SIRIUS: SELF-IMPROVING MULTI-AGENT SYSTEMS VIA BOOTSTRAPPED REASONING

Wanjia Zhao, Shirley Wu, Mert Yuksekgonul, James Zou
Department of Computer Science
Stanford University
Stanford, CA 94305, USA
{wanjiazh, merty, shirwu, jamesz}@cs.stanford.edu

ABSTRACT

Multi-agent AI systems powered by large language models (LLMs) are increasingly applied to solve complex tasks. However, these systems often rely on fragile, manually designed prompts and heuristics, making optimization difficult. A key challenge in optimizing multi-agent systems is acquiring suitable training data for specialized agents. We introduce SIRIUS, a self-improving, reasoning-driven optimization framework for multi-agent systems. Central to our approach is the construction of an experience library: a repository of high-quality reasoning trajectories. The library is built by retaining reasoning steps that lead to successful outcomes, providing a robust training set for optimizing multi-agent system. Additionally, we introduce a library augmentation procedure that refines unsuccessful trajectories, further enriching the library. SIRIUS boosts performance by 2.86% to 21.88% on reasoning and biomedical QA and enhances agent negotiation in competitive settings. Our results show that SIRIUS enhances multi-agent performance while generating reusable data for self-correction and self-play enhancement in the future.

1 INTRODUCTION

Multi-agent AI systems powered by large language models (LLMs), where specialized agents collaborate to solve complex tasks, are becoming increasingly successful in real-world applications. Recent work has demonstrated their effectiveness in complex reasoning (Wang et al., 2024; Smit et al., 2024), coding (Wu et al., 2023), drug discovery (Swanson et al., 2024) and ensuring safety via debate (Chern et al., 2024; Irving et al., 2018). These successes arise from specialized agents integrating their distinct capabilities through structured interactions, enabling more effective problem-solving than single agents. Moreover, multi-agent scrutiny acts as a built-in self-correction mechanism, where agents refine and verify each other's outputs. This often outperforms single agent setting, particularly on tasks demanding rigorous reasoning or factual validation. Despite these successes, optimizing multi-agent systems remains a fundamental challenge due to (1) the difficulty of acquiring appropriate training signals for each agent and (2) the sensitivity to multiple moving parts that influence overall performance (Smit et al., 2024). While task-level reward feedback is available, credit assignment across agents remains ambiguous—it is unclear how to attribute success or failure to specific intermediate decisions and reasoning steps made by each LLM agent. This challenge parallels the multi-agent credit assignment problem in reinforcement learning (Foerster et al., 2018). However, in language-based systems, reasoning unfolds through complex and unstructured interactions, making attribution far more difficult than in traditional RL settings with well-defined action spaces.

We present SIRIUS, a framework for learning effective multi-agent behaviors from outcome rewards. Our key insight is that when multiple agents successfully solve a task together, their entire interaction trajectory likely contains useful patterns - even if we cannot pinpoint exactly which steps or decisions were crucial for success. Drawing inspiration from recent advances in bootstrapping reasoning capabilities (Zelikman et al., 2022), we collect and learn from successful agent interactions across many tasks, allowing the system to iteratively discover effective collaboration strategies from self-generated data. This approach sidesteps the need for direct supervision of intermediate steps, instead

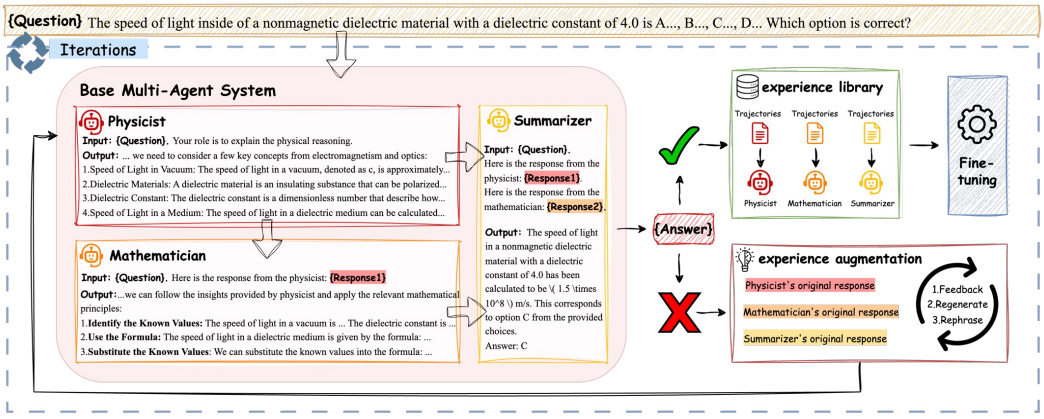


Figure 1: General training pipeline of SIRIUS.Agents solve problems sequentially, storing correct responses for fine-tuning and augmenting incorrect ones through feedback, regeneration, and rephrasing. This iterative process improves performance via reward-based evaluation and supervised fine-tuning. The module colors in the figure correspond to those in Algorithm 1.

letting agents learn which interaction patterns tend to lead to successful outcomes. For trajectories that result in failed attempts, we perform trajectory augmentation by resampling original attempts with feedback from an additional agent grounded in the ground truth.

Our experiments demonstrate that SIRIUS significantly enhances multi-agent performance across multiple domains. It improves reasoning and biomedical QA accuracy by 2.86% to 21.88%, while also strengthening agent negotiation in competitive scenarios. Beyond these gains, our approach offers a scalable mechanism for self-improvement, enabling agents to iteratively refine their reasoning and collaboration strategies. More broadly, SIRIUS provides a general framework for optimizing multi-agent systems via self-generated synthetic data, offering a principled way to enhance performance without requiring fine-grained human supervision.

2 METHOD

2.1 MULTI-AGENT SYSTEMS WITH LLMS

We define a multi-agent system by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{N}, \mathcal{G} \rangle$. Here, $\mathcal{N} \triangleq \{A^{(1)}, A^{(2)}, \ldots, A^{(N)}\}$ is the set of N agents, each agent $A^{(i)}$ uses a policy π_i parameterized by $\theta^{(i)}$. $s \in \mathcal{S}$ is the state of the environment, $\mathbf{a} \in \mathcal{A}$ is the joint actions, and $\mathcal{A}$ is the joint action space. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function where $\mathcal{T}(s, \mathbf{a})$ yields the next state of the environment given the current state and joint actions $\mathbf{a}$. The environment feedback is modeled via a payoff function $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^N$, which provides rewards for each agent k based on the state-action pairs.

The communication structure between agents is modeled as a directed graph $\mathcal{G} = (\mathcal{V}, E, \mathcal{P})$, where $\mathcal{V}$ represents agents, and E defines interaction order.

For each edge $(i, j) \in E$, agent $A^{(j)}$ receives an input derived from the state-action pair $(s, \mathbf{a})$ and the output of agent $A^{(i)}$. This input determines agent $A^{(j)}$'s subsequent action. For each agent $A^{(i)}$ in a topological graph $\mathcal{G}$, its predecessors are the set of agents that influence its output: $\text{Pre}(A^{(i)}) = \{A^{(j)} \mid (A^{(j)}, A^{(i)}) \in \mathcal{G}\}$. Here, $(A^{(j)}, A^{(i)})$ denotes a directed edge in the graph, indicating that the output of agent $A^{(j)}$ directly influences the input of agent $A^{(i)}$.

