# OPTAGENT: Optimizing Multi-Agent LLM Interactions Through Verbal Reinforcement Learning for Enhanced Reasoning

**Anonymous ACL submission**

## Abstract

Large Language Models (LLMs) have shown remarkable reasoning capabilities in mathematical and scientific tasks. To enhance complex reasoning, multi-agent systems have been proposed to harness the collective intelligence of LLM agents. However, existing collaboration structures are either predefined or rely on majority voting or round-table debates, which can suppress correct but less dominant agent contributions. Recent approaches model multi-agent systems as graph networks but optimize purely for agent performance, neglecting the quality of interactions. We hypothesize that effective agent communication is crucial for multi-agent reasoning and that debating quality plays a significant role. To address this, we propose OPTAGENT, a multi-agent verbal reinforcement learning algorithm that dynamically constructs and refines multi-agent collaboration structures. Our method defines action spaces and a feedback mechanism that evaluates communication robustness and coherence throughout the debate. The final decision is achieved through a majority vote over all the agents. We assess OPTAGENT on various reasoning tasks, including mathematical reasoning, creative writing, scientific reasoning, and numerical sorting. Results demonstrate that our approach significantly outperforms single-agent prompting methods and state-of-the-art multi-agent frameworks on diverse tasks.

## 1 Introduction

Large Language Models (LLMs) have exhibited significant potential in reasoning across various downstream tasks, including elementary mathematical reasoning, and fundamental science reasoning (Brown et al., 2020; Dubey et al., 2024; Wei et al., 2022; Wang et al., 2023b). Despite these initial successes, existing methodologies necessitate meticulously crafted prompt strategies that are often fixed for certain tasks (Yao et al., 2023; Besta et al., 2024). This approach lacks flexibility, as the users have to define different prompts under different scenarios, especially for complex reasoning tasks. A promising solution that mitigates the challenge is to explore multi-agent frameworks that capitalize on the strengths of LLM-based agents. Researchers proposed many multi-agent reasoning frameworks that enable collaborative debates among multiple LLM agents (Chan et al., 2023; Liang et al., 2024; Chen et al., 2023b; Wang et al., 2023a; Chen et al., 2023a), which are akin to human group problem-solving scenarios.

Despite these initial successes, existing multi-agent LLM reasoning methods often follow predefined or simple group chatting collaboration structures. For example, AutoGen (Wu et al., 2023) and ChatEval (Chan et al., 2023) employs pre-defined collaboration structures; ReConcile (Chen et al., 2023b) employs group discussion with confidence-based consensus decision; MAD (Liang et al., 2024) employs group debate with a meta-summarizer as the decision-maker. These methods do not account for the varying interactions of differently profiled agents, nor do they optimize the sequence of communications to ensure the most effective information flow for specific tasks. As a result, correct but less dominant agent contributions could be overlooked. We believe the interaction schemas should be more flexible and further optimized for task-specific communication efficacy.

Recent trends in multi-agent collaboration emphasize using graph optimization techniques to enable flexible, task-adaptable coordination among agents, enhancing efficacy and scalability in complex environments. Specifically, GPT-Swarm (Zhuge et al., 2024) conceptualizes the multi-agent framework as a computational graph. The inspiration is drawn from a "Society-of-Mind" concept and highlights the communication and collaboration among agents. For optimization, the authors use reinforcement learning to optimize the agent interactions. While previous methods show reasonable performance, they tend to overlook the agents' debate quality, an important aspect of a multi-agent framework. We hypothesize that the interaction quality between the agents should also play an important role in the optimization process. More specifically, we believe the optimization algorithms should also consider metrics like wording clarity and logical coherency apart from agent performance metrics.

