Multi-Agent Coordination Adaptation via Structure-Guided Orchestration

Anonymous ACL submission

Abstract

As large language model (LLM)-based multi-agent systems scale to handle increasingly complex tasks, balancing structural stability and dynamic adaptability becomes increasingly challenging. Existing systems typically adopt either structure-centric methods, committing to structures determined upfront that limit fine-grained control, or orchestration-centric methods, adapting decisions dynamically while leaving coordination structure implicit and unstable. To address this challenge, we revisit multi-agent coordination from a probabilistic perspective, casting it as posterior inference over the joint distribution of structure and orchestration. We introduce MACA, an automated coordination framework that learns a task- and budget-conditioned structural prior over agent participation and interactions. This prior guides a policy-based orchestration as an approximation to posterior inference, enabling efficient solutions with fine-grained control. Across benchmarks, MACA outperforms adaptive multi-agent baselines by an average of 8.42% while using 43.19% fewer tokens. Further investigation reveals that joint adaptation of structure and orchestration suppresses redundant interactions, converging coordination toward task-effective execution. The code is available at: `anonymous/MACA-53C9`.

1 Introduction

In an era marked by the maturation of foundation models (Tu et al., 2024), computational resources (Burns et al., 2016), and low-latency communication (Adhikari and Hazra, 2022), machine cognition is undergoing a paradigm shift from isolated computation to collective intelligence. As Minsky envisioned in Society of Mind (Minsky, 1986), intelligence arises from the interplay of simple agents, and this vision now unfolds at scale.

Within this context, LLM-based multi-agent systems (He et al., 2025; Li et al., 2024; Cheng et al., 2024) focus on harnessing collective intelligence through coordinated reasoning and organization to address increasingly complex challenges that transcend individual capability. Such LLM-based multi-agent systems have been demonstrated to be effective across a broad range of application domains, including question answering (Chen et al., 2025; Zhu et al., 2024; Zhang et al., 2024d), software development (Zhang et al., 2024c; Wang et al., 2025c), and data analysis (Xiao et al., 2024; Rasheed et al., 2024; Wang et al., 2025b), where structure and orchestration become critical for effective problem solving (Wu et al., 2023).

Early approaches to LLM-based multi-agent coordination, such as ChatDev (Qian et al., 2024a), MetaGPT (Hong et al., 2023), and AgentVerse (Chen et al., 2024), typically rely on handcrafted topologies and fixed interaction orders (Guo et al., 2024). While effective at encoding domain-specific structure (Yang et al., 2025a) and enabling role-level functional specialization (Lin et al., 2025; Naik et al., 2025), such designs rely heavily on manually engineered agent or coordination rules (Tang et al., 2024), and typically assume a fixed collaboration structure shared across tasks, prompting efforts for autonomous multi-agent systems. Zhang et al. (2024a) and Talebirad and Nadiri (2023) have formalized multi-agent systems as computational graphs, enabling structural and communication optimization. Methods such as GPTSwarm (Zhuge et al., 2024) and G-Designer (Zhang et al., 2024b) investigate learning adaptive communication structures or interaction patterns to reduce redundancy. A closer examination of recent autonomous mechanisms reveals two predominant paradigms, as illustrated in Figure 1: **(I) Structure-Centric Adaptation:** Given a query, the system adapts the multi-agent topology either by explicitly inferring a task-specific interaction structure (Zhang et al., 2025a; Yuan et al., 2025; Shang et al., 2024), or by refining a

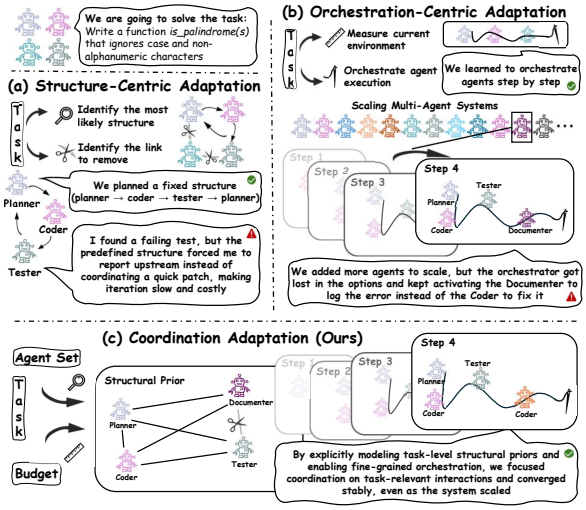

Figure 1: (a) **Structure-centric methods** fix a topology before inference, limiting adaptation as task states evolve. (b) **Orchestration-centric methods** offer step-by-step adaptability without a predefined structure, but incur high-variance coordination as scale increases. (c) **Our method** bridges these two paradigms by guiding adaptive orchestration with an explicit structural prior, achieving both stability and flexibility.

pre-defined structure via agent generation (Tian et al., 2025), agent selection (Zhang et al., 2024b), agent dropout (Wang et al., 2025d), and communication pruning (Zhang et al., 2024a). Since the system commits to a query-conditioned interaction structure prior to execution, it implicitly assumes that coordination efficiency is largely determined by this structural choice, leaving limited capacity for dynamic adaptation as task states evolve. **(II) Orchestration-Centric Adaptation:** Such methods adapt system behavior through sequential decision making, typically by selecting the next agent(s) at each step. Some works rely on heuristic or rule-based strategies (Rasal and Hauer, 2024; Rasal, 2024; Qayyum et al., 2025), while more recent methods formulate orchestration as an optimization problem, leveraging reinforcement learning (RL) (Sun et al., 2024; Dang et al., 2025; Zhang et al., 2025b; Liu et al., 2025) to optimize coordination policies. Despite their practical effectiveness, the absence of explicit interaction modeling induces role drift and high-variance credit assignment under increasing task complexity.

Revisiting these paradigms reveals a fundamental limitation of current adaptive multi-agent systems: **coordination is frequently treated as a single, monolithic process**. While Yang et al. (2025a) highlight the role of topological structure as a primary research objective and Bhatt et al. (2025) further investigate when orchestration is necessary, the separation between structure and orchestration prevents the system from jointly reasoning about how coordination should be organized and evolve over time, rendering coordination brittle and inefficient under scaling complexity (Cemri et al., 2025; Zhang et al., 2025c). Moreover, resource budgets (e.g., token limits) do not merely act as external constraints, but fundamentally shape the feasible coordination space and its inherent performance trade-offs. Ignoring these signals during either structural formulation or dynamic orchestration leads to unstable learning dynamics.

To address the above challenges, we propose a Multi-Agent Coordination Adaptation Framework (MACA) that casts multi-agent coordination as a posterior inference problem, where structural priors define a constrained interaction space and orchestration policies optimize within it. Specifically, our framework advances multi-agent coordination through two key innovations: **(I) Structural Prior Learning.** MACA models the structural prior as a task- and budget-conditioned variable by jointly estimating agent relevance and interaction plausibility. This process infers a probabilistic interaction graph that constrains agent participation and information flow, thereby yielding a principled foundation for downstream orchestration. **(II) Token-Aware Orchestration.** Given the inferred structure, MACA performs orchestration within the constrained space. The orchestration policy operates over a reduced space, where structural priors act as constraints through reward modulation, enabling flexible adaptation as task states evolve while maintaining scalability and efficiency.

Our contributions are summarized as follows:

- We are the first to cast multi-agent optimization as posterior inference over structural and orchestration variables. This elevates adaptive coordination from architecture selection or sampling to a principled probabilistic formulation.
- We introduce a novel and adaptive inference framework MACA, which learns a structural prior, subsequently optimizing orchestration within the generated prior to enable fine-grained and resource-efficient coordination.
- Extensive experiments across six benchmarks show that MACA yields robust and efficient coordination, surpassing adaptive multi-agent baselines with an average 8.42% performance boost and a 43.19% token reduction, thus achieving stronger task generalization and budget stability.

