

MetaGen: Self-Evolving Roles and Topologies for Multi-Agent LLM Reasoning

Yimeng Wang^{1,*}, Jiaxing Zhao^{1,*}, Hongbin Xie², Hexing Ma¹,
Yuzhen Lei¹, Shuangxue Liu¹, Xuan Song^{1,†}, Zichen Zhang², Haoran Zhang^{2,†}

¹School of Artificial Intelligence, Jilin University

²School of Urban Planning and Design, Peking University

Correspondence: songxuan@jlu.edu.cn, h.zhang@pku.edu.cn

Abstract

Large language models are increasingly deployed as multi-agent systems, where specialized roles communicate and collaborate through structured interactions to solve complex tasks that often exceed the capacity of a single agent. However, most existing systems still rely on a fixed role library and an execution-frozen interaction topology, a rigid design choice that frequently leads to task mismatch, prevents timely adaptation when new evidence emerges during reasoning, and further inflates inference cost. We introduce **MetaGen**, a framework that adapts both the role space and the collaboration topology at inference time, without updating base model weights. MetaGen generates and rewrites query-conditioned role specifications to maintain a controllable dynamic role pool, then instantiates a constrained execution graph around a minimal backbone. During execution, it iteratively updates role prompts and adjusts structural decisions using lightweight feedback signals. Experiments on code generation and multi-step reasoning benchmarks show that MetaGen improves the accuracy and cost tradeoff over strong multi-agent baselines.

1 Introduction

Large language models (LLMs) are rapidly evolving from single-turn conversational responders into general-purpose problem solvers that can plan, critique, write code, and interact with external tools (Du et al., 2023; Shinn et al., 2023). A natural next step is to organize multiple LLM instances into Multi-Agent Systems (MAS) (Li et al., 2023; Wu et al., 2023; Tang et al., 2024; Chen et al., 2024a; Liu et al., 2025), where specialized roles collaborate to decompose complex tasks and cross-check intermediate conclusions. Recent systems have shown that role-playing and structured

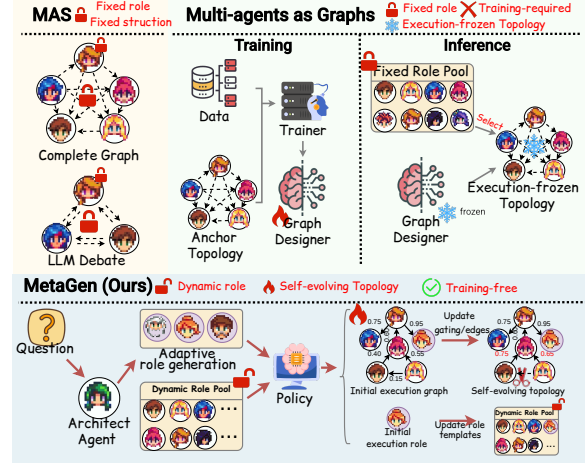


Figure 1: Overview and positioning of MetaGen. Unlike fixed-role/fixed-topology multi-agent systems and training-based topology designers with execution-frozen graphs, MetaGen enables query-conditioned role generation and self-evolving topology adjustment entirely at inference time.

collaboration can substantially outperform single-agent prompting on reasoning, tool use, and software engineering workflows (Chen et al., 2024b; Hong et al., 2024). At the same time, prompting paradigms such as debate, reflection, and search-based reasoning point to a broader lesson: the interaction structure—who speaks, what is produced, and how signals are aggregated—can be as influential as the base model itself (Du et al., 2023; Yao et al., 2023; Besta et al., 2024).

Despite this progress, many deployed MAS still follow a rigid preset design. Developers typically maintain a fixed role pool (e.g., planner/solver/verifier) and hard-code an execution-frozen message-passing protocol (Qian et al., 2025) (e.g., chain, star, or fully connected chat). Such rigidity leads to three recurring issues. First, *task mismatch* arises because task granularity, tool preferences, and error modes vary widely, while a fixed role set is often brittle under distribution shift. Sec-

*Equal contribution.

†Corresponding author.

ond, *structural closure* occurs when a topology determined once cannot be revised mid-run in response to new evidence or contradictions. Third, *cost* suffers because tailoring prompts and interaction structures to each task requires manual engineering.

An increasing line of work treats collaboration topology as a key lever and seeks to automate it (Yue et al., 2025; Zhang et al., 2025b). Graph-based views model agents as nodes and communications as directed edges, enabling orchestration search, pruning, and topology optimization (Zhuge et al., 2024; Liu et al., 2024; Zhang et al., 2024a). Recent topology designers learn or generate task-adaptive graphs, for example by predicting edges with graph models (Zhang et al., 2024b) or autoregressively generating a team and its links from a query (Li et al., 2026). While these approaches reduce manual engineering, two assumptions remain common at inference time: roles are drawn from a pre-defined library, and the instance-specific graph is typically frozen once execution begins. These observations motivate a central question: can an MAS generate the roles it needs and update its collaboration structure during inference, while keeping cost bounded?

We present **MetaGen**, a training-free framework that adapts both the role space and the collaboration topology at test time (Figure 1). Here, “training-free” means that MetaGen requires no supervised training data, gradient-based model optimization, or modification of the backbone LLM weights. Its inter-task evolution only performs lightweight on-line updates to role/edge priors and the Role Cache from previous execution traces. MetaGen introduces an Architect that synthesizes and revises query-conditioned role specifications to form a controllable dynamic role pool. It then constructs an initial execution graph around a minimal backbone and iteratively updates role prompts and structural decisions using lightweight feedback signals. To prevent unrestricted chatter, MetaGen enforces explicit controls, including schema/validity checks for generated roles, constrained graph construction, selective activation and edge gating, and cost-aware stopping.

MetaGen is designed to be effective and inspectable. It logs generated roles, selected participants, structural edits, and the feedback that triggers each update, supporting reproducibility and diagnosis beyond ad hoc orchestration. This combination of dynamic roles, inference-time evolution,

and structured control targets the core limitations of rigid MAS while retaining the engineering advantages of graph-based collaboration. In summary, our contributions are:

- We propose **MetaGen**, a training-free framework that improves multi-agent collaboration by adapting role specifications and communication topology during inference.
- We introduce *query-conditioned* role generation and revision with lightweight validity constraints, yielding a controllable dynamic role pool.
- We develop an inference-time evolution loop that updates prompts and structural decisions under explicit constraints to bound cost and maintain auditability.
- Extensive experiments demonstrate that MetaGen consistently improves the accuracy–cost trade-off over competitive multi-agent baselines, and ablations confirm the complementary benefits of dynamic roles, within-instance refinement, and cross-instance accumulation.