To tackle the above challenges, we propose OPTAGENT, an LLM-based Verbal Reinforcement Learning framework for Graph Optimization on multi-agent collaboration. The goal of OPTAGENT is to find the most effective interaction patterns in a multi-agent collaboration graph. OPTAGENT explicitly considers communication quality when identifying the most effective connections between agents. To refine the multi-agent collaboration structure, OPTAGENT contains a feedback agent that evaluates the quality of the agent interactions and an action agent that updates the multi-agent collaboration graph based on the feedback. The final decision is achieved through a majority vote over all the agents. We evaluate OPTAGENT on various downstream reasoning tasks, including mathematical reasoning, scientific reasoning, creative writing, and sorting tasks. Our experimental results demonstrate that OPTAGENT significantly outperforms single-agent prompting methods and state-of-the-art multi-agent debating schemas on diverse reasoning tasks across various LLM families. We also present a case study to illustrate the efficacy of our framework.

## 2 Related Work

**LLM Reasoning Prompting** The field of large language models (LLMs) has seen significant advancements in recent years, particularly in the area of reasoning prompting. Various prompt engineering methods have been developed, aiming to improve large language models' reasoning ability across various tasks and domains. Chain-of-thought (CoT) prompting (Wei et al., 2022) prompts the large language models (LLMs) to divide their reasoning process into smaller steps when solving a question, forming a chain of thoughts. Chain-of-thought self-consistency prompting (Wang et al., 2023b) improves on the CoT method by proposing different reasoning chains and ensembles on the final result. Tree-of-thought (ToT) prompting method (Yao et al., 2023) actively maintains a tree of thoughts, where each thought is a coherent language sequence that serves as an intermediate step toward problem-solving. Graph-of-thought (Besta et al., 2024) further improves ToT by constructing a Directed Graph instead of a tree. LLMs can loop over a thought to refine it and aggregate thoughts or chains. There are also other X-of-thought prompting methods developed for various different downstream tasks and datasets (Chen et al., 2023c; Sel et al., 2024; Bi et al., 2024; Jin et al., 2024). Another notable contribution to the field is the systematic survey on prompting techniques by the Prompt Engineering Guide (Schulhoff et al., 2024). This survey categorizes various prompting methods and their applications, emphasizing the importance of prompt design in enhancing LLM reasoning.

**Multi-Agent Reasoning** Recent advancements in large language model (LLM) multi-agent frameworks have garnered significant attention in the field of artificial intelligence. Studies such as (Wu et al., 2023; Chen et al., 2023a; Lu et al., 2024) have highlighted the impressive reasoning capabilities of LLMs, which have been leveraged to create autonomous agent systems that are capable of complex problem-solving and perform better than single agents.

The question is how researchers can design effective multi-agent reasoning frameworks. There have been several studies and analyses on the efficiency and effectiveness of multi-agent debating systems over reasoning tasks (Wang et al., 2023a, 2024; Pezeshkpour et al., 2024). However, most of the interaction schemas and decision strategies are either pre-defined (Wu et al., 2023; Chan et al., 2023), or follow a simple structure such as group debate, majority voting, summarizer decision, or a combination of the above strategies (Chen et al., 2023b; Liang et al., 2024; Chan et al., 2023). Recently, several researchers from KAUST proposed GPTSwarm (Zhuge et al., 2024), in which they suggest that the multi-agent system can be considered as a graph network and thus their interaction patterns can be optimized by optimization algorithms. They also conduct individual optimizations on agents by conducting prompt optimization. However, their optimization is heavily performance-oriented, overlooking the debating quality of the agents. This is something that should also be considered in LLM free generation.

## 3 OptAgent Framework

### 3.1 Problem Definition

Given a problem $P$, and $N$ LLM agents $A_1, A_2, ..., A_N$, our goal is to find the answer to question $P$. We achieve this goal through using LLMs as agents to conduct logical reasoning and structured discussions. Each agent is a distinctly prompted LLM capable of generating the answer and the corresponding CoT reasoning process.