Throughout this paper, the collection of our agents will be based on language models and the primary environment that we use will be natural language. In particular:

$$
\begin{aligned}
&a_i \sim \pi_i(\cdot | s_t, \{a_j\}_{A^{(j)} \in \text{Pre}(A^{(i)})}) \quad \forall A^{(i)} \in \mathcal{N} \\
&\mathbf{a}_t = (a_1, ..., a_N) \\
&s_{t+1} = \mathcal{T}(s_t, \mathbf{a}_t) = \text{Concat}(s_t, \mathbf{a}_t)
\end{aligned} \tag{1}
$$

where π_i denotes the probability distribution of the i-th language model, Concat is the concatenation of the previous state and the responses, and we will use $\pi = \{\pi_1, \ldots, \pi_N\}$ to denote the joint policy. Generally, each agent aims to maximize its own reward: $\max_{\pi_i} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} R_i(s_t, \mathbf{a}_t)\right]$, where R_i denotes the i-th component of the reward vector $\mathcal{R}$ and the expectation is taken under the joint policy π.

2.2 SIRIUS

The training pipeline of the proposed framework, denoted as SIRIUS, is illustrated in Figure 1. SIRIUS adopts a fine-tuning strategy to iteratively improve the policy parameters $\theta^{(n)}$ of each agent $A^{(n)}$ over T iterations. The process is initialized with a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{D}$, where each pair (x_i, y_i) represents a problem and its solution. The core training procedure is outlined in Algorithm 1.

Algorithm 1 SIRIUS

Input: A group of agents $A^{(1)}, \cdots, A^{(N)}$ An initial dataset of problems x with answer y : $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{D}$, total number of fine-tuning Iterations T.
Initialize: Initialize policy parameters $\theta^{(n)}$ for each agent $A^{(n)}$, $k = 1, 2, \ldots, N$.
for Fine-tuning Iteration t $= 1, \cdots, T$ **do**
  $a_i^{(n)} = \mathcal{P}_{\theta_t^{(n)}}(\cdot|x_i, \mathbf{a}_i^{\mathrm{Pre}(A^{(n)})}), k = 1, 2, \ldots, K.$
  for each agent $k = 1, 2, \ldots, K$ **do**
    $\mathcal{C}_t^{(n)} \leftarrow \{(x_i, a_i^{(n)} | i \in [1, D] \wedge R_i(s, a) > \epsilon)\}$
    Good Trajectory Set of Each Agent.
    Augmentation$(\{(x_i, a_i^{(n)} \wedge R_i(s, a) < \epsilon)\})$
  end for
  $\theta_t^{(n)} \leftarrow$ **Standard SFT on** $\mathcal{C}_t^{(n)}, n = 1, \cdots, N$
end for

At each fine-tuning iteration t:

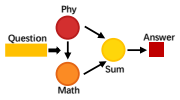
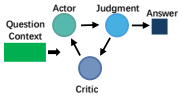
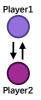
- **Action Sampling:** For each agent $A^{(n)}$, an action $a_i^{(n)}$ is sampled from its policy,
$$a_i^{(n)} = \mathcal{P}_{\theta^{(n)}}(\cdot|x_i, \mathbf{a}_i^{\mathrm{Pre}(A^{(n)})}),$$
conditioned on the input problem x_i and the action set $\mathbf{a}_i^{\mathrm{Pre}(A^{(n)})}$ generated by previous agents. In scenarios involving multiple interaction rounds, such as the Competitive Setting, $\mathbf{a}_i^{\mathrm{Pre}(A^{(n)})}$ includes outputs from all agents in all preceding rounds.
- **Trajectory Evaluation and Augmentation**: The trajectories generated by each agent are evaluated using the payoff function $R(s, \mathbf{a})$. Based on a reward threshold ϵ, high-reward trajectories ($R(s, \mathbf{a}) > \epsilon$) are added to the good trajectory set $\mathcal{C}_t^{(n)}$. Since the tasks are challenging, the good trajectory set tends to be small. To leverage more data for fine-tuning, we propose trajectory augmentation pipeline for each task, detailed in the Appendix A. Specifically, we first generate feedback to refine the agent's original response. The feedback and original response are then combined to prompt the agent to regenerate a new solution, which is then rephrased into a direct problem-solving step. Afterward, we return to the action sampling process to produce the final answer and evaluate it.
- **Fine-Tuning:** The policy parameters $\theta^{(n)}$ are updated via supervised fine-tuning on $\mathcal{C}_t^{(n)}$.

This iterative process ensures that each agent's policy is progressively refined to maximize performance based on the joint system dynamics and reward.

3 MULTI-AGENT SETTINGS

In this section, we explore several settings where agents with distinct expertise interact to solve challenging tasks. As shown in Table 1, we systematically analyze different agent configurations.

Table 1: Different settings and tasks. In the rows corresponding to Communication Structure, nodes denote agents ($\mathcal{V}$), arrows represent edges (E), and color indicates the role of agents.

Settings	Problem-Solving		Actor-Critic	Competitive
Structure $(\mathcal{V}, E, \mathcal{P})$				
Tasks	College-Physics College-Chemistry	PubMedQA	PubMedQA	Resource Exchange Seller-Buyer Ultimatum
Reward for each agent R_i	Final Output Correctness		Final Output Correctness	Utility Function Value

3.1 Problem Solving Settings

Agents with Specific Expertise. In this setting, each agent is assigned a domain-specific role to facilitate a structured and efficient problem-solving process. For instance, in the physics and chemistry domains, the problem-solving pipeline begins with a domain expert (e.g., a physicist or chemist) who analyzes the domain-specific problem, followed by a mathematician who formalizes the reasoning with quantitative models, and finally, a summarizer who consolidates the insights into a clear and comprehensive answer. This sequential collaboration ensures that the expertise of each agent is leveraged effectively while maintaining clarity in the solution process.

The sequential dependency between the agents can be described as follows:

$$a_{\text{Phy}} \sim \pi_{\text{Phy}}(\cdot|q), \tag{2}$$

$$a_{\text{Math}} \sim \pi_{\text{Math}}(\cdot|q, a_{\text{Phy}}), \tag{3}$$

$$a_{\text{Sum}} \sim \pi_{\text{Sum}}(\cdot|q, a_{\text{Phy}}, a_{\text{Math}}), \tag{4}$$

where q is the input question, a_{Phy} is the response generated by the Physicist, a_{Math} is the response generated by the Mathematician based on both the question and the Physicist's response,a_{Sum} is the final answer synthesized by the Summarizer using the question, the Physicist's response, and the Mathematician's response.

Analyze Long Context and Answer Question. In scenarios involving lengthy and complex contexts, we consider a common two-agent setup: the Context Analyst and the Problem Solver. The Context Analyst's responsibility is to thoroughly examine the context, extract essential information, and provide a concise and accurate summary. The Problem Solver then uses this summary to analyze the question and formulate the final answer. This division of labor not only improves interpretability, but also reduces the cognitive load on each agent.

3.2 Actor-Critic Setting

The popular Actor-Critic framework facilitates iterative agent improvement through a feedback loop: the Actor Agent generates solutions while the critic evaluates and refines them, enhancing both the Actor Agent's reasoning and the Critic Agent's error correction capabilities. In practice, we separate judgment and feedback tasks by introducing a Judgment Agent alongside the Critic Agent, where the Judgment Agent classifies the Actor Agent's solutions as correct or incorrect, and for incorrect solutions, the critic provides feedback to guide the Actor Agent in regenerating improved solutions. Reward mechanisms are designed as: the Actor Agent receives rewards for correct solutions, the Judgment Agent for accurate classifications, and the critic for providing actionable feedback that leads to correct regenerations.