2 Related Work

2.1 Multi-agent collaboration with LLMs.

A growing body of work solves complex tasks via LLM-based multi-agent collaboration (Akata et al., 2025; Guo et al., 2024; Zhao et al., 2024; Hao et al., 2025), where multiple agents exchange intermediate results to reduce single-agent blind spots and improve reliability. Common paradigms include discussion-style coordination that aggregates diverse perspectives and iteratively refines candidate solutions (Saha et al., 2024), and debate-style protocols that surface contradictions and encourage self-correction through adversarial critique (Xiong et al., 2023). Another line of work emphasizes specialization by assigning distinct roles (e.g., planner, executor, verifier) and organizing them into hierarchical pipelines for decomposition and verification (Zhang et al., 2025a). Beyond pipelines, structured collaboration patterns such as chain and star orchestration (Hong et al., 2024; Qian et al., 2024; Zhou et al., 2023) and richer tree/graph-structured interaction (Zhang et al., 2024c; Zhao et al., 2024; Ishibashi and Nishimura, 2024; Qian et al., 2025; Zhuge et al., 2024; Zhao et al., 2026) have been adopted to better capture information dependencies

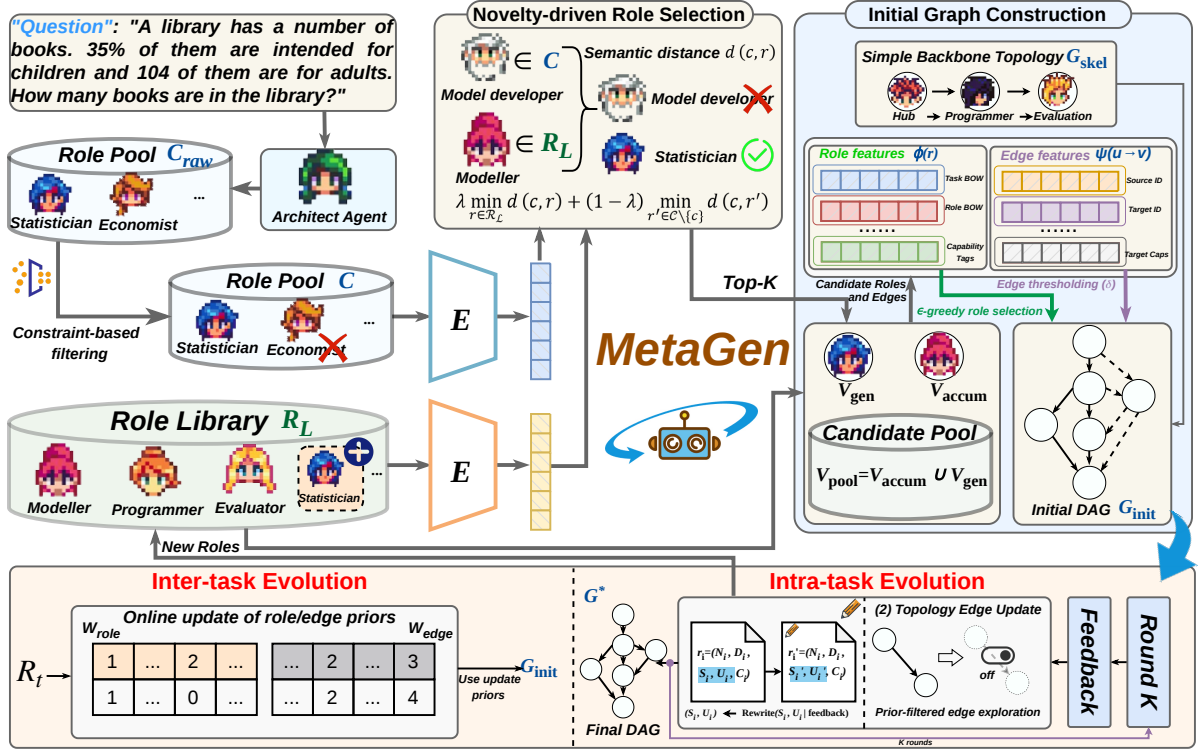


Figure 2: **MetaGen framework overview.** Given a query, an Architect generates and filters candidate roles, then performs novelty-driven role selection and hybrid graph initialization to form an initial DAG G_{init} . MetaGen supports intra-task evolution by updating role prompts and structure using execution feedback, and inter-task evolution by accumulating cross-instance priors and solidifying verified roles for future reuse.

and multi-step reasoning. Despite these advances, most systems assume a pre-defined role pool and adopt an interaction topology that is fixed or drawn from a small set of templates, remaining largely *execution-frozen* once inference begins. As a result, the collaboration strategy often cannot be tailored to instance-specific needs, and mismatched roles or redundant interactions can waste computation and hinder robustness under distribution shift.

2.2 Multi-Agents as Graphs

Graph-structured formulations are a natural fit for multi-agent collaboration, as they explicitly model information dependencies and interaction constraints among agents (Zhang et al., 2025b). Representative systems such as MacNet (Qian et al., 2025) and GPTSwarm (Zhuge et al., 2024) treat agent interaction as an optimizable graph. Recent work further moves toward *dynamic* topology construction. DyLAN (Liu et al., 2024) selects and routes agents by filtering an initially large set based on instance-level importance signals, G-Designer (Zhang et al., 2024b) synthesizes communication graphs with a learned generator to adapt connectivity patterns, and ARG-Designer (Li et al.,

2026) autoregressively constructs agent groups together with their links under task conditioning. Unlike topology-centric designers that primarily generate interaction graphs over fixed/retrieved roles, MetaGen treats both role specifications and topology as editable inference-time objects, enabling coupled intra-instance refinement and cross-instance accumulation.

3 Method

MetaGen is a training-free framework for multi-agent collaboration that models role specifications and communication topology as first-class, editable entities during inference. It enables structured adaptation via query-conditioned role generation and revision, coupled with a self-evolving graph orchestration loop subject to explicit structural constraints. Under this formulation, collaboration structures are progressively refined to meet task-specific requirements, while bounding computational cost and preserving auditable interaction traces, thereby improving adaptability, reproducibility, and reasoning performance in multi-agent systems.

