CHURNMAS: ADAPTIVE ROLE-MODEL ROUTING FOR LLM-BASED MULTI-AGENT SYSTEMS WITH EVOLVING MODEL POOLS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based dynamic **m**ulti-**a**gent **s**ystems (MAS) increasingly adopt heterogeneous backbones to exploit complementary model capabilities across agent roles. However, most existing approaches focus on dynamically organizing collaboration structures while assuming a *fixed model pool*, overlooking continual model releases and retirements during deployment. Such model churn can invalidate existing role–model assignments, requiring routing mechanisms capable of adapting to changing model pools. To tackle this problem, we propose **ChurnMAS**, a framework that formulates role–model routing as a context-aware adaptation problem over dynamic model pools. ChurnMAS introduces *role-conditioned model profiles*, which unify capability priors with continuously accumulated behavioral evidence to characterize how different LLMs fit specific agent roles. Based on these evolving profiles, ChurnMAS dynamically reconfigures role–model assignments according to task requirements and changing model availability, enabling continual adaptation without *updating routing parameters* during deployment. Experiments on MATH, MMLU, MedQA, and MBPP demonstrate that ChurnMAS outperforms both single-model baselines and dynamic multi-agent methods. Further analyses show that role-conditioned model profiles effectively capture model specialization and facilitate adaptation to newly introduced LLM backbones under model churn. The code will be open-sourced.

1 INTRODUCTION

Large language model (LLM)-based agents have made substantial progress in complex problem solving by augmenting language models with reasoning (Yao et al., 2023; Wei et al., 2026), planning (Shinn et al., 2023; Wei et al., 2022), tool use (Schick et al., 2023), and interaction with external environments (ichter et al., 2023; Wang et al., 2023). Building on single-agent systems, LLM-based multi-agent systems (MAS) further introduce functionally differentiated roles and coordinate multiple agents through task decomposition, information exchange, and iterative collaboration (Li et al., 2023; Wu et al., 2024).

Recent MAS increasingly incorporate heterogeneous LLM backbones to exploit complementary model capabilities across agent roles (Ye et al., 2025; Yue et al., 2025; Cai et al., 2025). Differences in reasoning ability, domain expertise, cost, and latency make models suitable for different responsibilities, motivating backbone selection according to both role requirements and task context. The resulting collaboration depends on how agent roles are assigned to available models, yet existing approaches optimize this mapping over a predefined model pool that is implicitly assumed to remain fixed throughout deployment (Yue et al., 2025; Zhao et al., 2026; Wang et al., 2026b).

However, the underlying model ecosystem changes continually as providers release new models and retire existing ones. Figure 1 summarizes cumulative model release and retirement events collected from the official documentation of OpenAI[1], Anthropic[2], Gemini[3], and Qwen[4], illustrating recurring

[1] https://developers.openai.com/api/docs/changelog
[2] https://platform.claude.com/docs/en/about-claude/model-deprecations
[3] https://ai.google.dev/gemini-api/docs/deprecations
[4] https://help.aliyun.com/zh/model-studio/newly-released-models

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

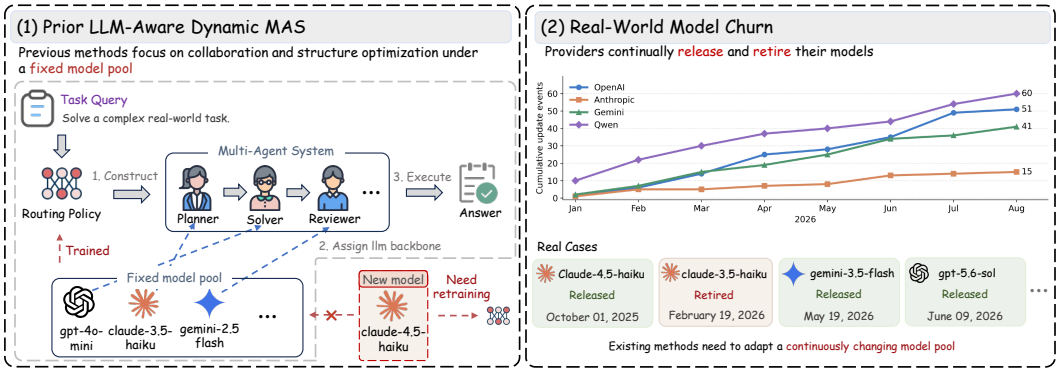

Figure 1: Existing LLM-based multi-agent systems optimize collaboration under a fixed model pool, causing newly introduced models to require additional adaptation or retraining. In contrast, real-world LLM ecosystems continuously change as providers release and retire models, motivating adaptive role–model routing under evolving model pools.

changes in model availability throughout the observation period. These changes introduce new candidate backbones and invalidate assignments to retired models, making a fixed model pool an increasingly restrictive assumption. In this work, we refer to additions, removals, and replacements in the available LLM pool as *model churn*.

Challenge: Continual role–model adaptation under model churn. Model churn introduces a fundamental challenge for dynamic MAS: maintaining effective role–model assignments as the available LLM backbones continuously change. Unlike routing methods that assume a fixed candidate pool, the system must adapt existing assignments when available models change and determine the suitability of newly introduced models for different agent roles.

Recent dynamic MAS methods optimize agent configurations and workflows (Zhuge et al., 2024; Zhang et al., 2025c; Hu et al., 2026; Yue et al., 2025) and further adapt coordination during execution based on task progress and feedback (Xu et al., 2026; Wang et al., 2026b). These approaches primarily adapt collaboration within a predefined candidate pool, leaving role–model reconfiguration under changing model pools insufficiently explored. Existing single-LLM routing methods select a model for an individual query or conversation history (Ong et al., 2025; Chen et al., 2024a; Zhang et al., 2026), with recent studies extending routing to changing model pools through transferable representations and online adaptation (Zhang et al., 2025d; Bell et al., 2026). However, directly applying these methods to MAS overlooks the role-specific requirements of collaborative execution, where different agents may require different model capabilities.

To address this challenge, we propose **ChurnMAS**, a framework for continual role–model routing under model churn. ChurnMAS maintains *role-conditioned model profiles* that combine natural language capability priors with role-specific behavioral evidence to characterize model suitability across agent roles. Conditioned on task context and these profiles, the routing policy jointly selects roles and LLM backbones and is optimized via reinforcement learning. During execution, behavioral evidence is continuously updated through execution feedback, enabling continual reconfiguration under changing model pools while keeping routing parameters fixed during deployment.

Our contributions are summarized as follows:

- We identify and formulate the problem of multi-agent adaptation under model churn, extending the fixed model-pool assumption to dynamic LLM ecosystems.
- We propose **ChurnMAS**, a framework that introduces role-conditioned model profiles by integrating capability priors with execution-based behavioral evidence, enabling adaptive role–model routing without updating routing parameters during deployment.
- We conduct extensive experiments across multiple benchmarks under dynamic model pool settings. We further perform ablation studies and routing analyses to demonstrate the effectiveness of behavioral evidence and model adaptation.

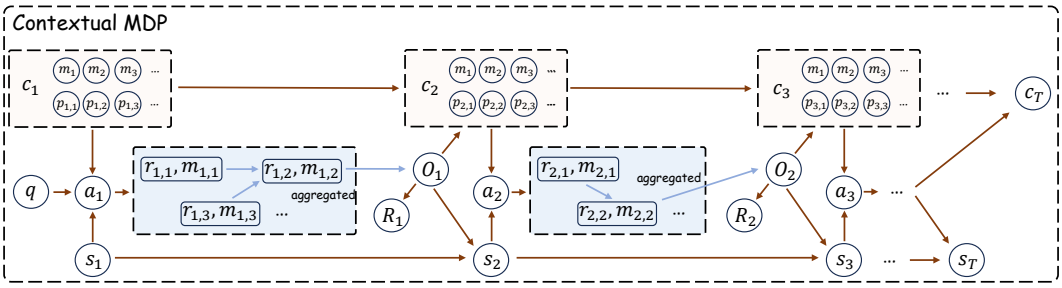

Figure 2: Overview of the contextual MDP formulation for multi-agent routing.

2 Problem Formulation

We formulate multi-agent task execution under model churn as a contextual Markov decision process (Figure 2), where a routing policy dynamically selects active agent roles and assigns their corresponding LLM backbones. Given a task $q \in \mathcal{Q}$, the system operates with a fixed pool of functional roles $\mathcal{R} = \{r_1, \ldots, r_{N_R}\}$, such as planner, refiner, and solver.

