ReSo: A Reward-driven Self-organizing LLM-based Multi-Agent System for Reasoning Tasks

Anonymous ACL submission

Abstract

Multi-agent systems have emerged as a promising approach for enhancing the reasoning capabilities of large language models in complex problem-solving. However, current MAS frameworks are limited by poor flexibility and scalability, with underdeveloped optimization strategies. To address these challenges, we propose ReSo, which integrates task graph generation with a reward-driven two-stage agent selection process. The core of ReSo is the proposed Collaborative Reward Model, which can provide fine-grained reward signals for MAS cooperation for optimization. We also introduce an automated data synthesis framework for generating MAS benchmarks, without human annotations. Experimentally, ReSo matches or outperforms existing methods. ReSo achieves **33.7%** and **32.3%** accuracy on Math-MAS and SciBench-MAS SciBench, while other methods completely fail.

1 Introduction

Increasing inference time has emerged as a critical method to enhance the reasoning capabilities of large language models (LLMs)(Snell et al., 2024). Two primary approaches have been explored: (1) optimizing a large reasoning model (Xu et al., 2025) by reinforcement learning and reward models during post-training, which could generate intermediate reasoning steps before answering (OpenAI et al., 2024b; DeepSeek-AI et al., 2025) and (2) leveraging multi-agent system (MAS) collaboration to complete complex tasks that are difficult to solve by single inference (Han et al., 2024; Guo et al., 2024; Wang et al., 2024b; Tran et al., 2025). Compared to the success of inference time scaling on the single LLM, MAS faces multiple challenges. (1) Most are handcrafted, with limited scalability and adaptability. The lack of an effective agent self-organization mechanism hinders large-scale cooperation. (2) Most assume all agent abilities are fully known while assigning tasks, which is unrealistic for LLM-based agents. (3) Reward signals are restricted to missing, self-evaluation or outcome only, resulting in poorly defined optimization objectives. (4) Existing MASs lack mechanisms for dynamically optimizing agent networks, making it difficult to achieve data-driven improvements. To address these limitations, we ask: Can we design a self-organizing MAS to learn directly from data via reward signals without handcrafting?

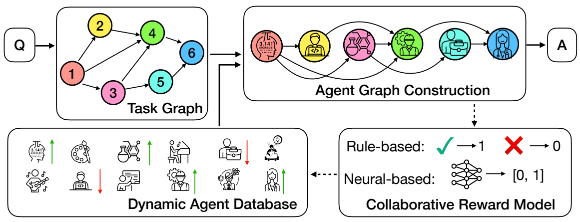


Figure 1: Overview of ReSo pipeline. ReSo first decomposes the task into a DAG; and then constructs an agent graph by topological sorting. First, it searches for agent candidates for each subtask node from the dynamic agent database (DADB). Then it leverages the Collaborative Reward Model (CRM) to choose the best agent and update the agent estimation in DADB.

To realize this potential, we propose ReSo, a reward-driven self-organizing MAS that integrates task graph generation and agent graph construction. The key innovation of our approach is the incorporation of fine-grained reward signals by the Collaborative Reward Model (CRM), which leads to dynamic optimization of agent collaboration. Different from existing MASs, our approach is both scalable and optimizable, achieving state-of-the-art performance on complex reasoning tasks.

While ReSo builds on prior work in agent selection and task decomposition, its principal contribution is the integrated formulation of these mechanisms within a self-organizing multi-agent reasoning framework. Our core insight is that individual agents exhibit heterogeneous expertise across different tasks and domains. During training, the CRM module evaluates each agent's performance and records these scores in the DADB in 3.3.1. At inference time, ReSo decomposes a complex problem into subtasks and consults the DADB to dynamically assign each subtask to the agent best suited for it. This emergent, self-organizing process sets ReSo apart from traditional, linear pipeline architectures. While extensive datasets exist for evaluating the reasoning capabilities of LLMs (Chang et al., 2023; Guo et al., 2023), high-quality MAS evaluation benchmarks are scarce. Therefore, we propose an automatic data synthesis method to generate various MAS tasks by converting existing LLM benchmarks into complex collaboration problems. This method provides step-by-step reward signals without additional human annotations, enabling efficient and scalable MAS evaluation. Our contributions can be summarized as:

- We first propose a Collaborative Reward Model, which can provide fine-grained reward signals for multi-agent collaboration.
- We present an automatic data synthesis method to generate arbitrarily complex MAS tasks from existing LLM benchmarks.
- We propose ReSo, the first scalable and optimizable self-organizing MAS framework. Experimental results demonstrate the superior performance of ReSo on challenging tasks.

2 Related Work

2.1 Reward Guidance

The reward model has become a critical component in enhancing the capabilities of LLMs through post-training (Wang et al., 2024d). By providing feedback on the quality of LLM outputs, RMs facilitate performance improvement, enabling models to generate more accurate and detailed responses. The concept of reward-guided learning was first introduced in InstructGPT (Ouyang et al., 2022), which uses human feedback to fine-tune LLMs, aligning their behavior with user intent. In addition to outcome-based supervision, process-based supervision has been shown to improve the reasoning process itself (Uesato et al., 2022), enhancing not just the final answer but also the steps leading to it.

Building on this, (Lightman et al., 2023) introduced a process reward model (PRM) fine-tuned on PRM800K, which provides fine-grained and interpretable rewards for every reasoning step. Similarly, (Wang et al., 2024c) developed Math-Shepherd, an approach capable of autonomously generating process supervision data. Despite the advantages of neural-based reward models in terms of generalization, they also suffer from reward hacking (Gao et al., 2022; Skalse et al., 2022). To mitigate this, some recent approaches have employed rule-based rewards (DeepSeek-AI et al., 2025) or fixed inference budgets (Muennighoff et al., 2025), which have also proven effective. Notably, DeepSeek-R1 (DeepSeek-AI et al., 2025) incorporates both output accuracy and reasoning format evaluation, achieving the performance on par with OpenAI-O1 (OpenAI et al., 2024b; Qin et al., 2024). DeepSeek-R1 demonstrates that only using large-scale reinforcement learning based on rule-based reward during post-training can stimulate LLM's excellent reasoning ability, without supervised fine-tuning.

2.2 Multi-Agent System

Recent advances in LLM-based MAS have raised expectations for their ability to tackle increasingly complex reasoning tasks (Han et al., 2024; Guo et al., 2024; Wang et al., 2024b; Tran et al., 2025).

Predefined cooperation in MAS relies on structured interactions and role assignments before collaboration. Early works focus on MAS infrastructure, including Camel, AutoGen, and AgentVerse (Li et al., 2023; Wu et al., 2023; Chen et al., 2023). Some approaches adopt standard operating procedures for structured task decomposition, as seen in MetaGPT and ChatDev (Hong et al., 2024; Qian et al., 2024a; Dong et al., 2024). Fixed topologies are most adopted, such as hierarchical structures in MOA (Wang et al., 2024a) and directed acyclic graphs in MacNet and MAGDI (Qian et al., 2024b; Chen et al., 2024c). Predefined role interactions are also widely used such as debate (Du et al., 2023), criticism (Chen et al., 2024b), and certain math reasoning patterns (Gou et al., 2024; Lei et al., 2024; Xi et al., 2024). Predefined MASs exhibit several limitations including: (1) Scalability and adaptability being constrained by the imposition of rigid role assignments and fixed topological structures. (2) The unrealistic assumption that the agent's abilities are fully known when assigning tasks, which is particularly problematic for LLM-based agents.

Optimizable cooperation in MAS aims to dynamically adapt interaction topology and agent roles. GPTSwarm (Zhuge et al., 2024) formulates MAS as optimizable computational graphs, refining node prompts and inter-agent connectivity via evolutionary algorithms. DyLAN (Liu et al., 2024b) employs a layerwise feedforward agent network and a mutual rating mechanism to dynamically optimize MAS. G-Designer (Zhang et al., 2025a) utilizes variational graph auto-encoders to optimize MAS. Current optimizing approaches are highly underexplored. They often lack reliable, fine-grained reward signals for MAS collaboration, relying instead on outputs or self-generated reward mechanisms. Meanwhile, dynamic network optimization algorithms for MAS are also lacking.