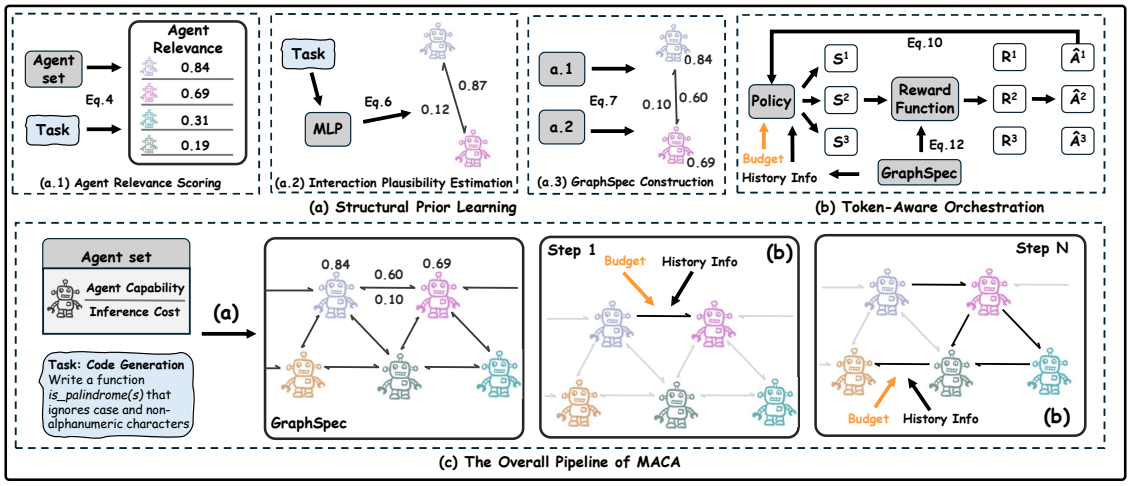

Figure 2: The overall framework of MACA. It consists of three main components: (a) Structural Prior Learning, (b) Token-Aware Orchestration, and (c) the overall pipeline for a given task.

2 Methodology

Figure 2 provides an overview of our proposed Multi-Agent Coordination Adaptation framework, namely MACA. We construct a structural prior from a space of agent compositions and interaction patterns. Building upon this prior, MACA approximates posterior inference over coordination decisions via reinforcement learning. The model leverages orchestration feedback to progressively refine system behaviors. In the following sections, Section 2.1 formally defines the coordination formulation and optimization objective of our framework, Section 2.2 presents the construction of structural prior, and Section 2.3 details the token-aware orchestration for adaptive multi-agent systems.

2.1 Preliminaries

(a) System Definition. We introduce the first unified definition for LLM-based multi-agent coordination systems, formulated as $\mathcal{M} = \{\mathcal{G}, \mathcal{T}\}$, where coordination is characterized by structure $\mathcal{G}$ and orchestration $\mathcal{T}$. Specifically, the structure is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \subseteq \mathcal{O}$ is a subset of a universal agent set $\mathcal{O}$, each node $v_i \in \mathcal{V}$ represents an agent as in previous practices (Zhang et al., 2024a). $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ specifies admissible interaction relations among agents, each edge $e_{ij} \in \mathcal{E}$ characterizes the potential for interaction from agent v_i to agent v_j, thereby defining a structured interaction space for downstream orchestration. The process $\mathcal{T}$ represents an ordered sequence of orchestration decisions. A particular realization τ can be modeled as a state-action trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$, where each state s_t summarizes the coordination context at step t, including the task input, intermediate agent outputs, and available resource signals. Each action a_t corresponds to an orchestration decision, such as selecting a subset of agents to invoke, routing information among agents, or terminating execution. The action selection process is guided by the structure G, i.e., $a_t \sim \pi(\cdot \mid s_t; G)$.

(b) Posterior Coordination. We refine the coordination mechanism to a posterior inference problem. Rather than treating the structure $\mathcal{G}$ and the orchestration $\mathcal{T}$ as fixed design choices, we model them as random variables that govern the solution process. Given a task instance x and a resource budget b, we model a hierarchical dependency. A structure G is first drawn from a task- and budget-conditioned structural space. This structure constrains the distribution of the execution orchestration τ. The resulting orchestration then determines the fidelity of producing the optimal output $y^\star$. Accordingly, coordination reduces to finding the joint posterior over structure and trajectory that best explains $y^\star$:

$$p(G, \tau \mid x, b, y^\star) \propto \underbrace{p(y^\star \mid \tau, x)}_{\text{Output Fidelity}} \cdot \underbrace{p(\tau \mid G, x, b)}_{\text{Orchestration}} \cdot \underbrace{p(G \mid x, b)}_{\text{Structural Prior}} \tag{1}$$

The posterior factorizes into (I) a structural prior $p(G \mid x, b)$ that assigns probability to valid agent topologies under the given conditions, (II) an orchestration $p(\tau \mid G, x, b)$ capturing how execution evolves within the structural constraints, and (III) output fidelity $p(y^\star \mid \tau, x)$ that measures how likely an orchestration yields the optimal solution.

(c) Problem Formulation. To compute the intractable posterior in Eq. 1 and thereby infer the

joint distribution of structure G and orchestration τ, we adopt variational inference (Levine, 2018) to approximate the target distribution $p(G, \tau \mid x, b, y^\star)$ with a parameterized generative policy $\pi_\theta(G, \tau \mid x, b)$. Specifically, the Kullback-Leibler (KL) divergence between the variational distribution and the target posterior is defined as follows:

$$D_{\mathrm{KL}}(\pi_\theta \parallel p) = \mathbb{E}_{(G,\tau)\sim\pi_\theta}\left[\log \pi_\theta(G, \tau \mid x, b) - \log p(G, \tau \mid x, b, y^\star)\right] \tag{2}$$

By minimizing the KL divergence, the inference process can be formulated into an optimization problem. We define a utility function $U(\tau, y) = \log p(y \mid \tau, x)$, representing the log-likelihood of the ground truth outcome with given orchestration and task instance. Assuming a fixed budget constraint implicit in the structural prior, the minimization of the KL divergence is mathematically equivalent to maximizing the evidence lower bound (ELBO), which yields the optimization objective:

$$\max_\theta \ \mathbb{E}_{\tau\sim\pi_\theta, G}\Big[U(\tau, y^\star) + \alpha \cdot \Omega(\pi_\theta \mid G, x, b)\Big] \tag{3}$$

where the regularization term $\Omega(\pi_\theta \mid G, x, b)$ encodes an objective that constrains the orchestration policy to remain aligned with the structural prior under the task input x and budget b. The coefficient $\alpha \in [0, 1]$ controls the trade-off between maximizing task utility and enforcing structural consistency.

2.2 Structural Prior Learning

Inspired by the success of learned priors for capturing data-driven regularities (Ulyanov et al., 2018; Wang et al., 2025a), we introduce a structural prior termed **GraphSpec**, which models coordination uncertainty by jointly inferring **Agent Relevance** and **Interaction Plausibility**.