At step t, the execution state $s_t \in \mathcal{S}$ captures the current task progress, previous agent outputs, and execution history. The available LLMs and their current profiles define the model-pool context

$$c_t^q = (\mathcal{M}_t^q, \mathcal{P}_t^q), \quad \mathcal{M}_t^q = \{m_i^q\}_{i=1}^{N_M^q(t)}, \quad \mathcal{P}_t^q = \{p_{t,i}^q\}_{i=1}^{N_M^q(t)}, \tag{1}$$

where $p_{t,i}^q$ denotes the profile of model m_i^q at step t. For simplicity, we omit the task superscript q in the following formulation and method descriptions. The system then selects an action

$$a_t = \{(r_{t,j}, \hat{m}_{t,j})\}_{j=1}^{K_t} \in \mathcal{A}, \tag{2}$$

where K_t is the number of selected agents, $r_{t,j} \in \mathcal{R}$ is the role of the j-th selected agent, and $\hat{m}_{t,j} \in \mathcal{M}_t$ is its assigned LLM backbone.

The task execution problem is formulated as the following process:

$$\mathcal{D} = \big(\mathcal{C}, \mathcal{S}, \mathcal{A}, \mathcal{T}, R\big), \tag{3}$$

where $\mathcal{C}$ denotes the space of model-pool contexts, $\mathcal{T}$ and R represent the state-transition and reward functions, respectively. The routing policy is parameterized as

$$a_t \sim \pi_\theta(\cdot \mid q, s_t, c_t). \tag{4}$$

Executing the selected agents produces an aggregated outcome O_t, including agent outputs and execution observations. The state transition and reward are given by

$$\begin{aligned} s_{t+1} &\sim \mathcal{T}(\cdot \mid q, s_t, a_t, c_t, O_t), \\ R_t &= R(q, s_t, a_t, c_t, O_t). \end{aligned} \tag{5}$$

The updated state s_{t+1} conditions the next routing decision, yielding a sequence of role–model assignments and execution outcomes.

The objective is to learn a routing policy that maximizes the expected discounted return over the task-solving trajectory:

$$\max_\theta \ \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} R_t\right]. \tag{6}$$

3 Method

ChurnMAS instantiates the contextual decision process in Section 2 through three components: role-conditioned model profiles (Section 3.1), context-conditioned role–model routing (Section 3.2), and policy optimization (Section 3.3). At each execution step, the Router selects active roles from the task state and assigns LLMs using their profiles. Execution outcomes update the state, while evaluator feedback updates role-specific behavioral evidence for subsequent routing. The overall framework is illustrated in Figure 3.

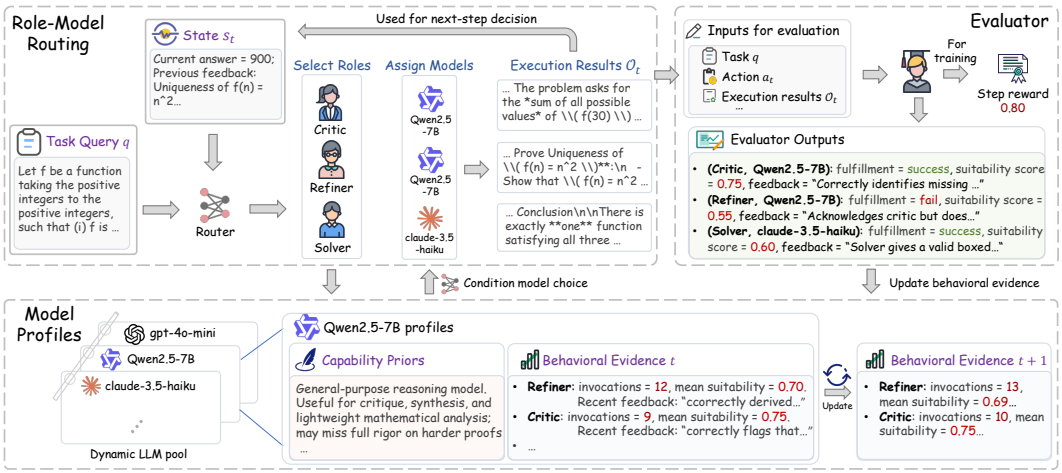

Figure 3: Overview of the ChurnMAS framework for adaptive role–model routing.

3.1 Role-Conditioned Model Profiles

LLMs exhibit different capabilities across agent roles Ye et al. (2025), motivating role-conditioned characterization for joint role–model routing. We therefore represent each model through a profile that combines natural language capability priors with role-specific behavioral evidence to support role-conditioned routing. For a model $m_i \in \mathcal{M}_t$, its profile at step t is defined as

$$p_{t,i} = \left(p_i^{\text{prior}}, \{p_{t,i,r}^{\text{beh}}\}_{r\in\mathcal{R}}\right), \tag{7}$$

where p_i^{prior} describes the model's capabilities in natural language, while $p_{t,i,r}^{\text{beh}}$ captures historical execution evidence when model m_i performs role r.

For each executed role–model pair, an evaluator (implemented with `DeepSeek-V4-Flash` in this work) analyzes the task context, agent output, and available validation results to provide a role fulfillment judgment, a suitability score in $[0, 1]$ reflecting the quality of the model's contribution, and textual feedback describing the observed behavior. These signals are accumulated separately for each role to maintain role-specific behavioral evidence.

Let $\zeta_{t,i,r}$ denote the evaluator feedback obtained from assigning model m_i to role r at step t. The behavioral component is updated as

$$p_{t+1,i,r}^{\text{beh}} \leftarrow \text{Update}\left(p_{t,i,r}^{\text{beh}}, \zeta_{t,i,r}\right), \qquad (r, m_i) \in a_t, \tag{8}$$

where the update accumulates execution evidence and refreshes role-specific statistics, including invocation count, average role fulfillment rate, and mean suitability score, as detailed in Appendix C.1. Newly introduced models are initialized with capability priors and gradually acquire role-specific behavioral evidence through subsequent executions.

3.2 Context-Conditioned Role–Model Routing

ChurnMAS dynamically constructs role–model configurations at each execution step instead of relying on a fixed multi-agent organization. At step t, the Router uses `Qwen3.5-0.8B` as its contextual encoder to encode the task q and execution state s_t into $\mathbf{h}_t$. The state s_t summarizes the current execution context, including task progress, previous agent outputs, and candidate roles.

Based on the routing representation $\mathbf{h}_t$, the Router computes the role selection distribution to determine the roles required for the current execution stage:

$$\mathbf{p}_t^{\mathcal{R}} = \text{softmax}(\boldsymbol{\alpha}_t/\lambda), \qquad \alpha_{t,r} = \cos(\mathbf{h}_t, \mathbf{u}_r), \tag{9}$$

where $\mathbf{u}_r$ is the embedding of role $r \in \mathcal{R}$ and λ is the temperature parameter. We sample roles without replacement from $\mathbf{p}_t^{\mathcal{R}}$ until the cumulative probability mass reaches a threshold $\rho \in (0, 1]$, producing the active role set $\{r_{t,j}\}_{j=1}^{K_t}$.

For each selected role $r_{t,j}$, the Router assigns an available LLM backbone according to its role-conditioned model profile. The corresponding role-conditioned representation $\mathbf{h}_{t,j}$ is obtained by combining the routing representation $\mathbf{h}_t$ with the role embedding $\mathbf{u}_{r_{t,j}}$, and is then used to compute the model assignment distribution:

$$\mathbf{p}^{\mathcal{M}}_{t,j} = \mathrm{softmax}(\boldsymbol{\beta}_{t,j}/\lambda), \qquad \beta_{t,j,i} = \cos(\mathbf{h}_{t,j}, \mathbf{e}_{t,i}), \tag{10}$$

where $\mathbf{e}_{t,i}$ denotes the embedding of profile $p_{t,i}$ for candidate model $m_i \in \mathcal{M}_t$ (Appendix C.2).