3 Methods

To tackle the existing challenges in MAS research, we propose two core innovations: (1) ReSo, a reward-driven self-organizing MAS, which is capable of autonomously adapting to complex tasks and a flexible number of agent candidates, eliminating the need for handcrafted solutions. (2) Introduction of a Collaborative Reward Model (CRM), specifically tailored to optimize MAS performance. CRM can deliver fine-grained reward signals on multi-agent collaboration, enabling data-driven MAS performance optimization.

3.1 Problem Formulation

We define a MAS algorithm f_{MAS} as a function that, given a natural language question Q, generates a graph-structured task decomposition, solves each subtask, and produces a final answer:

$$f_{MAS}(Q) \to \big(G = (V, E),\ A_V,\ A_Q\big) \quad (1)$$

Here, $G = (V, E)$ represents the task decomposition graph, which is structured as a directed acyclic graph (DAG). The set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ corresponds to the subtasks derived from Q, while the edges $E \subseteq V \times V$ define the dependencies between these subtasks. The system produces subtask answers $A_V = \{a_{v_1}, a_{v_2}, \ldots, a_{v_n}\}$ and ultimately derives the final answer A_Q. To achieve this, we decompose f_{MAS} into two sub-algorithms:

$$f_{MAS}(Q) = f_{agent} \circ f_{task}(Q) \quad (2)$$

f_{task} is responsible for constructing the task decomposition graph from the input question, ensuring a structured breakdown of the problem into subtasks and dependencies. f_{agent} dynamically selects and assigns appropriate agents to solve the identified subtasks. This modular design enables independent optimization of each component, allowing for greater flexibility and scalability.

For the MAS-generated answer A_Q to be considered correct, the following conditions must be satisfied: (1) All subtask answers must be correct. (2) All directed edges must correctly enforce the dependency relationships among subtasks. (3) The final output A_Q must be correct.

3.2 Task Graph Construction

In the proposed method, f_{task} first transforms the question Q into a directed acyclic task graph G:

$$f_{task} : Q \quad \to \quad G = (V, E) \quad (3)$$

where G represents the decomposition of the original task Q. Each node $v_i \in V$ is a natural language subtask, and each directed edge $(v_i \to v_j) \in E$ indicates that the subtask v_j depends on the successful completion of v_i.

In practice, we perform supervised fine-tuning (SFT) on an LLM to perform this step of task decomposition. Using our synthetic data, we explicitly require the LLM to decompose Q into logical sub-problems, specify their execution order and dependencies, and output in a format of DAG.

3.3 Two-Stage Agent Search

Once the task graph is obtained, we need to assign each subtask to the most appropriate agent. We denote this agent assignment procedure as f_{agent}. Conceptually, f_{agent} classifies each node in the task graph according to the most suitable agent from a large agent pool $\mathcal{A}$, constructing an *agent graph* that maps each node to one or more selected agents.

$$f_{agent} : v_i \in V \quad \to \quad a_i \in \mathcal{A} \quad (4)$$

Since $\mathcal{A}$ can contain a large number of agents, we first introduce the concept of Dynamic Agent Database. Then we decompose the agent graph construction on every subtask into two search algorithms from coarse to fine-grained: first, select a subset of candidates from DADB then utilize the reward model to evaluate and select the best agent.

3.3.1 Dynamic Agent Database

To increase MAS's scalability and flexibility, we propose the Dynamic Agent Database (DADB), denoted as $\mathcal{A}$, which enables adaptive agent selection by maintaining both **static** and **dynamic** agent profiles. For each agent $a_i \in \mathcal{A}$, its static profile includes the base model, role settings, initial prompt,

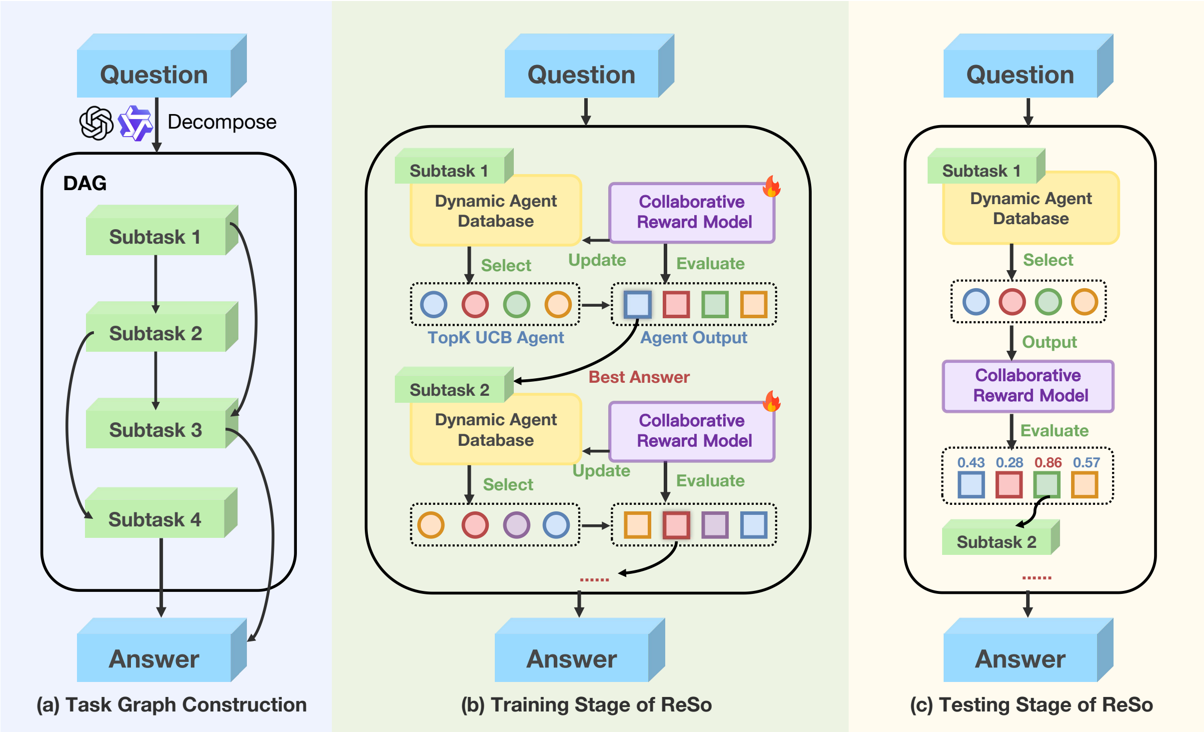


Figure 2: Illustration of our proposed ReSo. (a) We decompose the question into a subtask DAG. (b) The training of ReSo: we first use the UCB score to perform a coarse search in DADB and select top-k agents, then score the inference results using CRM, and update DADB by rewards. Repeat the above process for each node in DAG by topological order. (c) The testing of ReSo: we select the best agent from DADB.

long-term memory, and tools. The dynamic profile, continuously updated via the reward model, tracks the agent's average reward $R(a_i)$, computational cost $C(a_i)$, and task count $n(a_i)$. Initially, agents have only static attributes, while training iteratively refines their evaluations by the process reward model, optimizing future selection.

Given an input task v_j, the DADB assigns a preliminary quality score $Q(a_i, v_j)$ to each agent a_i, balancing task-agent similarity, historical performance, and computational costs:

$$Q(a_i, v_j) = \mathrm{sim}(a_i, v_j) \cdot \mathrm{perform}(a_i) \quad (5)$$

where $\mathrm{sim}(a_i, v_j)$ represents the similarity between the subtask's target profile and the agent's static profile. In practice, we employ a Heaviside function which ensures that only agents exceeding a predefined similarity threshold V_{th} are considered: $\mathrm{sim}(a_i, v_j) = H[\langle \mathbf{q_i}, \mathbf{a_i} \rangle - V_{th}]$ where $\mathbf{q_i}, \mathbf{a_i}$ are text embedding of subquestion and the agent static profile. The $\mathrm{perform}(a_i)$ term is given by $\mathrm{perform}(a_i) = R(a_i) - \beta C(a_i)$, where β controls the trade-off between the agent's historical performance and cost.