Agent Relevance Scoring. Agent suitability varies with task semantics and resource budgets. Recent findings (Yang et al., 2025b; Wang et al., 2024a) show that complex reasoning benefits from selective agent engagement. Accordingly, agent relevance is estimated by modeling the semantic compatibility between each agent representation and the task–budget context. For each agent $v_i \in \mathcal{O}$, we introduce a continuous variable $z_i \in [0, 1]$ that represents its participation strength in the coordination structure. Let $s_i = \cos\big(\mathbf{e}(x, b),\ \mathbf{e}_i\big)$ denote the resulting relevance score, where $\mathbf{e}(x, b)$ denotes the embedding of the input context and $\mathbf{e}_i$ represents the embedding of agent v_i, both obtained via a sentence encoder (Xiao et al., 2023). $\cos(\cdot, \cdot)$ denotes the cosine similarity. We then obtain a participation weight:

$$q_i = \sigma\left(\frac{s_i}{\beta(b)}\right), \qquad z_i = \begin{cases} q_i, & q_i \geq \gamma, \\ 0, & q_i < \gamma, \end{cases} \tag{4}$$

where $\beta(b)$ is a budget-dependent temperature. $\{q_i\}_{v_i\in\mathcal{O}}$ represents a continuous score. We apply a thresholded gating operation with parameter γ to suppress low-confidence agents. Agents with $q_i < \gamma$ are filtered out, while agents exceeding the threshold retain their participation strength. The resulting $Z_{prior} = \{z_i\}$ specifies agent relevance.

Interaction Plausibility Estimation. We model interaction plausibility as an edge-level prior over directed agent transitions $v_i \to v_j$. Intuitively, interactions that remain consistently effective under stochastic structural variations are more likely to reflect stable coordination patterns and thus should receive higher prior probability. We define a policy π_ϕ that directly parameterizes graph topology via learnable edge logits $\phi = \{\ell_{ij} \in \mathbb{R}\}$, which quantify the interaction plausibility between agents. This policy independently samples directed edges $e_{ij} \sim \mathrm{Bernoulli}(\sigma(\ell_{ij}))$. The parameters ϕ are optimized by minimizing the following loss function:

$$\mathcal{L}(\phi) = -\mathcal{U} \cdot \sum_{(v_i,v_j)\in\mathcal{O}} \log \sigma(\ell_{ij}) + \lambda \mathcal{R}_s \tag{5}$$

where $\mathcal{U} \in \{0, 1\}$ represents a utility signal. $\mathcal{R}_s = \frac{1}{|\mathcal{O}|}\sum_{(v_i,v_j)\in\mathcal{O}} \sigma(\ell_{ij})$ serves as a regularizer to prevent over-fitting. Subsequently, we maintain a buffer of the high quality interaction trajectories sampled from π_ϕ, filtering for instances that achieve correct reasoning. These filtered trajectories are used as pseudo-labels to train a Multi-Layer Perceptron (MLP):

$$P(v_i \to v_j | x) = \mathrm{MLP}_\psi([\mathbf{e}_i, \mathbf{e}_j, \mathbf{e}(x)]) \tag{6}$$

The MLP maps the task x to the edge interaction probability for any agent pair (v_i, v_j).

GraphSpec Construction. We synthesize the estimated agent relevance and interaction plausibility into GraphSpec, a unified probabilistic prior that adapts the structure to the task-budget context. We modulate the raw edge probabilities by the participation strengths of the incident nodes. The effective connection probability P_{prior} is defined as:

$$\tilde{p}_{ij} = p_{ij} \cdot q_j, \quad P_{prior} = [\tilde{p}_{ij}] \in [0, 1]^{N\times N} \tag{7}$$

This modulation suppresses links involving low-confidence agents. The resulting GraphSpec, denoted by $\mathcal{GS}(x, b) = (Z_{prior}, P_{prior})$, defines a parameterization of the structural prior $p(\mathcal{G} \mid x, b)$.

2.3 Token-Aware Orchestration

The orchestration problem is formulated as a Markov Decision Process (MDP), and the policy is optimized using a GRPO-based method (Guo et al., 2025). At each step t, the state $s_t \in \mathcal{S}$ encapsulates the task input x, history state h_t, and budget b_t. The objective is to learn a policy $\pi_\theta(a_t \mid s_t)$ that maximizes the expected return:

$$\max_\theta \ \mathbb{E}_{\tau\sim\pi_\theta}\Big[R - \alpha \cdot D_{\mathrm{KL}}\big(\pi_\theta \parallel \pi_{\mathrm{ref}}\big)\Big] \tag{8}$$

where $R = \sum_{t=1}^{T} r'_t$ denotes the total return of trajectory τ, with r'_t defined in Eq. (12). π_{ref} denotes the frozen reference policy, and α is a trade-off parameter. The policy π_θ selects an action a_t from an action space $\mathcal{A} = \mathcal{O} \cup \{\text{STOP}\}$. For each task, we sample a group of K trajectories $\{\tau^{(k)}\}_{k=1}^{K}$ using the current policy. Let $R^{(k)}$ denote the reward of trajectory k. Group-relative advantages are computed as:

$$A^{(k)} = R^{(k)} - \frac{1}{K}\sum_{k'=1}^{K} R^{(k')}, \quad \tilde{A}^{(k)} = \frac{A^{(k)}}{\mathrm{Std}(\{R^{(k')}\}) + \epsilon} \tag{9}$$

The overall loss function combines a clipped term $L_{\text{clip}}(\theta)$ with a KL divergence penalty, denoted as:

$$L_{\text{clip}}(\theta) = \min\Big(\rho_t(\theta)\tilde{A}, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\tilde{A}\Big),$$

$$\text{J}(\theta) = -\mathbb{E}\left[\frac{1}{K}\sum_k \left(L_{\text{clip}}^{(k)}(\theta) - \alpha \cdot D_{\mathrm{KL}}(\pi_\theta \| \pi_{\text{ref}})\right)\right] \tag{10}$$

where $\rho_t(\theta) = \exp(\log \pi_\theta - \log \pi_{\theta_{\text{old}}})$ is the likelihood ratio, ϵ is the clipping range. A KL divergence term is introduced to regularize π_θ, keeping it close to π_{ref}, which balances reward maximization with stability. However, relying solely on standard regularization leaves the vast multi-agent interaction space unconstrained, allowing the policy to waste tokens on spurious interactions. To address this, we introduce a hybrid supervision mechanism that incorporates both hard constraints and soft regularization into the orchestration. We first impose constraints on the action space to filter out structurally implausible interactions. Given the structural mask $\mathcal{H}$, we enforce the validity of the policy π_θ by masking out invalid actions. The masked policy distribution is formally defined as:

$$\pi_{masked}(a_t|s_t) = \frac{\exp(l(a_t|s_t)) \cdot \mathcal{H}_{a_t}}{\sum_{a'\in\mathcal{A}} \exp(l(a'|s_t)) \cdot \mathcal{H}_{a'}} \tag{11}$$

where $l(\cdot|s_t)$ denotes the raw logits generated by the policy network. By setting the probability of masked actions to zero, we restrict the policy to a prior-consistent subset of actions, constraining exploration to plausible connections.

To prevent the policy from engaging in aimless exploration among valid but low-value interactions, we incorporate the edge probabilities from the prior as a reference distribution $\pi_{\text{mix}} = \frac{1}{2}\Big(\pi_{\text{ref}} + \pi_{\text{prior}}\Big)$. Here, π_{prior} denotes the prior action distribution induced by GraphSpec. Distinct from GRPO, we leverage a KL penalty that encourages the policy to anchor its exploration around the learned structural prior. The reward function is formulated as:

$$r'_t = r_t - \lambda \cdot D_{KL}\left(\pi_\theta \parallel \pi_{mix}\right), \tag{12}$$

where $r_t = R_{\text{acc}} - \beta C_t^{\text{token}}$ denotes the extrinsic reward, which balances task utility R_{acc} against token cost C_t^{token} through the trade-off parameter β. λ is a regularization coefficient balancing task performance and structural adherence. This soft regularization keeps the policy close to the learned prior, stabilizing the training process in complex scenarios. More details of the training algorithm can be found in Appendix B.5.