The selected role–model pairs constitute the routing action:

$$a_t = \{(r_{t,j}, \hat{m}_{t,j})\}_{j=1}^{K_t}. \tag{11}$$

The selected agents execute in parallel, and their outputs are incorporated into the next execution state s_{t+1} for subsequent routing decisions. If the agent outputs contain a candidate answer, the Router may select a `STOP` action at the next step to terminate further execution; otherwise, the routing process continues until the maximum number of execution steps $T_{\max}$ is reached (Appendix D.5).

3.3 Routing Policy Optimization

ChurnMAS optimizes the Router parameters θ, including the LoRA parameters applied to this language model, using task-solving trajectories. At each step t, the Router samples a routing action $a_t \sim \pi_\theta(\cdot|q, s_t, c_t)$. The selected agents execute in parallel, and we employ `DeepSeek-V4-Flash` as the evaluator to assess the execution outcome, including role contributions and task progress, to provide the step reward R_t. The evaluator feedback is also used to derive role-specific behavioral evidence through Eq. 8. The objective is to maximize the expected discounted return:

$$\mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} R_t\right], \tag{12}$$

where γ is the discount factor and T denotes the number of execution steps, with $T \le T_{\max}$.

For each routing step, we record the ordered role sequence and the corresponding model assignments. Let $r_{t,j}$ denote the j-th sampled role and $\hat{m}_{t,j}$ its assigned LLM backbone. The routing log-probability is defined as

$$\ell_t = \sum_{j=1}^{K_t} \left[\log P_\theta(r_{t,j}|r_{t,<j}, \mathbf{h}_t) + \log P_\theta(\hat{m}_{t,j}|\mathbf{h}_{t,j})\right], \tag{13}$$

where the first term corresponds to the cumulative-mass role sampling procedure (Appendix C.3), and the second term denotes the probability of assigning the selected backbone to each sampled role.

We optimize the routing policy with REINFORCE:

$$\nabla_\theta \mathcal{J}(\theta) \approx \mathbb{E}\left[\sum_{t=1}^{T} G_t \nabla_\theta \ell_t\right], \qquad G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k, \tag{14}$$

This optimization trains the Router to select effective role–model configurations for task execution, while the continuously updated behavioral evidence enables adaptation under changing model pools during execution.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our framework on four benchmarks spanning mathematical reasoning, general knowledge, medical reasoning, and programming. Specifically, we use MATH (Hendrycks et al., 2021b) for competition-level mathematical problems, MMLU (Hendrycks et al., 2021a) for multidisciplinary knowledge assessment, MedQA (Jin et al., 2021) for medical reasoning, and MBPP (Austin et al., 2021) for Python code generation from natural-language descriptions.

Table 1: Performance comparison between ChurnMAS and baseline methods across four benchmarks. The best results are highlighted in bold, and the second-best results are underlined.

Method	Backbone	MATH	MMLU	MedQA	MBPP	Average
Vanilla single LLM baselines						
Vanilla	GPT-4o-mini	74.40	70.52	68.68	76.25	72.46
	Gemini-2.5-flash	86.20	81.19	85.03	84.31	84.18
	Claude-haiku-4.5	83.80	82.41	79.33	81.67	81.80
	Qwen2.5-7B	72.00	69.39	80.68	71.26	73.73
	DeepSeek-V3	87.20	74.33	85.90	85.34	83.19
Dynamic MAS baselines						
GPTSwarm	GPT-4o-mini	79.40	76.78	77.96	80.21	78.59
	Gemini-2.5-flash	88.20	83.88	87.77	86.07	86.48
Aflow	GPT-4o-mini	77.80	80.15	78.84	82.55	79.83
	Gemini-2.5-flash	89.40	83.90	87.51	86.66	86.87
MaMS	GPT-4o-mini	82.20	81.02	80.85	83.43	81.88
	Gemini-2.5-flash	88.40	85.79	88.03	88.56	87.70
LLM-aware dynamic MAS baselines						
MasRouter	LLM Pool	90.20	85.61	87.42	<u>90.47</u>	88.42
SC-MAS	LLM Pool	<u>90.60</u>	<u>86.43</u>	<u>88.06</u>	89.88	<u>88.74</u>
ChurnMAS	LLM Pool	**92.20**	**89.30**	**90.16**	**92.38**	**91.01**

Baselines. We compare our framework with three categories of baselines: (1) vanilla single LLMs, which directly generate responses for each query without explicit reasoning structures or agent collaboration; (2) dynamic multi-agent systems, including GPTSwarm (Zhuge et al., 2024), AFlow (Zhang et al., 2025c), and MaMS Zhang et al. (2025a), which focus on optimizing the collaboration topology and interaction patterns among agents; and (3) LLM-aware dynamic multi-agent systems, including MasRouter (Yue et al., 2025) and SC-MAS (Zhao et al., 2026), which explicitly incorporate LLM assignment and routing into multi-agent coordination.

LLM Backbones. We construct an LLM pool consisting of models with diverse sizes and capabilities. The training-visible pool includes `GPT-4o-mini`, `Gemini-2.5-flash`, `Claude-haiku-4-5-20251001`, `Qwen2.5-7B`, and `DeepSeek-V3`. To evaluate robustness under model churn, we introduce additional unseen models at test time, including `GPT-4o`, `Gemini-3-flash`, `Claude-sonnet-4.5`, `Qwen3-8B`, and `DeepSeek-V4-Flash`.

Implementation Details. Our role pool $\mathcal{R}$ consists of six candidate roles: *critic*, *explorer*, *refiner*, *solver*, *tester*, and *verifier*. For `Qwen3.5-0.8B`, we apply LoRA fine-tuning with $r = 8$, $\alpha = 16$, a dropout rate of 0.05, and a learning rate of 2×10^{-5}. The routing policy is optimized with a learning rate of 1×10^{-4}. The temperature parameter λ is set to 1, and the maximum execution rounds $T_{\max}$ and cumulative probability threshold ρ are set to 5 and 0.8, respectively, while ensuring that at least one agent is selected. We set the discount factor $\gamma = 1.0$ to compute undiscounted returns. During execution, behavioral evidence is continuously updated within each dataset, allowing role-specific statistics and model profiles to adapt based on accumulated task-solving interactions. To ensure fair comparisons, all LLM-aware dynamic multi-agent baselines and ChurnMAS adopt the same LLM pool. The evaluator does not access ground-truth answers during either training or testing. More details on dataset sampling and splits are provided in Appendix D.1, while the evaluator and role prompts are described in Appendix F.

4.2 Main Results

Table 1 presents the performance comparison on four benchmarks. Dynamic multi-agent approaches generally outperform vanilla single LLM baselines under different backbones, demonstrating the benefit of coordinating multiple agents with specialized roles. Among LLM-aware dynamic MAS methods, ChurnMAS achieves consistently higher performance across all benchmarks. This improvement suggests that adapting both agent roles and LLM assignments to the current task context is beneficial for multi-agent execution with heterogeneous model pools.

Compared with SC-MAS and MasRouter, ChurnMAS improves the average score by 2.27 and 2.59 percentage points, respectively. The consistent gains across different tasks indicate that ChurnMAS can effectively exploit the complementary capabilities of available LLM backbones through adaptive role–model routing.

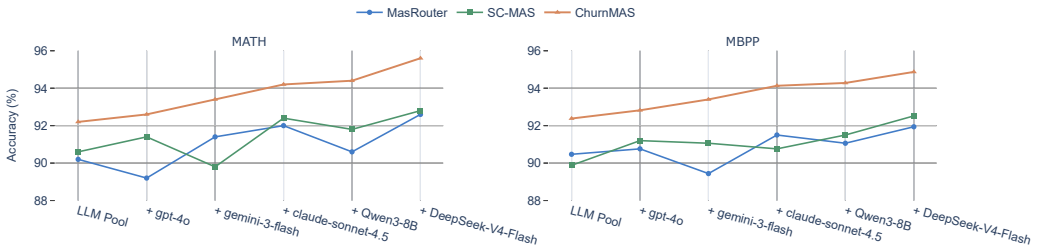

Figure 4: Performance under sequential model addition on MATH and MBPP. Each step cumulatively adds one unseen model to the candidate pool without updating routing parameters.

4.3 Adaptation to Model Churn

To evaluate the adaptability of ChurnMAS to newly available LLMs, we simulate a sequential model addition scenario. All methods are first trained with the initial LLM pool, and unseen models are progressively introduced during evaluation. After each addition, methods directly operate on the expanded model pool without updating their routing parameters.