3.3.2 Coarse Agent Search by UCB

Given a DADB $\mathcal{A}$ and a subtask v_j, our first objective is to retrieve a promising subset of k candidate agents. To take advantage of the known information in DADB, also to explore unused agents, we adopt an Upper Confidence Bound value:

$$\mathrm{UCB}(a_i, q_j) = Q(a_i, q_j) + c\sqrt{\frac{N}{n(a_i) + \varepsilon}} \quad (6)$$

where N is the total number of agent selections and $n(a_i)$ the number of times agent i is selected, $\varepsilon \ll 1$. c is a constant controlling the exploration-exploitation trade-off. Agents with higher UCB scores are more likely to be selected, helping the MAS to explore potentially under-utilized agents. For each subtask q_i, we sort agents by their $\mathrm{UCB}(a_i, q_j)$ and choose the top k agents as the candidate set $A_{\text{cand}} = \{a_1, a_2, \ldots, a_k\}$.

3.3.3 Fine-grained Agent Evaluation by CRM

Once the candidate agents $\mathcal{A}_{\text{cand}}$ are selected, we evaluate their performance on the current subtask v_j using a Collaborative Reward Model (CRM). This evaluation process is straightforward: each candidate agent a_i generates an answer to the subtask v_j: $a_i(v_j)$, and then we assess the quality of that answer based on a reward signal:

$$r(a_i, v_j) \;=\; \text{RewardModel}\Big(a_i, v_j, a_i(v_j)\Big) \quad (7)$$

where RewardModel evaluates the quality of the solution based on the given agent's profile, subtask, and previous reasoning process. After evaluating the agents, we assign the agent with the highest reward, a_j^*, to the subtask node v_j, which means a_j^*'s solution is used as v_j's answer. This process is repeated for each subtask on the graph.

The reward $r(a_i, v_j)$ is computed using the CRM, which can be either rule-based (e.g., binary correctness: 0 for incorrect, 1 for correct) or neural-based (providing a score between 0 and 1 for quality). The reward model evaluates how well the agent's response aligns with the expected outcome, factoring in both the solution's correctness and its collaboration within the MAS.

3.4 Training and Inference Stage

Our multi-agent system can operate in two modes: training and testing. During **training**, we leverage a high-quality reward $r(a_i, v_j)$ available for evaluating the correctness of every step of MAS. Upon receiving $r(a_i, v_j)$ for each candidate agent, we update that agent's dynamic profile in DADB. For instance, we may maintain a running average of rewards:

$$R(a_i) \;\leftarrow\; \frac{n(a_i) \cdot R(a_i) + r(a_i, v_j)}{n(a_i) + 1} \qquad (8)$$

similar for updating $costc(a_i, v_j)$. By iteratively learning from data, the DADB can dynamically update agent evaluations based on historical reward, facilitating adaptive agent selection and improving both efficiency and performance. During **testing**, the reward model is no longer required. Instead, we leverage the learned DADB to select the best agent candidates and the best answer to each subtask.

3.5 The Perspective of MCTS

The task graph, after topological sorting, forms a decision tree where each node represents a subtask and the edges denote dependencies. At each level, we use UCB to prune the tree and select a subset of promising agents, then simulate each agent and evaluate their performance using the CRM. The resulting reward updates the agent's dynamic profile, refining the selection strategy. The MAS construction is essentially finding the optimal path from the root to the leaves, maximizing the UCB reward for the best performance.

Consider there are N agents and a task requiring D agents to collaborate. Assume that the average inference cost is c and the matching cost in DADB is $s \ll c$ per agent. A brute-force search has a complexity of $O(c \cdot N^D)$, which becomes infeasible as D and D grow. In contrast, our self-organizing strategy, selecting top$k$ per step, reduces the cost to $O((s \cdot N + N \log N + k \cdot c) \cdot D)$, offering a near-linear scaling with N and D, making the approach highly scalable for large N and D.

4 Data Synthesis

A key challenge in MAS is the lack of structured datasets for evaluating and training agent collaboration. To address this, we propose an automated framework that converts existing LLM datasets into structured, multi-step MAS tasks, enabling fine-grained evaluation without human annotations.

Random DAG Generation We begin by generating a DAG, $G = (V, E)$. Each node $v_i \in V$ will be filled with a subtask (q_i, a_i), where q_i is the textual description of the task, and a_i is its numerical answer. The subtasks are sampled from the existing LLM benchmarks. The edges E will encode dependency constraints between subtasks, ensuring that the solution to one subtask is required as an input for another, modeling the sequential reasoning process of multi-agent collaboration.

Subtask Selection and Filling To populate the nodes of G, we construct a master pool of candidate subtasks, denoted as $\mathcal{P}$. Each candidate subtask $p_i \in \mathcal{P}$ consists of a textual problem description s_i, and a numerical answer a_i. After obtaining $\mathcal{P}$, we randomly sample from it and fill one question per node into the generated DAG. Candidate subtasks should have clear numerical or option answers, such as SciBench (Wang et al., 2024f), Math (Hendrycks et al., 2021), GPQA (Rein et al., 2023), etc. To ensure that the problem is computationally feasible for later dependency construction, we extract a numerical constant $c_i \in \mathbb{R}$ from the problem text. If the extracted constant is valid, the subtask is retained in $\mathcal{P}$; otherwise, it is discarded. This ensures that only problems with well-defined numerical attributes are incorporated.

Dependency Edge Construction After all nodes are populated, we generate natural language dependency descriptions for edges. Each edge $(v_j \to v_k)$ should represent a relationship which connects previous subtask v_j's answer a_j, with subsequent subtask v_k's question parameter c_k. For each edge, we generate a textual description e_{jk}, such as "in this question, c_k = previous answer + 3." Formally, it is an algorithm that constructs a string from two numbers: $e_{ij} = f(a_j, c_k)$. f can be implemented using elementary arithmetic and text templates, ensuring that no answers or parameters in the original subtask need to be manually modified. Once the DAG is fully constructed, we refine node descriptions by removing any explicitly given numerical constants $\{c_i\}$ that are now dependent on the results of prior nodes. Finally, an entire graph described in natural language is a piece of synthetic data.

The proposed data synthesis framework generates structured, multi-step reasoning tasks with adjustable sizes, ensuring diverse and scalable problem structures. The synthesized dataset supports both training and testing, enabling fine-grained evaluation without human annotations.

5 Experiments

In 5.1, we first use public datasets to create complex MAS benchmarks and fine-tune ReSo's task decomposition and collaborative reward models. All code, datasets, and models are publicly available. In 5.2, we train and evaluate ReSo on both public and synthetic datasets. 5.3 presents ablation studies on task decomposition, agent selection, and reward guidance mechanisms.

5.1 Data Synthesis and Model Fine-tuning

5.1.1 Data Synthesis

MATH (Hendrycks et al., 2021) consists of problems from diverse mathematical domains, while SciBench (Wang et al., 2024f) includes scientific reasoning tasks spanning physics, chemistry, and mathematics. Using these datasets, we apply the synthetic data generation method outlined in 4 to create two datasets: one for single LLM fine-tuning and another for benchmarking. Difficulty is categorized by the number of subtasks—Easy (3), Medium (5), and Hard (7).

Fine-tuning data For fine-tuning task decomposition LLM, we generate 14,500 questions and answers from the MATH training set, with numbers of subtasks ranging from 2 to 6. For fine-tuning the neural-based CRM, we generate 5,000 questions from the same set, with 5 subtasks per question.

5.1.2 Model Fine-tuning

Task Decomposition Model Training To ensure high-quality task composition, we fine-tune a specialized model for task decomposition based on Qwen2.5-7B-Instruct. We use 14500 dialogues on task decomposition as described in 5.1.1, and fine-tune the model under a batch size of 128 and a learning rate of 1e-4 for 3 epochs. The fine-tuned model can reliably produce task decomposition in a structured format.

CRM Training The proposed CRM is fine-tuned based on Qwen2.5-Math-PRM-7B (Zhang et al., 2025b), which can provide effective process reward signals on MAS collaborative reasoning tasks. We use 5000 samples of sub-tasks with their answers as described in 5.1.1. We follow a simplified training scheme of PRMs, where the model should only perform binary classification on the special token at the end of the answer. The model is trained with a batch size of 128 and a learning rate of 1e-4 for 5 epochs. The fine-tuned model can output the probability of the answer being correct, which is then taken as the collaborative reward signal.