3 Experiments

3.1 Experiment Setup

Benchmarks and Metrics. We comprehensively evaluate MACA across six benchmarks spanning three domains. **(I) Code generation**, HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021); **(II) Question Answering**, MMLU-Pro (Wang et al., 2024b) and ARC-C (Clark et al., 2018); and **(III) Math Reasoning**, SVAMP (Patel et al., 2021) and GSM-Hard (Gao et al., 2022). We evaluate models in terms of **Accuracy** and **Average Cost**. Accuracy is computed as $\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(\hat{y}_i \text{ is correct})$, where correctness follows task-specific criteria (e.g., Pass@1 or Exact Match). Average Cost is measured as $\frac{1}{N}\sum_{i=1}^{N} c_i$, where c_i denotes the token cost of task i, including both prompt tokens and completion tokens. The dataset statistics are in Appendix B.1.

Baselines. We compare MACA against baselines categorized by their coordination and adaptation mechanisms: **(I) single-agent methods** including CoT (Wei et al., 2022), ComplexCoT (Fu et al., 2022), Self-refine (Madaan et al., 2023), and Self-Consistency (Wang et al., 2022). **(II) hand-crafted multi-agent systems** including DyLAN (Liu et al., 2024), AgentVerse (Chen et al., 2024), and MacNet (Qian et al., 2024b). **(III) Adaptive multi-agent systems** including AgentPrune (Zhang et al., 2024a), MaAS (Zhang et al., 2025a) and Puppeteer (Dang et al., 2025). Additional details for baselines are provided in Appendix B.2.

Table 1: Performance comparison of baseline methods on Llama-3.1-8B. The best results are shown in **bold**, and the second-best are underlined. Avg Cost is compared among multi-agent methods to reflect their efficiency.

Model: Llama-3.1-8B	Structure Adaptation	Orchestration Adaptation	Code Generation				Question Answering				Math Reasoning			
			HumanEval		MBPP		MMLU-Pro		ARC-C		SVAMP		GSM-Hard	
			Acc (%)	Avg Cost	Acc (%)	Avg Cost	Acc (%)	Avg Cost	Acc (%)	Avg Cost	Acc (%)	Avg Cost	Acc (%)	Avg Cost
Vanilla	✗	✗	60.61	256.6	45.97	138.4	38.50	246.7	83.09	279.6	82.18	209.6	32.31	345.3
CoT	✗	✗	56.44	297.9	44.62	146.1	39.24	379.5	83.17	443.7	82.51	267.8	36.36	387.0
ComplexCoT	✗	✗	57.58	1154.4	44.18	247.4	38.54	1304.2	82.18	883.1	86.33	2658.9	32.83	2766.3
Self-Refine	✗	✗	62.12	1609.8	47.53	498.7	33.75	1646.5	83.16	1191.4	83.67	1195.3	26.26	2525.5
SC (CoT×5)	✗	✗	55.38	2389.6	46.36	1206.9	41.62	2406.9	84.16	1545.0	88.67	1080.7	40.40	1639.8
DyLAN	✗	✗	72.73	15256.7	12.50	11647.8	47.26	11874.6	83.33	3471.6	86.33	6675.6	33.63	15338.0
MacNet	✗	✗	68.18	8782.6	47.83	6794.8	25.47	10284.4	83.83	3596.3	79.74	6912.6	19.84	12468.3
AgentVerse	✗	✗	66.18	6826.9	48.94	5017.5	41.69	6193.2	85.61	3014.9	86.03	4184.7	38.19	6768.4
AgentPrune	✓	✗	62.50	2357.0	31.44	2787.8	43.53	2713.6	86.09	2691.3	88.44	2780.8	33.86	3815.8
MaAS	✓	✗	65.15	4796.9	42.05	3215.7	43.94	2978.4	43.83	2674.7	92.28	2827.7	**51.52**	2224.8
Puppeteer	✗	✓	71.49	2798.3	47.17	3696.5	51.25	4097.8	85.57	3753.3	94.18	4204.7	49.25	4196.6
MACA (**Ours**)	✓	✓	**75.76**	**2100.1**	**49.23**	**2412.3**	**52.67**	**2117.8**	**87.75**	**1656.3**	**96.00**	**2057.2**	50.30	**1602.5**

Table 2: Results on Llama-3.1-70B. The best and runnerup results are bolded and underlined, respectively. Avg Cost is compared among multi-agent methods.

Model: Llama-3.1-70B	MMLU-Pro		HumanEval		GSM-Hard	
	Acc (%)	Avg Cost	Acc (%)	Avg Cost	Acc (%)	Avg Cost
Vanilla	53.50	250.6	78.78	239.8	48.87	206.7
CoT	54.25	274.1	80.30	247.1	57.14	292.8
DyLAN	60.09	12068.9	84.25	12984.5	55.05	14186.4
AgentPrune	57.74	3128.7	80.25	3186.9	51.44	4372.5
MaAS	58.74	2868.2	83.33	3758.5	64.03	3125.8
MACA (**Ours**)	**64.57**	**2438.6**	**88.89**	**2672.2**	**67.37**	**2747.7**

3.2 Performance Comparison

MACA consistently outperforms baselines. As shown in Table 1, MACA achieves the highest average accuracy of 68.62% across six benchmarks. Compared to single-agent methods, MACA yields an average accuracy improvement of $9.19\% \sim 12.54\%$. Against adaptive multi-agent baselines, MACA delivers an 8.42% improvement in accuracy while simultaneously reducing token costs by 36.2% to 51.9%. These results confirm the effectiveness and cost-efficiency of MACA. Notably, this advantage remains consistent across model scales, from Llama-3.1-8B to Llama-3.1-70B.

RQ1: Can coordination itself induce reasoning capability? At the Llama-3.1-8B scale (Table 1), MACA yields clear gains over single-agent baselines, suggesting that the improvements arise from coordinated interaction. While methods such as CoT (Wei et al., 2022) improve reasoning by prompting models to generate explicit step-by-step rationales, they remain bounded by single-model limitations and may reinforce incorrect trajectories. By contrast, MACA mitigates the self-reinforcing error loops inherent in monolithic generation through coordination among diverse agents, achieving an average absolute gain of 13.69% over CoT on MMLU-Pro and GSM-Hard. More importantly, cross-scale results (Tables 1 and 2) show that MACA with Llama-3.1-8B can outperform a vanilla Llama-3.1-70B model on GSM-Hard (50.30% vs. 48.87%), while remaining competitive on MMLU-Pro and HumanEval (reaching ~97.3% of the vanilla 70B model's performance).

3.3 Framework Analysis

Ablation Study. Table 3 presents an ablation study on key components of MACA: (1) w/o Z_{prior}, removing agent relevance prior; (2) w/o P_{prior}, removing interaction plausibility prior; (3) w/o GraphSpec, eliminating the entire structural prior; and (4) w/o π_θ, removing the learnable policy.

Table 3: Ablation study of MACA.

Dataset	ARC-C		GSM-Hard	
Metric	Acc (%)	Avg Cost	Acc (%)	Avg Cost
MACA	87.75	1656.3	50.30	1602.5
MACA w/o Z_{prior}	82.64	1920.4	44.37	1885.2
MACA w/o P_{prior}	81.31	1896.7	43.92	1830.8
MACA w/o GraphSpec	72.35	2285.9	36.80	2050.1
MACA w/o π_θ	85.53	2109.6	47.96	1785.5

Removing GraphSpec degenerates the system into an orchestration-centric approach. Without a structural prior to constrain the vast interaction space, performance drops most severely (14.45% ↓ in accuracy). Conversely, removing the policy π_θ reduces the system to a structure-centric approach. While accuracy only drops slightly, the token cost surges drastically (19.40% ↑ in cost). This reveals a clear functional division: GraphSpec preserves reasoning fidelity by constraining the search space, while the token-aware policy π_θ optimizes efficiency. Focusing on the structural prior itself, we observe that dropping either agent relevance Z_{prior} or interaction plausibility P_{prior} leads to distinct accuracy degradation and increased overhead. This highlights that the joint effect of participating agents and their interaction patterns is essential for a robust coordination prior.