### 3.2 Framework Overview

In our setting, we view the multi-agent collaboration framework as a graph. Each agent is a node in the graph, denoted by $A_i$; the communications between agents are the edges, denoted by $e_{ij}$. We hypothesize that the interaction quality will be different for differently profiled agents, and the best connection order would allow the best information propagation pattern for a particular task. The goal of OptAgent is to optimize the connections between the agents and improve the overall performance of the multi-agent collaboration framework.

In our verbal reinforcement learning process, we design two meta agents, $LLM_{reflect}$ and $LLM_{act}$. which handle reflection and action processes, respectively. The training process involves selecting connections based on probability scores and updating them through reinforcement learning. Finally, a majority voting strategy is used to determine the final answer after executing the graph.

### 3.3 Initial Graph Setup

**Agent Profiling and Force Decoding** Given a group of LLM agents $A_1, ...A_i$, we ensure similar but different reasoning by assigning the agents with the same baseline reasoning prompt but different agent profiles in system prompts (see Appendix B). The seven agent profiles were manually crafted to reflect common reasoning strategies found in human problem-solving, such as deductive logic, intuition, and domain expertise. For the 3-agent and 5-agent scenarios, we randomly select 3 and 5 profiles from the proposed profiles, respectively. To promote versatility, we force the model to generate three different outputs for each agent profile and randomly choose one of the outputs as its initial answer to the input question.

**Connection Initialization** Given a group of agents $A_1, ...A_i$, and possible connections between the agents $e_{12}, ..., e_{ij}$, we first get the group of utility scores $u(A_i)$, which is the average self-evaluated confidence score given by the agent $A_i$ for the given task. We first randomly sample ten problems from the dataset, collect the confidence score from each agent on each question, and then calculate the average confidence score $u(A_i)$.

Then, we calculate the connection score of an edge, $s(e_{ij}) = u(A_i) * u(A_j)$, which is determined by the utility score of the two connecting nodes. We will update the connection scores during the reinforcement learning process. Based on all of the connection scores, we assign the probability, $p(e_{ij}) = \frac{s(e_{ij})}{\sum s(e_{ij})}$ to each connection $e_{ij}$, which is the proportion of the connection score $s(e_{ij})$ to the sum of the connection scores. The probabilities will serve as selection references in the first epoch of our training process.

### 3.4 Verbal Reinforcement Learning

Inspired by the Reflexion framework (Shinn et al., 2023), we design an LLM self-controlled verbal optimization for graph generation. First, we design two meta agents: $LLM_{reflect}$ and $LLM_{act}$. We also create a set of action spaces that $LLM_{act}$ can choose from to alter the current graph network.

**Reflection** $LLM_{reflect}$ is responsible for generating reflection text after $LLM_{act}$ makes a connection between two agents $(A_i, A_j)$. Here, a 'connection' means initiating direct communication between two agents, prompting them to exchange their initial reasoning and answers, debate their points of view, and revise their reasoning based on the exchange. To generate the feedback, $LLM_{reflect}$ takes in the reasoning arguments of $A_i$ and $A_j$ before and after the interaction process. Then, the reflection text is passed on to $LLM_{act}$ to guide its decision-making process. Specifically, the feedback that $LLM_{reflect}$ generates is determined by two criteria:

- **Criterion 1**: Both agents should answer the question correctly after making the connection;
- **Criterion 2**: Agents should be logical and coherent in their reasoning process.

For the first criterion, $LLM_{reflect}$ checks whether the connection helps agent $A_i$ and $A_j$ with answering the question. If both agents got the answer correct, then $LLM_{reflect}$ will give positive feedback. For the second criterion, $LLM_{reflect}$ checks whether the logical chains are sound and valid. If