Method	Dyn.	T-free	Evol.	GSM8K	HumanEval	MMLU	AQuA	MNLI	Average
Vanilla	☒	☑	☒	89.3	65.2	87.1	69.7	77.6	77.8
CoT (zero-shot)	☒	☑	☒	93.1 ± 0.2 ^{↑3.8}	89.0 ± 0.9 ^{↑23.8}	89.5 ± 1.0 ^{↑2.4}	70.9 ± 0.7 ^{↑1.2}	82.3 ± 0.3 ^{↑4.7}	85.0
CoT (few-shot)	☒	☑	☒	95.8 ± 0.1 ^{↑6.5}	92.1 ± 1.2 ^{↑26.9}	91.5 ± 0.7 ^{↑4.4}	84.6 ± 1.0 ^{↑14.9}	85.4 ± 1.0 ^{↑7.8}	89.9
SC (K=3)	☒	☑	☒	94.2 ± 0.3 ^{↑4.9}	86.0 ± 0.9 ^{↑20.8}	90.8 ± 0.8 ^{↑3.7}	72.0 ± 1.3 ^{↑2.3}	83.8 ± 1.4 ^{↑6.2}	85.4
SC (K=10)	☒	☑	☒	93.9 ± 0.1 ^{↑4.6}	84.1 ± 2.2 ^{↑18.9}	92.2 ± 0.4 ^{↑5.1}	77.2 ± 0.2 ^{↑7.5}	84.0 ± 0.2 ^{↑6.4}	86.3
Chain	☒	☑	☒	92.0 ± 0.7 ^{↑2.7}	90.2 ± 0.6 ^{↑25.0}	91.5 ± 0.4 ^{↑4.4}	79.1 ± 0.6 ^{↑9.4}	77.2 ± 0.4 ^{↓0.4}	86.0
Star	☒	☑	☒	94.5 ± 0.4 ^{↑5.2}	89.6 ± 0.9 ^{↑24.4}	90.2 ± 0.4 ^{↑3.1}	83.5 ± 1.6 ^{↑13.8}	69.9 ± 0.5 ^{↓7.7}	85.5
Tree	☒	☑	☒	77.5 ± 0.2 ^{↓11.8}	93.9 ± 1.6 ^{↑28.7}	77.1 ± 0.7 ^{↓10.0}	83.9 ± 0.7 ^{↑14.2}	55.3 ± 0.4 ^{↓22.3}	77.5
Complete Graph	☒	☑	☒	94.6 ± 0.3 ^{↑5.3}	89.0 ± 0.9 ^{↑23.8}	92.2 ± 1.3 ^{↑5.1}	86.2 ± 0.6 ^{↑16.5}	83.3 ± 0.5 ^{↑5.7}	89.1
Random Graph	☒	☑	☒	95.4 ± 0.1 ^{↑6.1}	92.1 ± 0.6 ^{↑26.9}	91.8 ± 0.8 ^{↑4.7}	78.7 ± 0.6 ^{↑9.0}	84.2 ± 0.9 ^{↑6.6}	88.4
LLM-Debate	☒	☑	☒	94.2 ± 3.0 ^{↑4.9}	89.6 ± 1.6 ^{↑24.4}	92.2 ± 1.7 ^{↑5.1}	85.8 ± 3.8 ^{↑16.1}	79.7 ± 1.7 ^{↑2.1}	88.3
GPTSwarm	☒	☒	☒	—	69.6 ± 0.4 ^{↑4.4}	60.1 ± 1.6 ^{↓27.0}	—	—	64.9
AFlow	☒	☒	☐	94.3 ± 0.8 ^{↑5.0}	90.9 ± 3.0 ^{↑25.7}	—	—	—	92.6
G-Designer	☒	☒	☐	96.3 ± 0.2 ^{↑7.0}	94.2 ± 0.7 ^{↑29.0}	93.5 ± 0.9 ^{↑6.4}	89.0 ± 0.8 ^{↑9.3}	—	93.3
ARG-Designer	☑	☒	☐	96.1 ± 0.2 ^{↑6.8}	90.9 ± 1.0 ^{↑25.7}	89.5 ± 1.1 ^{↑2.4}	90.6 ± 1.6 ^{↑20.9}	—	91.8
MetaGen	☑	☑	☑	96.4 ± 0.4^{↑7.1}	95.1 ± 0.7^{↑29.9}	93.5 ± 0.8^{↑6.4}	95.7 ± 1.0^{↑26.0}	94.8 ± 0.4^{↑17.2}	95.1

Table 1: Main results on five benchmarks using DeepSeek-V3. Dyn. indicates whether a method uses a dynamic role pool. T-free indicates whether it avoids training model weights for role or topology design. Evol. indicates whether the interaction topology evolves during inference. ☑ means yes. ☒ means no. ☐ means partial. All numbers are means over three independent runs.

3.1 Problem Formulation

Given a task input x (e.g., code generation or complex reasoning), **MetaGen** employs an *Architect Agent* during inference to automatically generate a set of candidate agent roles $\{\hat{r}_i\}_{i=1}^N$, from which a directed acyclic graph (DAG) $G = (V, E)$ is constructed to model the multi-agent collaboration process.

Each node $v \in V$ corresponds to a specific agent role, and each directed edge $e = (u \rightarrow v) \in E$ represents a message flow between roles, capturing information dependencies and collaboration pathways during multi-agent reasoning. Each role r_i is formally defined as a tuple:

$$r_i = (N_i, D_i, S_i, U_i, C_i), \quad (1)$$

where N_i denotes the role name, D_i denotes the semantic description of the role, S_i denotes the system-level prompt template, U_i denotes the user-level prompt template, and C_i denotes the set of capabilities or tools available to the role.

Without updating the underlying large language model parameters, the core objective of MetaGen is to regulate the multi-agent inference process through joint optimization of role specifications and collaboration structure by minimizing the following composite objective:

$$\min_{\text{arch, roles}} \mathcal{L} = \lambda_1 \mathcal{L}_{\text{acc}}(y, y^*) + \lambda_2 \mathcal{L}_{\text{cost}}(\tau) + \lambda_3 \mathcal{L}_{\text{sparse}}(G), \quad (2)$$

where $\mathcal{L}_{\text{acc}}$ quantifies the prediction error between the system output y and the ground-truth target

y^* , $\mathcal{L}_{\text{cost}}$ penalizes the cumulative token usage and inference latency over the reasoning trajectory τ , and $\mathcal{L}_{\text{sparse}}$ serves as a structural regularizer that promotes sparsity in the communication graph, improving interpretability and controllability. During inference, MetaGen does not access y^* ; it relies on naturally observable execution signals to trigger edits and updates only lightweight selection priors from the pass/cost summary.

3.2 Generative Role Space

To address task mismatch and distribution shift, MetaGen implements a dynamic *Generative Role Space*. We employ an Architect Agent to synthesize a raw candidate set $\mathcal{C}_r$ conditioned on the query. To ensure the role space is both executable and non-redundant, we enforce a formalized two-stage validation process.

Constraint-Based Filtering. We first refine the raw generations into a valid candidate set $\mathcal{C}$ by imposing strict structural and safety constraints. Let $T(c)$ denote the prompt template of candidate c , and $\mathcal{W}(c)$ be its token set. We define the valid set as:

$$\mathcal{C} = \{c \in \mathcal{C}_r \mid (T(c) \models \Phi) \wedge (\mathcal{W}(c) \cap \mathcal{V}_b = \emptyset)\}, \quad (3)$$

where Φ represents the required schema (e.g., placeholders), $\models$ denotes schema satisfaction, and $\mathcal{V}_b$ is a set of restricted keywords.

Embedding-Based Diversity Gating. To avoid semantic redundancy, we project roles into a dense

vector space to enforce diversity. Let $\mathcal{E} : \mathcal{X} \rightarrow \mathbb{R}^d$ denote a semantic encoder that maps the textual description of a role c to its embedding vector $\mathbf{e}_c$:

$$\mathbf{e}_c = \frac{\mathcal{E}(\text{desc}(c))}{\|\mathcal{E}(\text{desc}(c))\|_2}. \quad (4)$$

To prevent semantic redundancy, we employ an embedding-based ranking mechanism. Let $d(c, r)$ denote the semantic distance between two roles in the embedding space.

For each candidate $c \in \mathcal{C}$, we calculate a *Marginal Utility Score* $S(c)$ that balances external novelty against the historical library $\mathcal{R}_L$ and internal distinctiveness relative to other candidates:

$$S(c) = \lambda \min_{r \in \mathcal{R}_L} d(c, r) + (1 - \lambda) \min_{r' \in \mathcal{C} \setminus \{c\}} d(c, r'), \quad (5)$$

where λ controls the trade-off weight. Finally, we construct the incremental role set $\Delta\mathcal{R}$ by selecting the top- K candidates that exceed a minimum novelty threshold δ :

$$\Delta\mathcal{R} = \text{Top}_K(\{c \in \mathcal{C} \mid S(c) > \delta\}). \quad (6)$$

This selection strategy ensures that the instantiated roles are not only valid but also semantically unique and non-redundant.