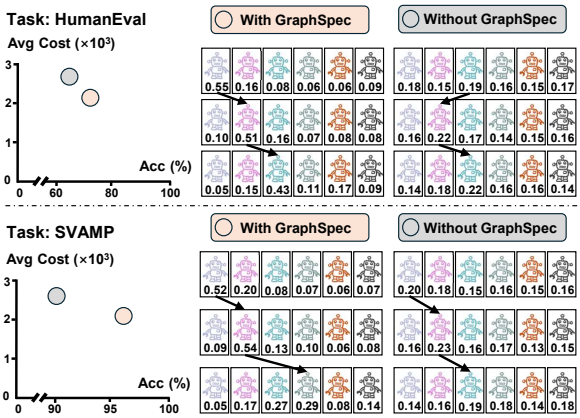

Figure 3: Impact of the prior on orchestration.

RQ2: How does the prior affect orchestration decisions? Figure 3 shows that GraphSpec imposes a structural bias on orchestration. Without it, coordination is highly unconstrained, with probability mass diffusely spread across transitions. With GraphSpec, the distribution concentrates on a few dominant, task-relevant paths. Even when the same agent is selected, the prior still reshapes the confidence of execution. For example, the top-2 transition mass increases from 0.37 to 0.71 on HumanEval and from 0.38 to 0.72 on SVAMP, showing that GraphSpec turns diffuse routing into concentrated, task-relevant orchestration. This concentration makes orchestration more selective and cost-efficient, improving MACA's efficiency.

3.4 Cost Analysis

RQ3: How does MACA balance cost and task performance? We evaluate cost–performance by jointly considering task accuracy and token consumption during inference, as shown in Figure 4.

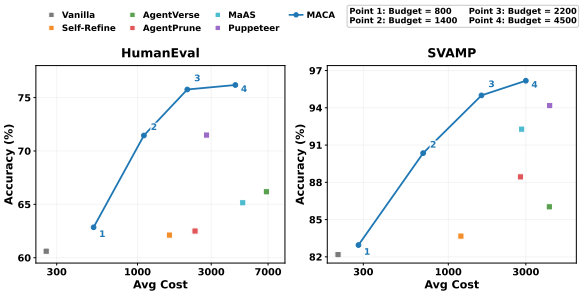

Figure 4: Cost–performance trade-offs of MACA and baseline approaches across datasets.

MACA exhibits a clear cost-performance trade-off: higher cost consistently yields higher accuracy. MACA is tunable with respect to budget, enabling practitioners to flexibly trade computation for performance under different constraints. In practice, this allows accuracy to improve in a predictable manner. More importantly, MACA achieves a more favorable cost–performance frontier than existing multi-agent baselines. This advantage stems not merely from budget tunability, but from how the budget is utilized. MACA allocates computation through structured and selective coordination, converting additional cost into accuracy gains more effectively while avoiding unnecessary interaction.

3.5 Coordination Mechanisms Analysis

RQ4: What coordination patterns emerge in MACA? Analysis of agent transition probabilities and coordination patterns on GSM-Hard trajectories reveals the following mechanisms:

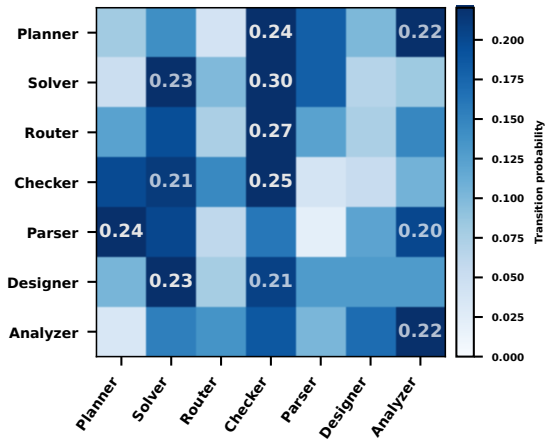

Figure 5: Column-to-row Transition Probabilities.

Self-Correction (X → Checker): Figure 5 shows a coupling between generation and verification modules, particularly transitions such as *Solver* → *Checker* (0.30) and *Router* → *Checker* (0.27), with iterative feedback (e.g., *Checker* → *Solver*, 0.21). This pattern mirrors the **Self-Refine** (Madaan et al., 2023), where candidate solutions are followed by verification and revision. However, MACA goes beyond fixed self-refine loops by learning an orchestration policy over when and where verification should occur: verification is invoked more frequently along complex reasoning while being skipped for simpler steps. This mechanism allows MACA to balance accuracy and token efficiency.

Hierarchical Coordination: Figure 6 aggregates the most frequent three-agent transition sequences, revealing clear hierarchical coordination patterns. The first layer, *Router* and *Designer*, is responsible for task decomposition and strategy selection. The second layer, *Parser* and *Analyzer*, translates high-level intent into structured representations. The final layer, *Solver* and *Checker*, carries out computational reasoning and verification. Rather than

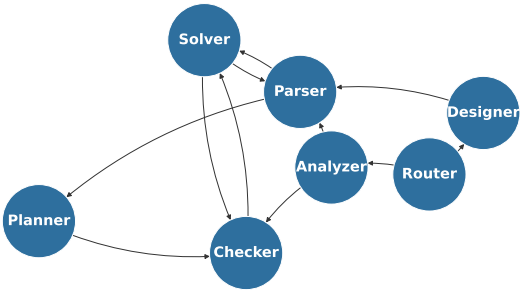

Figure 6: Dominant Three-Agent Coordination Patterns.

relying on manually designed workflows, these coordination strata emerge from optimizing the structural prior and orchestration policy. This process structures agent capabilities into a cascaded reasoning framework, ensuring efficient task execution.

3.6 Sensitivity Analysis

Figure 7 illustrates the sensitivity of MACA to two core parameters: the threshold γ in Eq. (4), and the regularization coefficient λ in Eq. (12).

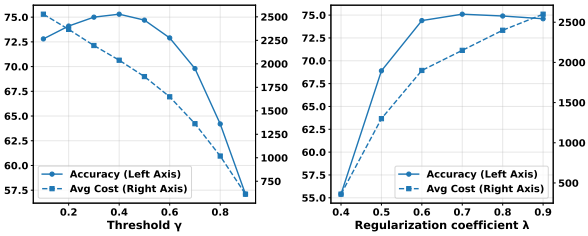

Figure 7: Parameter sensitivity analysis of MACA.

Threshold γ. An accuracy-efficiency trade-off emerges as γ varies. As γ increases from 0.1 to 0.4, accuracy rises from 72.8% to 75.2% while token cost drops, showing that moderate thresholding removes redundant coordination. Further increasing γ causes accuracy to decline steadily, indicating that overly aggressive sparsification discards useful coordination paths. We therefore set $\gamma = 0.4$.

Regularization coefficient λ. Increasing λ from 0.4 to 0.7 improves accuracy from 55.4% to 75.1%, but also raises token cost, indicating that stronger regularization improves orchestration quality at the expense of computation. Beyond this point, accuracy saturates while cost continues to grow, suggesting diminishing returns from overly large regularization. Accordingly, we adopt $\lambda = 0.7$.

4 Related Work

LLM-based Multi-Agent Systems. Advances in language models (Guo et al., 2025; Hurst et al., 2024) have driven progress in LLM-based agents (Zhao et al., 2024; Li et al., 2023a; Xie et al., 2024). As tasks grow complex, research has increasingly turned to LLM-based multi-agent systems (Bo et al., 2024; Du et al., 2023). Prior work demonstrates empirical performance in such systems, exemplified by MacNet (Qian et al., 2024b), ChatDev (Qian et al., 2024a), and AutoGen (Wu et al., 2023). Despite progress, early approaches rely on handcrafted structures (Hong et al., 2023), limiting the exploration of structural optimality and transferability (Cemri et al., 2025). Hence, research has begun to explore adaptive agentic systems.

