designer** excels at constructing an effective Hetero-MAS. Specifically, (i) **Hetero-Designer** achieves the best performance across all five benchmarks, consistently outperforming a wide range of baselines. The superior performance demonstrates the effectiveness of Hetero-MAS designed by **Hetero-Designer**. (ii) Compared to the strongest automated baseline, MaAS, **Hetero-Designer** achieves an average improvement of 3.29%. This markedly outperforms homogeneous systems, demonstrating that the efficient integration of heterogeneous LLMs leads to superior collective intelligence. (iii) Compared to router-based methods, the success of **Hetero-Designer** suggests that MAS benefits more from query-adaptive topology synthesis rather than relying on static, predefined structures.

Token-economical. **Hetero-Designer** leverages hetero-LLMs to construct adaptive topologies, thereby mitigating over-engineering and reducing operational costs. Consequently, it achieves the best cost-performance trade-off on the HumanEval Pareto front (Figure 3). Notably, in Math benchmarks, our model exhibits a notable decline in expenditures (from 1.59$ to 0.68$ on GSM8K and 3.59$ to 0.73$ on MATH, compared to MasRouter), alongside a 0.47%–4.1% accuracy improvement. This underscores **Hetero-Designer**'s capacity to achieve superior intelligence with significantly higher resource efficiency.

4.3 Ablation Study

We conduct an ablation study on the key modules of **Hetero-Designer**: (i) ***w/o* q**, the query embeddings are set to zero vectors (Li et al., 2026). (ii) ***w/o* LLM Pool**, which replaces the Heterogeneous LLMs Pool with the same backbone (`gpt-4o-mini`). (iii) ***w/o* BSFormer**, which uses the raw semantic embeddings without BSFormer. (iv) ***w/o* Cold Start**, which removes the cold start training process. (v) ***w/o*** $C(\cdot)$, which removes the cost penalty in Eq.1. As shown in Table 2, removing the query information, the model fails to adaptively design MAS based on task difficulty, leading to suboptimal performance and inflated costs. Significant performance degradation is observed when using homogeneous LLM, underscoring its inherent limitations. The exclusion of BSFormer precludes effective modeling of intricate connections, confirming that raw semantic features alone are inadequate for synthesizing high-performance MAS topologies. Furthermore, the absence of a cold-start process hinders effective exploration of superior policies during the reinforcement learning phase. Meanwhile, removing the cost penalty results in uneconomical designs with inflated costs.

4.4 Inductive & Extensible Ability Analysis

In this section, we validate the inductive and extensible capabilities of **Hetero-Designer**. Specifically, we demonstrate that our model consistently generalizes to unseen LLMs and roles without the need for intensive re-training.

Generalization to Unseen LLMs. Figure 4 illustrates the evolution of system performance and cost on MATH as additional LLMs are progressively

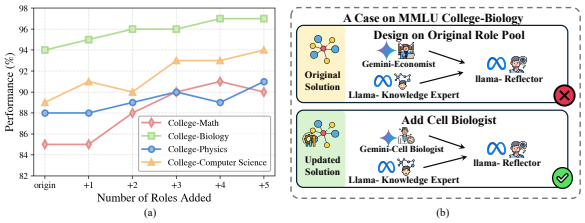


Figure 6: (a) Performance evolution on MMLU subsets with the incremental addition of new roles. (b) A design case of expanded agent role pool on College Biology.

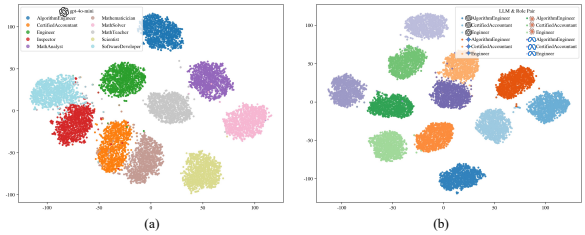


Figure 7: (a) Embedding of different roles for the same LLM. (b) Embedding of different LLM-role pair.

incorporated into the LLM pool. Correspondingly, Figure 5 presents the selection distribution across available LLMs. With the continuous integration of new models, the system's performance exhibits a steady increase, rising from 79.52% to 93.01%. Furthermore, the system does not experience a cost explosion, instead, it maintains a reasonable equilibrium and even exhibits a downward trend with the incorporation of `Gemini-3-Flash`. The distribution reveals that new models are effectively utilized, with legacy versions exhibiting a clear substitution trend (*e.g.*, `GPT-5-mini` accounts for 19.3%, whereas `GPT-4o-mini` decreases to 8.5%).

Extensibility to Novel Roles. Figure 6 illustrates the performance across MMLU subsets as new roles are progressively integrated, alongside a case of designed MAS on expanded roles. Specifically, we generated diverse domain-specific roles for each subset (*e.g.* Cell Biologist for Biology, Programming Expert for Computer Science). Increasing role diversity boosts MAS performance by 6% in Math and 5% in Computer Science, confirming that **`Hetero-Designer`** successfully capitalizes on specialized roles to strengthen system synergy. The case study demonstrates that the system effectively performs role substitution and seamlessly integrates these roles into the original topological design, thereby yielding superior responses.

4.5 Representation Analysis

To verify whether **`Hetero-Designer`** captures the fine-grained features of LLM-role pairs and forms discriminative representations, we employ t-SNE (Maaten and Hinton, 2008) in Figure 7 to visualize the learned embeddings by BSFormer of LLM-role pairs of different queries in MATH. In Figure 7(a), the representations of various roles within `gpt-4o-mini` retain a clear separation, suggesting that the model effectively captures and distinguishes the unique functional characteristics of each role. Figure 7(b) further demonstrates a distinct separation in the representations space of various LLM-role pairs, suggesting that the embeddings generated by BSFormer effectively capture the implicit associations between LLMs and roles, thereby facilitating subsequent MAS design.

4.6 Case Study & sensitivity Analysis

In Appendix C.1, we showcase various Hetero-MAS examples constructed by **`Hetero-Designer`** across five datasets, illustrating how it adaptively tailors MAS architectures to tasks of varying difficulty. Furthermore, we provide a detailed sensitivity analysis of three core parameters—the number of maximum agents γ, the cost penalty λ, and the sampling count K in Appendix C.2.