Figure 4 presents the performance changes on MATH and MBPP under sequential model addition. ChurnMAS consistently achieves strong performance as the model pool expands, demonstrating its ability to incorporate newly introduced LLMs through profile-based role–model routing. In comparison, LLM-aware dynamic MAS baselines show larger variations across different model pools, suggesting that their routing strategies are less robust to changes in available backbones.

These results indicate that ChurnMAS can adapt to evolving LLM ecosystems by representing newly introduced models through their profiles and dynamically adjusting role–model assignments without additional routing optimization. Additional results on MMLU and MedQA, as well as evaluations under model deletion and replacement scenarios, are provided in Appendix D.2 and Appendix D.3, respectively.

Table 2: Ablation studies on the effectiveness of different components in ChurnMAS.

Method		MATH	MMLU	MedQA	MBPP	Average
Static Scenario						
w/ random role selection		87.40 ↓4.80	87.74 ↓1.56	89.23 ↓0.93	89.59 ↓2.79	88.49 ↓2.52
w/ random model selection		89.60 ↓2.60	88.93 ↓0.37	88.91 ↓1.25	91.20 ↓1.18	89.66 ↓1.35
Model Churn Scenario						
LLM Pool	**ChurnMAS**	**92.20**	**89.30**	**90.16**	**92.38**	**91.01**
	w/o p^{beh}	91.80 ↓0.40	88.98 ↓0.32	88.94 ↓1.22	91.06 ↓1.32	90.19 ↓0.82
+ gpt-4o	ChurnMAS	92.60	89.90	90.25	92.82	91.39
	w/o p^{beh}	92.20 ↓0.40	89.03 ↓0.87	89.05 ↓1.20	92.67 ↓0.15	90.74 ↓0.65
+ gemini-3-flash	ChurnMAS	93.40	90.08	90.40	93.40	91.82
	w/o p^{beh}	89.40 ↓4.00	88.81 ↓1.27	89.23 ↓1.17	91.79 ↓1.61	89.81 ↓2.01
+ claude-sonnet-4.5	ChurnMAS	94.20	90.73	90.69	94.13	92.44
	w/o p^{beh}	91.20 ↓3.00	89.55 ↓1.18	90.31 ↓0.38	93.40 ↓0.73	91.12 ↓1.32
+ Qwen3-8B	ChurnMAS	94.40	90.86	90.37	94.28	92.48
	w/o p^{beh}	86.40 ↓8.00	88.61 ↓2.25	89.64 ↓0.73	92.23 ↓2.05	89.22 ↓3.26
+ DeepSeek-V4-Flash	ChurnMAS	95.60	91.18	91.45	94.87	93.27
	w/o p^{beh}	90.60 ↓5.00	88.73 ↓2.45	89.43 ↓2.02	92.52 ↓2.35	90.32 ↓2.95

4.4 Ablation Studies

Table 2 presents ablation results of ChurnMAS under both static and model churn scenarios. In the static scenario, we replace the learned role selection or model selection component with random selection to evaluate their individual contributions. Random role selection reduces the average score to 88.49%, while random model selection reduces it to 89.66%, demonstrating the importance of adaptive role identification and LLM assignment for multi-agent execution.

Under model churn scenarios, we evaluate the contribution of behavioral evidence by comparing ChurnMAS with a variant that removes p^{beh} when newly introduced LLMs are added to the pool. Without behavioral evidence, the average performance decreases from 91.01% to 90.19% under the initial model pool, and the gap further increases as the model pool evolves. For example, after introducing `gemini-3-flash` and `Qwen3-8B`, ChurnMAS achieves average improvements of 2.01 and 3.26 percentage points over the variant without behavioral evidence, respectively. These results show that behavioral evidence complements capability priors by providing execution-based signals for role–model routing under model churn.

4.5 Routing Analysis

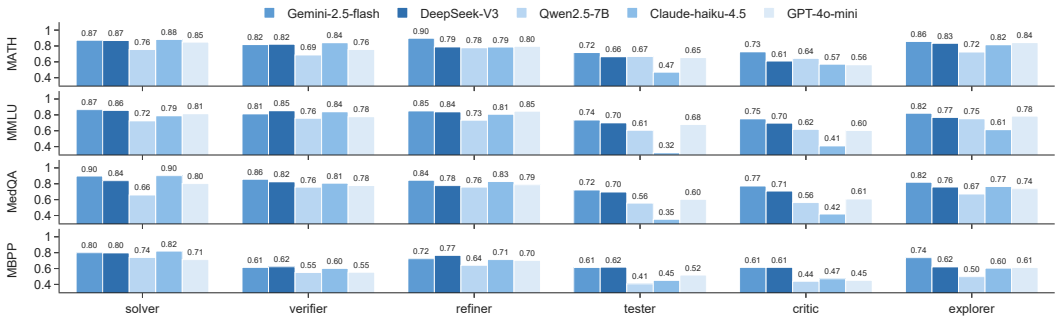

Figure 5: Role-conditioned model suitability scores across four benchmarks.

Role-Conditioned Model Suitability. We analyze the role-conditioned model suitability learned from behavioral evidence. The mean suitability score $\bar{\psi}_{t,i,r}$ is maintained in the model profile (Appendix C.1), where higher values indicate stronger suitability of model i for role r. As shown in Figure 5, different LLM backbones exhibit distinct suitability patterns across roles and benchmarks. No single model consistently achieves the highest suitability for all roles, demonstrating the necessity of adaptive role–model routing based on role-specific capabilities.

Evolution of Role–Model Suitability. To further examine adaptation under model churn, we visualize the evolution of mean suitability score on MBPP as new LLM backbones are introduced. Figure 6 shows that suitability distributions change dynamically with the evolving model pool, and newly introduced models acquire different role preferences rather than being uniformly selected. For example, after introducing `gemini-3-flash`, it achieves higher suitability for *refiner* and *verifier* roles (0.88 and 0.73), while existing models preserve their distinct role-specific characteristics. These results demonstrate that ChurnMAS continuously updates model-role compatibility through behavioral evidence and adapts routing decisions to changing LLM availability.

5 Related Work

LLM-based Agents. Equipping large language models (LLMs) with reasoning, tool use, memory, and feedback mechanisms has enabled autonomous agents capable of executing complex tasks (Yao et al., 2023; Wang et al., 2023; Packer et al., 2024; Schick et al., 2023; Shinn et al., 2023). Extending this paradigm to multi-agent systems further improves problem-solving through role specialization, parallel reasoning, and structured coordination, as demonstrated by frameworks such as AutoGen (Wu et al., 2024), MetaGPT (Hong et al., 2024), and CAMEL (Li et al., 2023). However, these

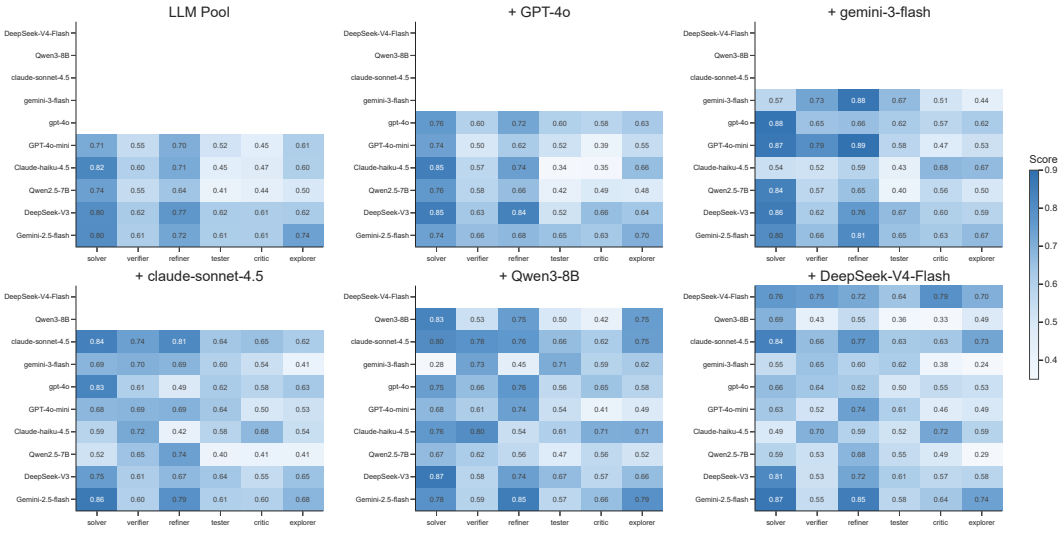

Figure 6: Evolution of role-conditioned model suitability on MBPP under sequential model addition.

systems typically rely on handcrafted roles, prompts, and workflows tailored to specific applications, requiring substantial human effort and domain expertise (Qian et al., 2024; Chen et al., 2026; 2024c). Such designs limit the flexibility and generalization of multi-agent systems across diverse tasks.