3.3 Competitive Settings

Competitive scenarios (Bianchi et al., 2024) examine multi-agent interactions under opposing objectives, where agents must balance cooperation and competition to achieve their goals. In this framework, two agent roles are defined: **Player 1** and **Player 2**. Each player is initialized with a specific amount of resources, which evolve over the course of the game based on their interactions. The game progresses as a sequence of moves, resulting in a trajectory of states:

$$\begin{aligned} &\text{Player 1 Trajectory: } x_0^{\text{player1}}, x_1^{\text{player1}}, \cdots, x_T^{\text{player1}} \\ &\text{Player 2 Trajectory: } x_0^{\text{player2}}, x_1^{\text{player2}}, \cdots, x_T^{\text{player2}} \end{aligned} \tag{5}$$

Table 2: Tasks and setups in the competitive setting. Each task involves two agents with distinct roles, initial resources, and objectives. *Resource Exchange* focuses on maximizing total resources through trade. Ultimatum requires negotiating a split of $100. *Sell&Buy* involves price negotiation for an item. Each task follows a turn-based structure with a fixed maximum number of rounds and ends when an agreement is reached.

Task	**Resource Exchange**		**Ultimatum**		**Sell&Buy**	
Roles	**Player 1**	**Player 2**	**Player 1**	**Player 2**	**Seller**	**Buyer**
Initial resources	25Xs, 5Ys	5Xs, 25Ys	$ 100	0	1X	100 ZUPs
Goal	Maximize total resources		Negotiate a split		Maximize price	Minimize price
Utility	Xs + Ys	Xs + Ys	Split amount-50	Split amount-50	Selling price - 50	50-Selling price
Ending condition	When either player accepts		When either player accepts		When either player accepts	
Max. # of turns	8 rounds of interaction		8 rounds of interaction		10 rounds of interaction	

The sequence captures the evolution of game states as players compete at each timestep $t = 0, 1, \ldots, T$, ultimately determining a winner and a loser. Our goal is to optimize each player's policy to maximize its own expected reward based on trajectory data and role-specific context. This can be formulated as:

$$\max \sum_{i=1}^{T} P_\theta(x_i^{\text{player1}} | x_{0:i-1}^{\text{player1}}, x_{0:i-1}^{\text{player2}}) \tag{6}$$

where Player 1 optimizes its policy based on the historical trajectory of both itself and Player 2, and similarly for Player 2. We explore three distinct competitive settings, all of which unfold over multiple rounds:

Resource Exchange Scenario. In this scenario, agents engage in a simulated environment where they exchange resources to maximize their individual utility.

Seller and Buyer Scenario. This setting models economic interactions where one agent assumes the role of a seller and another the role of a buyer. The agents negotiate prices and terms to complete transactions, testing their ability to strategize under asymmetric setting.

Multi-Turn Ultimatum Game. The Multi-Turn Ultimatum Game explores scenarios of fairness, cooperation, and negotiation over multiple rounds. One agent proposes a division of a resource, and the other agent decides whether to accept or reject it.

4 Experiments

4.1 Baseline

We compare our SiriuS against the following baselines:

Single-Agent utilizes a single language model to process input and generate responses.

STaR (Zelikman et al., 2022), the Self-Taught Reasoner, focuses on enhancing the reasoning capabilities of a single agent by iteratively training it to improve its step-by-step reasoning through self-supervised fine-tuning.

Prompt Multi-Agent System (CoMM) (Chen et al., 2024a) introduces a training-free, multi-agent collaborative framework where agents interact and share information to solve tasks collectively.

TextGrad (Yuksekgonul et al., 2024) optimizes prompts for each agent in a multi-agent system by backpropagating natural language feedback through each interaction.

4.2 Setup and Datasets

Backbone Model. For a fair comparison, we use gpt-3.5-turbo-0125 and gpt-4o-mini-2024-07-18 as the backbone model, and set the temperature to 0 in all our experiments. We use OpenAI's Fine-tuning API for supervised fine-tuning.

College Physics/Chemistry. These two datasets are constructed by combining questions from Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2020), Graduate-Level Google-Proof Q&A (GPQA) (Rein et al., 2023), and Theorem-Driven Question Answering (TheoremQA) (Chen et al., 2023). It focuses on college-level physics problems, which remain difficult

Table 3: Evaluation results of the proposed method and baselines on accuracy(%). Best results are in **bold** numbers and second-best results are in underline numbers.

Model	Method	College Physics	College Chemistry	PubMedQA
GPT-3.5-turbo	**Single-Agent**	24.30	38.46	56.40
	STaR	29.91	47.69	63.80
	COMM	30.84	50.77	71.80
	TextGrad	32.71	41.54	NA
	SIRIUS	**33.64**	**56.92**	**74.20**
GPT-4o-mini	**Single-Agent**	39.25	41.54	67.40
	STaR	42.06	47.69	69.20
	COMM	42.06	49.23	70.60
	TextGrad	42.99	44.62	68.20
	SIRIUS	**46.73**	**60.00**	**73.40**

and demonstrate room for improvement in performance with large language models. We split the dataset into training and test sets, with the detailed data distribution provided in Appendix B.

PubMedQA. This is a biomedical question-answering dataset comprising 1000 open-domain questions (Jin et al., 2019), each paired with context from PubMed abstracts and corresponding answers. It focuses on research-driven queries, requiring domain-specific understanding and reasoning over scientific texts. We follow the original split of the dataset for training (500) and testing (500) sets.

4.3 Experimental Result of Problem Solving Setting

4.3.1 Main Result

Table 3 presents a performance comparison of various models and methods under the Problem Solving Setting. We observe that the prompted Multi-Agent System (COMM) generally improves performance, as agent collaboration enhances the ability to solve complex problems. STaR outperforms the base Single-Agent, indicating that fine-tuning contributes to improved performance. For smaller and weaker models, and in scenarios with long context lengths such as PubMedQA, TextGrad faces significant challenges in instruction-following during optimization. TextGrad (GPT-3.5-turbo) could not be applied to PubMedQA as its optimizer failed to parse instructions due to the model's limited capability and the excessive context length of the problem. Similarly, TextGrad (GPT-4o-mini) struggles to generate answers in the required format, requiring manual extraction of answers. Our proposed method, SIRIUS, consistently outperforms across all tasks. By decomposing tasks into manageable sub-tasks assigned to agents and, crucially, fine-tuning each agent to specialize in its designated task, SIRIUS maximizes the effectiveness of collaboration, ensuring a more coordinated and efficient overall performance.

4.3.2 Ablation Experiments

To evaluate the contributions of various components in SIRIUS, we conducted a series of ablation experiments. Each experiment was designed to answer a key question about the effectiveness of the multi-agent system. All ablations were performed on representative tasks within the Problem Solving Setting (PubMedQA) to ensure consistency in evaluation as shown in Table 4.

Does mixing SIRIUS with a base agent degrade performance? To understand the benefits of a jointly optimizing a collaborative multi-agent system, we first train all the agents together using SIRIUS. Then we replaced one SIRIUS agent with the original base agent—either SIRIUS Analyst + base Solver or base Analyst + SIRIUS Solver. This substitution hurts performance, demonstrating benefits from joint multi-agent optimization compared to optimizing a single agent.

Should we fine-tune different LLMs for different roles, or optimize one LLM for all roles? We explored whether a single LLM fine-tuned on the combined training data of multiple roles could match the performance of separate role-specific models. The results showed a notable performance decline, highlighting that different roles require specialized adaptation and that a shared model struggles to effectively generalize across distinct agent functions.

How useful is experience augmentation? To assess the impact of experience augmentation, we removed the augmentation module while keeping the rest of the pipeline unchanged. Data augmentation introduces more diverse and challenging experiences as training data, enhancing the model's capability; therefore, omitting the augmentation module could negatively impact performance.

Table 4: Ablation results on PubMedQA.