5 Related Work

Multi-Agent System Design. Contemporary LLM-based MAS has evolved through three stages: (i) **Static Topology.** Initial approaches adopted static, manually crafted architectures, from chain (Wei et al., 2022; Hong et al., 2024), tree (Yao et al., 2023; Wu et al., 2024), complete graph (Qian et al., 2025), sparse graph (Li et al., 2024; Zhang et al., 2026b), LLM-Debate (Du et al., 2024) to optimizable graph (Zhuge et al., 2024). (ii) **Domain-Specific Design.** Subsequently, research shifted toward automated design universal topologies for specific domain, including: edge pruning (Zhang et al., 2025d), joint node-and-edge pruning (Wang et al., 2025b; Zhang et al., 2026a), and search-based paradigms (Zhang et al., 2025g; Hu et al., 2025a). (iii) **Query-specific Design.** Recently, the focus increasingly evolved to the query-specific MAS design, such as GNN-based network (Zhang et al., 2025e), agent supernet search (Zhang et al., 2025c) and autoregressive generation (Wang et al., 2025a; Li et al., 2026), provide unparalleled flexibility and results. Inspired by this, we build MAS in autoregressive generation manner.

Heterogeneous LLM Collaboration. However,

contemporary MAS design approaches remain LLM-homogeneous, failing to effectively organize heterogeneous LLMs-based agents. Ye et al. (2025) highlights promising prospects of hetero-MAS through a comprehensive analysis. Despite progress in routing and assignment strategies (Yue et al., 2025; Ye et al., 2025; Zhang et al., 2025a; Hegazy, 2025), these methods fail to achieve fully topological flexibility, as they dependent on predefined topology rather than automated design. In light of this limitation, we define the ADHM framework for the first time and identify its core challenge: modeling the intricate, implicit correlations between queries, LLMs, agent roles, and topologies. Toward this challenging goal, we propose **Hetero-Designer**, a novel pipeline designed for the effective construction of hetero-MAS.

6 Conclusion

In this paper, we for the first time define the concept of **Automated Design of Heterogeneous-LLMs-based MAS (ADHM)**, which aims to automatically design the MAS topology alongside the assignment of agent roles with heterogeneous LLM backbones to suit various queries. Toward this challenging goal, we propose **Hetero-Designer**, a novel pipeline that captures complex interactions among query, LLMs and roles via a Binary-Star Transformer and construct Hetero-MAS through an autoregressive graph generation process. Extensive experimental results demonstrate that **Hetero-Designer** is: (i) high-performing across various benchmarks, (ii) cost-effective in reducing overhead, and (iii) highly extensible to previously unseen models and roles.

Limitations

We argue that the main limitations of our work are mainly twofold:

(i) Fixed Agent Roles Selection. Following existing literature (Zhang et al., 2025d; Wang et al., 2025b; Li et al., 2026; Zhang et al., 2025e), our study assumes a set of predefined agent roles. However, in complex real-world scenarios, agent roles may need to be dynamically generated and optimized to meet evolving task requirements. While our framework currently lacks the capability for autonomous role discovery, our experiments demonstrate that **Hetero-Designer** possesses robust role-modeling and extensibility capabilities. This foundation paves the way for integrating our heterogeneous MAS design with role-optimization techniques proposed by Wang et al. (2024); Yuan et al. (2025), thereby further enhancing the openness and adaptability of the system.

(ii) Lack of Collaborative Memory Accumulation. The collaboration among heterogeneous LLMs in our framework does not explicitly model the accumulation and evolution of long-term memory (Hu et al., 2025b; Wei et al., 2025; Zhang et al., 2025b; Fu et al., 2026). While **Hetero-Designer** focuses on optimizing collaboration topologies within a single task, it does not yet account for experience retention or capability evolution across multiple tasks. This limitation hinders the system from developing continuously refined coordination patterns. Future work could incorporate shared memory architectures or evolutionary parameter updates to support the collaborative co-evolution of multi-agents during long-term interactions.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.U24A2033, No.62276264, and No.62276095).

References

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

Bradley P Carlin and Thomas A Louis. 2000. Empirical bayes: Past, present and future. *Journal of the American Statistical Association*, 95(452):1286–1289.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. 2024. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Advances in neural information processing systems*.

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

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning*, pages 11733–11763. PMLR.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2025. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*.

Muxin Fu, Xiangyuan Xue, Yafu Li, Zefeng He, Siyuan Huang, Xiaoye Qu, Yu Cheng, and Yang Yang. 2026. Latentmem: Customizing latent memory for multi-agent systems. *Preprint*, arXiv:2602.03036.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1315–1325, Minneapolis, Minnesota. Association for Computational Linguistics.

Mahmood Hegazy. 2025. Diversity of thought elicits stronger reasoning capabilities in multi-agent debate frameworks. *International Journal of Computer Science and Mobile Applications*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

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

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, and 28 others. 2025b. Memory in the age of ai agents. *Preprint*, arXiv:2512.13564.

Junkai Li, Yunghwei Lai, Weitao Li, Jingyi Ren, Meng Zhang, Xinhui Kang, Siyu Wang, Peng Li, Ya-Qin Zhang, Weizhi Ma, and Yang Liu. 2025. Agent hospital: A simulacrum of hospital with evolvable medical agents. *Preprint*, arXiv:2405.02957.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. 2026. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(28):23142–23150.

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

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

Yixin Liu, Guibin Zhang, Kun Wang, Shiyuan Li, Shirui Pan, and Bo An. 2026. Graph-Augmented Large Language Model Agents: Current Progress and Future Prospects . *IEEE Intelligent Systems*, 41(02):45–55.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Yohei Nakajima. 2023. Babyagi. *GitHub repository*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA. Association for Computing Machinery.

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

Toran Bruce Richards. 2023. Auto-gpt: An autonomous gpt-4 experiment.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5977–6043, Suzhou, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*.

Song Wang, Zhen Tan, Zihan Chen, Shuang Zhou, Tianlong Chen, and Jundong Li. 2025a. AnyMAC: Cascading flexible multi-agent collaboration via next-agent prediction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11555–11567, Suzhou, China. Association for Computational Linguistics.

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

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

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2024. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 257–279, Mexico City, Mexico. Association for Computational Linguistics.

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

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

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. 2025. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

Jiaxi Yang, Mengqi Zhang, Yiqiao Jin, Hao Chen, Qingsong Wen, Lu Lin, Yi He, Weijie Xu, James Evans, and Jindong Wang. 2025. Topological structure learning should be a research priority for llm-based multi-agent systems. *Preprint*, arXiv:2505.22467.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. 2025. X-mas: Towards building multi-agent systems with heterogeneous llms. *Preprint*, arXiv:2505.16997.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. 2025. EvoAgent: Towards automatic multi-agent generation via evolutionary algorithms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6192–6217, Albuquerque, New Mexico. Association for Computational Linguistics.

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

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

Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng YAN. 2025b. G-memory: Tracing hierarchical memory for multi-agent systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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

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

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2025e. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *ICLR 2025 Workshop on Foundation Models in the Wild*.

Hangfan Zhang, Zhiyao Cui, Jianhao Chen, Xinrun Wang, Qiaosheng Zhang, Zhen Wang, Dinghao Wu, and Shuyue Hu. 2025f. Stop overvaluing multi-agent debate – we must rethink evaluation and embrace model heterogeneity. *Preprint*, arXiv:2502.08788.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, and 1 others. 2025g. Aflow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024. CodeAgent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13643–13658, Bangkok, Thailand. Association for Computational Linguistics.

Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. 2019. D-vae: A variational autoencoder for directed acyclic graphs. *Advances in neural information processing systems*.

Ruijia Zhang, Xinyan Zhao, Ruixiang Wang, Sigen Chen, Guibin Zhang, An Zhang, Kun Wang, and Qingsong Wen. 2026a. Safesieve: From heuristics to experience in progressive pruning for llm-based multi-agent communication. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(35):29892–29900.

Zhiheng Zhang, Yuanzhe Zhang, Daojian Zeng, Kang Liu, and Jun Zhao. 2026b. Beware of the woozle effect: Exploring and mitigating hallucination propagation in multi-agent debate. *IEEE Transactions on Audio, Speech and Language Processing*, 34:2021–2032.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 62743–62767. PMLR.

A Notations

We conclude the commonly used notations in Table 3 for reference.

B Model supplementary details

In this section, we elaborate the extended technical details of model architectures, the specifics of training and inference, and the formal algorithmic workflow.

B.1 Multi-Head Attention & Cross Attention

Multi-Head Attention. The update process of BSFormer is built upon the attention mechanism (Vaswani et al., 2017). Specifically, given an input sequence $\mathbf{Z} \in \mathbb{R}^{k \times d_{mha}}$, where k and d_{mha} denote the sequence length and hidden dimension, a query vector $\mathbf{q}_{mha} \in \mathbb{R}^{1 \times d}$ is used to extract relevant features via scaled dot-product attention:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathsf{T}}}{\sqrt{d_{mha}}}\right)V \quad (12)$$

where $[Q; K; V] = [\mathbf{q}_{mha}W_q; \mathbf{Z}W_k; \mathbf{Z}W_v]$, and W_q, W_k, W_v denote learnable parameters.

The multi-head attention layer generalizes the scaled dot-product attention by executing h attention operations in parallel. The resulting outputs are subsequently concatenated and linearly projected to aggregate the multi-faceted information:

$$\begin{aligned} &\text{MHAttn}(\mathbf{q}_{mha}, \mathbf{Z}) = [\text{head}_1, ..., \text{head}_h]W_o \\ &\text{head}_i = \text{Attn}(Q_i, K_i, V_i), i \in [1, h], \end{aligned} \quad (13)$$

Notation	Definition
$\mathbb{M} = \{m_1, m_2, \ldots, m_{N_m}\}$	Pool of available LLM backbones, where N_m is the number of LLMs in $\mathbb{M}$
$\mathbb{R} = \{r_1, r_2, \ldots, r_{N_r}\}$	Set of predefined agent roles, where N_r is the number of roles in $\mathbb{R}$
$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\},\ \mathcal{V} = \{v_1, v_2, \ldots, v_{N_g}\}$	A DAG representation of the MAS, where v_i is an agent and N_g is the number of agents in $\mathcal{G}$
v_{end}	Termination token
$\mathcal{N}_{in}(v_i) = \{v_j \mid (v_j, v_i) \in \mathcal{E}\}$	in-neighbors of agent v_i
$a = \text{Aggregate}\{o_i \mid v_i \in \mathcal{V}\}$	a is the output of $\mathcal{G}$, which aggregates all agent output o_i
$\mathcal{Q}$	Input query to the multi-agent system
$U(\mathcal{G}; \mathcal{Q}, \hat{a})$	Utility function measuring MAS performance, $\hat{a}$ is the oracle answer corresponding to $\mathcal{Q}$
$C(\mathcal{G}; \mathcal{Q})$	Cost function quantifying token expenditure, computed based on the API token price
$P(\mathcal{G} \mid \mathcal{Q}, \mathbb{M}, \mathbb{R})$	The probability of MAS $\mathcal{G}$
$\mathbf{q}$,$\mathbf{m}_i$,$\mathbf{r}_j$,$\mathbf{e}$	Semantic embedding of query, LLM, role and end token
$\tilde{\mathbf{q}}^l_{llm}, \tilde{\mathbf{q}}^l_{role}$	Relay node representation of BSFormer at l layer
$\tilde{\mathbf{S}}^l_{llm}, \tilde{\mathbf{S}}^l_{role}$	Satellite node representation of BSFormer at l layer
N_l	The number of BSFormer layer
$\mathbf{x}^t_{i,j} = \text{MLP}([\tilde{\mathbf{m}}_i^{N_l}; \tilde{\mathbf{r}}_j^{N_l}])$	graph encoder input representation of t-th agent
$\hat{\mathbf{x}} = [\mathbf{x}^t; \mathbf{PE}(v^t)]$	graph encoder input representation concat with position encodings of t-th agent $\mathbf{H}^t, \mathbf{g}^t$
node and graph embeddings of $\mathcal{G}_{<t}$	
$\mathcal{U}(\cdot), \mathcal{A}(\cdot)$	Update function and Aggregation function of graph encoder
$\varphi(\cdot), \sigma(\cdot), \phi(\cdot)$	mapping network, gating network and readout function of graph encoder
$s^t_{i,j}, s^t_{end}$	node prediction score of llm-role pair m_i&r_j and end token v_{end}
$s^{i,t}_{edge}$	edge prediction score of edge $i \to t$
$\mathcal{D}, \mathcal{D}_{cold}, \mathcal{D}_{RL}, D_{replay}$	training dataset, cold start dataset, reinforcement learning dataset and replay dataset
γ	Maximum number of agents
$f_s(\cdot)$,$f_{BS}(\cdot), f_{DAG}(\cdot)$	Semantic encoder, Binary-Star Transformer and DAG encoder

Table 3: The notations that are commonly used throughout the manuscript.

where $[\mathbf{Q}_i; \mathbf{K}_i; \mathbf{V}_i]$ represent the i-th head partitioned from $[\mathbf{Q}; \mathbf{K}; \mathbf{V}]$ with dimension d/h, and $\mathbf{W}_o$ is the learnable output projection matrix. we set all the number of attention heads to $h = 4$.

Cross Attention. To enable effective information exchange between different star Transformers, we introduce a cross-attention mechanism that allows one Transformer to attend to the representations produced by another. Specifically, given a query vector $\mathbf{q}_{ca} \in R^{1 \times d_{ca}}$ and a source representation $\mathbf{Z}^s \in R^{k_s \times d_{ca}}$ from another sequence (*e.g.*, query from the relay node of Role-Former and source representation contains information from LLM-Former), cross attention is computed by projecting queries from $\mathbf{q}_{ca}$ and keys and values from $\mathbf{Z}^s$:

$$\text{CrossAttn}(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathsf{T}}}{\sqrt{d_{ca}}}\right)V, \tag{14}$$

$[Q; K; V] = [\mathbf{q}_{ca}W_q; \mathbf{Z}^sW_k; \mathbf{Z}^sW_v]$. Similarly, we employ multi-head attention to capture multi-faceted and fine-grained features across different representation subspaces.