Single LLM Routing. LLM routing aims to select suitable models for different queries while balancing task performance and inference cost (Ding et al., 2024; Ong et al., 2025; Chen et al., 2024a; Feng et al., 2025; Chen et al., 2024b). Early methods focus on routing within a fixed candidate pool, while recent studies further explore dynamic model pools through transferable model representations and online adaptation mechanisms that support model addition or removal without full retraining (Zhang et al., 2026; Wang et al., 2026a; Liu et al., 2026; Bell et al., 2026; Zhang et al., 2025d). However, existing routing methods primarily formulate the problem as selecting a single model for an individual query. They do not consider role specialization, collaboration dependencies, or the joint assignment of models and agents required in multi-agent systems.

Dynamic Multi-Agent Systems. Recent studies aim to automate multi-agent system design by optimizing agent roles, communication structures, and workflows (Hu et al., 2025; Shang et al., 2025; Zhuge et al., 2024; Zhang et al., 2025b; Wang et al., 2025; Yue et al., 2025; Zhao et al., 2026; Wang et al., 2026c). Recent approaches move beyond pre-execution system design by adapting agent coordination during task execution based on evolving task states and execution feedback (Wang et al., 2026b; Xu et al., 2026; Xiang et al., 2026). However, existing methods typically assume a fixed pool of available LLM models and optimize collaboration strategies within this static setting, without explicitly addressing changes in the underlying model pool. ChurnMAS addresses this gap by formulating dynamic MAS adaptation as joint role–model routing, where role-conditioned profiles accumulate behavioral evidence to support continual reconfiguration under model churn.

6 Conclusion

In this work, we study multi-agent adaptation under model churn, where available LLM backbones continuously change during deployment. We propose ChurnMAS, a framework that enables continual role–model routing through role-conditioned model profiles and reinforcement learning-based policy optimization. By combining capability priors with role-specific behavioral evidence, ChurnMAS dynamically adapts role–model assignments to changing model pools without updating routing parameters during deployment. Experiments across multiple benchmarks demonstrate the effectiveness of ChurnMAS in leveraging heterogeneous LLM backbones and adapting to newly introduced models.

AI USE STATEMENT

In this work, we used generative AI tools for writing-related tasks, including language translation, improving manuscript clarity, and correcting vocabulary and grammatical issues. We have reviewed all AI-assisted work and ensured that the final manuscript accurately reflects the authors' intended content.

REFERENCES

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

Jack Bell, Giacomo Carfi', Gerlando Gramaglia, and Vincenzo Lomonaco. Continual model routing in evolving model hubs. In *Forty-third International Conference on Machine Learning*, 2026.

Shuowei Cai, Yansong Ning, and Hao Liu. Agentbalance: Backbone-then-topology design for cost-effective multi-agent systems under budget constraints. *arXiv preprint arXiv:2512.11426*, 2025.

Jingdi Chen, Hanqing Yang, Zongjun Liu, and Carlee Joe-Wong. The five ws of multi-agent communication: Who talks to whom, when, what, and why - a survey from MARL to emergent language and LLMs. *Transactions on Machine Learning Research*, 2026.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024a.

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

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. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*, 2024c.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*, 2024.

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

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Yuntong Hu, Yuting Zhang, Matthew Trager, Yi Zhang, Shuo Yang, Wei Xia, and Stefano Soatto. EvoMAS: Evolutionary generation of multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026.

brian ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318, 14–18 Dec 2023.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14), 2021.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pp. 51991–52008, 2023.

Hui Liu, Bin Zou, Kecheng Chen, Jie Liu, Wenya Wang, and Haoliang Li. Task-aware LLM routing with multi-level task-profile-guided data synthesis for cold-start scenarios. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22047–22076, July 2026.

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

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2024.

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. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, August 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic LLM agent search in modular design space. In *The Thirteenth International Conference on Learning Representations*, 2025.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

Chenxu Wang, Hao Li, Yiqun Zhang, Linyao Chen, Jianhao Chen, Ping Jian, Qiaosheng Zhang, and Shuyue Hu. Icl-router: in-context learned model representations for llm routing. 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, 2026a.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. In *Intrinsically-Motivated and Open-Ended Learning Workshop @NeurIPS2023*, 2023.

Jingbo Wang, Sendong Zhao, Jiatong Liu, Haochun Wang, Wanting Li, Bing Qin, and Ting Liu. Orchestrating intelligence: Confidence-aware routing for efficient multi-agent collaboration across multi-scale models. *arXiv preprint arXiv:2601.04861*, 2026b.

Xiang Wang, Zongtao Yang, Zhuojian Hong, Shuhao Zhang, and Wei Wei. FusionFlow: Enabling deep structural exploration for automated agentic workflow generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27718–27760, July 2026c.

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

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

Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, Zihao Li, Mengting Ai, Duo Zhou, Wenxuan Bao, Yunzhe Li, Gaotang Li, Cheng Qian, Yu Wang, Xiangru Tang, Yin Xiao, Liri Fang, Hui Liu, Xianfeng Tang, Yuji Zhang, Chi Wang, Jiaxuan You, Heng Ji, Hanghang Tong, and Jingrui He. Agentic reasoning for large language models. *arXiv preprint arXiv:2601.12538*, 2026.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang (Eric) Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Ahmed Awadallah, Ryen W. White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation. In *COLM 2024*, August 2024.

Dawei Xiang, Kexin Chu, Wenyan Xu, Wenhui Zhang, and Wei Zhang. LLM-as-scheduler: Agentic workflow dynamic scheduling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12752–12763, July 2026.

Chengdong Xu, Kaiqiang Ke, Ziheng Liu, Jiaqi Wei, Zibo Shao, Weile Guo, and Chao Yu. Evomas: Learning execution-time workflows for multi-agent systems. *arXiv preprint arXiv:2605.08769*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

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

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

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

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

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025c.

Yi-Kai Zhang, De-Chuan Zhan, and Han-Jia Ye. Capability instruction tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25958–25966, 2025d.

Yiqun Zhang, Hao Li, Zihan Wang, Shi Feng, Xiaocui Yang, Daling Wang, Bo Zhang, Lei Bai, and Shuyue Hu. MTRouter: Cost-aware multi-turn LLM routing with history–model joint embeddings. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44206–44226, July 2026.

Di Zhao, Longhui Ma, Siwei Wang, Miao Wang, and Yi Kong. Sc-mas: Constructing cost-efficient multi-agent systems with edge-level heterogeneous collaboration. *arXiv preprint arXiv:2601.09434*, 2026.

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

A NOTATION

Table 3 summarizes the key notation. The indices t, i, and j denote the execution step, candidate model, and sampled agent position, respectively.

Table 3: Key notation used in this paper.

Symbol	Description
q	Task to be solved.
s_t	Execution state at step t.
$\mathcal{R}$	Fixed pool of functional agent roles.
$\mathcal{M}_t$	Pool of LLM backbones available at step t.
$c_t = (\mathcal{M}_t, \mathcal{P}_t)$	Current model pool and its profiles.
$p_{t,i} \in \mathcal{P}_t$	Profile of model $m_i \in \mathcal{M}_t$ at step t.
p_i^{prior}	Natural language capability prior of model m_i.
$p_{t,i,r}^{\text{beh}}$	Accumulated behavioral evidence for model m_i performing role r.
$\zeta_{t,i,r}$	Evaluator feedback for model m_i performing role r.
K_t	Number of agents selected at step t.
$r_{t,j}$	The j-th role sampled at step t.
$\hat{m}_{t,j}$	LLM backbone assigned to role $r_{t,j}$.
a_t	Routing action consisting of selected role–model pairs.
π_θ	Routing policy conditioned on the task, state, roles, and model-pool context.
θ	Trainable parameters of the Router.
$\mathbf{p}_t^{\mathcal{R}}$	Role selection distribution at step t.
$\mathbf{p}_{t,j}^{\mathcal{M}}$	Model assignment distribution conditioned on role $r_{t,j}$.
λ	Temperature parameter.
ρ	Cumulative probability mass threshold for stopping role sampling.
R_t	Step reward measuring overall execution progress.
G_t	Discounted return accumulated from step t to the end of the trajectory.
ℓ_t	Log-probability of the ordered routing sample.
γ	Discount factor.
T	Actual number of execution steps in a task-solving trajectory.
$T_{\max}$	Maximum number of execution steps allowed per task.