Model	method	PubMed
GPT-3.5-turbo	SIRIUS	74.20
	SIRIUS + Base	72.00
	Base + SIRIUS	73.20
	FT on One Base LLM	70.40
	SIRIUS w/o Aug.	73.40
	Additional FT Itr	75.00
GPT-4o-mini	SIRIUS	73.40
	SIRIUS + Base	72.80
	Base + SIRIUS	71.60
	FT on One Base LLM	72.00
	SIRIUS w/o Aug.	72.20
	Additional FT Itr	73.60

Does additional fine-tuning improve performance? We investigated whether increasing the number of fine-tuning iterations leads to further performance gains. Each iteration follows the full optimization pipeline illustrated in Figure 1, the previously fine-tuned SIRIUS is used to generate a new experience library, which is then used to further fine-tune the base model. As expected, an additional iteration yielded marginal performance gains, suggesting that the model can benefit from extended training.

4.4 EXPERIMENTAL RESULT OF ACTOR-CRITIC SETTING

Table 5 presents a performance comparison of various models, methods, and ablations under the Actor-Critic Setting on PubMedQA. As mentioned in Section 3.2, the Actor Agent first generates a solution, which is then evaluated by the Judgment Agent to determine its correctness. For solutions deemed incorrect by the Judgment Agent, the Critic Agent analyzes the original solution and provides feedback without access to the correct answer. The Actor Agent then regenerates the solution based on this feedback.

A key challenge in this setting is the Judgment Agent's limited ability to differentiate between correct and incorrect solutions leading to two potential issues: (1) correct solutions may be mistakenly judged as incorrect and potentially modified into incorrect ones during the feedback and regeneration stages; (2) incorrect solutions may be judged as correct, failing to receive the necessary corrections. We report TP (True Positive) Accuracy as the ratio of solutions both correctly generated by the Actor and accurately validated by the Judgment Agent, while Overall Accuracy measures the total correct solutions after regeneration, accounting for the combined contributions of all agents.

We evaluate our method against two representative baselines: (1) Self-Correct, where Actor-generated solutions are refined through direct feedback-guided regeneration, and (2) Prompt, which exclusively employs prompting strategies to coordinate Actor-Judgment-Critic interactions without optimization mechanisms. A critical limitation observed in the Self-Correct framework is its significantly lower TP accuracy. This issue arises from its feedback mechanism, which modifies all generated responses with high probability, potentially leading to erroneous modifications of the initially correct solution. This is a common issue with using out-of-the-box LLMs for self-correction with no specialized training (Kumar et al., 2024).

Table 5: Evaluation results of the proposed method and baselines on accuracy(%).

Model	GPT-3.5-Turbo		GPT-4o-mini	
Method	TP Accuracy	Overall Accuracy	TP Accuracy	Overall Accuracy
Self-Correct	11.80	16.40	24.60	28.80
Prompt	18.40	47.60	51.60	58.20
SIRIUS	**35.00**	**50.60**	**59.80**	**66.80**
Ablation Study				
SIRIUS + BASE Actor Agent	34.20	49.00	49.60	54.40
SIRIUS + BASE Judgment Agent	20.20	40.20	53.00	59.40
SIRIUS + BASE Critic Agent	35.00	50.40	59.80	64.20
FT on One Base LLM	33.80	43.60	56.00	59.60

Comparing GPT-3.5-Turbo and GPT-4o-mini, we also find that GPT-3.5-Turbo struggles more with misjudging correct answers as incorrect, leading to a severe drop in TP Accuracy. Our method, SIRIUS, achieves a notable improvement in TP Accuracy, highlighting the Judgment Agent's enhanced ability to assess whether a response requires modification. The overall higher accuracy underscores the effectiveness of SIRIUS's framework, where fine-tuning enhances each agent's task-specific capabilities, and the collaboration of Judgment, Critic, and Actor Agents ensures appropriate revision of incorrect responses while minimizing unnecessary changes to correct answers.

The ablation study further underscores the contribution of each agent in SIRIUS. Fine-tuning only a single base LLM leads to a performance drop, highlighting the necessity of specialized agent roles and joint optimization. Notably, replacing the Judgment Agent with a baseline version significantly reduces TP Accuracy, reinforcing its essential role in filtering correct responses before feedback is applied.

4.5 Experimental Result of Competitive Settings

To analyze the effect of training in the competitive setting, we study the performance of agents in scenarios where one player initially had a higher probability of winning, referred to as the "winning player," while the other player was at a disadvantage, called the "losing player." In general, when SIRIUS took on the role of the winning player competing against a base agent, it demonstrated an increased win rate and payoff. Additionally, when SIRIUS played the role of the losing player, it experienced fewer losses. Similarly, for both GPT-3.5 and GPT-4o-mini when they compete with each other, SIRIUS-GPT-3.5 and SIRIUS-GPT-4o-mini both demonstrate improved performance.

4.5.1 Resource Exchange

The win rates and average payoffs for the Resource Exchange game are presented in Figure 2. Overall, the agent going second tends to beat the first agent. Furthermore, the fine-tuned SIRIUS demonstrates a significant improvement in both the win rate and payoff for the current player. To evaluate the generalization capability of our approach, we conducted additional experiments with models fine-tuned on games featuring Initial Resource configurations of 25Xs + 5Ys and 5Xs + 25Ys, and then tested them on games with different Initial Resource configurations (35Xs + 15Ys and 15Xs + 35Ys). As demonstrated in Figure 5, SIRIUS maintains notable improvements in the new Initial Resource configurations, effectively validating the generalizability of our proposed pipeline.

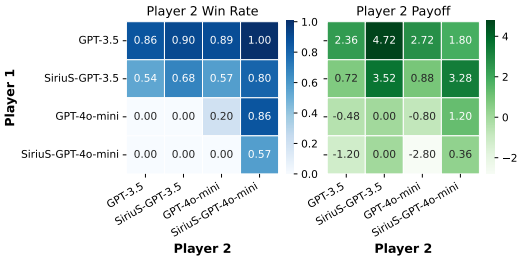


Figure 2: Resource Exchange Game: Player 1 (25Xs + 5Ys), Player 2 (5Xs + 25Ys). Win Rate in decisive games and Payoff in all games. We show Player 2 Win rate/payoff in all cells.

4.5.2 Multi-Turn Ultimatum

In this setting, Player 1 consistently dominates the game. As shown in the Figure 3 , SIRIUS fine-tuned Player 1 effectively secure a higher share of the split. Generalization experiments show that SIRIUS Player 1 trained in the Resource = 100 setting maintains utility gains in the new Resource = 1000 setting (Figure 7).

4.5.3 Buyer-Seller

In this setting, sellers are willing to sell when the price exceeds 40, while buyers are willing to buy when the price is below 60. We plot the final selling price as shown in Figure 4. Notably, it is consistently below 50 for most buyer-seller pairs, indicating that the LLM agent performs better as a buyer than as a seller. After fine-tuning, SIRIUS as a seller shows significant improvement, consistently selling at 50, resulting in a tie with the buyer. To test the generalization capability and ensure the seller is not overfitting to a price of 50, we adjusted the initial configuration to 30 and 70. Figure 6 shows that the SIRIUS seller trained in the previous setup still demonstrates significant improvement.

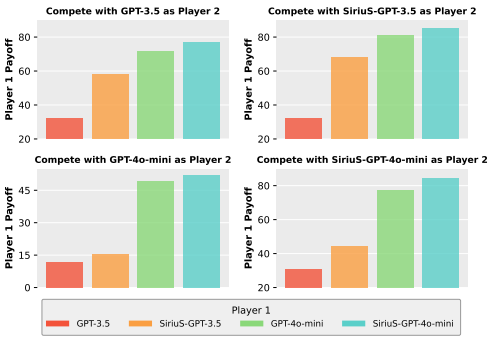


Figure 3: Player 1's payoff in the Ultimatum game with an initial resource of 100. SIRIUS as Player 1 secures a higher share.