B.2 Supplementary details of the DAG encoder

DAG Positional Encoding. To encode the topological positions of nodes within a DAG, we implement a structure-aware positional encoding (PE) scheme designed to capture both hierarchical depth and intra-layer ordering. Given that DAG nodes are only partially ordered, traditional sequential PE methods are inherently inapplicable. Our approach addresses this by assigning each node a dual-component embedding, derived from its topological depth and its relative rank within the corresponding level. Given a DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we first compute a topological ordering of the nodes. During this process, the depth of each node $v_i \in \mathcal{V}$ is defined as the length of the longest path from any source node to v:

$$\text{depth}(v_i) = \max_{v_j \in \mathcal{N}_{in}(v_i)} \big(\text{depth}(v_j) + 1\big), \tag{15}$$

We let the first node as the initial vertex v_0. For any node v_i with an empty in-degree set $\mathcal{N}_{in}(v_i) = \emptyset$, we establish a directed edge from v_0 to v_i, where v_0 encapsulates the query-LLM-role information to provide necessary task context. After computing

Algorithm 1: MAS Sample & Inference & Forward pass Workflow of **Hetero-Designer**

Input : Task Query $\mathcal{Q}$, Semantic encoder $f_s(\cdot)$, Binary-star Transformer $f_{BS}(\cdot)$, DAG encoder $f_{DAG}(\cdot)$, LLM pool $\mathbb{M}$, Role Pool $\mathbb{R}$, termination token v_{end},(Option: Forward pass need a generated MAS $\hat{\mathcal{G}}$ as Input)

// Input Representation

Encode $\mathcal{Q}, \mathbb{M}, \mathbb{R}, v_{end}$ through f_s into $\mathbf{q}, \mathbf{M}, \mathbf{R}, \mathbf{e}$ // Section 3.1

// Binary-Star Transfomer Encoding

// Feature Initialization

Initialize $\tilde{\mathbf{q}}^0_{llm}, \tilde{\mathbf{q}}^0_{role} \leftarrow \mathbf{q}, \mathbf{q}$

Initialize $\tilde{\mathbf{S}}^0_{llm}, \tilde{\mathbf{S}}^0_{role} \leftarrow [\mathbf{M}; \mathbf{e}], [\mathbf{R}; \mathbf{e}]$

for $l = 1$ to N_l **do**

  // Update the satellite nodes of LLM-former and Role-former

  for $i = 1$ to $N_m + 1$ **do**

    $\mathbf{C}^l_{llm,i} = [\tilde{\mathbf{s}}^{l-1}_{llm,i}; \tilde{\mathbf{s}}^0_{llm,i}; \tilde{\mathbf{q}}^{l-1}_{llm}]$

    $\tilde{\mathbf{s}}^l_{llm,i} = \text{MHAttn}(\tilde{\mathbf{s}}^{l-1}_{llm,i}, \mathbf{C}^l_{llm,i})$

    $\tilde{\mathbf{s}}^l_{llm,i} = \text{Layernorm}(\text{GeLU}(\tilde{\mathbf{s}}^l_{llm,i}))$

  for $j = 1$ to $N_r + 1$ **do**

    $\mathbf{C}^l_{role,i} = [\tilde{\mathbf{s}}^{l-1}_{role,i}; \tilde{\mathbf{s}}^0_{role,i}; \tilde{\mathbf{q}}^{l-1}_{role}]$

    $\tilde{\mathbf{s}}^l_{role,i} = \text{MHAttn}(\tilde{\mathbf{s}}^{l-1}_{role,i}, \mathbf{C}^l_{role,i})$

    $\tilde{\mathbf{s}}^l_{role,i} = \text{Layernorm}(\text{GeLU}(\tilde{\mathbf{s}}^l_{role,i}))$

  // Update the relay node

  $\tilde{\mathbf{q}}^l_{llm} = \text{MHAttn}(\tilde{\mathbf{q}}^{l-1}_{llm}, [\tilde{\mathbf{q}}^{l-1}_{llm}; \tilde{\mathbf{S}}^l_{llm}]), \tilde{\mathbf{q}}^l_{role} = \text{MHAttn}(\tilde{\mathbf{q}}^{l-1}_{role}, [\tilde{\mathbf{q}}^{l-1}_{role}; \tilde{\mathbf{S}}^l_{role}])$

  $\tilde{\mathbf{q}}^l_{llm} = \text{Layernorm}(\text{GeLU}(\tilde{\mathbf{q}}^l_{llm})), \tilde{\mathbf{q}}^l_{role} = \text{Layernorm}(\text{GeLU}(\tilde{\mathbf{q}}^l_{role}))$

  $\tilde{\mathbf{q}}^l_{llm} = \text{CrossAttn}(\tilde{\mathbf{q}}^l_{llm}, [\tilde{\mathbf{q}}^l_{llm}; \tilde{\mathbf{q}}^0; \tilde{\mathbf{q}}^l_{role}]), \tilde{\mathbf{q}}^l_{role} = \text{CrossAttn}(\tilde{\mathbf{q}}^l_{role}, [\tilde{\mathbf{q}}^l_{role}; \tilde{\mathbf{q}}^0; \tilde{\mathbf{q}}^l_{llm}])$

  $\tilde{\mathbf{q}}^l_{llm} = \text{Layernorm}(\text{GeLU}(\tilde{\mathbf{q}}^l_{llm})), \tilde{\mathbf{q}}^l_{role} = \text{Layernorm}(\text{GeLU}(\tilde{\mathbf{q}}^l_{role}))$ // Section 3.2

// Autoregressive MAS Generation

Initialize $\mathcal{V} \leftarrow \{v^0_{start}\}, \mathbf{X^0} \leftarrow \hat{\mathbf{x}}^0, t \leftarrow 1, \log P_G = 0$

while $t \leq \gamma$ **do**

  Graph Encoding: $\mathbf{H^t}, \mathbf{g^t} = f_{DAG}(\mathcal{G}_{<t})$

  // Node Generation

  if Inference **then**

    $v^t = \arg\max[P(v^t|\mathcal{G}_{<t}, \mathcal{Q}, \mathbb{M}, \mathbb{R})] = \arg\max(\text{softmax}(\{s^t_{i,j}\} \cup \{s^t_{end}\}))$

  else if Sample **then**

    Sample $v^t \sim P(v^t|\mathcal{G}_{<t}, \mathcal{Q}, \mathbb{M}, \mathbb{R}) = \text{softmax}(\{s^t_{i,j}\} \cup \{s^t_{end}\})$

  else

    $v^t = \hat{v}^t$ in $\hat{\mathcal{G}}$

  $\log P_G \leftarrow \log P_G + \alpha \cdot \log(P(v^t|\mathcal{G}_{<t}, \mathcal{Q}, \mathbb{M}, \mathbb{R}))$

  if $v^t = v_{end}$ **then**

    break

  else

    $\mathcal{V} \leftarrow \mathcal{V} \cup \{v^t\}, \mathbf{X^t} \leftarrow [\mathbf{X}^{t-1}; \hat{\mathbf{x}}^t]$