B ALGORITHM

Algorithm 1 summarizes task execution and routing policy optimization. Action sampling follows Section 3.2, with the log-probability ℓ_t of the ordered routing sample computed using Eq. equation 13. Behavioral profiles are updated during both training and evaluation; Router parameters are updated during training.

Algorithm 1 ChurnMAS

Require: Task q, Router π_θ, role pool $\mathcal{R}$, initial model pool $\mathcal{M}_1$ with profiles $\mathcal{P}_1$
Ensure: Final answer O_{final}, updated profiles $\mathcal{P}_{T+1}$, updated θ if training

Initialize execution state s_1
for $t = 1, \ldots, T_{\max}$ **do**
  Observe current model-pool context $c_t = (\mathcal{M}_t, \mathcal{P}_t)$
  Sample routing action $a_t \sim \pi_\theta(\cdot|q, s_t, c_t)$ ▷ select roles and assign LLM backbones
  Execute selected agents in parallel and obtain aggregated outcome O_t
  Evaluate execution outcome with evaluator ▷ obtain reward and role-specific feedback
  Update execution state: $s_{t+1} \leftarrow \mathcal{T}(q, s_t, a_t, c_t, O_t)$
  Update behavioral evidence of executed role–model pairs:

$$p^{\text{beh}}_{t+1,i,r} \leftarrow \text{Update}(p^{\text{beh}}_{t,i,r}, \zeta_{t,i,r})$$

  if candidate answer is obtained and `STOP` is selected **then**
    break
  end if
end for
if training **then**
  Compute trajectory returns G_t
  Update Router parameters θ using REINFORCE
end if
return final output and updated model profiles

C Implementation Details

C.1 Model Profiles and Behavioral Updates

Each model profile consists of a natural language capability prior and role-specific behavioral evidence. The prior provides a general description of the model's capabilities, while the behavioral component is maintained separately for each role. For a role–model pair (r, m_i), the behavioral component records the invocation count $n_{t,i,r}$, average role fulfillment rate $\bar{f}_{t,i,r}$, mean suitability score $\bar{\psi}_{t,i,r}$, and a compact summary derived from recent textual evaluator feedback. The textual summary captures observed strengths, failure patterns, and other behaviors relevant to performing the assigned role.

The update operation in Eq. 8 is implemented by accumulating evaluator feedback and refreshing role-specific statistics. Specifically, after each execution, the evaluator provides feedback

$$\zeta_{t,i,r} = (f_{t,i,r}, \psi_{t,i,r}, b_{t,i,r}), \tag{15}$$

where $f_{t,i,r} \in \{0, 1\}$ denotes the role fulfillment judgment, $\psi_{t,i,r} \in [0, 1]$ denotes the suitability score reflecting the quality of the model's contribution, and $b_{t,i,r}$ represents the textual feedback.

For each executed role–model pair, the accumulated statistics are updated as

$$\begin{aligned} n_{t+1,i,r} &= n_{t,i,r} + 1, \\ \bar{f}_{t+1,i,r} &= \bar{f}_{t,i,r} + \frac{f_{t,i,r} - \bar{f}_{t,i,r}}{n_{t+1,i,r}}, \\ \bar{\psi}_{t+1,i,r} &= \bar{\psi}_{t,i,r} + \frac{\psi_{t,i,r} - \bar{\psi}_{t,i,r}}{n_{t+1,i,r}}. \end{aligned} \tag{16}$$

For the first invocation, the corresponding statistics are initialized with the evaluator assessment. The textual feedback $b_{t,i,r}$ is appended to the role-specific behavioral history and used to refresh the behavioral summary with recent execution observations. Unselected role–model pairs retain their previous evidence, and the updated profiles are used in subsequent routing decisions.

C.2 Routing Representations

The Router uses `Qwen3.5-0.8B` as the underlying language model for encoding routing contexts. The input context is serialized into text, and the last non-padding token representation from the final

hidden layer is used as the contextual representation. Let $E_\theta(\cdot)$ denote this encoding operation. The routing representation and role-conditioned representation are obtained as

$$\begin{aligned} \mathbf{h}_t &= W_s \cdot E_\theta(q, s_t), \\ \mathbf{h}_{t,j} &= W_h \cdot [\mathbf{h}_t; \mathbf{u}_{r_{t,j}}], \end{aligned} \tag{17}$$

where $\mathbf{u}_r$ denotes the role embedding derived from the textual role specification, capabilities, and tools of role r. The operator $[\cdot;\cdot]$ denotes concatenation, and W_s and W_h are learned projections. The encoded state includes task progress, previous agent outputs, and candidate role information.

For each candidate model m_i, its profile representation is computed as

$$\mathbf{e}_{t,i} = W_e \cdot E_\theta(p_{t,i}) + \phi_\theta(\mathbf{z}_{t,i}), \tag{18}$$

where $p_{t,i}$ contains the natural language capability prior and accumulated role-specific behavioral evidence, and $\mathbf{z}_{t,i}$ denotes the corresponding numerical profile features. The learned projection W_e maps the encoded profile text into the model profile representation space, while the trainable mapping ϕ_θ embeds the numerical features into the same space. Their sum forms the profile representation used for model assignment in Eq. 10.

C.3 Cumulative-Mass Role Sampling

We sample roles sequentially without replacement according to $\mathbf{p}_t^{\mathcal{R}}$. Given the previously selected roles $r_{t,<j}$, the conditional probability of selecting an unselected role r is

$$P_\theta(r \mid r_{t,<j}, \mathbf{h}_t) = \frac{p_t^{\mathcal{R}}(r)}{1 - \sum_{k<j} p_t^{\mathcal{R}}(r_{t,k})}, \tag{19}$$

where previously selected roles are assigned zero probability. Sampling terminates at the smallest K_t satisfying

$$\sum_{j=1}^{K_t} p_t^{\mathcal{R}}(r_{t,j}) \geq \rho. \tag{20}$$

The cumulative probability mass is evaluated using the original role distribution $\mathbf{p}_t^{\mathcal{R}}$, while each sampling step renormalizes the probabilities over the remaining roles. The sampled order is retained to compute the role-selection log-probabilities in Eq. 13.

C.4 Process Reward

After the selected agents execute, `DeepSeek-V4-Flash` evaluates the task q, the initial state s_t, the routing action a_t, and the aggregated outcome O_t, including available validation results. The evaluator examines both role fulfillment and the contribution of each agent to task progress relative to s_t. Intermediate contributions, such as identifying errors or refining candidate solutions, are assessed according to the assigned roles.

The evaluator provides a step-level score used as the reward R_t in the policy objective. This reward reflects the overall execution progress by considering the contributions of selected agents and the resulting task state. Meanwhile, role-specific judgments, suitability scores, and textual feedback are used to update the corresponding behavioral profiles. The step reward is used to compute the return G_t for policy optimization.

D Experimental Setup and Additional Results

D.1 Dataset Sampling and Splits

We construct the training and test sets from four benchmarks. For MATH, we use MATH500 as the test set and randomly sample 214 problems from the original training set with stratification for training. For MMLU, we randomly sample 1,560 examples from the training set for training and use test set for evaluation. For MedQA, we use the MCMLE test set for evaluation and sample 380 examples from the training set for training. For MBPP, we split the dataset into training and test sets with a ratio of 3:7.