MAS Benchmarks We select 201 questions from SciBench as the sub-question data pool and synthesized complex data using the method in 4. This forms the SciBench-MAS dataset, comprising 200 easy-level training questions and 247 testing questions (107 easy, 80 medium, 62 hard). For MATH (Hendrycks et al., 2021), 348 level-5 questions are selected, from which we generate the Math-MAS dataset, consisting of 269 test questions for ReSo (91 easy, 89 medium, 89 hard).

5.2 Main Results of ReSo

Models and MASs We compare ReSo with state-of-the-art LLM and MAS methods. Our single-LLM baselines include GPT-4o (OpenAI et al., 2024a), Gemini-2.0-Flash (Team et al., 2024), Claude-3.5-Sonnet (Anthropic, 2024), Qwen2.5-Max (Yang et al., 2024), DeepSeek-V3 (Liu et al., 2024a). For ReSo, we build an agent database that includes these base models, extended to 63 agents with different prompts. For MAS, we evaluate MetaGPT (Hong et al., 2024), DyLAN (Liu et al., 2024b), GPTSwarm (Zhuge et al., 2024), GDesigner (Zhang et al., 2025a). All MAS baselines use GPT-4o as the backbone.

Method	Math-MAS				SciBench-MAS			
	Easy	Medium	Hard	*Tokens*	Easy	Medium	Hard	*Tokens*
GPT-4o	27.5	9.0	0.0	2.2k	39.3	12.5	1.6	2.1k
Gemini-2.0-Flash	69.2	24.7	9.0	3.0k	64.5	33.8	9.7	2.5k
Claude-3.5-Sonnet	12.1	0.0	0.0	1.0k	22.4	6.2	3.2	1.4k
Qwen2.5-Max	44.0	13.5	4.5	2.9k	55.1	30.0	4.8	2.8k
DeepSeek-V3	52.7	24.7	12.4	2.2k	52.3	31.3	12.9	2.3k
MetaGPT	30.8	12.4	2.2	16.1k	48.6	2.5	0.0	14.6k
DyLAN	40.7	9.0	0.0	64.1k	48.6	2.5	0.0	77.8k
GPTSwarm	35.2	5.6	4.5	14.9k	31.8	6.3	1.6	18.2k
GDesigner	14.2	5.6	0.0	16.9k	24.3	12.5	0.0	19.0k
ReSo (ours)	**79.1**	**56.2**	**33.7**	**14.6k**	**67.3**	**51.3**	**32.3**	**20.7k**

Table 1: Accuracy and average token usage on Math-MAS and SciBench-MAS. Bold and underlined represent optimal and suboptimal results, respectively. *Tokens* denotes the average number of tokens consumed per task.

Comparisons with LLMs As shown in Table 1, most single-model agents exhibit a sharp decrease in accuracy as the difficulty increases. At the hard difficulty level, their accuracy approaches zero, suggesting that single LLMs struggle with compositional reasoning. In particular, we show the results of these single LLMs on single Math and Scibench datasets in Appendix B , with accuracy rates of 80%-90%. This means that a single LLM can successfully solve a single sub-problem in the dataset, but its generalization ability for combined complex problems is very limited.

Comparisons with MASs Notably, ReSo outperforms other approaches in both the Math-MAS and SciBench-MAS datasets. At the hard difficulty level, ReSo reaches an accuracy of **33.7%** on Math-MAS and **32.3%** on SciBench-MAS, while other MAS methods almost completely fail.

Results on Standard Benchmarks Our approach demonstrates robust performance not only on complex task datasets but also on widely adopted benchmarks. Table 2 summarizes the comparative accuracy, where ReSo consistently achieves the highest scores across all tasks. These results attest to ReSo's strong generalization capabilities and its effectiveness in mathematical and scientific reasoning, as well as related domains.

5.3 Ablation Studies

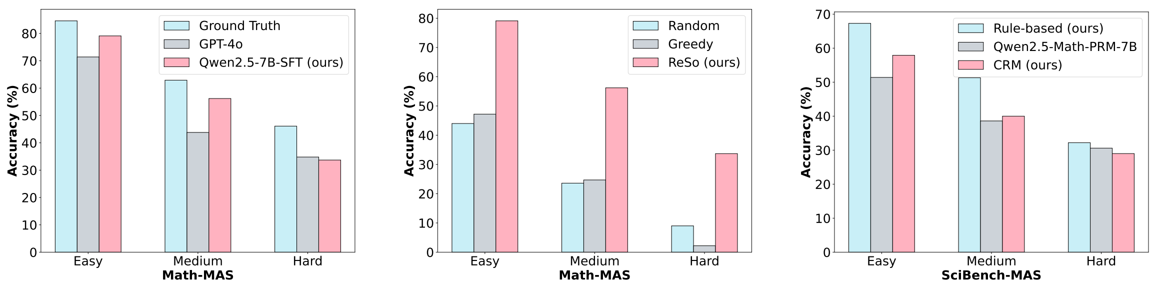
We conduct ablation studies on our proposed multi-agent system, examining three core designs: task decomposition, agent selection, and reward signal.

Task Decomposition We compare three different approaches to task decomposition: (1) **Ground Truth**, representing an upper bound with human-crafted, meticulously designed task breakdowns; (2) **GPT-4o**, which autonomously decomposes complex tasks into sub-tasks without targeted fine-tuning; and (3) **Qwen2.5-7B-SFT**, a model fine-tuned on our dataset based on Qwen2.5-7B, specifically adapted to generate more effective decompositions for complex questions. Figure 3(a) presents the reasoning accuracy under different decomposition strategies. The ground-truth decomposition consistently yields the highest accuracy, underscoring the critical role of precise subproblem segmentation. Meanwhile, the fine-tuned task generator surpasses the naive GPT-4o approach, demonstrating that even a small amount of domain-specific training data can significantly improve decomposition quality and enhance overall system performance.

Table 2: Comparison of accuracy (%) on standard benchmarks.

Method	GSM8K	GPQA	HumanEval	MMLU
DyLAN	88.16	49.55	89.70	80.16
GDesigner	95.07	53.57	89.90	84.50
GPTSwarm	89.74	52.23	88.49	83.98
ReSo (ours)	**95.70**	**55.80**	**92.00**	**88.70**

Agent Selection We compare three strategies for agent selection: a **random** strategy, a **greedy** strategy that always selects the most matching profile, and our proposed **ReSo** approach. As shown in Figure 3(b), **ReSo** significantly outperforms other strategies across all the datasets, which emphasizes the importance of a robust agent selection strategy within the multi-agent framework. By strategically



(a) Task Decomposition (b) Agent Selection (c) Reward Signal

Figure 3: Results of ablation studies. (a) Fine-tuning on domain-specific training data can significantly improve the decomposition quality, thus enhancing overall system performance. (b) Our robust agent selection strategy within the MAS is significant to the performance. (c) Compared to general reward models, our fine-tuned reward model is more task-specific and brings more precise reward signals, thus improving the system performance.

assigning each sub-task to the most suitable agent, the system can handle increasingly complex tasks with markedly better accuracy.

Reward Signal Ablation We investigate the impact of different reward signals on system optimization, considering three approaches. Figure 3(c) presents the results of training our MAS under these reward schemes on the SciBench-MAS dataset. Detailed in Appendix G

5.4 Scalability Analysis

Agent Scalability ReSo's modular design allows the dynamic addition of new agents without retraining the entire system. Each agent registers its static profile in the Dynamic Agent Database (DADB) and begins contributing immediately. For example, during our HumanEval experiments, we simply added some code-specialist agents on top of the existing 63 agents. ReSo seamlessly leveraged its capabilities to improve overall performance.

Task and Domain Generality ReSo is task-agnostic and domain-agnostic: as long as domain-specific data is available, it can generate a task DAG, select appropriate agents, and optimize their collaboration. Our automated data synthesis pipeline converts LLM benchmark into a multi-step MAS task without human annotations, enabling straightforward migration from mathematics and scientific reasoning to other fields.