    // Edge Generation

    for $i = 1$ to $t - 1$ **do**

      if Inference **then**

        if $P(e_{i,t} = 1|v^t, \mathcal{G}_{<t}, \mathcal{Q}) = \text{Sigmoid}(s^{i,t}_{edge}) \geq 0.5$ **then**

          $e_{i,t} = 1$

        else

          $e_{i,t} = 0$

      else if Sample **then**

        Sample $e_{i,t} \sim P(e_{i,t} = 1|v^t, \mathcal{G}_{<t}, \mathcal{Q})$

      else

        $e_{i,t} = \hat{e}_{i,t}$

      if $e_{i,t} = 1$ **then**

        $\mathcal{V} \leftarrow \mathcal{V} \cup \{e_{i,t} = 1\}, P_G + (1 - \alpha) \cdot \log P(e_{i,t} = 1|v^t, \mathcal{G}_{<t}, \mathcal{Q})$

      else

        $\mathcal{V} \leftarrow \mathcal{V} \cup \{e_{i,t} = 0\}, P_G + (1 - \alpha) \cdot \log(1 - P((e_{i,t} = 1|v^t, \mathcal{G}_{<t}, \mathcal{Q})))$

      // Section 3.3

node depths, we group nodes into layers according to their depth values. For nodes within the same layer, we assign an intra-layer rank based on their prediction order, which serves to break symmetry among nodes at the same depth level.

The final positional encoding for each node v_i is constructed by concatenating two sinusoidal embeddings that encode its depth and intra-layer rank:

$$\mathbf{PE}(v_i) = \left[\mathbf{PE}_{\text{depth}}(v_i); \mathbf{PE}_{\text{rank}}(v_i)\right] \tag{16}$$

where each component occupies $d/2$ dimensions. Specifically, for $k = 0, \ldots, \frac{d}{4} - 1$, the sinusoidal encodings are defined as:

$$\begin{aligned}
\mathbf{PE}^{(2k)}_{\text{depth}}(v) &= \sin\left(\frac{\text{depth}(v)}{10000^{2k/(d/2)}}\right), \\
\mathbf{PE}^{(2k+1)}_{\text{depth}}(v) &= \cos\left(\frac{\text{depth}(v)}{10000^{2k/(d/2)}}\right), \\
\mathbf{PE}^{(2k)}_{\text{rank}}(v) &= \sin\left(\frac{\text{rank}(v)}{10000^{2k/(d/2)}}\right), \\
\mathbf{PE}^{(2k+1)}_{\text{rank}}(v) &= \cos\left(\frac{\text{rank}(v)}{10000^{2k/(d/2)}}\right).
\end{aligned} \tag{17}$$

This simple yet effective positional encoding enhances the topological awareness of the DAG, facilitating the seamless integration of nodal features with the structural information. This encoding allows the model to more distinctly localize specific roles within the topology. For instance, a `Reflector` node typically occupies the terminal position (*i.e.*, the last node of the final layer).

Network details. Following Zhang et al. (2019), we implement the mapping network and gating network using simple linear layers. Specifically, the mapping network $\varphi(\cdot)$ is defined as:

$$\varphi(\mathbf{h}^k) = W_\varphi \mathbf{h}^k \tag{18}$$

and gating network $\sigma(\cdot)$ is defined as:

$$\sigma(\mathbf{h}^k) = \text{sigmoid}(W_\sigma \mathbf{h}^k + b_\sigma) \tag{19}$$

In addition, we employ $\phi(\mathbf{H}^t) = \text{MHAttn}(\mathbf{q}, \mathbf{H}^t)$, to compute a global readout of the entire graph.

B.3 Training Specifics

The DAG probability of Hetero-Designer. While Eq. 2 defines the basic probability of the DAG structure, we here provide a more comprehensive formulation for **`Hetero-Designer`** by incorporating the terminal node v_{end}. The joint probability, augmented by the existence of v_{end}, is defined as:

$$P(\mathcal{G}|\mathcal{Q}, \mathbb{M}, \mathbb{R}) = P(v_{end}|\mathcal{G}, \mathcal{Q}, \mathbb{M}, \mathbb{R}) \times \prod_{i=1}^{|\mathcal{V}|} \underbrace{P(v_i|\mathcal{G}_{<i}, \mathcal{Q}, \mathbb{M}, \mathbb{R})}_{\textit{node probability}} \times \prod_{j=1}^{i-1} \underbrace{P(e_{ji}|v_i, \mathcal{G}_{<i}, \mathcal{Q})}_{\textit{edge probability}} \tag{20}$$

We introduce a balancing factor α to weight node and edge log-probabilities, counteracting the numerical dominance of edges (up to $\frac{n(n-1)}{2}$) over nodes (n) to stabilize training. In addition, we can leverage Teacher Forcing for off-policy probability re-evaluation, which helps maximize data utility and avoid premature sample discarding.

Details of Cold Start Training. We split the training data into two subsets using a ratio of 2 : 3: $\mathcal{D}_{cold}$ for cold-start pre-training and $\mathcal{D}_{RL}$ for reinforcement learning. For $\mathcal{Q}$ in $\mathcal{D}_{cold}$, we collect four successful samples for each query through random sampling. We iterate through the number of agents starting from one, continuing until either the maximum number of agents γ is reached or four samples are collected, with a maximum of four iterations allowed for each specific agent count. The final training objective for the cold-start phase is:

$$\mathcal{L}_{cold} = \alpha \cdot \mathcal{L}_{node} + (1 - \alpha) \cdot \mathcal{L}_{edge} \tag{21}$$

The individual loss terms are defined as:

$$\begin{aligned}
\mathcal{L}_{\text{node}} &= -\sum_{(\mathcal{G},\mathcal{Q},\mathcal{C})\in\mathcal{D}_{cold}} (1 - \lambda'\mathcal{C}) \\
&\quad \sum_{t=1}^{|\mathcal{V}|} \log P(v^t|\mathcal{G}_{<t}, \mathcal{Q}, \mathbb{M}, \mathbb{R}) \\
\mathcal{L}_{\text{edge}} &= -\sum_{(\mathcal{G},\mathcal{Q},\mathcal{C})\in\mathcal{D}_{cold}} (1 - \lambda'\mathcal{C}) \\
&\quad \sum_{t=1}^{|\mathcal{V}|} \sum_{i=1}^{t-1} \log P(e_{i,t}|v^t, \mathcal{G}_{<t}, \mathcal{Q}).
\end{aligned} \tag{22}$$

where $\mathcal{C}$ is the cost of graph $\mathcal{G}$, and $\lambda' \in \{15, 25\}$ is the cost penalty for cold start.