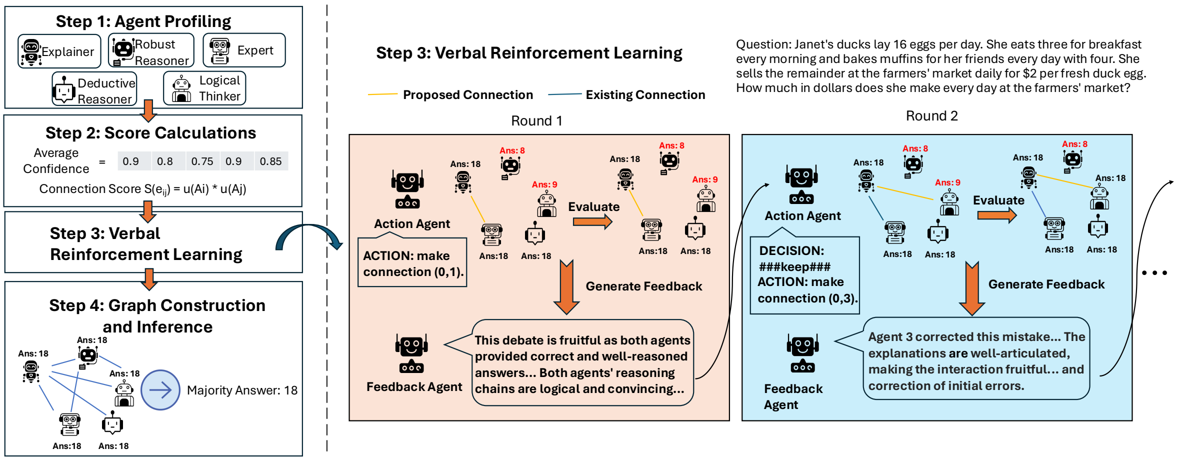


Figure 1: Overview of OPTAGENT framework. The overall pipeline is on the left side; an example process for verbal reinforcement learning is shown on the right.

both agents demonstrate good reasoning quality during the interaction process after seeing each other's reasoning, $LLM_{reflect}$ will give out good feedback. Otherwise, $LLM_{reflect}$ will have negative feedback on the connection $(A_i, A_j)$. Detailed instruction prompts for $LLM_{reflect}$ are provided in Appendix B.

**Action** $LLM_{act}$ is responsible for conducting actions at each step, from the pre-defined action pool:

- Make a connection between the two agents $(A_i, A_j)$ to initiate debate;
- Keep a previously made connection $(A_n, A_m)$;
- Delete a previously made connection between the two agents $(A_n, A_m)$ to prohibit debate.

After $LLM_{act}$ receives the verbal feedback, it will make a decision to keep or delete the previously made connection. For instance, if $LLM_{act}$ decided to make a connection $(A_i, A_j)$ but consequently received negative feedback in this round, then $LLM_{act}$ would remove the connection. We decrease their connection score $s(e_{ij})$ for removed connections. If $LLM_{act}$ receives positive feedback, it will keep the connection $(A_i, A_j)$ in the graph, and we increase the connection score $s(e_{ij})$. Before deciding whether or not to keep the current edge, $LLM_{act}$ would also look back at the feedback history of the current edge in previous rounds.

After the decision, $LLM_{act}$ makes a connection that hasn't been explored during the current training epoch. The result of the newly created connection will be evaluated and passed on to $LLM_{reflect}$ for the next round of reflection text generation.

### 3.5 Training Process

To start the Reinforcement learning process, we perform weighted random sampling to select a connection $(A_i, A_j)$ based on the probability score of the connections. At later epochs, $LLM_{act}$ is responsible for choosing a connection $(A_i, A_j)$. After $LLM_{act}$ takes action, we execute the debate process between $A_i$ and $A_j$, and then pass the results to $LLM_{reflect}$ for feedback, which is then given to $LLM_{act}$ for decision-making. We update the connection score $e_{ij}$ after $LLM_{act}$ has decided whether to keep the connection $(A_i, A_j)$. The connection score $s(e_{ij})$ is increased by $\alpha * \hat{s}(e_{ij})$ if $LLM_{act}$ chooses to keep it and decreases otherwise, where $\alpha$ is the learning rate we set, and $\hat{s}(e_{ij})$ is the current connection score of connection $e_{ij}$. We repeat the above process in the current epoch until every connection is visited once for an update. The pseudocode algorithm is provided in Algorithm 1 in the Appendix.