Adaptive Agentic Systems Optimization. Recent work explores how agent compositions (Zhuge et al., 2024; Zhang et al., 2024b) and execution policies (Yuan et al., 2025) can be optimized, broadly categorized into two streams: (I) Structure-Centric Adaptation methods (Yang et al., 2025b; Wang et al., 2025d) adapt multi-agent topology, with methods such as AgentPrune (Zhang et al., 2024a) and MaAS (Zhang et al., 2025a) parameterizing agent interactions (Li et al., 2023b; Yan et al., 2021). (II) Orchestration-Centric Adaptation methods (Qayyum et al., 2025; Rasal, 2024) introduce adaptivity at the execution level. Structure-centric methods favor stability, whereas orchestration-centric methods provide finer-grained control, yet both remain limited in isolation.

RL as Inference. Reinforcement Learning (RL) has been widely used in LLM-based multi-agent coordination (Sun et al., 2024). Applications include structural reasoning like BAMAS (Yang et al., 2025b) and DyLAN (Liu et al., 2024), as well as execution orchestration such as Puppeteer (Dang et al., 2025) and OSC (Zhang et al., 2025b). MAPRO (Zhang et al., 2025d) uses posterior inference for multi-agent prompt optimization. More broadly, prior research has established the concept of RL as Inference (Levine, 2018; O'Donoghue et al., 2020; Tarbouriech et al., 2023). This view casts control as probabilistic inference and provides a principled foundation for execution optimization.

5 Conclusion

In this paper, we introduce MACA, a probabilistic framework that rethinks automated multi-agent system design from a posterior inference perspective. MACA explicitly factorizes system adaptation into a task- and budget-conditioned structural prior and a token-aware orchestration policy, enabling principled uncertainty modeling and fine-grained control. MACA enables adaptive multi-agent systems that are effective across diverse tasks. The probabilistic perspective provides a foundation for future research on adaptive multi-agent systems.

Limitations

Although MACA demonstrates clear gains in both task performance and cost-efficiency, the current evaluation is still centered on relatively structured benchmarks, so its generalizability to more open-ended, interactive, or domain-specific settings is not yet fully established. In addition, MACA depends on a predefined agent pool with manually specified capabilities and role descriptions, which means that part of its effectiveness may still come from careful agent initialization rather than from coordination alone. Another limitation lies in the way the structural prior is learned: GraphSpec is trained from filtered high-quality trajectories and supervision signals derived from final outcome quality, which can favor coordination patterns that are easy to verify on benchmark tasks while making it harder to capture useful but less immediately rewarded interactions. This may reduce robustness in settings where credit assignment is noisy, intermediate collaboration is important, or success cannot be cleanly reflected by final answers alone.

References

Mainak Adhikari and Abhishek Hazra. 2022. 6g-enabled ultra-reliable low-latency communication in edge networks. *IEEE Communications Standards Magazine*, 6(1):67–74.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Umang Bhatt, Sanyam Kapoor, Mihir Upadhyay, Ilia Sucholutsky, Francesco Quinzan, Katherine M Collins, Adrian Weller, Andrew Gordon Wilson, and Muhammad Bilal Zafar. 2025. When should we orchestrate multiple agents? *arXiv preprint arXiv:2503.13577*.

Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. 2024. Reflective multi-agent collaboration based on large language models. *Advances in Neural Information Processing Systems*, 37:138595–138631.

Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. 2016. Borg, omega, and kubernetes. *Communications of the ACM*, 59(5):50–57.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, and 1 others. 2025. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*.

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, and 1 others. 2024. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *ICLR*.

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. 2025. Multi-agent evolve: Llm self-improve through co-evolution. *arXiv preprint arXiv:2510.23595*.

Yuheng Cheng, Ceyao Zhang, Zhengwen Zhang, Xiangrui Meng, Sirui Hong, Wenhao Li, Zihao Wang, Zekai Wang, Feng Yin, Junhua Zhao, and 1 others. 2024. Exploring large language model based intelligent agents: Definitions, methods, and prospects. *arXiv preprint arXiv:2401.03428*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, and 1 others. 2025. Multi-agent collaboration via evolving orchestration. *arXiv preprint arXiv:2505.19591*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based

multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Junda He, Christoph Treude, and David Lo. 2025. Llm-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–30.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Sergey Levine. 2018. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*.

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

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. 2024. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9.

Zhixun Li, Liang Wang, Xin Sun, Yifan Luo, Yanqiao Zhu, Dingshuo Chen, Yingtao Luo, Xiangxin Zhou, Qiang Liu, Shu Wu, and 1 others. 2023b. Gslb: the graph structure learning benchmark. *Advances in Neural Information Processing Systems*, 36:30306–30318.

Yi-Cheng Lin, Kang-Chieh Chen, Zhe-Yan Li, Tzu-Heng Wu, Tzu-Hsuan Wu, Kuan-Yu Chen, Hung-yi Lee, and Yun-Nung Chen. 2025. Creativity in llm-based multi-agent systems: A survey. *arXiv preprint arXiv:2505.21116*.

Shuo Liu, Tianle Chen, Zeyu Liang, Xueguang Lyu, and Christopher Amato. 2025. Llm collaboration with multi-agent reinforcement learning. *arXiv preprint arXiv:2508.04652*.

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

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Marvin Minsky. 1986. *Society of mind*. Simon and Schuster.

Akshat Naik, Patrick Quinn, Guillermo Bosch, Emma Gouné, Francisco Javier Campos Zabala, Jason Ross Brown, and Edward James Young. 2025. Agentmisalignment: Measuring the propensity for misaligned behaviour in llm-based agents. *arXiv preprint arXiv:2506.04018*.

Brendan O'Donoghue, Ian Osband, and Catalin Ionescu. 2020. Making sense of reinforcement learning and probabilistic inference. *arXiv preprint arXiv:2001.00805*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Adnan Qayyum, Abdullatif Albaseer, Junaid Qadir, Ala Al-Fuqaha, and Mohamed Abdallah. 2025. Llm-driven multi-agent architectures for intelligent self-organizing networks. *IEEE Network*.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, and 1 others. 2024a. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, and 1 others. 2024b. Scaling large language model-based multi-agent collaboration. *arXiv preprint arXiv:2406.07155*.

Sumedh Rasal. 2024. Llm harmony: Multi-agent communication for problem solving. *arXiv preprint arXiv:2401.01312*.

Sumedh Rasal and EJ Hauer. 2024. Navigating complexity: Orchestrated problem solving with multi-agent llms. *arXiv preprint arXiv:2402.16713*.

Zeeshan Rasheed, Muhammad Waseem, Aakash Ahmad, Kai-Kristian Kemell, Wang Xiaofeng, Anh Nguyen Duc, and Pekka Abrahamsson. 2024. Can large language models serve as data analysts? a multi-agent assisted approach for qualitative data analysis. *arXiv preprint arXiv:2402.01386*.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. 2024. Agentsquare: automatic llm agent search in modular design space. 2024.

Chuanneng Sun, Songjun Huang, and Dario Pompili. 2024. Llm-based multi-agent reinforcement learning: Current and future directions. *arXiv preprint arXiv:2405.11106*.

Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*.

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.

Jean Tarbouriech, Tor Lattimore, and Brendan O'Donoghue. 2023. Probabilistic inference in reinforcement learning done right. *Advances in Neural Information Processing Systems*, 36:33687–33725.

Chunhao Tian, Yutong Wang, Xuebo Liu, Zhexuan Wang, Liang Ding, Miao Zhang, and Min Zhang. 2025. Agentinit: Initializing llm-based multi-agent systems via diversity and expertise orchestration for effective and efficient collaboration. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11870–11902.