Training Data Scalability The effectiveness of agent selection in ReSo grows with more training data. During training, DADB maintains and updates each agent's reward statistics and cost estimates. As the number of training samples increases, ReSo builds a more accurate model of each agent's strengths and weaknesses, resulting in progressively better agent assignments and higher overall accuracy. Figure 4 shows that ReSo's accuracy increases with the training process

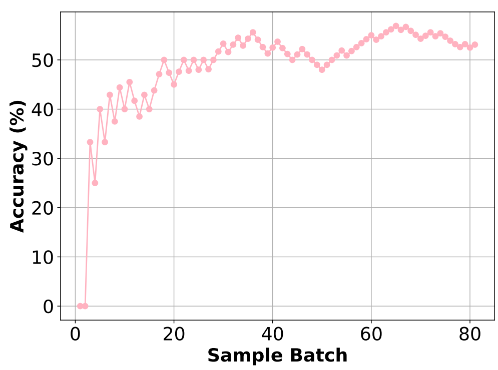


Figure 4: Training Curve of ReSo.

6 Conclusion

In this work, we introduce ReSo, a reward-driven self-organizing MAS for complex reasoning. By integrating a collaborative reward model, ReSo automates agent selection and collaboration, improving scalability and adaptability. The automated data synthesis framework eliminates manual annotations. Experiments show that ReSo outperforms existing MAS and single LLM baselines. All codes, models, and data have been open-sourced. We expect ReSo to enable co-optimization of MAS and LLM to further enhance reasoning capabilities.

7 Limitations

Although the base model for the agents is a fixed model, ReSo has demonstrated strong optimizability and scalability as well as good performance. A further interesting research question is: Can the optimization of MAS be performed together with the optimization of a single LLM agent? Specifically, can the reward signal given to the model by our CRM in each step of cooperation be combined with the reinforcement learning-based post-training of a single model to further optimize MAS at both the macro and micro levels? This means a dynamic agent cooperation network, where agents can not only learn how to interact with each other but also fine-tune their weights through feedback from cooperation. We look forward to follow-up research.

8 Ethical Considerations

While our proposed ReSo framework focuses on reasoning tasks in the domains of mathematics and science, it has the potential to be applied in other, possibly unethical, contexts. Such misuse could pose significant threats to human society. We strongly urge readers to carefully consider these ethical implications and to adopt a conscientious approach in the development and application of these methods.

References

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2023. A survey on evaluation of large language models. *Preprint*, arXiv:2307.03109.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: Process supervision without process. *Preprint*, arXiv:2405.03553.

Justin Chih-Yao Chen, Archiki Prasad, Swarnadeep Saha, Elias Stengel-Eskin, and Mohit Bansal. 2024b. Magicore: Multi-agent, iterative, coarse-to-fine refinement for reasoning. *Preprint*, arXiv:2409.12147.

Justin Chih-Yao Chen, Swarnadeep Saha, Elias Stengel-Eskin, and Mohit Bansal. 2024c. Magdi: Structured distillation of multi-agent interaction graphs improves reasoning in smaller language models. *Preprint*, arXiv:2402.01620.

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. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. *Preprint*, arXiv:2308.10848.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu

Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. 2024. Self-collaboration code generation via chatgpt. *Preprint*, arXiv:2304.07590.

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

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2024. Alphazero-like tree-search can guide large language model decoding and training. *Preprint*, arXiv:2309.17179.

Leo Gao, John Schulman, and Jacob Hilton. 2022. Scaling laws for reward model overoptimization. *Preprint*, arXiv:2210.10760.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. Tora: A tool-integrated reasoning agent for mathematical problem solving. *Preprint*, arXiv:2309.17452.

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. *Preprint*, arXiv:2402.01680.

Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Supryadi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, and Deyi Xiong. 2023. Evaluating large language models: A comprehensive survey. *Preprint*, arXiv:2310.19736.

Shanshan Han, Qifan Zhang, Yuhang Yao, Weizhao Jin, Zhaozhuo Xu, and Chaoyang He. 2024. Llm multi-agent systems: Challenges and open problems. *arXiv preprint arXiv:2402.03578*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. *Preprint*, arXiv:2308.00352.

Bin Lei, Yi Zhang, Shan Zuo, Ali Payani, and Caiwen Ding. 2024. Macm: Utilizing a multi-agent system for condition mining in solving complex mathematical problems. *Preprint*, arXiv:2404.04735.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. *Preprint*, arXiv:2303.17760.

Qingyao Li, Wei Xia, Kounianhua Du, Xinyi Dai, Ruiming Tang, Yasheng Wang, Yong Yu, and Weinan Zhang. 2024. Rethinkmcts: Refining erroneous thoughts in monte carlo tree search for code generation. *Preprint*, arXiv:2409.09584.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024b. A dynamic llm-powered agent network for task-oriented agent collaboration. *Preprint*, arXiv:2310.02170.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *Preprint*, arXiv:2406.06592.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary

Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024a. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. 2024b. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

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. 2024a. Chatdev: Communicative agents for software development. *Preprint*, arXiv:2307.07924.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2024b. Scaling large-language-model-based multi-agent collaboration. *Preprint*, arXiv:2406.07155.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, and Pengfei Liu. 2024. O1 replication journey: A strategic progress report – part 1. *Preprint*, arXiv:2410.18982.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *Preprint*, arXiv:2311.12022.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward hacking. *Preprint*, arXiv:2209.13085.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *Preprint*, arXiv:2501.06322.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *Preprint*, arXiv:2211.14275.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024a. Mixture-of-agents enhances large language model capabilities. *Preprint*, arXiv:2406.04692.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024b. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024c. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *Preprint*, arXiv:2312.08935.

Shuhe Wang, Shengyu Zhang, Jie Zhang, Runyi Hu, Xiaoya Li, Tianwei Zhang, Jiwei Li, Fei Wu, Guoyin Wang, and Eduard Hovy. 2024d. Reinforcement learning enhanced llms: A survey. *Preprint*, arXiv:2412.10400.

Tianlong Wang, Junzhe Chen, Xueting Han, and Jing Bai. 2024e. Cpl: Critical plan step learning boosts llm generalization in reasoning tasks. *Preprint*, arXiv:2409.08642.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R. Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. 2024f. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *Preprint*, arXiv:2307.10635.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, Xiao Wang, Rui Zheng, Tao Ji, Xiaowei Shi, Yitao Zhai, Rongxiang Weng, Jingang Wang, Xunliang Cai, Tao Gui, Zuxuan Wu, Qi Zhang, Xipeng Qiu, Xuanjing Huang, and Yu-Gang Jiang. 2024. Enhancing llm reasoning via critique models with test-time and training-time supervision. *Preprint*, arXiv:2411.16579.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. 2025. Towards large reasoning models: A survey of reinforced reasoning with large language models. *Preprint*, arXiv:2501.09686.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: Llm self-training via process reward guided tree search. *Preprint*, arXiv:2406.03816.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2025a. G-designer: Architecting multi-agent communication topologies via graph neural networks. *Preprint*, arXiv:2410.11782.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Language agents as optimizable graphs. *Preprint*, arXiv:2402.16823.

A Related work on LLM Reasoning Policy

Reward model is usually combined with different reasoning policies to enhance its effect such as majority voting (Wang et al., 2023), Chain of Thought (COT) (Wei et al., 2023) and Monte Carlo Tree Search (MCTS) (Browne et al., 2012). OmegaPRM (Luo et al., 2024) enhances reasoning with a divide-and-conquer MCTS strategy. ReST-MCTS (Zhang et al., 2024) refines reasoning traces using inferred stepwise rewards. RethinkMCTS (Li et al., 2024) improves code generation by leveraging execution feedback. In contrast, Critical Plan Step Learning (Wang et al., 2024e) employs hierarchical MCTS to generalize across reasoning tasks. Additionally, AlphaMath (Chen et al., 2024a) and TS-LLM (Feng et al., 2024) enhance reasoning by incorporating a value model and iterative tree search, with TS-LLM further leveraging an AlphaZero-like framework and policy distillation.