Replay Data. We perform random sampling of trajectories from both $\mathcal{D}_{cold}$ and the historical replay buffer $\mathcal{D}_{replay}$ during reinforcement learning to enhance data utilization. By executing a new forward pass through these trajectories, we derive the action probabilities under the current policy, thereby supporting efficient off-policy updates.

Following the methodology established by Yue et al. (2025), we performed stratified sampling on the MATH to select 519 problems across various

Method	Prompt	Completion	Cost ($)
AFlow	321,813,314	28,083,445	21.75
MasRouter	3,235,288	2,499,530	3.56
Hetero-Designer	2,656,711	1,549,446	2.87

Table 4: Token usage and cost comparison on MATH.

difficulty levels. The sampled data was then partitioned into training and test sets using a 1:4 ratio.

B.4 Inference Specifics

During inference, we employ a greedy decoding strategy. Specifically, the LLM and role of next node is determined by selecting the candidate with the maximum conditional probability $P(v^t|\mathcal{G}_{<t}, \mathcal{Q}, \mathbb{M}, \mathbb{R})$. As for edge generation, an edge $e_{i,t}$ is instantiated if its predicted probability $P(e_{i,t} = 1|v^t, \mathcal{G}_{<t}, \mathcal{Q})$ meets or exceeds the threshold of 0.5.

B.5 Algorithm Workflow

We summarize the overall algorithmic workflow in Algorithm 1, which details the MAS sampling procedure, the inference process, and the forward pass of the **Hetero-Designer**.

C Case Study & sensitivity Analysis

C.1 Case Study

As shown in Figures 9 to 13, we present the customized MAS designed by **Hetero-Designer** across diverse query complexities for the five benchmarks. For straightforward queries, the model exhibits a preference for utilizing cost-effective and well-balanced LLMs alongside specialized agent roles to achieve an efficient solution. Conversely, for more challenging instances, the model tends to architect increasingly sophisticated systems.

Furthermore, we observe that the model's decision-making process is closely aligned with the semantic context of the query. For instance, **Hetero-Designer** incorporates an "Economist" role for economics-related questions in the MMLU benchmark and strategically deploys a "Programmer" to handle MATH problems that are amenable to algorithmic solutions. This behavior demonstrates that **Hetero-Designer** possesses a fine-grained domain awareness, allowing it to dynamically reconfigure both the agent composition and the interaction topology based on the underlying task semantics.

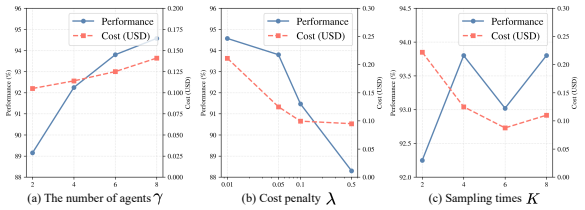


Figure 8: Parameter sensitivity analysis of **Hetero-Designer** on HumanEval. The unit of cost (right) and performance (left) is $ and pass@1 (%), respectively.

C.2 Sensitivity Analysis

We analyze the sensitivity of **Hetero-Designer** to three core parameters: the maximum number of agents γ, the cost penalty coefficient λ in Eq.2, and the sampling count K in Eq.11. The results are presented in Figure 8. While performance scales with the agent count γ, the associated computational cost exhibits only a marginal increase. This efficiency is primarily attributed to the introduction of the termination token v_{end}, which enables dynamic termination and prevents the generation of redundant agent nodes. As the penalization strength λ increases, the model prioritizes structural sparsity and simplicity, leading to a more parsimonious topology at the expense of task performance. Regarding the sample size K, insufficient sampling leads to unstable performance-cost trade-offs, primarily due to restricted exploration and high gradient variance. As K increases, the model more effectively navigates the strategy space, eventually achieving a superior balance.

D Training cost statistics

Table 4 presents the training costs on the MATH benchmark, where we observe that expenditures remain relatively modest. This cost-efficiency is primarily attributed to the early acquisition of economical strategies during the cold-start phase and the improved sample efficiency afforded by the experience replay mechanism. These components allow the model to refine its policy using high-quality historical trajectories, thereby achieving convergence with minimal financial overhead.

E The Module Profile

In this section, we present the profiles of each module. Follow Yue et al. (2025), we use the same role pool and llm pool for fair comparison. Furthermore, we utilize `GPT-4o-mini` to generate a dedicated termination token v_{end}, providing the model with an

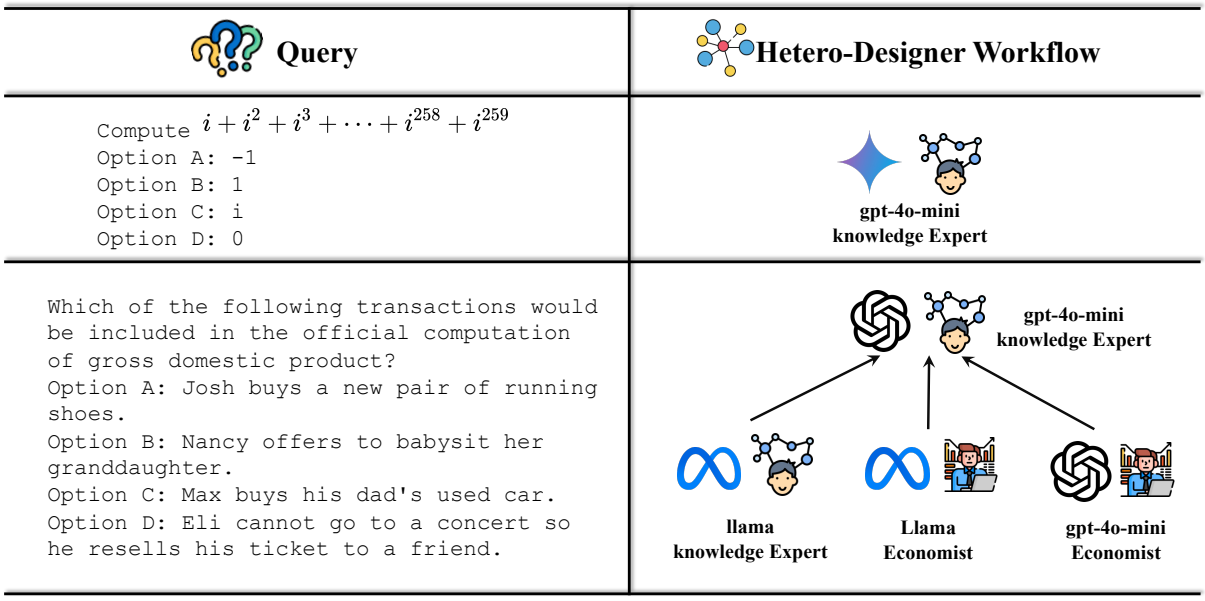


Figure 9: Case study on MMLU dataset.

explicit signal to conclude the generation process. We provide a series of profile examples for each category in Figures 14, 15, 16, 17 and 18.