### 3.6 Inference Process

After the framework is trained with the connection weights updated, we construct the final graph before doing inference. Connections with higher scores are established first. The construction process continues until all agents have been visited. We consider the information flow within the graph as complete when each agent $A_i$ has interacted with at least one other agent $A_j$. The final decision is determined using a majority voting strategy as the final answer $Ans_{final} = mode(Ans_1, ..., Ans_n)$, where $Ans_1, ..., Ans_n$ are answers provided by different agents in the graph. The pseudocode algorithm is provided in Algorithm 2 in the Appendix.

## 4 Experiments

### 4.1 Experimental Setup

**Dataset and Tasks** We experiment OPTAGENT on four downstream tasks: math reasoning, creative writing, science reasoning, and sorting. All experiments were tested on publicly available datasets. For the math reasoning task, we use two datasets: GSM8K (Cobbe et al., 2021), which contains grade school arithmetic questions, and MATH (Hendrycks et al., 2021), which contains high school-level mathematical questions spanning six different fields. We also include two adversarial reasoning datasets that are built on GSM8K: AdversarialGSM (Xie et al., 2024) in which we will refer to as AdvGSM in Table 3, and GSM-PLUS (Li et al., 2024). AdvGSM contains questions that are changed only in number magnitude, and have three levels of difficulties, with M3 being the easiest using same magnitude with GSM8K, and M1 being the hardest. For each of the reasoning datasets except AdvGSM, we randomly select 100 questions from the dataset for evaluation. For AdvGSM, we randomly select 100 questions from each magnitude for evaluation. For creative writing, we follow the setup in (Yao et al., 2023), where we test on 10 examples. For sorting, we randomly generate 100 numerical sequences at length 8, 16, 32.

**Model and Implementation** We experiment the baselines and OPTAGENT utilizing GPT-3.5-turbo (Brown et al., 2020), GPT-4o (OpenAI, 2023), or the LLaMa 3.1-70B model (Dubey et al., 2024). We direct call model APIs for prompting. For all models, We set the temperature to 0.5, and $top_k$ to 1.0. For GPT-Swarm and OPTAGENT, we use a total of three data points to train the framework. All agents, including the baselines, are prompted with the 0-shot CoT prompt. We train OPTAGENT on three randomly sampled data points from the dataset and report the performance on randomly sampled evaluation datasets mentioned before. We run OPTAGENT three times and report the mean performance. We use majority voting as our final decision strategy and a random choice when there is a tie. We provide a cost analysis under the 5-agent scenario for some baselines in AppendixC.

**Baselines** We compared OPTAGENT with six single-agent prompting methods and state-of-the-art multi-agent baseline methods as below:

- **Single Model Prompts** in which we include 3 prompts: **DirectIO**, where we ask the model for a direct answer without explanations; **0-Shot CoT**, where we ask the model to provide step-by-step reasoning without providing any demonstrating examples; **ToT**, where we follow (Yao et al., 2023) and implement their framework.

- **Simple Debate**, where we initiate several instances of non-profiled agents with the same 0-shot CoT prompt. The agents are provided with each other's reasonings and answers, and are asked to reflect on their own reasoning. We let models debate for 2 rounds and utilize a majority voting to decide the final answer.

- **GPTSwarm** (Zhuge et al., 2024), where we follow the original implementation. We train the framework using three randomly sampled data points from the dataset and report the performance. We run GPTSwarm three times and report the mean performance.

- **ReConcile** (Chen et al., 2023b), where we follow the original implementation, using GPT-3.5-turbo and gpt-4o models as backbone, respectively. We report their performance in mathematical reasoning datasets. We run ReConcile three times and report the mean performance.