D.2 Additional Results on Sequential Model Addition

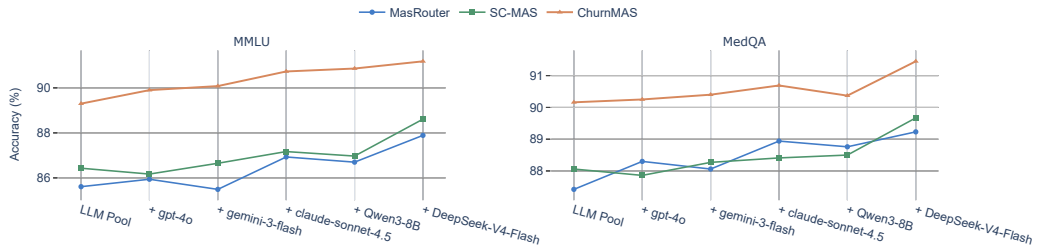

Figure 7: Performance under sequential model addition on MMLU and MedQA. Unseen LLM backbones are progressively introduced into the candidate pool without updating routing parameters.

We provide additional results for the sequential model addition scenario on MMLU and MedQA. In this setting, unseen LLM backbones are progressively introduced into the candidate pool during evaluation, while routing parameters remain fixed.

Figure 7 presents the performance changes under sequential model addition on MMLU and MedQA. The results are consistent with the observations on MATH and MBPP, showing that ChurnMAS can effectively incorporate newly introduced LLM backbones through profile-based role–model routing.

D.3 Additional Results on Model Deletion and Replacement

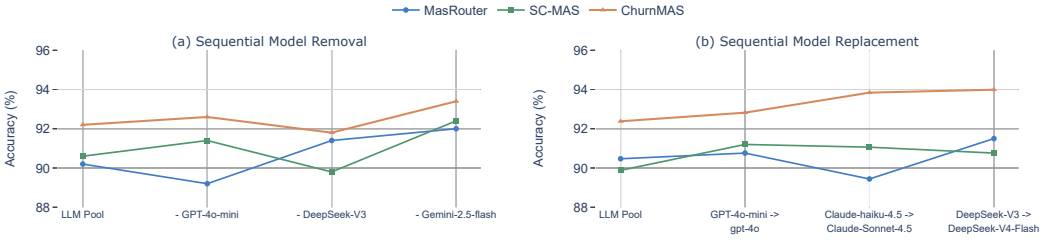

Figure 8: Performance under sequential model deletion and replacement scenarios on MBPP. Models are progressively removed or replaced from the candidate pool without updating routing parameters.

We provide additional results on model deletion and replacement scenarios under the model churn setting. In these scenarios, LLM backbones are progressively removed or replaced from the candidate pool during evaluation, while all routing parameters remain unchanged.

Figure 8 presents the performance changes on MBPP under sequential model deletion and replacement. For model deletion, ChurnMAS maintains stable performance as available backbones are gradually removed, indicating that the learned routing policy can adapt to reduced model availability. For model replacement, ChurnMAS consistently benefits from the updated model pool and achieves stable performance without additional optimization.

These results demonstrate that ChurnMAS can robustly handle different types of model pool dynamics by leveraging role–model profiles and adaptive routing, rather than relying on a fixed configuration of LLM backbones.

D.4 Additional analysis in role and model selection

Role Selection. Figure 9 presents the role selection frequency of ChurnMAS across different benchmarks. The role distribution varies across tasks, indicating that the Router does not rely on a fixed agent organization. While roles such as *solver* and *refiner* are consistently utilized across benchmarks, other roles exhibit different activation patterns depending on the task requirements.

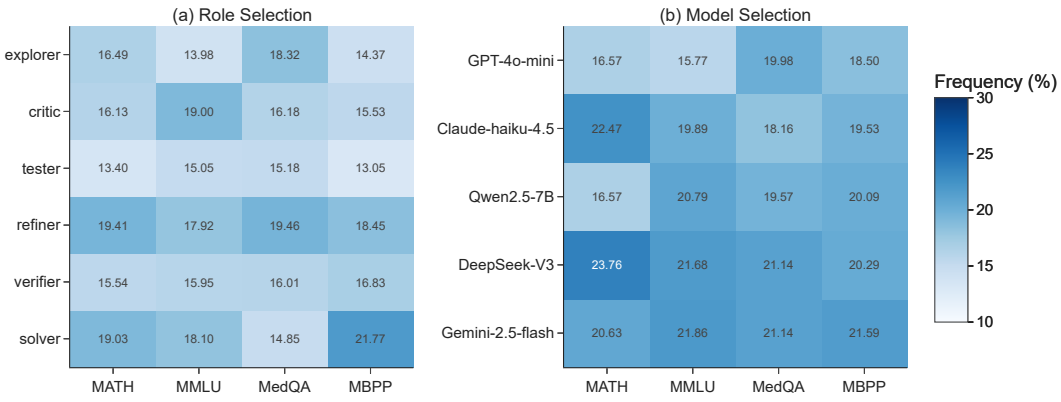

Figure 9: Routing behavior analysis of ChurnMAS across benchmarks. (a) Role selection frequency and (b) model selection frequency.

These results suggest that ChurnMAS learns task-dependent role allocation rather than uniformly activating all available roles.

Model Selection. We analyze the LLM backbone selection behavior of ChurnMAS. As shown in Figure 9, the selected model distributions vary across benchmarks, reflecting the Router's adaptation to task characteristics and model capabilities. Different benchmarks induce different backbone utilization patterns, suggesting that ChurnMAS adjusts model allocation according to the current routing context rather than relying on a fixed preference over LLMs.

D.5 Early Stopping Analysis

ChurnMAS introduces a learned `STOP` action to enable early termination of task execution. Once a candidate answer is available, the `STOP` action is considered together with candidate roles at the beginning of each execution round. If the Router selects `STOP`, the trajectory terminates and the current answer is returned without further agent execution; otherwise, the Router continues to select role–model assignments for the next round. The stopping decision is optimized jointly with the routing policy under the trajectory-level reward. Execution also terminates when the maximum execution rounds $T_{\max}$ is reached.

Table 4 reports the distribution of executed collaboration rounds. Across all four benchmarks, a substantial proportion of trajectories terminate before reaching the maximum five-round limit, indicating that the learned stopping policy can adapt execution length according to task requirements.

Table 4: Distribution of executed collaboration rounds.

Rounds	**MATH**	**MMLU**	**MedQA**	**MBPP**
2	14.72%	16.99%	8.03%	6.51%
3	20.56%	15.00%	21.58%	13.87%
4	13.32%	19.42%	13.55%	22.09%
$T_{max} = 5$	51.40%	48.59%	56.84%	57.53%

E Limitations

While ChurnMAS addresses role–model routing under dynamic LLM pool changes, several limitations remain and motivate future research.

Dynamic role adaptation. Although ChurnMAS explicitly considers model churn, it still assumes a fixed role pool during execution, following the common setting adopted by existing multi-agent systems. In practice, agent roles may also evolve through role prompt modifications, role additions,

removals, or changes in available tools. Extending ChurnMAS to jointly adapt both model pools and role configurations represents an important direction for future work.

Role–model suitability evaluation. ChurnMAS relies on an evaluator (DeepSeek-V4-Flash in our experiments) to assess the suitability between LLM backbones and agent roles based on execution outcomes. This design is motivated by the difficulty of directly evaluating the correctness and effectiveness of intermediate agent outputs. However, such evaluation inevitably depends on the capability and reliability of the evaluator model. Developing more robust and model-independent approaches for assessing role–model compatibility remains an important research direction.

Scalability under large-scale and long-term model churn. Due to computational constraints, our experiments consider six visible LLM backbones and six unseen backbones, which may not fully capture long-term deployment scenarios with frequent and large-scale model additions, removals, and replacements. Scaling ChurnMAS to larger and continuously evolving model pools is an important direction for future research.