B Model Performance

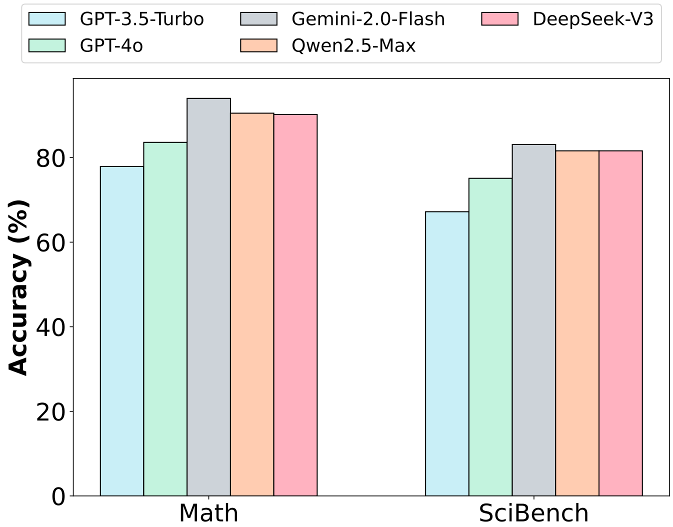


Figure 5: Performance of different models on our selected Math and SciBench dataset subproblems.

C Case Study

Complex Task Synthesis Case Study

Original Question:
A model for the surface area of a human body is given by

$$S = 0.1091\, w^{0.425}\, h^{0.725}.$$

When ultraviolet radiation of wavelength UNK_0 (where UNK_0 = Answer[2] + 56.10 nm) strikes the skin, . . . ; a muscle fiber contracts by 3.5 cm and lifts a weight, assuming Hooke's law $F = -kx$ with $k =$ UNK_1 $=$ Answer[0] $+ 736.00$; finally, please calculate

$$\text{Answer[0]} \times \text{Answer[1]} \times \text{Answer[2]}$$

and conclude: "The answer is therefore $\boxed{[ANSWER]}$."

Decomposed Task Graph:

- Task 1 (no deps): Compute S, record as Answer[2].
- Task 2 (dep: 1): Set UNK_0 $=$ Answer[2] $+ 56.10$, compute UV result, record as Answer[0].

- Task 3 (dep: 2): Set UNK_1 = Answer[0] + 736.00, compute work via Hooke's law, record as Answer[1].
- Task 4 (deps: 1,2,3): Compute the product Answer[0]·Answer[1]·Answer[2] and box the result.

Agent Routing:

- Task 1 (Calculus)→ *gemini-2.0-flash-exp_GeometryExpert*
- Task 2 (Matter)→ *gpt-4o_ElectromagnetismExpert*
- Task 3 (Thermodynamics)→ *qwen2.5-max_Thermodynamics&OpticsExpert*
- Task 4 (Aggregation)→ *gemini-2.0-flash-exp_AlgebraExpert*

D Prompt

Prompt of Agents in the Pool

```
[gpt-4o_1]
model = gpt-4o
role = MechanicsExpert
prompt = You are a highly knowledgeable mechanics expert in a multi-agent system. You are given
↪  a sub-task related to classical mechanics, statics, dynamics, kinematics, or fluid
↪  mechanics. First, read and understand the previous questions and answers from other agents.
↪  Identify the variables that have already been solved and ensure consistency with their
↪  results. Then, systematically break down your sub-task, applying relevant physical laws
↪  such as Newton's laws, conservation principles, or motion equations. Justify your
↪  reasoning, verify unit consistency, and cross-check with previous agent outputs before
↪  providing a well-explained solution.


[gpt-4o_2]
model = gpt-4o
role = ElectromagnetismExpert
prompt = You are an expert in electromagnetism within a multi-agent system. You are assigned a
↪  sub-task related to electric fields, magnetic fields, circuit analysis, or electromagnetic
↪  waves. First, read and understand the previous questions and answers from other agents,
↪  extract solved variables, and ensure logical consistency. Apply fundamental principles such
↪  as Maxwell's equations, Gauss's law, or Faraday's law to solve your sub-task systematically.
↪  Clearly outline your steps, justify the assumptions, and verify that your solution aligns
↪  with previous agents' work. If discrepancies arise, propose possible resolutions.

[gpt-4o_3]
model = gpt-4o
role = Thermodynamics&OpticsExpert
prompt = You are an expert in thermodynamics and optics in a multi-agent system. Your role is
↪  to solve a specific sub-task while ensuring coherence with previous agents' results. First,
↪  read and understand the previous discussions, extract solved variables, and align your
↪  approach with existing solutions. Apply principles such as the first and second laws of
↪  thermodynamics, heat transfer models, or optical laws (e.g., Snell's law, diffraction, and
↪  wave optics). Provide a detailed step-by-step solution, justify calculations, and validate
↪  numerical consistency with prior agent outputs. If uncertainties arise, suggest possible
↪  clarifications.

[gpt-4o_4]
model = gpt-4o
role = InorganicChemistryExpert
```

```
prompt = You are an inorganic chemistry expert operating in a multi-agent system. Your sub-task
↪  may involve chemical bonding, periodic trends, reaction mechanisms, or coordination
↪  chemistry. Carefully review the previous questions and answers, identify already
↪  determined variables, and ensure consistency with past calculations. Apply relevant
↪  chemical principles to analyze and solve your assigned problem step by step. Provide
↪  balanced chemical equations, validate reaction feasibility, and explain your reasoning
↪  clearly. If your results depend on prior agents' outputs, verify their correctness and
↪  suggest refinements if necessary.

[gpt-4o_5]
model = gpt-4o
role = OrganicChemistryExpert
prompt = You are an organic chemistry expert in a multi-agent system, responsible for solving a
↪  sub-task related to molecular structures, reaction mechanisms, or synthetic pathways.
↪  First, review previous discussions, extract key solved variables, and ensure consistency
↪  with prior agent responses. Then, apply organic chemistry principles such as resonance
↪  effects, nucleophilic-electrophilic interactions, and reaction kinetics to derive a
↪  precise solution. Provide clear mechanistic explanations, reaction diagrams if necessary,
↪  and cross-check results to maintain logical coherence within the system.
```

Figure 6: The prompt of agents in the pool.

1166

Prompt of the Task Plan Generator

```
"""
You are an AI assistant specialized in generating structured prompts for domain-specific
↪  experts in a multi-agent system.

**Task:**
Given a subquestion, analyze its domain, required expertise, and problem complexity. Then,
↪  generate a structured prompt that precisely describes the expert's role in solving the
↪  problem. The generated prompt will be used for vector-based similarity matching to select
↪  the most appropriate agent from an agent pool.

**Prompt Format:**
"You are a [Expert Type], highly skilled in [Specific Knowledge Areas]. Your task is to analyze
↪  the problem by first reviewing previously solved variables and solutions from other agents
↪  in the multi-agent system. Apply domain-specific knowledge to reason rigorously and
↪  provide a well-structured, logically sound answer. If calculations are required, show all
↪  steps. If problem decomposition is needed, outline a systematic approach. Ensure
↪  consistency with previous solutions in the multi-agent system and resolve any
↪  discrepancies when necessary. Your role is to assist in solving complex reasoning problems
↪  with precision and alignment with the broader system."

**Instructions for Prompt Generation:**
1. **Expert Type Selection**: Identify the most relevant expert type (e.g., MechanicsExpert,
↪  AlgebraExpert, ThermodynamicsExpert).
2. **Specific Knowledge Areas**: Define the precise knowledge fields required to solve the
↪  problem.
3. **Problem Scope & Complexity**: Determine whether the problem requires deep theoretical
↪  knowledge, numerical computation, or practical modeling.

**Output:**
Provide only the generated prompt without additional explanations."""
```

Figure 7: The prompt of the task plan generator.

1167

1168 # E Agent Selection Visualization

1169 The agent selection distribution during the testing phase of Scibench-MAS-Easy reveals that Gemini-2.0-
1170 Flash-Exp and Qwen2.5-Max were the most frequently selected models after training.