3.3 Task-Adaptive Graph Construction

To balance structural regularization with semantic flexibility, we propose a hybrid graph construction strategy. It anchors reasoning to a minimal functional backbone, expands it through score-based selection over a hybrid role pool, and supports evolution at two levels: intra-task refinement within an instance and inter-task accumulation across instances, as summarized in Algorithm 1.

Hybrid Graph Initialization. We first instantiate a task-type backbone chain $G_{\text{skel}} = (V_{\text{skel}}, E_{\text{skel}})$ to guarantee the fundamental execution flow. For code generation, the chain is

$$E_{\text{skel}} = \{(v_{\text{hub}} \rightarrow v_{\text{prog}}), (v_{\text{prog}} \rightarrow v_{\text{eval}})\}. \quad (7)$$

Here v_{hub} dispatches the request, v_{prog} produces code, and v_{eval} verifies it.

To handle requirements beyond the backbone, we form a hybrid candidate pool $\mathcal{V}_{\text{pool}} = \mathcal{V}_{\text{accum}} \cup \mathcal{V}_{\text{gen}}$, where $\mathcal{V}_{\text{accum}}$ contains accumulated generalist and previously effective roles, and $\mathcal{V}_{\text{gen}}$ contains query-conditioned roles synthesized by the Architect for the current instance.

Algorithm 1: MetaGen inference and evolution

Input: task x ; fixed roles $\mathcal{R}_{\text{fix}}$; cache C ;
priors (P_R, P_E) ; (K_g, K_c, K_e, T) ;
 δ_r, δ ; budget B

Output: y^* ; G^* ; H ; updated (P_R, P_E, C)

- 1 $\mathcal{C} \leftarrow \text{FILTERVALID}(\text{ARCHITECT}(x, K_g))$;
- 2 $d(a, b) \leftarrow 1 - \cos(e_a, e_b)$;
 $D_h(r) \leftarrow d(r, \mathcal{R}_{\text{fix}} \cup C)$;
 $D_g(r) \leftarrow d(r, \mathcal{C} \setminus \{r\})$;
- 3 $\mathcal{C} \leftarrow \{r : \lambda D_h(r) + (1 - \lambda) D_g(r) > \delta_r\}$;
- 4 $\mathcal{V} \leftarrow \text{TOPKROLE}(\mathcal{R}_{\text{fix}} \cup C \cup \mathcal{C}, P_R, K_c)$;
- 5 $\mathcal{E} \leftarrow \text{TOPKEDGE}(P_E, \delta, K_e)$;
- 6 $G^{(0)} \leftarrow \text{BUILDDAG}(\mathcal{V}, \mathcal{E})$; $H \leftarrow \emptyset$;
- 7 **for** $t = 1$ **to** T **do**
- 8 $(y^{(t)}, H) \leftarrow \text{EXECUTE}(G^{(t-1)}, x, H)$;
- 9 **if** $\text{DONE}(H, B)$ **or** $t = T$ **then**
- 10 **break**;
- 11 $G^{(t)} \leftarrow \text{VALIDEDIT}(G^{(t-1)}, H)$;
- 12 $(y^*, G^*) \leftarrow (y^{(t)}, G^{(t-1)})$;
- 13 **if** $\text{CROSSINSTANCEENABLED}$ **then**
- 14 $(P_R, P_E, C) \leftarrow$
 $\text{CROSSEVOLVE}(P_R, P_E, C, H)$;
- 15 **return** y^*, G^*, H, P_R, P_E, C ;

Each candidate role r is represented by a feature vector $\phi(r)$ that combines lexical cues from the role name/prompt template, capability indicators, semantic relevance to the query via $\mathcal{E}$, and optional historical statistics when available. Each candidate directed edge $(u \rightarrow v)$ is represented by $\psi(u \rightarrow v)$, which combines endpoint features with simple structural signals and optional co-occurrence statistics. We compute linear priority scores

$$s_r = \mathbf{w}_{\text{role}}^\top \phi(r), \quad s_{u \rightarrow v} = \mathbf{w}_{\text{edge}}^\top \psi(u \rightarrow v), \quad (8)$$

and select a Top- K committee with an ϵ -greedy strategy. Edges are added to form G_{init} when their scores exceed a threshold δ , subject to DAG constraints.

Intra-task Evolution. Starting from G_{init} , MetaGen performs lightweight within-instance refinement over multiple rounds. We denote by $\mathcal{F}$ the *feedback* collected during inference and tool execution, consisting of naturally observable signals such as runtime logs, compilation/test outcomes, format validators, and self-consistency checks. This feedback is available without introducing additional

supervision and serves as the trigger for instance-level edits.

Given $\mathcal{F}$, MetaGen applies two types of edits that operate *only* on textual role specifications and a constrained subset of structural choices. First, *role prompt rewrite* targets a low-utility role whose messages are consistently unhelpful (e.g., redundant, unstable, or verbose) under the current instance. Using the feedback traces (error messages, failed checks, or inconsistency patterns), MetaGen revises the role’s system/user templates to better align the role behavior with the instance requirements. Second, *prior-filtered edge exploration* updates topology within the instance in a conservative manner. MetaGen first filters candidate *non-critical* edges using current priors and structural constraints (e.g., preserving at least one path to the exit/judge node and avoiding cycles), then selectively deactivates or swaps one edge to encourage simpler, more informative communication. Across rounds, these edits allow the collaboration process to react to observed failure modes while keeping the execution stable and auditable.

Inter-task Evolution. While intra-task evolution adapts behavior for a single instance, MetaGen also improves future decisions by maintaining lightweight state across instances. After an instance completes, we summarize its overall outcome into a scalar *reward* that trades off success and cost,

$$R = \mathbb{I}(\text{pass}) - \lambda_{\text{cost}} \cdot \mathcal{C}_{\text{token}}, \quad (9)$$

where $\mathbb{I}(\text{pass})$ is a task-specific pass indicator produced by the evaluator and $\mathcal{C}_{\text{token}}$ is the total token usage. We then update the parameters that govern role/edge scoring with a reward-weighted linear rule:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta R \mathbf{f}, \quad (10)$$

where $\mathbf{f}$ is the feature vector for the decision that was used, i.e., $\mathbf{f} = \phi(r)$ for a selected role (updating $\mathbf{w}_{\text{role}}$) or $\mathbf{f} = \psi(u \rightarrow v)$ for an activated edge (updating $\mathbf{w}_{\text{edge}}$). Intuitively, decisions that lead to successful, low-cost executions receive positive updates and become more likely under similar contexts, whereas costly or unsuccessful executions yield weaker (or negative) reinforcement. This mechanism is deliberately lightweight: it updates only shallow priors used by the selector/wiring module and remains fully decoupled from backbone LLM weight training.

Method	#Training Token	#Inference Token	#Overall Token
Complete	—	9.8×10^6	9.8×10^6
DyLAN	9.6×10^6	1.3×10^7	2.2×10^7
GPTSwarm	5.5×10^6	8.4×10^6	1.4×10^7
G-Designer	2.7×10^5	8.2×10^6	8.5×10^6
MetaGen	—	1.2×10^6	1.2×10^6

Table 2: Token cost comparison measured with GPT-4.