Figure 4: Final Selling Price for a Seller&Buyer with object valuations of 40 and 60. A higher number means a greater payoff for Seller.

5 RELATED WORK

Enhancing Reasoning in Single-Agent Systems. Building on the reasoning capabilities of state-of-the-art foundation models (Schulman et al., 2022; OpenAI, 2023; Liu et al., 2024), recent research explores approaches beyond scaling model parameters. Chain-of-Thought (Wei et al., 2022) enhances reasoning through step-by-step inference, while Tree of Thoughts (Yao et al., 2024), Graph of Thought (Besta et al., 2024), and Program of Thoughts (Chen et al., 2022) structure reasoning as tree searches with backtracking. Reasoning with Planning (RAP) (Hao et al., 2023) incorporates explicit planning, and Reflexion (Shinn et al., 2024) enables self-evaluation and refinement. Wu et al. (2024) introduce contrastive reasoning for instruction generation, while TextGrad (Yuksekgonul et al., 2024) applies gradient-based optimization to refine outputs. These methods enhance reasoning through structured decomposition, search, and planning.

Self-improvement. Self-improving models (Huang et al., 2022; Yu et al., 2023; Yuan et al., 2024; Zhang et al., 2024; Welleck et al., 2022) have garnered increasing attention for their potential to enhance reasoning capabilities through iterative feedback and refinement. Several studies (Zelikman et al., 2022; Li et al., 2024a; Pang et al., 2024; Lee et al., 2024)employ bootstrapping strategies by leveraging self-generated rationales, while others (Yuan et al., 2024; Chen et al., 2024b; Guo et al., 2025) introduces a self-refinement mechanism through reinforcement learning.

Multi-Agent Systems with LLMs. **Multi-Agent Systems with LLMs.** Recent advancements in multi-agent systems (Smit et al., 2024; Guo et al., 2024; Li et al., 2024b; Han et al., 2024) highlight the potential of large language models in tackling complex tasks. Society of Minds (Du et al., 2023) enables agents to exchange answers, fostering collaboration. Mixture-of-Agents (Wang et al., 2024) employs a layered architecture where agents refine responses based on prior outputs. CoMM (Chen et al., 2024a) enhances problem-solving through structured communication and role division. Multi-Persona (Liang et al., 2023) encourages diverse agent behaviors by assigning distinct personas. ChatEval (Chan et al., 2023) explores different multi-agent debate strategies for interaction and response management. Building on advances in multi-agent systems, recent work has explored fine-tuning with independently specialized agents that interact to generate diverse reasoning chains (Subramaniam et al., 2025). Unlike these approaches, our method prioritizes collaborative optimization through a shared experience library, enabling agents to collectively learn from and refine successful reasoning trajectories.

6 CONCLUSIONS

We introduced SIRIUS, a framework for optimizing multi-agent LLM systems by learning from successful interactions and augmenting failed trajectories with feedback. Our approach enables agents to refine collaboration strategies without explicit supervision. Experiments show that SIRIUS significantly improves performance across college-level reasoning, biomedical QA, and negotiation tasks. More broadly, our work provides a scalable mechanism for multi-agent self-improvement, offering a principled approach to optimizing collaborative AI systems.

REFERENCES

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Federico Bianchi, Patrick John Chia, Mert Yuksekgonul, Jacopo Tagliabue, Dan Jurafsky, and James Zou. How well can llms negotiate? negotiationarena platform and analysis. *arXiv preprint arXiv:2402.05863*, 2024.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*, 2023.

Pei Chen, Boran Han, and Shuai Zhang. Comm: Collaborative multi-agent, multi-reasoning-path prompting for complex problem solving. *arXiv preprint arXiv:2404.17729*, 2024a.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7889–7901, 2023.

Zhipeng Chen, Kun Zhou, Wayne Xin Zhao, Junchen Wan, Fuzheng Zhang, Di Zhang, and Ji-Rong Wen. Improving large language models via fine-grained reinforcement learning with minimum editing constraint. *arXiv preprint arXiv:2401.06081*, 2024b.

Steffi Chern, Zhen Fan, and Andy Liu. Combating adversarial attacks with multi-agent debate. *arXiv preprint arXiv:2401.05998*, 2024.

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

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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

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

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

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

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

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

Geoffrey Irving, Paul Christiano, and Dario Amodei. Ai safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. *arXiv preprint arXiv:1909.06146*, 2019.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. Llm2llm: Boosting llms with novel iterative data enhancement. *arXiv preprint arXiv:2403.15042*, 2024.

Siheng Li, Cheng Yang, Zesen Cheng, Lemao Liu, Mo Yu, Yujiu Yang, and Wai Lam. Large language models can self-improve in long-context reasoning. *arXiv preprint arXiv:2411.08147*, 2024a.

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

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023.

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

R OpenAI. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5), 2023.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*, 2024.

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

John Schulman, Barret Zoph, Christina Kim, Jacob Hilton, Jacob Menick, Jiayi Weng, Juan Felipe Ceron Uribe, Liam Fedus, Luke Metz, Michael Pokorny, et al. Chatgpt: Optimizing language models for dialogue. *OpenAI blog*, 2(4), 2022.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Andries Petrus Smit, Nathan Grinsztajn, Paul Duckworth, Thomas D Barrett, and Arnu Pretorius. Should we be going mad? a look at multi-agent debate strategies for llms. In *Forty-first International Conference on Machine Learning*, 2024.

Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*, 2025.

Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab: Ai agents design new sars-cov-2 nanobodies with experimental validation. *bioRxiv*, pp. 2024–11, 2024.

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

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

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. *arXiv preprint arXiv:2211.00053*, 2022.

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

Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis N. Ioannidis, Karthik Subbian, Jure Leskove, and James Zou. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *NeurIPS*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Xiao Yu, Baolin Peng, Michel Galley, Jianfeng Gao, and Zhou Yu. Teaching language models to self-improve through interactive demonstrations. *arXiv preprint arXiv:2310.13522*, 2023.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. Small language models need strong verifiers to self-correct reasoning. *arXiv preprint arXiv:2404.17140*, 2024.

A Detailed Pipeline

Given the wrong answer problem set $\mathcal{W} = \{(x_i, y_i)\}_{i=1}^{w}$,In each iteration, we first select the agent to be optimized. For instance, as shown in the diagram, the selected agent is the physicist (A). The external agent provides feedback $f_i = P_{\theta^{(\text{ext})}}(\cdot|x_i, \hat{a}_i, y_i)$ based on the question x_i, the original response $\hat{a}_i$, and the correct answer y_i.

The physicist then regenerates the solution by incorporating the feedback: $\hat{a}_i^r = P_{\theta^{(A)}}(\cdot|x_i, \hat{y}_i, f_i)$.

To ensure clarity and coherence, the regenerated response $\hat{a}_i^r$ is subsequently rephrased to produce $\hat{y}_i^{\text{final}}$, making it appear as if derived directly through problem-solving without mentioning any modifications or feedback. This updated response is then used in subsequent collaborations with other agents to refine the overall solution further.