### 4.2 Evaluation Metrics

**Math and Science reasoning** We report the performance in terms of accuracy following prior benchmarks and papers. The datasets include GSM8K, AdvGSM, GSM-PLUS, MATH, ARC and GPQA. We report the detailed post-processing and evaluation description in the Appendix.

**Creative Writing** We follow the metrics in (Yao et al., 2023) and report the performance in terms of Coherence score, which another GPT-4 model evaluates. We provide the evaluation prompt in Appendix B.

**Sorting** We follow the metrics in (Besta et al., 2024) and report the performance in terms of error scope, defined by the sum of the number of wrongly sorted elements and missing elements.

### 4.3 Main Results

**Math Reasoning** We compare OPTAGENT with multi-agent simple debating baselines on Math Reasoning datasets in Table 1. The backbone LLMs (i.e., the primary large language model underlying all agents) include GPT-3.5-turbo and LLaMa 3.1-70B. OPTAGENT performs better on the original

| Model | Prompt Class | Framework Type | GSM8K | AdvGSM-M3 | AdvGSM-M2 | AdvGSM-M1 | GSM-PLUS | MATH |
|---|---|---|---|---|---|---|---|---|
| GPT-3.5-turbo | Single Agent | DirectIO | 35.0 | 52.0 | 28.0 | 15.0 | 27.0 | 8.0 |
| | | 0-Shot CoT | 73.0 | 87.0 | 75.0 | 30.0 | 59.0 | 22.0 |
| | | ToT | 80.0 | 89.0 | 76.0 | 30.0 | 61.0 | 25.0 |
| | 3-Agent | Simple Debate | 77.0 | 90.0 | 79.0 | 31.0 | 62.0 | 25.0 |
| | | GPT-Swarm | 79.6 | **91.3** | 80.6 | 33.6 | 63.0 | 28.0 |
| | | ReConcile | 80.6 | 90.3 | 80.0 | **34.3** | 63.6 | 29.0 |
| | | OPTAGENT | **81.3** | 91.0 | 81.3 | 34.0 | **64.3** | **29.3** |
| | | Accuracy Only | 81.0 | 90.0 | 81.0 | 33.0 | 64.0 | 29.0 |
| | | No Forced Sampling | 79.0 | 89.0 | 81.0 | 32.0 | 63.0 | 29.0 |
| | | Reconsider Minority | 78.0 | 88.0 | 81.0 | 30.0 | 61.0 | 28.0 |
| | | Split Action Agents | 81.0 | 90.0 | **82.0** | 34.0 | 64.0 | 29.0 |
| | 5-Agent | Simple Debate | 78.0 | 91.0 | 82.0 | 33.0 | 62.0 | 30.0 |
| | | GPT-Swarm | 81.3 | 92.6 | 85.3 | 35.3 | 66.6 | 32.6 |
| | | ReConcile | 82.3 | 93.6 | **86.3** | 36.3 | 66.3 | 33.3 |
| | | OPTAGENT | **87.3** | **95.6** | 85.3 | **38.6** | 66.0 | 34.6 |
| | | Accuracy Only | 84.0 | 94.0 | 84.0 | 36.0 | 64.0 | 33.0 |
| | | No Forced Sampling | 84.0 | 94.0 | 84.0 | 37.0 | 65.0 | 32.0 |
| | | Reconsider Minority | 85.0 | 95.0 | 86.0 | 36.0 | **69.0** | **37.0** |
| | | Split Action Agents | 86.0 | 95.0 | 85.0 | 38.0 | 66.0 | 34.0 |
| GPT-4o | 5-Agent | Simple Debate | 97.0 | 98.0 | 85.0 | 42.0 | 86.0 | 41.0 |
| | | GPT-Swarm | 97.0 | 98.0 | 87.0 | 44.0 | 88.0 | 42.0 |
| | | ReConcile | **98.0** | **99.0** | 87.0 | 44.0 | **89.0** | 42.0 |
| | | OPTAGENT | **98.0** | 98.0 | **88.0** | **45.0** | 88.0 | **45.0** |