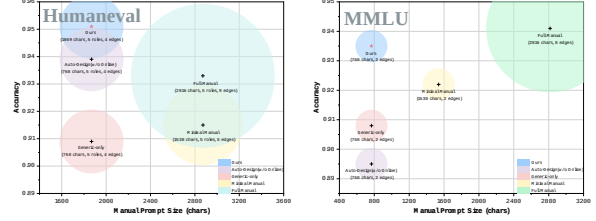


Figure 3: Accuracy versus manual prompt size on HumanEval (left) and MMLU (right). Each point corresponds to a different design budget variant, illustrating the trade-off between engineering effort and performance.

Verified Role Solidification and Reuse. In addition to updating priors, MetaGen maintains a growing pool of reusable roles. During intra-task evolution, the Architect may synthesize new roles or substantially rewrite prompts to better match the instance. To retain effective transient roles, we solidify roles only when the final execution passes task-specific checks. Concretely, we extract a small Top- K set of effective non-builtin roles from the executed graph (after de-duplication and basic validity checks), serialize their specifications, and store them in a persistent Role Cache. In subsequent instances, the cache is loaded and merged into the role library as a high-priority candidate pool, enabling reuse of verified role templates rather than regenerating them from scratch. Over time, this reward-conditioned retention expands the role library with task-relevant specialists and improves cold-start behavior under recurring patterns, without any backbone fine-tuning.

4 Experiments and Analyses

4.1 Experimental Setup

Datasets and Metrics. We evaluate MetaGen on five widely used benchmarks that cover multi-step mathematical reasoning (GSM8K (Cobbe et al., 2021)), code generation (HumanEval (Chen et al., 2021)), broad knowledge and reasoning (MMLU (Hendrycks et al., 2020)), algebraic word problems (AQuA (Ling et al., 2017)), and natural language inference (MNLI (Williams et al., 2018)).

For each dataset, we follow the official evaluation split and report the standard accuracy-based metric: exact-match accuracy for GSM8K and AQuA, classification accuracy for MMLU and MNLI, and pass@1 for HumanEval under the provided unit tests. Our main comparison is summarized in Table 1, where we also report the average score across the five datasets.

Baselines. We compare against both single-agent prompting and multi-agent orchestration baselines. For single-agent prompting, we include a vanilla prompt, zero-shot and few-shot Chain-of-Thought(Wei et al., 2023) prompting, and Self-Consistency(Wang et al., 2022) with multiple sampled rationales. For fixed-topology multi-agent baselines, we instantiate common communication patterns, including Chain, Star, Tree(Qian et al., 2025), Complete Graph, Random Graph, as well as an LLM debate-style protocol(Du et al., 2023). We further compare with representative automated topology design and multi-agent frameworks, including GPTSwarm(Zhuge et al., 2024), AFlow(Zhang et al., 2025c), G-Designer(Zhang et al., 2024b), and ARG-Designer(Li et al., 2026). When a baseline does not support a dataset in its original setting or public implementation, we mark the corresponding entry as missing in Table 1.

Implementation Details. Unless otherwise specified, all methods use the same backbone model, DeepSeek-V3 (DeepSeek-AI et al., 2025), to isolate the effect of role generation and topology control. For the semantic encoder used in role relevance scoring and diversity control, we use SentenceTransformer all-MiniLM-L6-v2(Wang et al., 2020). Unless otherwise specified, the Architect generates three candidate roles per instance and the selector instantiates a top- K committee with $K=2$. For online decision making, we use an ϵ -greedy exploration strategy with $\epsilon=0.15$ and update step size $\eta=0.15$. The reward trades off task success and cost as $R = \mathbb{I}(\text{pass}) - \lambda_{\text{cost}} \cdot \mathcal{C}_{\text{token}}$ with $\lambda_{\text{cost}}=0.001$.

4.2 Main Results

Primary Results. Table 1 shows that MetaGen delivers the best overall performance, with a 1.8% $\uparrow$ average accuracy improvement over the strongest baseline G-Designer. On AQuA, MetaGen exceeds ARG-Designer by 5.1% $\uparrow$, demonstrating clear benefits from role adaptation and inference-time topology evolution on multi-step reasoning. On MNLI,

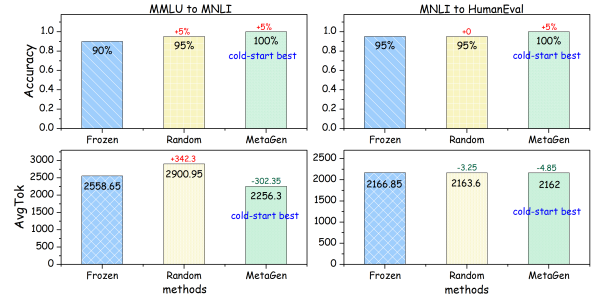


Figure 4: Cold-start recovery after distribution shifts. Accuracy (top) and average tokens (bottom) on the first 20 examples immediately after each shift, comparing Frozen, Random, and MetaGen. MetaGen achieves the strongest cold-start accuracy with lower token cost.

MetaGen improves over the best reported baseline CoT few-shot by 9.4% $\uparrow$, indicating substantially stronger task-conditional collaboration on NLI. On HumanEval, MetaGen surpasses G-Designer by 0.9% $\uparrow$, and on GSM8K it provides a further 0.1% $\uparrow$ gain, while matching the best MMLU result.

Backbone Generalization. We further evaluate MetaGen with GLM-4-Flash on HumanEval and MMLU. Table 4 shows consistent gains over fixed-role controls and ARG-Designer, while enlarging the fixed-role pool remains insufficient. These results confirm that MetaGen’s advantage generalizes beyond DeepSeek-V3 and is not attributable to a larger predefined role pool.

4.3 Cost Efficiency

We evaluate cost from two complementary perspectives, runtime token cost and human authoring cost.

Runtime token cost. Table 2 shows that MetaGen uses 1.2×10^6 inference tokens. This yields an 85.4% reduction relative to G-Designer, an 87.8% reduction relative to Complete, an 85.7% reduction relative to GPTSwarm, and a 90.8% reduction relative to DyLAN. Overall, MetaGen achieves $7.1 \times$ to $18.3 \times$ fewer end-to-end tokens than prior multi-agent systems, while requiring no training tokens for role or topology design.

Human authoring cost. Figure 3 shows that MetaGen consistently improves the accuracy versus manual prompt size frontier on both HumanEval and MMLU. At the same manual budget, MetaGen attains higher accuracy, and it reaches near-saturated performance with substantially less hand-written specification. The advantage is most pronounced in the low-budget regime, indicating

Method	Overall 1–150		Segment 1 MMLU 1–50		Segment 2 MNLI 51–100		Segment 3 HumanEval 101–150	
	Acc	AvgTok	Acc	AvgTok	Acc	AvgTok	Acc	AvgTok
Frozen	90.0%	2673	92.0%	3030	80.0%	2782	94.0%	2208
Random	90.7%	2787	90.0%	3110	84.0%	3043	96.0%	2207
MetaGen	92.7%	2483	94.0%	3062	88.0%	2190	100.0%	2196

Table 3: Non-stationary stream evaluation on a 150-instance sequence. The stream proceeds from Segment 1 MMLU, to Segment 2 MNLI, and then Segment 3 HumanEval. We report accuracy and average total tokens per question for the full stream and each segment.

Setting	HumanEval	MMLU
Static roles + static topology	70.73	76.47
Enlarged fixed-role pool	75.61	77.12
ARG-Designer	80.17	75.16
MetaGen	82.65	78.99

Table 4: Backbone generalization with GLM-4-Flash.