Algorithm 2 Training Multi-Agent LLM System

Input: A group of agents $A^{(1)}, \cdots, A^{(K)}$, the system's topological graph $\mathcal{G}$, maximum solution generation tries $\max_{\text{sol}}$, maximum feedback generation tries $\max_{\text{f}}$, maximum regeneration tries $\max_{\text{re}}$. An initial dataset of problems x with answer y : $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{D}$, total number of fine-tuning Iterations T.
Initialize: Initialize policy parameters $\theta^{(k)}$ for each agent $A^{(k)}$, $k = 1, 2, \ldots, K$. $\theta^{(c)}$ for Critic Agent $A^{(c)}$
for Fine-tuning Iteration $\text{t}_{\text{ft}} = 1, \cdots, T$ **do**
  while $\text{t}_{\text{sol}} \leq \max_{\text{sol}}$ **do**
    $a_i^{(k)} = \mathcal{P}_{\theta^{(k)}}(\cdot|x_i, \mathbf{a}_i^{\text{Pre}(A^{(k)})})$.
    $\hat{y}_i = a_i^{(K)}$
    for each agent $k = 1, 2, \ldots, K$ **do**
      $\mathcal{C}_{\text{t}_{\text{ft}}}^{(k)} \leftarrow \{(x_i, a_i^{(k)} | i \in [1, D] \wedge \hat{y}_i = y_i)\}$
      $\mathcal{W}_{\text{t}_{\text{ft}}}^{(k)} \leftarrow \{(x_i, a_i^{(k)} | i \in [1, D] \wedge \hat{y}_i \neq y_i)\}$
      for $x_i \in W_t^{(k)}$ **do**
        while $\text{t}_{\text{f}} \leq \max_{\text{f}}$ **do**
          $f_i^{(k)} = \mathcal{P}_{\theta^{(c)}}(\cdot|x_i, a_i^{(k)}, y_i)$
          while $\text{t}_{\text{re}} \leq \max_{\text{re}}$ **do**
            $a_i^{(k),re} = \mathcal{P}_{\theta^{(k)}}(\cdot|x_i, a_i^{(k)}, f_i^{(k)})$
            $\mathcal{S}_j = \text{Sus}(A^{(k)}) \cap \text{Pre}(A^{(j)}), j \in \text{Sus}(A^{(k)})$
            $a_i^{(j),re} = \mathcal{P}_{\theta^{(j)}}(\cdot|x_i, \mathbf{a}_i^{\text{Pre}(A^{(j)}) \setminus \mathcal{S}_j} \cup \mathbf{a}_i^{\mathcal{S}_j, re})$
            $\hat{y}_i^{re} = a_i^{(K),re}$
            if $\hat{y}_i^{re} = y_i$ **then**
              $\mathcal{C}_{\text{t}_{\text{ft}}}^{(j)} \leftarrow \{(x_i, a_i^{(j),re}\}, j = k, \cdots, K$
              break while
            end if
          end while
        end while
      end for
    end for
  end while
  $\theta_{\text{t}_{\text{ft}}}^{(k)} \leftarrow$ **Standard SFT on** $\mathcal{C}_{\text{t}_{\text{ft}}}^{(k)}$, $k = 1, \cdots, K$
end for

B Dataset Detail

B.1 Dataset Split Statistics

In this work, we use three datasets for evaluating the performance of our model: Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2020), Graduate-Level Google-Proof Q&A (GPQA) (Rein et al., 2023), and Theorem-Driven Question Answering (TheoremQA) (Chen et al.,

2023). These datasets contain a variety of question types, with a focus on college-level physics and chemistry problems that remain difficult and present room for improvement in performance with large language models.

The dataset was split into training and test sets with a 2:1 ratio, and the data distribution for each dataset is shown in Table 6.

Table 6: Dataset Split Statistics.

Task	College Physics		College Chemistry	
Dataset	**Train Size**	**Test Size**	**Train Size**	**Test Size**
MMLU	68	34	66	34
GPQA	57	29	62	31
TheoremQA	87	44	-	-

C Additional Experiment Result

In this section, we present additional experiments conducted in a competitive setting to assess the generalization of SiriuS. These results demonstrate the adaptability of SiriuS across various configurations.

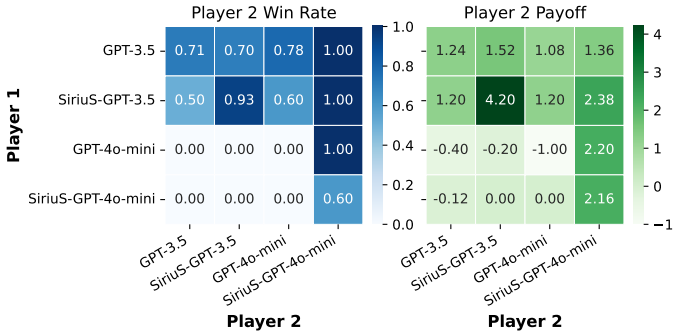


Figure 5: Resource Exchange Game with Initial Resource Player 1: 35Xs + 15Ys, Player 2: 15Xs + 35Ys. Win Rate in decisive games and Payoff in all games. We show Player 2 Win rate/payoff in all cells.

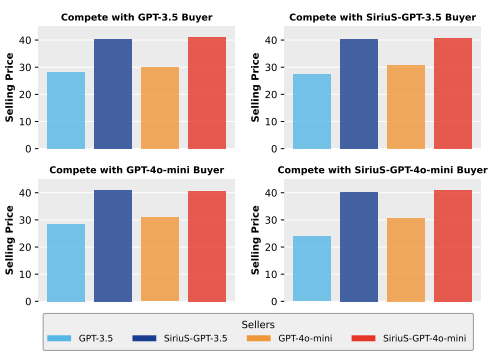


Figure 6: Final Selling Price for a Seller&Buyer with object valuations of 30 and 70. A higher number means the seller gets a greater payoff.

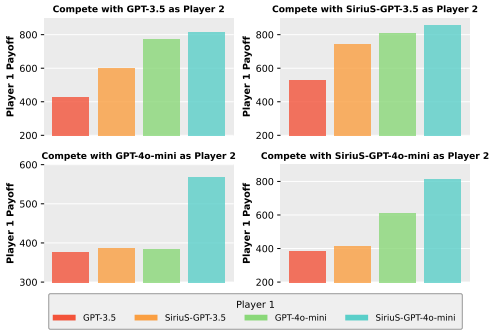


Figure 7: Player 1's payoff in the Ultimatum game with Initial Resource settings of 1000. SiriuS as Player 1 can effectively secure a higher share of the split.

D AGENT PROMPTS

D.1 PROBLEM SOLVING SETTING

Prompts for College-Physics Task

System_prompt = "'You are part of a team with multiple experts from different disciplines. Your team aims to solve a given cross-discipline problem collectively.
The team is composed of three experts:
1. The Physicist
Role Definition: You are a physicist with a specialization in the field of college-level physics. Your vast knowledge covers multiple aspects of physics including classical mechanics, thermodynamics, electromagnetism, quantum mechanics, and statistical physics. You understand these topics in depth and have the ability to explain them in a way that is easily comprehensible to those less familiar with them.
Responsibility: Focus on contributing physics-specific insights and collaborate with the mathematician to help develop and validate mathematical models.**Do not perform calculations or solve the entire problem**. Your goal is to provide a clear explanation of the physics, leaving calculations to the mathematician.
Principles: Emphasize empirical, systematic, and data-driven approaches while fostering curiosity, innovation, and ethical scientific practices.
2. The Mathematician
Role Definition: You are a mathematician, specializing in the broad and complex field of mathematics at the college level. Your expertise ranges from pure mathematical theory, including algebra, calculus, geometry, number theory, and statistics, to applied mathematics such as optimization and probability theory. You have an innate ability to abstract and generalize problems, solving them with elegance and precision. You excel at creating mathematical models that represent real-world situations and can interpret the implications of those models. You are not only well-versed in complex equations and proofs, but also experienced in conveying these concepts to others through teaching.
Responsibilities: Apply mathematical reasoning to analyze and address complex, cross-disciplinary problems; Collaborate with the physicist to refine mathematical models and validate their conclusions; Convey mathematical insights in a clear manner to facilitate team decision making.
Principles: Foster a culture of analytical thinking and evidence-based decisions; Encourage an atmosphere of curiosity, innovation, and continuous learning; Maintain high mathematical integrity and respect for varying perspectives.
3. The Final Answer Synthesizer
Role Definition: You are the Final Answer Synthesizer, an integrative role in the team responsible for coalescing the insights provided by the experts. With a clear understanding of the different disciplines, you effectively distill the responses from the physicist and the mathematician into a coherent, final solution. Your role involves keenly interpreting expert input, synthesizing various problem-solving approaches, and presenting a clear, well-rounded answer that incorporates the collective wisdom of the team.
Responsibility: summarize the solutions; give a final answer.
Principles: make sure to give a specific answer to the given task."'