Table 1: Main results table on Math Reasoning Task. The best-performing methods on each dataset under each number-of-agent scenario are bolded, and the second-best are underlined. The results below OPTAGENT represent the variants of OPTAGENT framework. The detailed setting and discussion are presented in Section 4.4.

| Multi-Agent Framework | GSM8K | GSM8K-M3 | GSM8K-M2 | GSM8K-M1 | GSM-PLUS | MATH |
|---|---|---|---|---|---|---|
| 3 GPT-3.5-turbo | 82.0 | **91.0** | 82.0 | 34.0 | **65.0** | 29.0 |
| 1 LLaMa3.1 70B + 2 GPT-3.5-turbo | 83.0 | 87.0 | **84.0** | **35.0** | 63.0 | 33.0 |
| 2 LLaMa3.1 70B + 1 GPT-3.5-turbo | 84.0 | 83.0 | 73.0 | 34.0 | 61.0 | **34.0** |
| 3 LLaMa3.1 70B | **92.0** | 71.0 | 56.0 | 26.0 | 62.0 | 33.0 |

Table 2: Mixture of Model Ablation Task. All the multi-agent frameworks are optimized with OPTAGENT.

datasets like GSM8K and MATH than the simple debating baselines, and significantly outperforms the single-agent baselines. The performance increase is more prominent in 5-agent scenarios compared with 3-agent scenarios. We present also present the results of two adversarial datasets in column 5 to 8. OPTAGENT demonstrates robustness in the adversarial math reasoning datasets, outperforming the baseline scheme and frameworks by a similar margin compared with the original datasets.

We also conduct experiments on the mathematical datasets with GPT-4o as the backbone model. With enhanced reasoning ability, even the simple debating method performs near-perfectly on basic math reasoning datasets. We still see a slight performance increase using the multi-agent debating frameworks on more challenging datasets.

**Creative Writing** Results for creative writing task is reported in Figure 2. OPTAGENT increase the coherence score by an average of at least 0.1 points across different settings under this task.

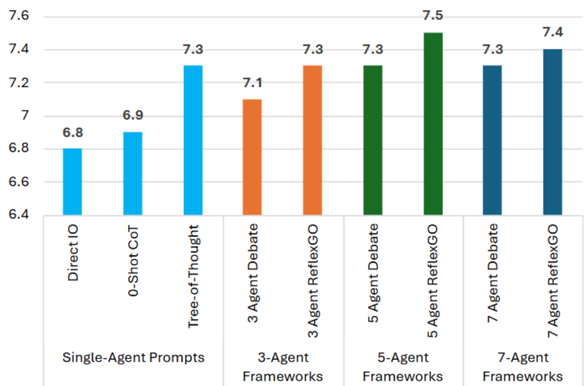


Figure 2: Results on Creative Writing, measured in terms of coherence scores.

Compared with Tree-of-Thought, which used a single model to explore different branches, OPTAGENT achieves slightly better performance. Increasing the number of agents only brings marginal performance improvement, and adding more agents from 5 to 7 does not seem to help with the performance of the multi-agent framework.