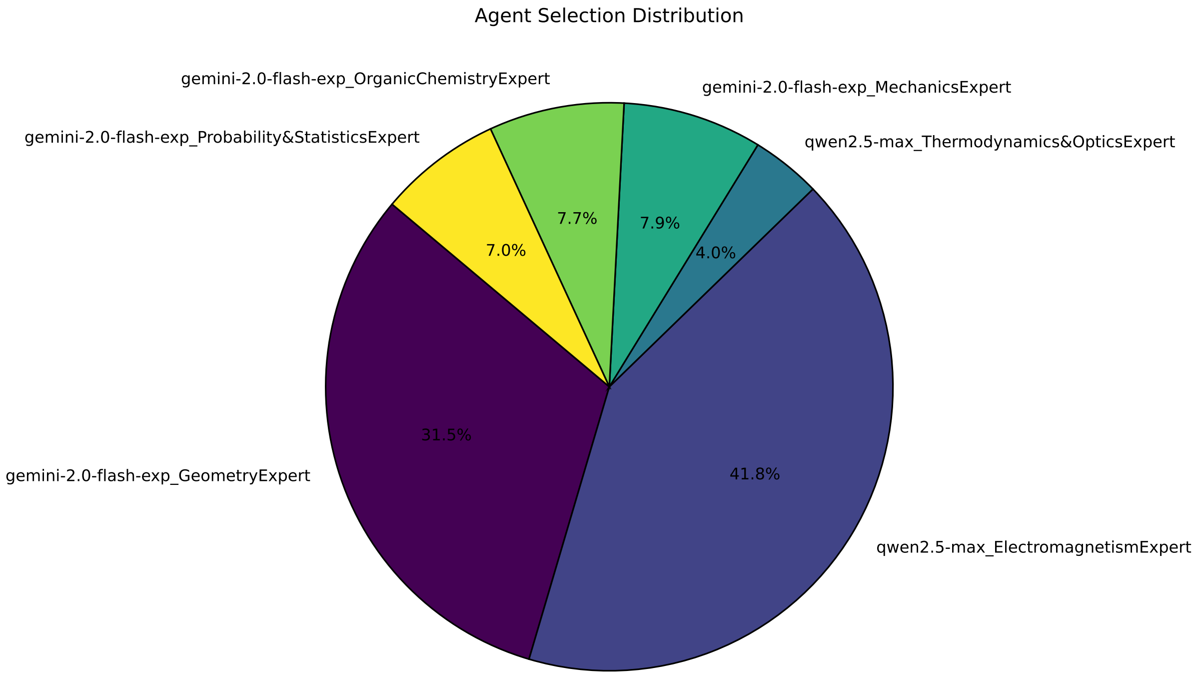


Figure 8: Testing stage on the easy-level tasks in Scibench-MAS.

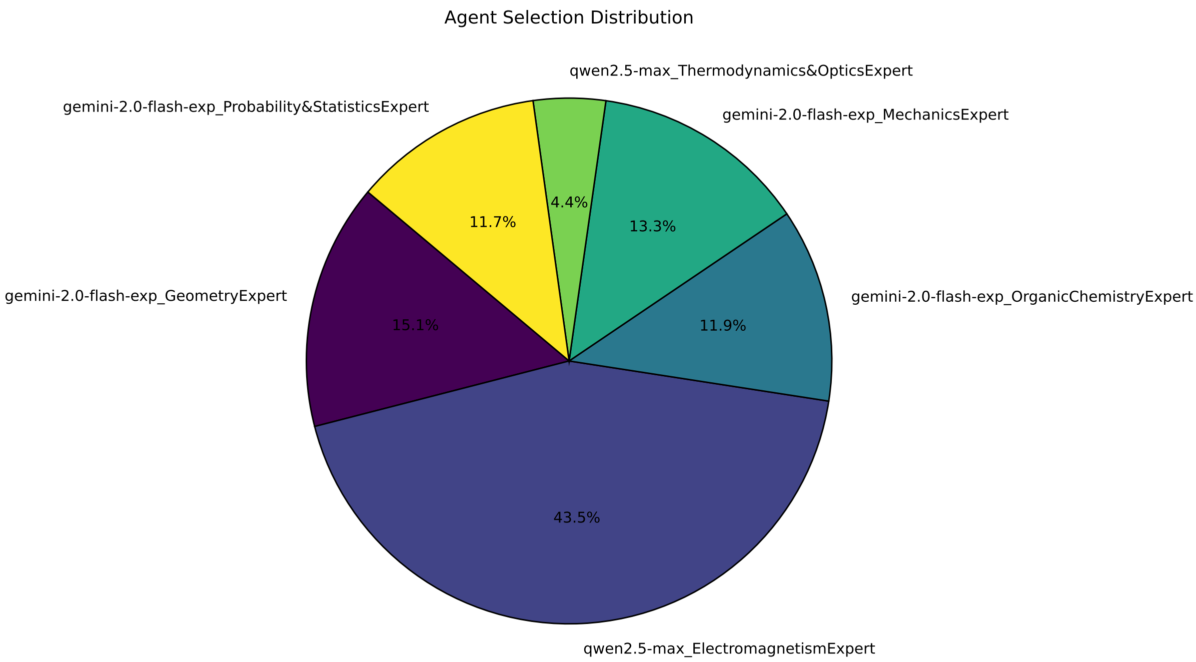


Figure 9: Testing stage on the hard-level tasks in Scibench-MAS.

F Hyperparameters

During both training and testing, a set of weighted factors and constraints guide agent selection, allowing for dynamic adjustments. Specifically, `similarity_weight` = 0.6 regulates the influence of subproblem-agent similarity, `reputation_weight` = 1.0 balances agent selection based on past performance, and `cost_weight` = 1.0 accounts for computational overhead. A `THRESHOLD` = 0.6 establishes the similarity cutoff for specialized handling of certain subproblems, while `EXPLORATION_CONST` = 0.3 encourages periodic assignments to underutilized agents. During testing, hyperparameters can be adjusted to fine-tune the selection process—modifying `similarity_weight` and `THRESHOLD` controls the search scope, adjusting `reputation_weight` increases the weight of agent reputation in scoring, and tweaking `cost_weight` alters the impact of computational overhead, enabling a flexible trade-off between efficiency and performance. Finally, `TOP_K` = 3 restricts the number of candidate agents per subproblem, balancing exploration and efficiency in the selection process.

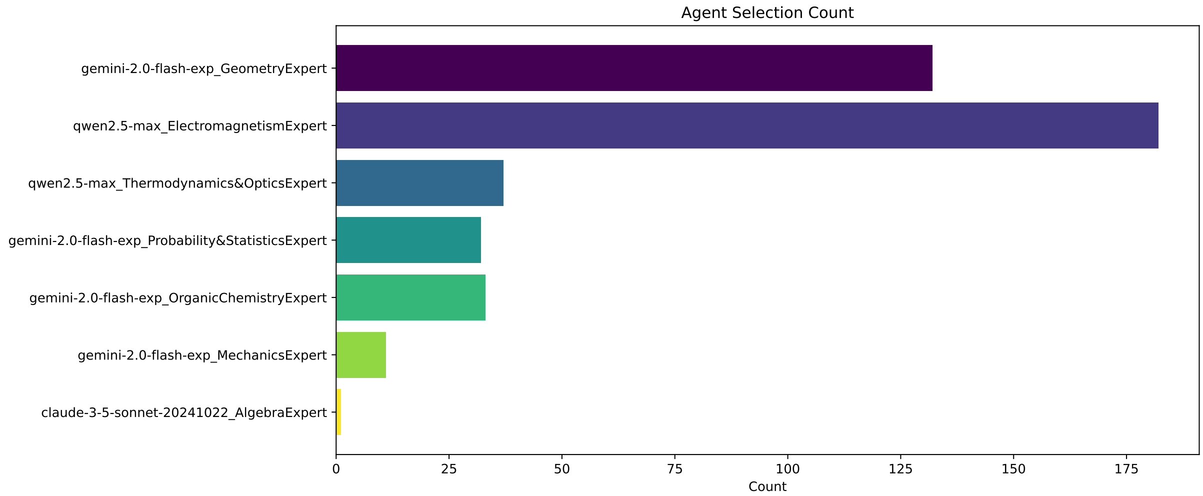


Figure 10: Testing stage on the medium-level tasks in Scibench-MAS using reputation_weight 1.