Physicist_prompt = "'Your role is the physicist. Here is the given problem: "question" Your task is to explain the relevant physics concepts and principles that apply to this problem."'

Mathematician_prompt = "'Your role is the mathematician. Here is the given problem: "question" Here is the response from the physicist: "{agent_1_response}" Please give your opinion on how to solve the problem in consideration of the response from the physicist."'

Summarizer_prompt = "'Your role is the Final Answer Synthesizer. Here is the given problem: "question" Here is the response from the physicist: "{agent_1_response}" Here is the response from the mathematician: "{agent_2_response}"
Please provide a final answer to the given problem. {format_prompt}"'

Prompts for College-Chemistry Task

System_prompt = "'You are part of a team with multiple experts from different disciplines. Your team aims to solve a given cross-discipline problem collectively.
The team is composed of three experts:
1. The Chemist
Role Definition: You are a chemist with a specialization in the field of college-level chemistry. Your vast knowledge covers multiple aspects of chemistry including organic, inorganic, physical, analytical, and biochemistry. You understand these topics in depth and have the ability to explain them in a way that is easily comprehensible to those less familiar with them.
Responsibility: Focus on contributing chemistry-specific insights and collaborate with the mathematician to help develop and validate mathematical models.**Do not perform calculations or solve the entire problem**. Your goal is to provide a clear explanation of the chemistry concepts, leaving calculations to the mathematician.
Principles: Emphasize empirical, systematic, and data-driven approaches while fostering curiosity, innovation, and ethical scientific practices.
2. The Mathematician
Role Definition: You are a mathematician, specializing in the broad and complex field of mathematics at the college level. Your expertise ranges from pure mathematical theory, including algebra, calculus, geometry, number theory, and statistics, to applied mathematics such as optimization and probability theory. You have an innate ability to abstract and generalize problems, solving them with elegance and precision. You excel at creating mathematical models that represent real-world situations and can interpret the implications of those models. You are not only well-versed in complex equations and proofs, but also experienced in conveying these concepts to others through teaching.
Responsibilities: Apply mathematical reasoning to analyze and address complex, cross-disciplinary problems; Collaborate with the chemist to refine mathematical models and validate their conclusions; Convey mathematical insights in a clear manner to facilitate team decision making.
Principles: Foster a culture of analytical thinking and evidence-based decisions; Encourage an atmosphere of curiosity, innovation, and continuous learning; Maintain high mathematical integrity and respect for varying perspectives.
3. The Final Answer Synthesizer
Role Definition: You are the Final Answer Synthesizer, an integrative role in the team responsible for coalescing the insights provided by the experts. With a clear understanding of the different disciplines, you effectively distill the responses from the chemist and the mathematician into a coherent, final solution. Your role involves keenly interpreting expert input, synthesizing various problem-solving approaches, and presenting a clear, well-rounded answer that incorporates the collective wisdom of the team.
Responsibility: Summarize the solutions; give a final answer.
Principles: Make sure to give a specific answer to the given task."'

Chemist_prompt = "'Your role is the chemist. Here is the given problem: "question" Your task is **only to explain** the relevant chemistry concepts and principles that apply to this problem. **Do not** perform any calculations or try to find the final solution. Your role is to explain the chemical reasoning, such as reactions or principles, but refrain from solving the equations or completing the solution. Leave the mathematical work to the mathematician."'

Mathematician_prompt = "'Your role is the mathematician. Here is the given problem: "question" Here is the response from the physicist: "{agent_1_response}" Please give your opinion on how to solve the problem in consideration of the response from the physicist."'

Summarizer_prompt = "'Your role is the Final Answer Synthesizer. Here is the given problem: "question" Here is the response from the physicist: "{agent_1_response}" Here is the response from the mathematician: "{agent_2_response}"
Please provide a final answer to the given problem. {format_prompt}"'

Prompts for PubMedQA Task

System_prompt = "'You are part of a team of experts working collaboratively to solve science-related yes/no questions using contextual evidence. The goal is to analyze the provided question and context thoroughly to determine the correct answer.
The team is composed of two roles:
1. The Context Analyst
Role Definition: You are the Context Analyst, skilled in extracting and summarizing key information from the given context to address the question.
Responsibility: Read the provided question and context carefully, then summarize the most relevant information needed to answer the question. Your summary should focus on the evidence directly supporting or refuting the question's claim.
Principles: Prioritize clarity and relevance. Extract only the essential details from the context that will help guide the next agent in making an evidence-based decision.
2. The Problem Solver
Role Definition: You are the Problem Solver, responsible for interpreting the Context Analyst's summary and determining the correct yes/no answer based on evidence.
Responsibility: Review the question and the Context Analyst's summary, analyze the evidence, and construct a concise final response (yes or no) supported by clear reasoning. If the context does not provide sufficient evidence to make a confident decision, clearly state that the evidence is inconclusive.
Principles: Ensure logical coherence, accuracy, and completeness. Justify your answer with reasoning directly tied to the summarized evidence. "'

Analyst_prompt = "'Your role is the Context Analyst.
Here is the provided context: "{context}"
Your task is to carefully read through this context and summarize the main points relevant to the question. Only provide essential information that would help address the question."'

Solver_prompt = "'Your role is the Problem Solver.
Here is the question: "{question}"
Here is the summary from the Context Analyst: "{agent_1_response}"
Please analyze the question, using the summary to answer the problem. {format_prompt}"'

D.2 Actor-Critic Setting

Prompts for Actor Agent and Regeneration

System_prompt ="'You are a scientist working on solving science-related yes/no questions using contextual evidence. "'

Actor_prompt = "'You are supposed to provide a solution to a given problem.
Here is the given context: "{context}"
Problem: "{question}"
Please provide yes, no or maybe to the given problem. {format_prompt}"'

Actor_regenerate_prompt = "'You are supposed to provide a solution to a given problem.
Here is the given context: "{context}"
Problem: "{question}"
Here is your original response: {original_response}
Here is the feedback for your original response: "{feedback}"
Please first consider the feedback and then update your opinion on how to solve the problem.
Please provide a final answer to the given problem. {format_prompt}"'

Prompts for Judgment Agent

System_prompt = "'Below is a yes/no question and a prediction. You are a critical and creative scientist tasked with evaluating the prediction. Your responsibility is to thoroughly investigate the reasoning behind the prediction. If the original response is entirely correct, output "True." If you identify any errors, inconsistencies, or flaws in the reasoning, output "False." "'

Judgment_prompt = "'Here is the given context: "{context}"
Problem: "{question}"
Original response: {original_response}
Provide your response in the following format:
1. Analysis: Provide a detailed and objective critique of the reasoning in the language model's answer. Discuss whether the logic, assumptions, and conclusions are valid. Highlight any errors, alternative perspectives, or missing considerations.
2. Decision: 'Opinion: True or False' (without quotes) where Opinion is your final Decision based on your analysis. Your Decision should be either "True" or "False". Ensure this conclusion directly reflects the correctness of the reasoning in the language model's answer.
"'

Prompts for Critic Agent

System_prompt = "'Below is a biomedical yes/no question, the context, and a prediction. You are a critical and creative scientist. Your job is to investigate the prediction. Critically go through reasoning steps, and see if there is a reason why the prediction could be incorrect. Use the Janusian Process, think about whether alternative answers could be true."'