Xiaoguang Tu, Zhi He, Yi Huang, Zhi-Hao Zhang, Ming Yang, and Jian Zhao. 2024. An overview of large ai models and their applications. *Visual Intelligence*, 2(1):34.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2018. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454.

Hanchen Wang, Dawei Cheng, Ying Zhang, and Wenjie Zhang. 2025a. Aigc for graphs: Current techniques and future trends. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pages 4504–4508. IEEE.

Peiran Wang, Yaoning Yu, Ke Chen, Xianyang Zhan, and Haohan Wang. 2025b. Large language model-based data science agent: A survey. *arXiv preprint arXiv:2508.02744*.

Qian Wang, Tianyu Wang, Zhenheng Tang, Qinbin Li, Nuo Chen, Jingsheng Liang, and Bingsheng He. 2025c. Megaagent: A large-scale autonomous llm-based multi-agent system without predefined sops. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4998–5036.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024a. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? *arXiv preprint arXiv:2402.18272*.

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*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. 2025d. Agentdropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration. *arXiv preprint arXiv:2503.18891*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 3(4).

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Yihang Xiao, Jinyi Liu, Yan Zheng, Xiaohan Xie, Jianye Hao, Mingzhi Li, Ruitao Wang, Fei Ni, Yuxiao Li, Jintian Luo, and 1 others. 2024. Cellagent: An llm-driven multi-agent framework for automated single-cell data analysis. *arXiv preprint arXiv:2407.09811*.

Chengxing Xie, Canyu Chen, Feiran Jia, Ziyu Ye, Shiyang Lai, Kai Shu, Jindong Gu, Adel Bibi, Ziniu Hu, David Jurgens, and 1 others. 2024. Can large language model agents simulate human trust behavior? *Advances in neural information processing systems*, 37:15674–15729.

Zhicheng Yan, Xiaoliang Dai, Peizhao Zhang, Yuandong Tian, Bichen Wu, and Matt Feiszli. 2021. Fp-nas: Fast probabilistic neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15139–15148.

Jiaxi Yang, Mengqi Zhang, Yiqiao Jin, Hao Chen, Qingsong Wen, Lu Lin, Yi He, Srijan Kumar, Weijie Xu, James Evans, and 1 others. 2025a. Topological structure learning should be a research priority for llm-based multi-agent systems. *arXiv preprint arXiv:2505.22467*.

Liming Yang, Junyu Luo, Xuanzhe Liu, Yiling Lou, and Zhenpeng Chen. 2025b. Bamas: Structuring budget-aware multi-agent systems. *arXiv preprint arXiv:2511.21572*.

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

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025a. 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. *arXiv 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, and 1 others. 2024c. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*.

Jusheng Zhang, Yijia Fan, Kaitong Cai, Jinzhou Tang, Xiaofei Sun, and Keze Wang. 2025b. Osc: Cognitive orchestration through dynamic knowledge alignment in multi-agent llm collaboration. *Rn*, 100(R1):R2.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and 1 others. 2025c. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. *arXiv preprint arXiv:2505.00212*.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. 2024d. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237.

Zheyuan Zhang, Lin Ge, Hongjiang Li, Weicheng Zhu, Chuxu Zhang, and Yanfang Ye. 2025d. Mapro: Recasting multi-agent prompt optimization as maximum a posteriori inference. *arXiv preprint arXiv:2510.07475*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Jun-Peng Zhu, Peng Cai, Kai Xu, Li Li, Yishen Sun, Shuai Zhou, Haihuang Su, Liu Tang, and Qi Liu. 2024. Autotqa: Towards autonomous tabular question answering through multi-agent large language models. *Proceedings of the VLDB Endowment*, 17(12):3920–3933.

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*.

Appendix

A Probabilistic Assumptions for Posterior Inference

We adopt a probabilistic view of coordination in which the coordination structure G affects the target outcome $y^\star$ only through the induced execution trajectory τ. Formally, conditioned on the input x and trajectory τ, the output is independent of G, i.e., $y^\star \perp G \mid (\tau, x)$. Under this assumption, the output fidelity term in Equation (1) depends only on the realized trajectory and the given task, while G remains important because it constrains which trajectories are plausible under the task and budget.

From this perspective, once a suitable structural prior is learned, posterior inference favors trajectories with higher utility. In MACA, this intractable inference is approximated by a learnable parameterized policy, while the structural prior induces a reference distribution π_{mix} to guide exploration toward structurally plausible and high-value coordination patterns. In this sense, MACA can be interpreted as an approximation to posterior coordination inference under a learned structural prior.

B Experimental Details

B.1 Dataset

Following practice in prior work (Zhang et al., 2025a; Yang et al., 2025b), we split each benchmark into training and test sets. As shown in Table 4, our experiments cover three task domains: code generation, math reasoning, and question answering. For code generation, we use HumanEval and MBPP, evaluated by pass@1. For math reasoning, we include GSM-Hard and SVAMP, both evaluated by accuracy. For question answering, we use MMLU-Pro and ARC-Challenge (ARC-C), also measured by accuracy.

Table 4: Dataset Statistics.

Domain	Dataset	#Train	#Test	Metric
Code Generation	HumanEval	96	68	pass@1
	MBPP	587	387	pass@1
Math Reasoning	GSM-Hard	611	389	Accuracy
	SVAMP	593	407	Accuracy
Question Answering	MMLU-Pro	731	487	Accuracy
	ARC-Challenge	689	483	Accuracy

B.2 Baseline Setups

To ensure fair comparison, all baselines are rerun under a unified experimental protocol. All methods use the same backbone LLM, the same dataset split, and the same evaluation metrics as in the experiments. For methods with public implementations, we follow their original core design as closely as possible; otherwise, we reproduce them according to the descriptions in their papers. The mean performance over three independent trials is reported. We describe the configurations of the baseline methods in detail:

Vanilla. A single LLM directly produces the final answer without explicit deliberation or interaction.

CoT. We use standard Chain-of-Thought prompting (Wei et al., 2022).

ComplexCoT. We adopt complexity-based prompting (Fu et al., 2022) with more elaborate intermediate reasoning than standard CoT.

Self-Refine. We follow Self-Refine (Madaan et al., 2023) and let a single LLM iteratively generate, critique, and refine its answer.

SC (CoT×5). We apply self-consistency (Wang et al., 2022) by sampling five CoT reasoning paths and aggregating the final answer.

DyLAN. We follow the collaborative discussion setting of DyLAN (Liu et al., 2024).

MacNet. We use MacNet (Qian et al., 2024b) with a fixed fully connected communication topology.

AgentVerse. We follow the original role-based collaboration framework of AgentVerse (Chen et al., 2024).

AgentPrune. We implement AgentPrune (Zhang et al., 2024a) as a structure-centric adaptive baseline that prunes agents or communication edges before execution.

MaAS. We follow MaAS (Zhang et al., 2025a) as a structure-adaptive baseline that dynamically selects task-relevant agents and interaction patterns.

Puppeteer. We follow Puppeteer (Dang et al., 2025) as an orchestration-centric baseline where a controller dynamically selects which agent to invoke at each step.

B.3 Computational Resources

All experiments are conducted on servers equipped with **8 NVIDIA A800 GPUs**, and mixed-precision training is used throughout. The underlying large language model is deployed through **vLLM** (Kwon et al., 2023) during both training and evaluation. Unless otherwise specified, all reported results are obtained under the same hardware setting.

Our framework is trained in two stages. The prior learning stage typically takes about **3–6 hours** per dataset. The policy optimization stage is more

computationally intensive due to rollout sampling, reward computation, and parameter updates, and usually requires about **12–24 hours** per dataset. During evaluation, a full benchmark run typically takes about **1–3 hours**.