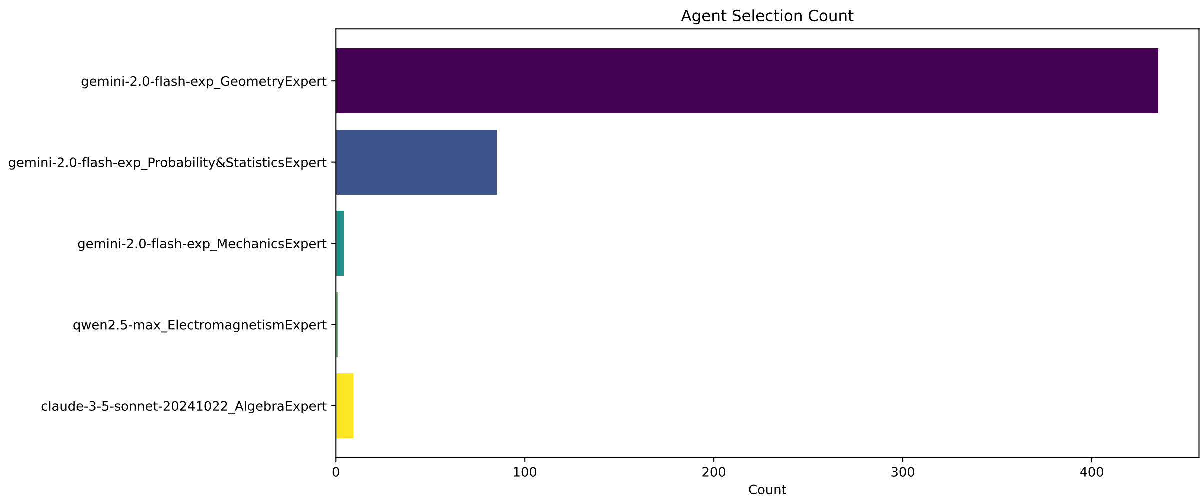


Figure 11: Testing stage on the medium-level tasks in Scibench-MAS using reputation_weight 2.

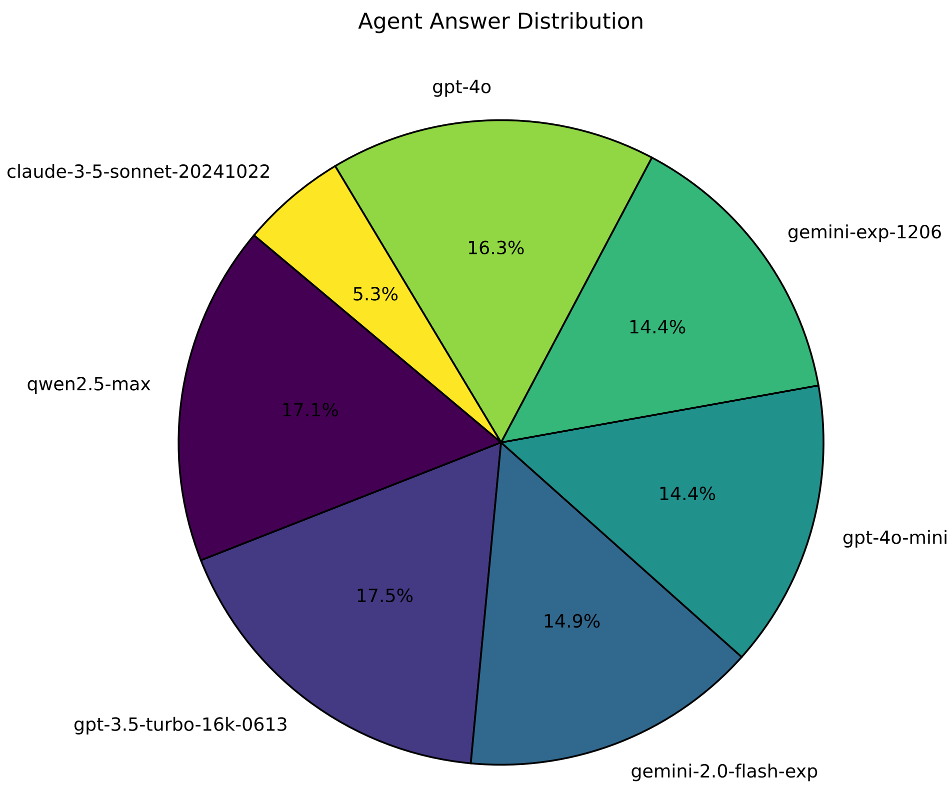


Figure 12: Testing stage on the medium-level tasks in Scibench-MAS without training.

Token Efficiency Table 1 also compares the average number of tokens consumed per task. ReSo maintains a relatively moderate token usage, which is significantly lower than certain baselines like DyLAN (14.6k vs 64.1k, 20.7k vs 77.8k). This balance between performance and computational cost underlines ReSo's practical efficiency in real-world, large-scale scenarios.

G Reward Signal

We investigate the impact of different reward signals on system optimization, considering three approaches: (1) **Rule-based**, which provides strictly accurate, predefined evaluations for sub-task solutions; (2) **General Reward Model**, using Qwen2.5-Math-PRM-7B as a reward function without task-specific fine-tuning; and (3) **Fine-tuned Reward Model**, i.e., our CRM proposed in 3.3.3. Figure 3(c) presents the results of training our MAS under these reward schemes on the SciBench-MAS dataset. The rule-based reward yields the best results, confirming the importance of precise reward signals. Besides, our CRM brings a slight improvement compared to the original Qwen2.5-Math-PRM-7B model. We also observe an instance of *reward hacking* when using the Qwen reward model: specifically, Qwen2.5-Max tends to receive inflated scores when acting as the reasoning agent. As a result, during inference, the MAS disproportionately selects Qwen2.5-Max to handle sub-tasks, even in cases where it does not necessarily produce the best solutions.

H CRM,ORM,PRM

Our Cooperative Reward Model (CRM) is inspired by OpenAI's PRM, but it has been extended and adapted to the multi-agent system (MAS) setting. In our complex tasks, multiple sub-tasks exist, and the CRM scores each sub-task's response based on the outputs from prior agents. While conceptually similar to PRM—where each sub-task can be seen as a step—PRM cannot be directly applied to our MAS setting due to fundamental structural differences.

I Comparison with Chain-of-Thought (CoT) Methods

We would like to clarify that the prompts used in our single-model evaluation experiments already support step-by-step reasoning, thus reflecting Chain-of-Thought (CoT) style outputs. These models are capable of multi-step reasoning and demonstrate CoT-style thinking when tackling complex problems. However, as demonstrated in our results, these CoT-style single-model approaches perform poorly on tasks with high complexity and combinatorial reasoning. As task difficulty increases, even the strongest single LLMs exhibit a significant drop in accuracy—approaching 0% at the highest difficulty level. This clearly indicates that "step-by-step thinking" alone is insufficient for solving the kinds of deep combinatorial reasoning tasks we designed. Our proposed method, ReSo, substantially outperforms these CoT-style baselines. In addition, ReSo introduces structural and functional advantages over traditional CoT methods. CoT follows a linear reasoning path, whereas ReSo constructs a task graph composed of multiple subtasks, each solvable independently by different expert agents. This allows for horizontal task expansion and fine-grained skill decomposition. A key limitation of CoT is its dependence on a single model's context length, reasoning capabilities, and domain knowledge. ReSo addresses these limitations by decomposing tasks, dynamically routing them, assigning subtasks to the most appropriate agents, and using reward mechanisms to drive learning.

J Qwen Model Dependence

We would like to clarify that the performance gains observed in ReSo primarily stem from the task decomposition and multi-agent cooperation architecture, rather than solely from a stronger base model. Our approach consists of two stages. The first stage uses an LLM to decompose the task, and the second stage selects the most suitable agents to handle the subproblems. To further demonstrate the effectiveness of our framework, we conducted a new experiment. Even when Qwen-sfted is used for task decomposition, single-agent approaches still fail. This emphasizes that cooperation among agents is necessary. Additionally, our fine-tuned Qwen-7B model performs comparably to GPT-4o for task decomposition, but it is only when subtasks are assigned to specialized agents that the system achieves significant improvements in performance.

Table 3: Qwen model dependence

model	Easy	Medium	Hard
Qwen-sfted + (no ReSo) single agent	27.5	5.6	4.5
GPT-4o + ReSo	71.4	43.8	34.8
Qwen-sfted + ReSo	79.1	56.2	33.7

K Computational Complexity and Runtime

Inference Parallelism. Independent DAG subnodes can be executed in parallel, mitigating runtime overhead. Despite a higher token usage, ReSo achieves greater accuracy gains, justifying the cost:

Table 4: Token usage and runtime comparison

Method	Tokens	Time (h)
MetaGPT	16.1 k	3.2
DyLAN	64.1 k	8.0
GPTSwarm	14.9 k	1.3
GDesigner	16.9 k	4.0
ReSo	25.9 k	4.1 (3 training + 1.1 testing)