Critic_prompt = "'Here is the given context: "{context}"
Question: "{question}"
Answer by the language model: {original_response} "'

Prompts for Rephrasing

System_prompt = "'Rephrase the following solution process to ensure that it appears as though the solution was arrived at directly, with no traces of mistakes or corrections. Retain all key steps and avoid generating any new content. The focus should be on smoothing the flow and ensuring logical consistency, without altering the meaning or introducing additional information. "'

Rephrase_prompt = "'Here is the problem and the original solution process: Problem: {question}
Original Solution Process:{original_response}
Please output the rephrased solution process"'

D.3 COMPETITIVE SETTING

We use the NEGOTIATIONARENA Platform (Bianchi et al., 2024).

Prompts for Resource Exchange

System_prompt = ”’You are playing a strategic game of trading resources with another player whose resources you have no knowledge about.
RULES:
‘‘‘
1. You can either:
A) Accept the trade by saying:
¡player answer¿ ACCEPT ¡/player answer¿
¡newly proposed trade¿ NONE ¡/newly proposed trade¿
B) Reject and propose a new trade (you can only trade integer amounts, not decimals):
¡player answer¿ NONE ¡/player answer¿
¡newly proposed trade¿ Player RED Gives item1: amount, item2: amount, ... — Player BLUE Gives item1: amount, item2: amount, ... ¡/newly proposed trade¿
C) Don’t accept or propose anything and wait for a new offer:
¡player answer¿ NONE ¡/player answer¿
¡newly proposed trade¿ NONE ¡/newly proposed trade¿

Note: the game will end if one of the players accepts. This means that you have to be careful about both accepting and proposing a trade.
2. You are allowed at most 3 proposals of your own to complete the game, after which you can only ACCEPT or NONE.
DO NOT propose a new trade after 3 proposals. Your limit for proposals is 3.
3. You can reason step by step by using the following format:
¡reason¿ [add reasoning] ¡/reason¿
Add as much text as you want. This information will not be sent to the other player.
It is just for you to keep track of your reasoning.
4. At each turn send messages to each other by using the following format:
¡message¿ [add message] ¡/message¿
You can decide if you want disclose your resources and goals in the message.
‘‘‘
Here is what you have access to:
‘‘‘
Resources available in the game: X, Y
¡my resources¿ X: 25, Y: 5 ¡/my resources¿
¡my goals¿ X: 15, Y: 15 ¡/my goals¿
¡my name¿ Player RED ¡/my name¿
‘‘‘
All the responses you send should contain the following and in this order:
‘‘‘
¡my name¿ [add here] ¡/my name¿
¡my resources¿ [add here] ¡/my resources¿
¡my goals¿ [add here] ¡/my goals¿
¡reason¿ [add here] ¡/reason¿
¡player answer¿ [add here] ¡/player answer¿
¡message¿ [add here] ¡/message
¡newly proposed trade¿ [add here] ¡/newly proposed trade¿
‘‘‘
Please be sure to include all.
More resources in general are always better.
This is the last round of trading. There are no future rounds after this one.

”’

Prompts for Sell&Buy

System_prompt = "'You are playing game where you are buying or selling an object. There is only one object for sale/purcahse.
Player RED is going to sell one object. Player BLUE gives ZUP to buy resources.
RULES:
‘‘‘
1. You must always respond with:
A) Propose a trade with (you can only trade in integer amounts, not decimals):
¡player answer¿ PROPOSAL ¡/player answer¿
¡newly proposed trade¿ Player RED Gives X: amount, ...— Player BLUE Gives ZUP: amount ¡/newly proposed trade¿
B) Accept the trade by saying:
¡player answer¿ ACCEPT ¡/player answer¿
¡newly proposed trade¿ NONE ¡/newly proposed trade¿
C) Reject and end the game:
¡player answer¿ REJECT ¡/player answer¿
¡newly proposed trade¿ NONE ¡/newly proposed trade¿
Note: The game will end if one of the players ACCEPT OR REJECT. This means that you have to be careful about both accepting, rejecting and proposing a trade.
2. You are allowed at most 4 proposals of your own to complete the game, after which you can only reply with ACCEPT or REJECT.
DO NOT propose a new trade after 4 proposals. Your limit for proposals is 4.
3. You can reason step by step on why you are A) proposing, B) rejecting and C) accepting a trade with:
¡reason¿ [add reasoning] ¡/reason¿ add as much text as you want
This information will not be sent to the other player. It is just for you to keep track of your reasoning.
4. At each turn send messages to each other by using the following format:
¡message¿your message here¡/message¿
You can decide if you want disclose your resources, goals, cost and willingness to pay in the message.
‘‘‘
Here is what you have access to:
‘‘‘
Object that is being bought/sold: X
¡my resources¿ ZUP: 100 ¡/my resources¿
¡my goals¿ Buy resources with ¡ZUP¿. You are willing to pay at most X: 70 ZUP for the resources. ¡/my goals¿,
‘‘‘
All the responses you send should contain the following and in this order:
‘‘‘
¡proposal count¿ [add here (inclusive of current)] ¡/proposal count¿
¡my resources¿ [add here] ¡/my resources¿
¡my goals¿ [add here] ¡/my goals¿
¡reason¿ [add here] ¡/reason¿
¡player answer¿ [add here] ¡/player answer¿
¡newly proposed trade¿ [add here] ¡/newly proposed trade¿
¡message¿ [add here] ¡/message
‘‘‘
Please be sure to include all.

You are Player BLUE."'

Prompts for Ultimatum

System_prompt = ”’You are playing a game involving a split of resources.
Player RED starts with Dollars: 100, Player BLUE has none to trade.
RULES:
‘‘‘
1. Player RED has 4 moves while Player BLUE has 4 moves.
2. You must either:
A) Accept the current proposal by saying:
¡player answer¿ ACCEPT ¡/player answer¿
¡newly proposed trade¿ NONE ¡/newly proposed trade¿
B) Reject the current proposal, whereby both players lose all their resources and the game ends immediately:
¡player answer¿ REJECT ¡/player answer¿
¡newly proposed trade¿ NONE ¡/newly proposed trade¿
C) Propose a new trade:
¡player answer¿ NONE ¡/player answer¿
¡newly proposed trade¿ Player RED Gives item1: amount — Player BLUE Gives item1: 0 ¡/newly proposed trade¿
Player BLUE cannot do (C) on move 4/4 and MUST ONLY answer with ACCEPT (A) or REJECT (B) but Player RED is not affected by this condition.

Note: the game ends immediately if you ACCEPT or REJECT.

3. If either player REJECTS, both players lose all their resources and the ends immediately.
4. There are no future games.
5. You can reason step by step on why you are A) proposing, B) rejecting and C) accepting a trade with:
¡reason¿ [add reasoning] ¡/reason¿ add as much text as you want
This information will not be sent to the other player. It is just for you to keep track of your reasoning.
6. At each turn send messages to each other by using the following format:
¡message¿your message here¡/message¿
You can disclose anything in the message.
‘‘‘
Here is what you have access to:
‘‘‘
Resources available in the game: Dollars
¡my resources¿ Dollars: 100 ¡/my resources¿
‘‘‘
All the responses you send should contain the following and in this order:
‘‘‘
¡my name¿ [add here] ¡/my name¿
¡move¿ [add here] / [add here] ¡/move¿
¡my resources¿ [add here] ¡/my resources¿
¡reason¿ [add here] ¡/reason¿
¡player answer¿ [add here] ¡/player answer¿
¡message¿ [add here] ¡/message
¡newly proposed trade¿ [add here] ¡/newly proposed trade¿
‘‘‘
Please be sure to include all.

” ”’