B.4 Agent Setups

We instantiate a pool of functionally specialized agents with distinct roles. Table 5 summarizes the candidate agent pools used for different task families, while Figures 8–11 provide the corresponding role prompts.

Table 5: Candidate agent pools.

Task Family	Candidate Agent Pool
Question Answering [Fig.9]	TaskRouter, AnalyzeAgent, ChoiceEliminator, EvidenceChecker, Skeptic, QASynthesizer
Math Reasoning [Fig.10]	WordProblemParser, MathSolver, ArithmeticChecker, StepChecker, AlgebraSimplifier, GeneralCritic
Code Generation [Fig.11]	AlgorithmDesigner, CodeWriting, CodeReviewer, UnitTestWriter, EdgeCaseHunter, BugFixer
Auxiliary Roles [Fig.8]	TaskPlanner, Summarizer, BudgetController, RedTeamCritic

In our framework, prior shaping is achieved by first estimating task- and budget-conditioned agent relevance scores to identify which roles are most useful for the current input, and then modeling interaction plausibility to characterize which inter-agent communications are likely to be beneficial. Based on these two components, the framework organizes their communication according to the learned orchestration structure.

During analysis, we maintain a lightweight mapping that canonicalizes implementation-specific agent names into shared functional roles (e.g., mapping planning-related agents to Planner) to enable consistent role-level statistics and comparisons.

B.5 Algorithm

This section summarizes the training procedure of MACA, as detailed in Algorithm 1. Training consists of two stages: learning a task- and budget-conditioned structural prior, and optimizing a token-aware orchestration policy over execution trajectories. The policy is trained with policy-gradient updates to favor task-effective coordination under explicit budget constraints.

Figure 8: Auxiliary role prompts.

TaskPlanner

TaskPlanner: Produces a coarse multi-step execution plan before detailed reasoning begins.
Prompt summary: Break the task into a small number of necessary steps, identify dependencies between them, and suggest a compact execution plan.

Summarizer

Summarizer: Compresses intermediate discussion into a concise state summary for downstream agents.
Prompt summary: Summarize the most useful intermediate conclusions, unresolved uncertainties, and next-step recommendations in a compact form.

BudgetController

BudgetController: Monitors communication cost and recommends more economical coordination behavior.
Prompt summary: Estimate whether the current discussion is becoming unnecessarily expensive and suggest cheaper alternatives when possible.

RedTeamCritic

RedTeamCritic: Stress-tests the current reasoning by searching for overlooked risks, counterexamples, or failure modes.
Prompt summary: Deliberately challenge the current solution, identify brittle assumptions, and point out scenarios in which the answer may fail.

Figure 9: Question-answering role prompts.

TaskRouter

TaskRouter: Determines the task type and proposes a minimal, task-appropriate coordination team.
Prompt summary: Classify the task as code, math, or QA, and provide a concise routing rationale without solving the task.

AnalyzeAgent

AnalyzeAgent: Extracts task constraints, key entities, hidden assumptions, and salient evidence from the input and peer outputs.
Prompt summary: Analyze the problem structure carefully and summarize the most decision-relevant information.

ChoiceEliminator

ChoiceEliminator: Reduces the answer space by explicitly ruling out implausible or contradicted options.
Prompt summary: Eliminate incorrect choices one by one and then provide the final answer letter.

EvidenceChecker

EvidenceChecker: Verifies whether candidate answers are supported by the question facts and flags contradictions.
Prompt summary: Match each option against the evidence, identify factual mismatches, and select the most supported answer.

Skeptic

Skeptic: Challenges intermediate reasoning and exposes weak assumptions or unsupported jumps.
Prompt summary: Critique other agents' reasoning, highlight likely errors, and question overconfident conclusions.

QASynthesizer

QASynthesizer: Produces the final concise answer by aggregating validated evidence from upstream agents.
Prompt summary: Return a short, direct final answer grounded in the strongest available evidence.

Figure 10: Math reasoning role prompts.

WordProblemParser

WordProblemParser: Transforms a math word problem into explicit quantities, variables, and equations before solving.
Prompt summary: List known values, define variables, and derive equations or constraints without computing the final answer.

MathSolver

MathSolver: Carries out the main step-by-step derivation for numerical reasoning tasks.
Prompt summary: Solve the problem sequentially, show the key arithmetic or algebraic steps, and produce the final numeric answer.

ArithmeticChecker

ArithmeticChecker: Recomputes arithmetic operations to detect calculation mistakes in intermediate or final steps.
Prompt summary: Recalculate sums, products, ratios, and final values, then confirm or correct the numeric answer.

StepChecker

StepChecker: Identifies the first invalid step in a reasoning chain and repairs it.
Prompt summary: Locate the first incorrect step, explain why it is wrong, and provide a corrected derivation with unit consistency.

AlgebraSimplifier

AlgebraSimplifier: Simplifies symbolic expressions and rewrites equations into cleaner, easier-to-verify forms.
Prompt summary: Rearrange equations and simplify intermediate expressions while preserving correctness.

GeneralCritic

GeneralCritic: Detects hidden assumptions, missing constraints, and logically weak solution paths.
Prompt summary: Review the proposed solution critically and point out logical gaps, overlooked cases, or unjustified assumptions.

Figure 11: Code generation role prompts.

AlgorithmDesigner

AlgorithmDesigner: Designs the algorithmic strategy and data structures before implementation.
Prompt summary: Outline the intended approach, complexity, and edge cases without writing the full program.

CodeWriting

CodeWriting: Produces the executable implementation for the target coding task.
Prompt summary: Write the complete Python solution, preserve the required function signature, and output valid code only.

CodeReviewer

CodeReviewer: Inspects correctness, boundary conditions, and code quality after implementation.
Prompt summary: Review the candidate program for logical bugs, edge cases, and maintainability issues.

UnitTestWriter

UnitTestWriter: Constructs representative and adversarial tests for validating candidate programs.
Prompt summary: Provide concise assertions or test cases that cover both standard and corner-case behavior.

EdgeCaseHunter

EdgeCaseHunter: Searches for pathological inputs and counterexamples likely to break a candidate solution.
Prompt summary: List tricky boundary conditions or adversarial inputs and explain why they are risky.

BugFixer

BugFixer: Repairs incorrect implementations using minimal but targeted modifications.
Prompt summary: Identify the failure mode and provide a corrected version of the code with minimal necessary changes.

Algorithm 1 Algorithm of MACA

Require: Training dataset $\mathcal{D}_{train}$, Agent set $\mathcal{O}$, Generation group size g, Reference policy π_{ref}, Budget constraint b, Threshold γ, Regularization coefficient λ

Output: Orchestration policy π_θ

Structural Prior Learning:

for each episode **do**
  Sample $\{G^{(k)}\}_{k=1}^{K} \sim \pi_\phi(\cdot \mid x)$
  Update ϕ accordingly ▷ Eq. 5
  Derive ψ from the sampled structures $G^{(k)}$ via π_ϕ
end for

Token-Aware Orchestration Policy Learning:

Initialize parameters θ
for each episode **do**
  Sample a task $x \sim \mathcal{D}_{train}$
  Compute agent relevance prior Z_{prior} given (x, b, γ) ▷ Eq. 4
  Compute interaction plausibility prior P_{prior} with MLP_ψ ▷ Eq. 7
  Construct GraphSpec
  Construct State s and Action mask $\mathcal{M}$
  Sample outputs $\{\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(g)}\} \sim \pi_\theta(\cdot|s, \mathcal{M})$
  Obtain rewards $\{r_\lambda^{(1)}, r_\lambda^{(2)}, \ldots, r_\lambda^{(g)}\}$ ▷ Eq. 12
  Compute advantage for each group member ▷ Eq. 9
  Compute loss w.r.t. π_θ ▷ Eq. 10
  Update θ accordingly
end for

Return π_θ