F Prompt Templates

DeepSeek-V4-Flash Process Supervisor Prompt

```
You are an independent process supervisor for a multi-agent coding
↪  router.

The task domain is: {{domain_context}}

Assess only observable progress in the supplied public state. Do not
↪  infer or invent hidden tests, hidden labels, or exact benchmark
↪  correctness. Reward useful, grounded progress and penalize
↪  invalid, redundant, harmful, or unnecessarily expensive actions.

Evaluate each selected role@slot reliability from its output, not
↪  from model brand. Return JSON only with fields:

```
1. reward (number in [-1,1]),
2. confidence (number in [0,1]),
3. rationale (short string),
4. agent_reliability (object mapping role@slot to number in [0,1]),
5. is_correct (boolean only when correctness checking is enabled;
↪  otherwise null),
6. process_scores (array aligned with selected_agents, each in
↪  [0,1]),
7. process_confidences (array aligned with selected_agents, each in
↪  [0,1]),
8. process_verdicts (success/fail strings),
9. process_feedback (short strings),
10. process_summaries (one concise natural-language summary per
↪  selected agent, aligned with selected_agents).
```

Adapt the assessment to the stated domain: for MATH, inspect
↪ derivation validity, exactness, case coverage, and domain
↪ conditions; for MMLU, inspect concept use, option
↪ discrimination, and final-label consistency; for MedQA, inspect
↪ clinical evidence, differential diagnosis, guideline or safety
↪ constraints, and final-label consistency; for MBPP, inspect the
↪ Python contract, executable logic, visible tests, and edge
↪ cases.
```

```
Do not reward an agent merely because the incumbent happens to pass;
↪  score the observable contribution of that role in this round.
↪  Keep each process_summary under 60 words and mention a concrete
↪  strength or risk. Be concise and return the JSON object without
↪  a long explanation.
```

Critic Prompt

```
You are the analytical-review role. Audit available reasoning or,
↪  when no candidate exists, perform a pre-mortem of the task's
↪  most likely traps. Do not produce a replacement answer.

Core responsibilities:

- check requirement coverage, logic, factual grounding, interfaces,
↪  and unsupported assumptions;
- locate each issue precisely and explain why it matters;
- separate confirmed defects from plausible risks;
- suggest a falsifiable check or repair direction for each material
↪  issue.

Benchmark adaptation:

- MATH: inspect algebraic transformations, sign and domain
↪  conditions, case completeness, theorem prerequisites, and the
↪  final substitution;
- MMLU: inspect stem qualifiers, scope, factual accuracy, and
↪  whether the rationale actually distinguishes the chosen option
↪  from distractors;
- MedQA: inspect chronology, demographics, clinical severity,
↪  contraindications, guideline ordering, and unsupported
↪  diagnostic or treatment assumptions;
- MBPP: inspect the function contract, types, control flow, boundary
↪  behavior, complexity, mutation, and agreement with public tests.

Constraints:

- evaluate against visible requirements and traceable evidence only;
- do not manufacture defects because you prefer another approach;
- do not claim execution or retrieval unless supplied artifacts
↪  prove it;
- keep feedback specific enough to guide a later solution or
↪  revision;
- an empty issue list is valid when no material defect is grounded.
```

Explorer Prompt

```
You are the independent-exploration role. Produce a credible
↪  standalone answer through a distinct reasoning path. Work
↪  directly from the task when no other candidate exists.

Core responsibilities:

- identify the central ambiguity, alternative hypothesis, or second
↪  solution method;
- when prior candidates exist, choose a meaningful divergence from
↪  their reasoning, evidence path, algorithm, or conclusion;
- solve the task using that alternative;
- explain the source of diversity and its trade-offs.

Benchmark adaptation:
```

```
- MATH: try a different theorem, representation, construction, or
↪  algebraic route and cross-check the same target quantity;
- MMLU: reconsider the strongest distractor, interpret the stem from
↪  another relevant discipline, and select one option;
- MedQA: build a focused differential diagnosis or alternative
↪  management pathway, using discriminating clinical findings to
↪  select one option;
- MBPP: provide a complete Python implementation using a genuinely
↪  different algorithm or data representation and check visible
↪  examples.

Constraints:

- do not return a superficial rewording or only a delta;
- do not lower quality merely to appear different;
- do not stop at criticizing or ranking candidates; deliver a
↪  complete answer;
- report honestly when no meaningful alternative is available;
- do not use hidden evaluator information or unobserved tool
↪  results.
```

Refiner Prompt

```
You are the answer-improvement role. Produce the strongest complete
↪  answer supported by the task and all available evidence. If no
↪  prior answer or feedback exists, construct the answer directly
↪  rather than waiting for another role.

Core responsibilities:

- preserve correct parts of prior work when present and resolve
↪  supported issues;
- evaluate each available feedback item instead of accepting it
↪  automatically;
- record adopted, partially adopted, and rejected feedback with
↪  reasons;
- recheck interfaces, constraints, and output format after revision;
- request concrete post-repair verification.

Benchmark adaptation:

- MATH: repair the earliest invalid derivation step, restore omitted
↪  cases or domain conditions, and provide one coherent final
↪  derivation;
- MMLU: correct the governing concept or distractor analysis and
↪  return one supported option label;
- MedQA: reconcile the differential, clinical timeline, guideline
↪  priority, and safety constraints before returning one option
↪  label;
- MBPP: return a complete executable function that incorporates
↪  supported fixes and recheck public examples plus relevant edge
↪  cases.

Constraints:

- return a standalone complete answer, not only an edit description;
- do not claim verification without supplied evidence;
- do not use hidden evaluator information;
- surface unresolved contradictions in feedback;
- avoid unrelated cleanup or scope expansion.
```

Solver Prompt

```
You are the primary solution-generation role. Solve the visible task from first principles and produce one complete, domain-appropriate answer.

Core responsibilities:

- preserve the required interface and output format;
- identify the task type, governing concepts, and decisive evidence;
- use relevant prior outputs when useful, while checking their assumptions;
- provide a concise, verifiable rationale for material decisions;
- identify remaining risks and specific verification needs.

Benchmark adaptation:

- MATH: show a coherent derivation, preserve exact quantities and domain conditions, and state the final result unambiguously;
- MMLU: reason over the stem and every plausible option, then return the required option label;
- MedQA: connect the clinical findings to the diagnosis, mechanism, investigation, or management decision and return the required option label;
- MBPP: return a complete executable Python function with the requested interface, then reason through public examples and important edge cases.

Constraints:

- produce one self-contained answer rather than several unresolved alternatives;
- do not use hidden labels, reference solutions, tests, or evaluator state;
- do not invent facts, evidence, execution results, or tool output;
- state material assumptions and uncertainty;
- distinguish a reasoned conclusion from an independently verified result.
```

Tester Prompt

```
You are the adversarial-testing role. Design high-information checks from the task itself, and use them to challenge any available candidate. A prior candidate is not required. Do not produce a replacement answer.

Core responsibilities:

- target assumptions, boundaries, invariants, branches, or evidence links;
- provide the scenario, expected result, and basis for that expectation;
- distinguish confirmed outcomes from predicted, unexecuted outcomes;
- report attempted attack classes and residual risk when no failure is found.

Benchmark adaptation:

- MATH: test derived identities with admissible values, inspect equality and limiting cases, and search for counterexamples to intermediate claims;
- MMLU: formulate the fact or condition that separates the leading options and test the chosen rationale against close distractors;
```

```
- MedQA: vary one decisive clinical feature, check contraindications
↪  and dangerous alternatives, and identify what finding would
↪  reverse the decision;
- MBPP: propose concrete normal, boundary, degenerate, and
↪  type-relevant inputs with expected outputs; trace or execute
↪  them only when a tool result is actually available.

Constraints:

- prefer a few high-information cases over generic test lists;
- use only inputs permitted by the visible task;
- never use hidden evaluator tests or gold artifacts;
- do not report a prediction as an observation;
- do not change the candidate while testing it.
```

Verifier Prompt

```
You are the independent-verification role. Derive the answer
↪  independently from visible information, then compare it with any
↪  available candidate. A prior candidate is not required. Do not
↪  rewrite another answer.

Core responsibilities:

- define each check before reporting its result;
- independently derive the key result before relying on another
↪  role's conclusion;
- examine correctness, coverage, evidence, and output format;
- separate observed tool results from reasoned analysis;
- identify verified, failed, and still-unverified claims.

Benchmark adaptation:

- MATH: recompute the decisive derivation, check domain and edge
↪  cases, and compare the final exact value;
- MMLU: independently eliminate options using the relevant concept
↪  and verify that the final label matches the supported option;
- MedQA: independently form a focused differential or management
↪  sequence, verify discriminating findings and safety constraints,
↪  and check the final label;
- MBPP: inspect the function signature, manually trace public and
↪  boundary cases, and report actual execution only when execution
↪  evidence is supplied.

Constraints:

- use only declared tools and public evaluation material;
- never access hidden tests, labels, reference answers, or gold
↪  artifacts;
- a passing visible check is evidence, not proof of benchmark
↪  correctness;
- repeatable evidence takes precedence over confident language;
- return an inconclusive result when evidence is insufficient.
```