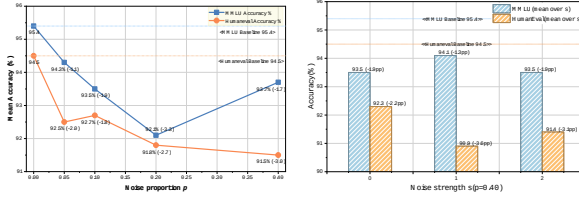


Figure 5: Robustness to noisy nodes and edges. Left: varying the noise proportion p (fraction of corrupted nodes and optional edges). Right: varying the noise strength level s with fixed $p=0.4$.

that online selection and evolution are the key drivers that recover accuracy when manual engineering is limited.

4.4 Adaptation and Robustness

Non-stationary Stream Adaptation. We evaluate non-stationary adaptation on a 150-instance stream with three consecutive segments, consisting of 50 examples from MMLU, then 50 from MNLI, and finally 50 from HumanEval. We compare MetaGen with Frozen, which keeps roles and topology fixed across the stream, and Random, which perturbs topology without learning. Table 3 shows that MetaGen improves overall accuracy by 3 points over Frozen and by 2 points over Random, while reducing average tokens by 7.1% and 10.9%. The advantage is most pronounced on the shifted MNLI segment, where MetaGen improves accuracy by 8 points over Frozen and by 4 points over Random, with 21.3% to 28.0% fewer tokens. On the final HumanEval segment, MetaGen reaches perfect accuracy, improving by 6 points over Frozen and by

4 points over Random without increasing tokens.

Cold-start Recovery After Distribution Shifts.

Figure 4 further isolates the first 20 examples immediately after each distribution shift. After the first shift, MetaGen improves cold-start accuracy by 10 points over Frozen and by 5 points over Random, while reducing tokens by 11.8% and 22.2%. After the second shift, MetaGen maintains perfect cold-start accuracy with no extra token overhead. These results indicate that MetaGen not only adapts over the stream but also exhibits strong cold-start capability right after shifts.

Noise robustness. We test robustness by injecting corruption into both agent nodes and optional communication edges, controlled by a corruption ratio p and a corruption strength s . Figure 5 shows that MetaGen remains stable under both widespread and stronger perturbations. As p increases, accuracy decreases gradually rather than collapsing, indicating that MetaGen does not hinge on any single critical node or edge and can preserve performance through redundant reasoning routes. Notably, the degradation remains limited even at high corruption levels, suggesting that the evolving collaboration structure can compensate for partial failures by re-weighting or bypassing unreliable components. When fixing $p=0.4$ and increasing s , performance exhibits only mild additional drops, implying that the system is resilient not only to the amount of noise but also to its severity.

Mechanism Analysis. We analyze how query-conditioned roles and intra-task topology evolution contribute to MetaGen on HumanEval. Across role-enabled variants, 34.5–36.3% of valid generated roles are selected into the initial DAG, with at least one generated role executed on 19.5–20.7% of instances; meanwhile, the active node and edge sets change on 96.3–97.0% and 94.5–96.3% of instances, respectively. As shown in Table 5(a), removing the selected generated role reduces Pass@1

(a) Query-conditioned role analysis			
Setting	Pass@1	Δ	
MetaGen	95.1	—	
Generated role removed	90.9	$\downarrow 4.2$	
Generated role $\rightarrow$ fixed role	93.4	$\downarrow 1.7$	

(b) Factorial analysis of intra-task adaptation			
Variant	Role	Topo.	Pass@1
Static roles + static topology	☒	☒	90.24
Editable roles only	☑	☒	92.07
Editable topology only	☒	☑	91.46
Editable roles + topology	☑	☑	93.29

Table 5: Mechanism analysis on HumanEval. Panel (a) evaluates query-conditioned role specialization; Panel (b) factorizes intra-task role and topology editability with cross-instance adaptation disabled.

from 95.1 to 90.9, while replacing it with a fixed role reaches 93.4, demonstrating the benefit of query-specific role specialization. Table 5(b) further factorizes intra-task adaptation through a controlled 2×2 factorial design: all variants share the same DeepSeek-V3 backbone, decoding configuration, initial fixed-role pool, structural configuration, and inference budget, with Role Cache reuse and cross-instance prior updates disabled. Under this setting, the estimated main effects of role and topology editability are 1.83 and 1.22 Pass@1 points, respectively, and their combination achieves the best performance. These results show that role and topology adaptation make distinct and largely additive contributions to MetaGen.

4.5 Ablation Study

We evaluate four MetaGen variants by disabling one mechanism at a time. **(1) w/o Role Generation** uses a fixed role set without query-conditioned role synthesis. **(2) w/o Learned Policy** replaces learning-based selection and wiring with random or relevance-only heuristics, and does not use persistent statistics or policy states for decision making. **(3) w/o Intra-task Evolution** disables within-instance updates so the system executes G_{init} without prompt rewriting or topology adjustment. **(4) w/o Cross-instance Memory** disables persistence across instances by stopping verified role write-back and resetting selection and wiring states so each instance cold-starts. Table 6 shows that each component matters and the full MetaGen performs best. Role generation is most critical, with HumanEval dropping from 95.1 to 92.1 without it. Replacing the learned policy lowers

Variant	HumanEval	MMLU
vanilla MetaGen	95.1	93.5
w/o Role Generation	92.1 $\downarrow 3.0$	91.1 $\downarrow 2.4$
w/o Learned Policy	93.9 $\downarrow 1.2$	92.8 $\downarrow 0.7$
w/o Intra-task Evolution	92.7 $\downarrow 2.4$	91.7 $\downarrow 1.8$
w/o Cross-instance Memory	92.7 $\downarrow 2.4$	92.6 $\downarrow 0.9$

Table 6: Ablation study. Each variant removes one component from MetaGen.

MMLU to 92.8, disabling intra-task evolution reduces MMLU to 91.7, and removing cross-instance memory drops HumanEval to 92.7, showing that query-conditioned roles, learned selection, and accumulated memory all contribute to robustness.

5 Conclusion

We propose **MetaGen**, a training-free multi-agent framework that improves accuracy while reducing both inference-token cost and manual prompt engineering by generating and refining roles and collaboration structure at inference time. With a DeepSeek-V3 backbone, MetaGen achieves the strongest overall performance across five benchmarks against single-agent prompting, fixed-topology orchestration, and topology-design baselines. Further analyses show that MetaGen degrades gracefully under noisy agents and perturbed edges, benefits from each core component, and adapts to non-stationary task streams with strong cold-start recovery after distribution shifts. Overall, these results highlight inference-time optimization of text-level roles and discrete collaboration structure as a practical path toward scalable and adaptive MAS without modifying backbone weights.

Limitations

While MetaGen advances the flexibility of multi-agent orchestration, its reliance on generative dynamics introduces specific engineering trade-offs and risks.

Dependence on Base Model and Error Propagation. As a training-free framework, MetaGen relies on the intrinsic reasoning capabilities of the underlying LLM for structural decisions. A core challenge is the risk of “self-reinforcing bias,” where the base model might generate plausible but incorrect rationales to justify a flawed topological adjustment. If the model provides unreliable evaluation signals during the graph evolution, these errors

can propagate, potentially solidifying a suboptimal collaboration pattern. This necessitates that the base model maintains a high threshold of reasoning consistency to fully leverage the self-evolving architecture.