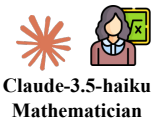
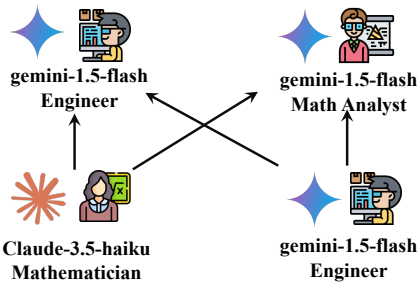
Query	Hetero-Designer Workflow
Adam has $100 and wants to spend it to open a rock stand. He can buy rocks for $5 each and sell them for $7 each. If he invests all his money in the rock stand but only sells 60% of his inventory, how much money does he lose?	
Grandma walks 3 miles every day on her favorite walking trail, which includes 2 miles of walking on the beach and 1 mile of walking on the sidewalk. On the sidewalk, Grandma walks at twice the rate of speed that she does on the beach. If 40 minutes of her walk is spent on the beach, how long does it take for her to complete the entire 3-mile walk, in minutes?'	

Figure 10: Case study on GSM8K dataset.

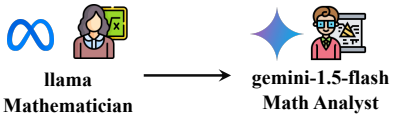
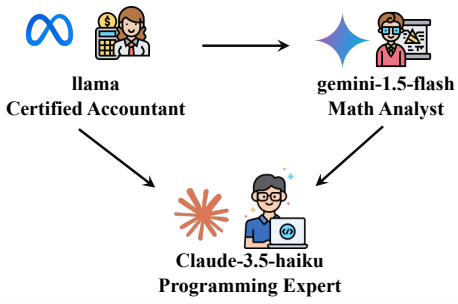
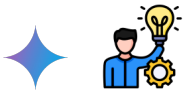
Query	Hetero-Designer Workflow
What is the value of x in the equation $\frac{17^6 - 17^5}{16} = 17^x$	
For any number x, we are told that $x\& = 7 - x$ and $\&x = x - 7$. What is the value of $15\&$?	

Figure 11: Case study on MATH dataset.

Query	Hetero-Designer Workflow

def anti_shuffle(s: str) -> str:
    Write a function that takes a string and returns an ordered version of it.
    Ordered version of string, is a string where all words (separated by space)
    are replaced by a new word where all the characters arranged in
    ascending order based on ascii value.
    Note: You should keep the order of words and blank spaces in the sentence.
    For example:
    >>> anti_shuffle('Hi')
    'Hi'
    >>> anti_shuffle('hello')
    'ehllo'
    >>> anti_shuffle('Hello World!!!')
    'Hello !!!Wdlor'
	
	gemini-1.5-flash
	Plan Solver

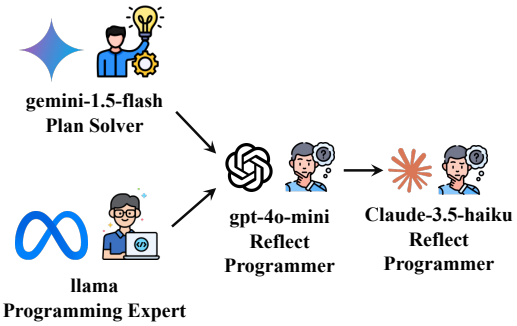
from typing import List
def sort_even(l: List[int]) -> List[int]:
    This function takes a list l and returns a list l' such that
    l' is identical to l in the odd indicies, while its values at the even indicies are equal
    to the values of the even indicies of l, but sorted.
    >>> sort_even([1, 2, 3])
    [1, 2, 3]
    >>> sort_even([5, 6, 3, 4])
    [3, 6, 5, 4]
	
	gemini-1.5-flash
	Plan Solver
	gpt-4o-mini
	Reflect
	Programmer
	Claude-3.5-haiku
	Reflect
	Programmer
	llama
	Programming Expert

Figure 12: Case study on humaneval dataset.

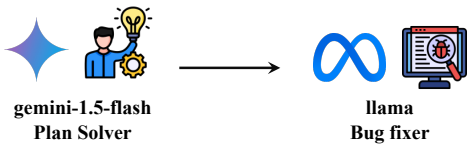
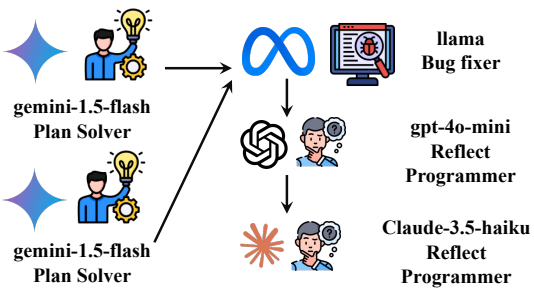
Query	Hetero-Designer Workflow
Task: Write a function to remove all elements from a given list present in another list. Your code should pass these tests: assert remove_elements([1,2,3,4,5,6,7,8,9,10],[2,4,6,8])==[1, 3, 5, 7, 9, 10] assert remove_elements([1, 2, 3, 4, 5, 6, 7, 8, 9, 10],[1, 3, 5, 7])==[2, 4, 6, 8, 9, 10] assert remove_elements([1, 2, 3, 4, 5, 6, 7, 8, 9, 10],[5,7])==[1, 2, 3, 4, 6, 8, 9, 10]	 gemini-1.5-flash Plan Solver llama Bug fixer
Task: Write a python function to check whether one root of the quadratic equation is twice of the other or not. Your code should pass these tests: assert Check_Solution(1,3,2) == "Yes" assert Check_Solution(1,2,3) == "No" assert Check_Solution(1,-5,6) == "No"	 gemini-1.5-flash Plan Solver gemini-1.5-flash Plan Solver llama Bug fixer gpt-4o-mini Reflect Programmer Claude-3.5-haiku Reflect Programmer

Figure 13: Case study on MBPP dataset.

LLM Profile

```
[
  {
    "Name": "gpt-4o-mini",
    "Description": "GPT-4o Mini is a smaller version of the GPT-4o language
     model, designed for faster inference and reduced memory usage. It retains
     the same capabilities as the full-size model, but with fewer parameters.
     The model costs $0.15 per million input tokens and $0.6 per million output
     tokens.
     In General Q&A Benchmark MMLU, GPT-4o-mini achieves an accuracy of 77.8.
     In Reasoning Benchmark GPQA, GPT-4o-mini achieves an accuracy of 40.2.
     In Coding Benchmark HumanEval, GPT-4o-mini achieves an accuracy of 85.7.
     In Math Benchmark MATH, GPT-4o-mini achieves an accuracy of 66.09."
  },
  {
    "Name": "claude-3-5-haiku-20241022",
    "Description": "The new Claude 3.5 Haiku combines rapid response times
     with improved reasoning capabilities, making it ideal for tasks that
     require both speed and intelligence. Claude 3.5 Haiku improves on its
     predecessor and matches the performance of Claude 3 Opus.
     The model costs $0.1 per million input tokens and $0.5 per million output
     tokens.
     In General Q&A Benchmark MMLU, claude-3-5-haiku achieves an accuracy of
     67.9.
    In Reasoning Benchmark GPQA, claude-3-5-haiku achieves an accuracy of 41.6.
     In Coding Benchmark HumanEval, claude-3-5-haiku achieves an accuracy of
     86.3.
     In Math Benchmark MATH, claude-3-5-haiku achieves an accuracy of 65.9."
  },
  {
    "Name": "gemini-1.5-flash-latest",
    "Description": "Gemini 1.5 Flash was purpose-built as our fastest, most
     cost-efficient model yet for high volume tasks, at scale, to address
     developers feedback asking for lower latency and cost.
     The model costs $0.15 per million input tokens and $0.6 per million output
     tokens.
     In General Q&A Benchmark MMLU, gemini-1.5-flash achieves an accuracy of
     80.0.
    In Reasoning Benchmark GPQA, gemini-1.5-flash achieves an accuracy of 39.5.
     In Coding Benchmark HumanEval, gemini-1.5-flash achieves an accuracy of
     82.6.
     In Math Benchmark MATH, gemini-1.5-flash achieves an accuracy of 74.4."
  },
  {
    "Name": "Meta-Llama-3.1-70B-Instruct",
    "Description": "The Meta Llama 3.1 multilingual large language model (LLM)
     is a pretrained and instruction tuned generative model in 70B (text
     in/text out).
     The model costs $0.2 per million input tokens and $0.2 per million output
     tokens.
     In General Q&A Benchmark MMLU, Llama 3.1 achieves an accuracy of 79.1.
     In Reasoning Benchmark GPQA, Llama 3.1 achieves an accuracy of 46.7.
     In Coding Benchmark HumanEval, Llama 3.1 achieves an accuracy of 80.7.
     In Math Benchmark MATH, Llama 3.1 achieves an accuracy of 60.3."
  }
]
```

Figure 14: The LLM profile of main backbones.

Math_role_Profile

```
[
  {
    "Name": "MathAnalyst",
    "MessageAggregation": "Normal",
    "Description": "You are a mathematical analyst. You will be given a math problem ,
analysis and code from other agents. You need to first analyze the problem solving
process , where the variables are represented by letters. Then you substitute the
values into the analysis process to perform calculations and get the results.",
    "OutputFormat": "Calculation",
    "PostProcess": "None",
    "PostDescription": "None",
    "PostOutputFormat": "None"
  },
  {
    "Name": "Inspector",
    "MessageAggregation": "Normal",
    "Description": "You are an Inspector. You will be given a math problem , analysis
and code from other agents. Check whether the logic/calculation of the problem
solving and analysis process is correct(if present). Check whether the code
corresponds to the solution analysis(if present). Give your own solving process step
by step based on hints",
    "OutputFormat": "Answer",
    "PostProcess": "None",
    "PostDescription": "None",
    "PostOutputFormat": "None"
  }
]
```

Figure 15: Example profile of math role.

Coding_role_Profile

```
[
  {
    "Name": "BugFixer",
    "Description": "You are a programming expert. You will be given a function
signature and its docstring by the user. Use a Python code block to write your full
implementation (restate the function signature).",
    "OutputFormat": "CodeCompletion",
    "PostProcess": "PythonInnerTest",
    "PostDescription": "You need to provide modified and improved python code based
on the current code implementation and problems that arise during testing. You can
refer to specific examples.Write your full implementation (restate the function
signature). ",
    "PostOutputFormat": "CodeCompletion"
  },
  {
    "Name": "Test Analyst",
    "MessageAggregation": "PythonInnerTest",
    "Description": "You are a Test Analyst. You will be given a function signature
and its docstring by the user. You need to provide problems in the current code or
solution based on the test data and possible test feedback in the question. You need
to provide additional special use cases , boundary conditions , etc . that should be
paid attention to when writing code. You can point out any potential errors in the
code.Your reply should be more concise.Preferably within fifty words.",
    "OutputFormat": "Text",
    "PostProcess": "None",
    "PostDescription": "You are a programming expert. You will be given a function
signature and its docstring by the user. Give your own answers to problems that arise
in other implementations. Use a Python code block to write your full implementation
(restate the function signature).",
    "PostOutputFormat": "CodeCompletion"
  }
]
```

Figure 16: Example profile of coding role.

Commensense_role_Profile

```
[
  {
    "Name": "Critic",
    "MessageAggregation": "Normal",
    "Description": "You are an excellent critic. Please point out potential issues in
other agent's analysis point by point. Give your critical opinion. Finally give the
final result",
    "OutputFormat": "Answer",
    "PostProcess": "None",
    "PostDescription": "None",
    "PostOutputFormat": "None"
  },
  {
    "Name": "WikiSearcher",
    "MessageAggregation": "Normal",
    "Description": "Please give several key entities that need to be searched in
wikipedia to solve the problem. ",
    "OutputFormat": "Keys",
    "PostProcess": "Wiki",
    "PostDescription": "You are a knowlegable expert in question answering. Please
answer the question based on the explanation of the question keywords obtained from
the wikipedia search.",
    "PostOutputFormat": "Answer"
  }
]
```

Figure 17: Example profile of commensense role.

END Termination_token_Profile

```
[
    {
  "Name": "EndTokenRoleController",
  "Description": "You are responsible for determining whether the multi-agent system
autoregressive generation process should terminate. If the current system can already
provide a correct response for the given query, you trigger an immediate stop to
avoid unnecessary token cost. If the system is able to provide a sufficiently correct
response, you trigger an immediate stop to avoid unnecessary token usage. If the
collaboration among roles remains uncertain or incomplete, you allow further
generation to continue. Your role is to balance correctness and efficiency,
dynamically deciding the optimal stopping point."
    }
]
```

Figure 18: Profile of termination token v_{end}.