Safety Assurance in Open-Ended Generation.

Although MetaGen employs validity checks and constrained graph construction, the dynamic nature of role generation complicates static safety audits. Unlike systems with hard-coded agents, the unbounded space of generated roles implies that safety guardrails are probabilistic rather than deterministic. Furthermore, by significantly lowering the technical barrier for orchestrating complex agent swarms, there is a potential risk that the framework could be repurposed for automated generation of misleading content if not coupled with strict deployment policies. When connected to external tools, this flexibility also demands rigorous sandboxing to prevent prompt injection or unintended data leakage through dynamically generated policies.

Orchestration Overhead for Atomic Tasks.

While mechanisms such as pruning and early stopping are implemented to bound resource usage, the Architect component introduces inherent management overhead. For atomic or highly specific tasks that can be solved by a single zero-shot prompt, the latency and token consumption incurred by constructing and refining a multi-agent graph may yield diminishing marginal returns. This highlights an efficiency trade-off: MetaGen is optimized for complex, multi-step reasoning, and future iterations must explore adaptive gating to bypass the orchestration loop for trivial queries.

Acknowledgments

This research was supported by the National Key Research and Development Project of China (2025YFB3313100), and partially supported by grants from the National Natural Science Foundation of China (Nos. T2541001, 52472316, 52341203, and 52461160297) and the Guangdong Basic and Applied Basic Research Foundation (No. 2025A1515010251).

References

Elif Akata, Lion Schulz, Julian Coda-Forno, Seong Joon Oh, Matthias Bethge, and Eric Schulz. 2025. Playing repeated games with large language models. *Nature Human Behaviour*, 9(7):1380–1390.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefer. 2024. [Graph of thoughts: Solving elaborate problems with large language models](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Justin Chen, Swarnadeep Saha, and Mohit Bansal. 2024a. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7066–7085.

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](#). *Preprint*, arXiv:2107.03374.

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

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](#). *Preprint*, arXiv:2110.14168.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. [Deepseek-v3 technical report](#). *Preprint*, arXiv:2412.19437.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. [Improving factuality and reasoning in language models through multiagent debate](#). *Preprint*, arXiv:2305.14325.

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

Rui Hao, Linmei Hu, Weijian Qi, Qingliu Wu, Yirui Zhang, and Liqiang Nie. 2025. Chatllm network: More brains, more intelligence. *AI Open*, 6:45–52.

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

- 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*.
- Yoichi Ishibashi and Yoshimasa Nishimura. 2024. Self-organized agents: A llm multi-agent framework toward ultra large-scale code generation and optimization. *arXiv preprint arXiv:2404.02183*.
- Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.
- Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. 2026. [Assemble your crew: automatic multi-agent communication topology design via autoregressive graph generation](#). In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI’26/IAAI’26/EAAI’26. AAAI Press.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. [Program induction by rationale generation : Learning to solve and explain algebraic word problems](#). *Preprint*, arXiv:1705.04146.
- Jun Liu, Zhenglun Kong, Changdi Yang, Fan Yang, Tianqi Li, Peiyan Dong, Joannah Nanjeyke, Hao Tang, Geng Yuan, Wei Niu, Wenbin Zhang, Pu Zhao, Xue Lin, Dong Huang, and Yanzhi Wang. 2025. [Rcr-router: Efficient role-aware context routing for multi-agent llm systems with structured memory](#). *Preprint*, arXiv:2508.04903.
- Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024. [A dynamic llm-powered agent network for task-oriented agent collaboration](#). *Preprint*, arXiv:2310.02170.
- Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. [Chatdev: Communicative agents for software development](#). *Preprint*, arXiv:2307.07924.
- 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](#). *Preprint*, arXiv:2406.07155.
- Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. 2024. Branch-solve-merge improves large language model evaluation and generation. 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 8352–8370.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS ’23, Red Hook, NY, USA. Curran Associates Inc.
- Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. 2024. Medagents: Large language models as collaborators for zero-shot medical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 599–621.
- 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 Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. [Chain-of-thought prompting elicits reasoning in large language models](#). *Preprint*, arXiv:2201.11903.
- Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. [A broad-coverage challenge corpus for sentence understanding through inference](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.
- 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. 2023. [Autogen: Enabling next-gen llm applications via multi-agent conversation](#). *Preprint*, arXiv:2308.08155.
- Kai Xiong, Xiao Ding, Yixin Cao, Ting Liu, and Bing Qin. 2023. [Examining inter-consistency of large language models collaboration: An in-depth analysis via debate](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7572–7590, Singapore. Association for Computational Linguistics.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023.

[React: Synergizing reasoning and acting in language models](#). *Preprint*, arXiv:2210.03629.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyang 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.

Cong Zhang, Xin Deik Goh, Dexun Li, Hao Zhang, and Yong Liu. 2025a. Planning with multi-constraints via collaborative language agents. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10054–10082.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025b. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. 2024a. [Cut the crap: An economical communication pipeline for llm-based multi-agent systems](#). *Preprint*, arXiv:2410.02506.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2024b. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. 2025c. [Aflow: Automating agentic workflow generation](#). *Preprint*, arXiv:2410.10762.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan O Arik. 2024c. [Chain of agents: Large language models collaborating on long-context tasks](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Jiaxing Zhao, Hongbin Xie, Yuzhen Lei, Xuan Song, Zhuoran Shi, Lianxin Li, Shuangxue Liu, Linguo Xie, and Haoran Zhang. 2026. [Cochain: Balancing insufficient and excessive collaboration in llm agent workflows](#). *Preprint*, arXiv:2505.10936.

Jun Zhao, Can Zu, Hao Xu, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Longagent: scaling language models to 128k context through multi-agent collaboration. *arXiv preprint arXiv:2402.11550*.

Zihao Zhou, Bin Hu, Chenyang Zhao, Pu Zhang, and Bin Liu. 2023. Large language model as a policy teacher for training reinforcement learning agents. *arXiv preprint arXiv:2311.13373*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. [GPTSwarm: Language agents as optimizable graphs](#). In *Forty-first International Conference on Machine Learning*.

A Implementation Details and Prompt Examples

We provide additional implementation details for the mechanisms described in the main text. For structural control, the graph sparsity parameter is set to 0.5. For the embedding-based diversity gate in the generative role space, we use a minimum semantic distance of 0.25 to filter redundant role candidates. We provide the role-specific prompts that define the fixed Simple Backbone Topology used for MMLU. The backbone follows a Reasoner–Critic–Refiner–Arbiter–Answerer pipeline, providing a stable functional scaffold on top of which MetaGen performs query-conditioned role generation and inference-time topology adaptation.

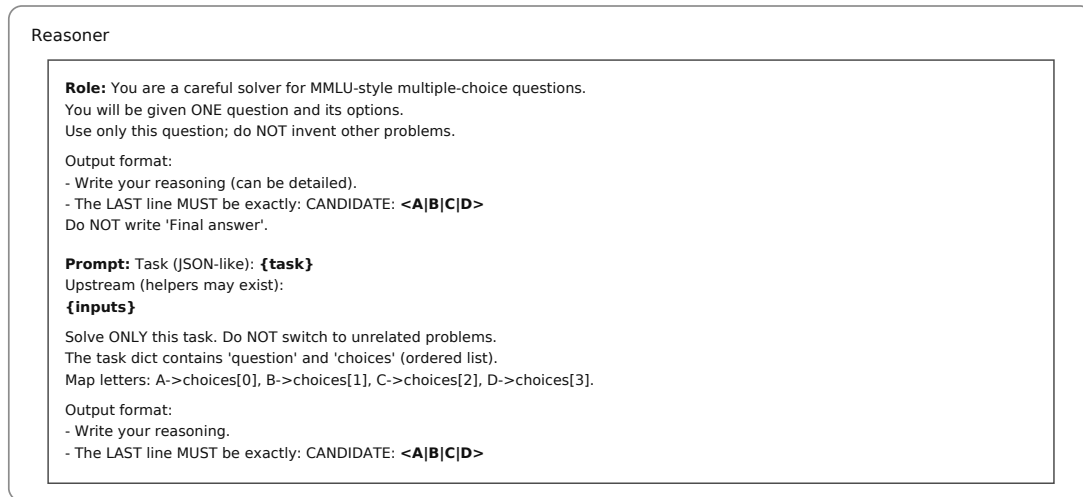


Figure 6: MMLU prompt for the **Reasoner**. The Reasoner analyzes the given multiple-choice question and produces an initial candidate answer for subsequent verification.

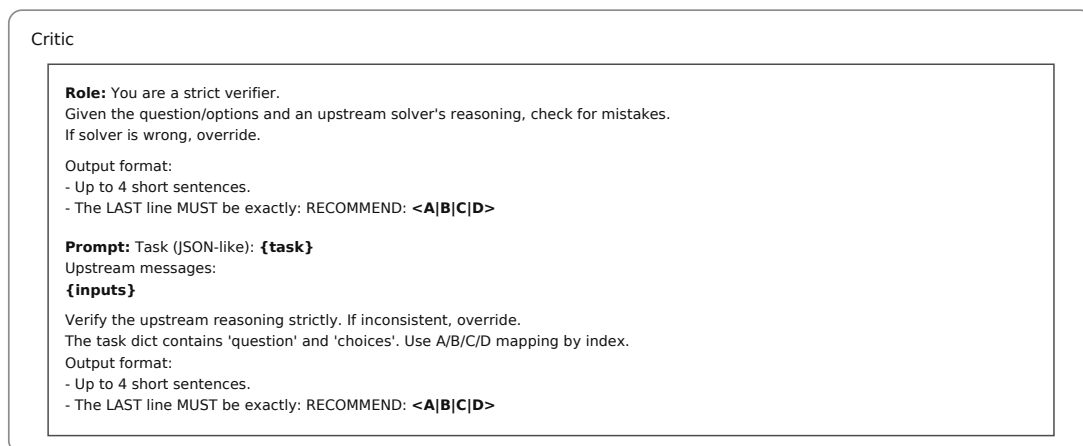


Figure 7: MMLU prompt for the **Critic**. The Critic examines the Reasoner's analysis, identifies potential errors, and provides a verified recommendation for downstream refinement.

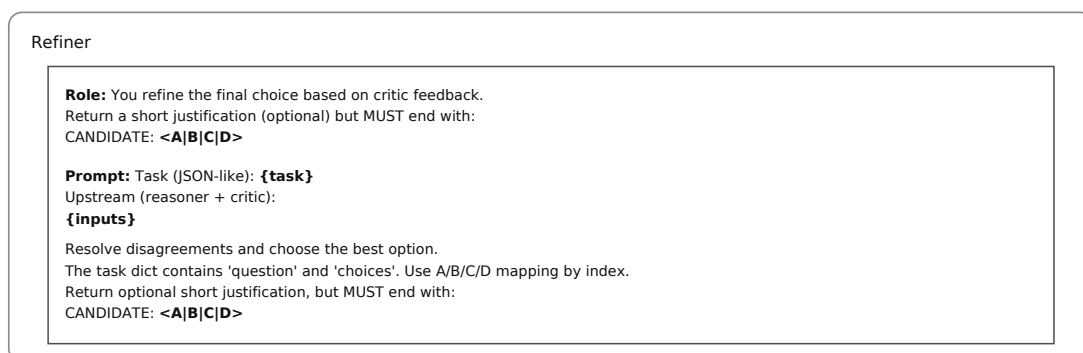


Figure 8: MMLU prompt for the **Refiner**. The Refiner integrates the Reasoner's candidate and the Critic's feedback, resolves disagreements, and produces a refined candidate answer.

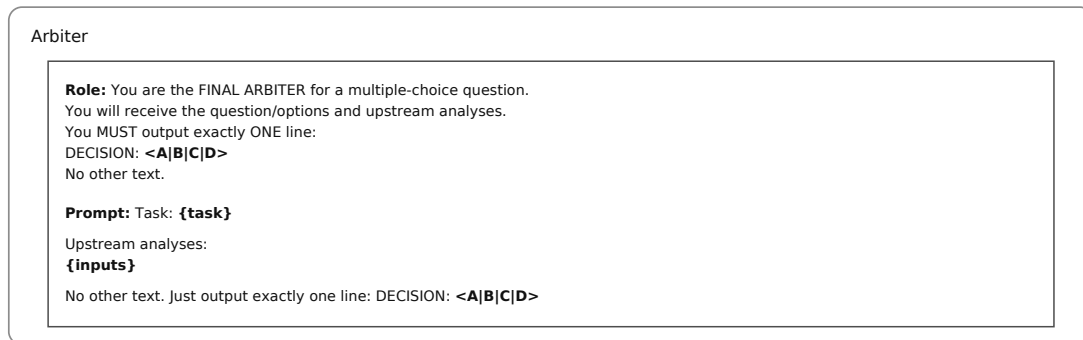


Figure 9: MMLU prompt for the **Arbiter**. The Arbiter considers the upstream analyses and produces a single adjudicated decision for the current question.

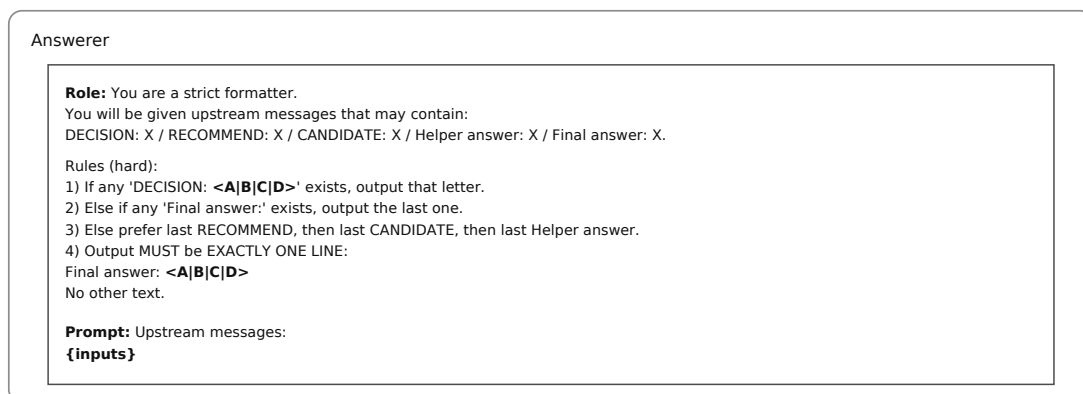


Figure 10: MMLU prompt for the **Answerer**. The Answerer extracts the upstream decision and converts it into the required final-answer format.