| Number of Agents | Framework Type | GSM8K | GSM8K-M3 | GSM8K-M2 | GSM8K-M1 | GSM-PLUS | MATH |
|---|---|---|---|---|---|---|---|
| 3-Agent | Simple Debate | 77.0 | 90.0 | 79.0 | 31.0 | 62.0 | 25.0 |
| | +Profiling | 82.0 (+5.0) | 90.0 (+0.0) | 82.0 (+3.0) | 33.0 (+2.0) | 64.0 (+2.0) | 29.0 (+4.0) |
| | OPTAGENT | 82.0 (+5.0) | 91.0 (+1.0) | 82.0 (+3.0) | 34.0 (+3.0) | 65.0 (+3.0) | 29.0 (+4.0) |
| 5-Agent | Simple Debate | 78.0 | 91.0 | 82.0 | 33.0 | 62.0 | 30.0 |
| | +Profiling | 83.0 (+5.0) | 94.0 (+3.0) | 84.0 (+2.0) | 35.0 (+2.0) | 66.0 (+4.0) | 31.0 (+1.0) |
| | OPTAGENT | **87.0 (+9.0)** | 96.0 (+5.0) | **86.0 (+4.0)** | **38.0 (+5.0)** | 67.0 (+5.0) | **34.0 (+4.0)** |
| 7-Agent | Simple Debate | 78.0 | 92.0 | 81.0 | 34.0 | 62.0 | 30.0 |
| | +Profiling | 83.0 (+5.0) | 95.0 (+3.0) | 85.0 (+4.0) | 35.0 (+1.0) | 65.0 (+3.0) | 31.0 (+1.0) |
| | OPTAGENT | 85.0 (+7.0) | **98.0 (+6.0)** | **86.0 (+5.0)** | 37.0 (+4.0) | **68.0 (+6.0)** | 33.0 (+2.0) |

Table 3: Performance of OPTAGENT on GPT-3.5-turbo under 3, 5, and 7-agent scenarios. "Simple Debate" refers to agents debating without profiles and forced generation. "+Profiling" refers to debating with added profiles. OPTAGENT contains both Profiling and Verbal Reinforcement Learning. We bold the best performing variant. The deltas stand for differences between variant from simple debate baseline.

### 4.4 Ablation Study

**Train on Accuracy Only** In this experiment, we study the effect of considering interaction quality by asking $LLMact$ to consider only correctness instead of interaction quality when training. The results are demonstrated in Table 1. Under the 5-agent scenario, considering only accuracy in training time would hurt the performance, suggesting that considering interaction quality between agents $LLM_{act}$ plays a vital role in the training process. Under the 3-agent scenario, the performance stayed roughly the same, since the agents' profiles and interactions between the agents are more limited than in the 5-agent scenario.

**Forced Generation and Random Initial Output Sampling** We examine the impact of forced generation, where each agent generates multiple outputs using stochastic decoding, and one is randomly selected. The results are demonstrated in Table 1. Removing this (i.e., using greedy decoding) significantly reduced reasoning diversity and performance under both 3-agent and 5-agent scenarios.

**Split Agent $LLM_{act}$** In this study, we split $LLM_{act}$ into two agents: $LLM_{propose}$, which is responsible for proposing the new connections; and $LLM_{decide}$, which is responsible for deciding whether or not to keep an edge. $LLM_{reflect}$ will interact with $LLM_{decide}$ only. $LLM_{propose}$ would be provided with a summary of the conversation history between $LLM_{reflect}$ and $LLM_{decide}$. We do not see much performance difference across datasets under this setting compared with OPTAGENT, which used a single agent $LLM_{act}$, for the 3-agent and the 5-agent scenario.

**Reconsidering Minority** In this setup, if one agent gets a unique answer while the other agents all got the same majority answer, the unique answer would be considered as a "minority answer", and we would prompt a group discussion on the unique answer first before executing the graph. From the results in Table 1 as well as the upper-bound analysis results in Table 8, we can see that this strategy brings up the performance in datasets where we have a bigger gap between OPTAGENT and the theoretical upper-bound performance. It suggests that the models that had the wrong reasoning will be able to catch their mistakes from this discussion process. However, this approach does not work under the 3-agent scenario, where there are many instances where one agent has the wrong answer. This suggests that the agents are also prone to overthinking and would be misled by the wrong answer.

**Mixture of Models as Agents** Table 2 shows the results of using different backbone models as agents in OPTAGENT under the 3-agent setting. On adversarial datasets where GPT-3.5-turbo performs better than LLaMa3.1, we observe that the performance of OPTAGENT using GPT-3.5-turbo as the backbone model is better than using LLaMa3.1 as the backbone model. This suggests that the communication quality is heavily affected by the performance of the backbone models.

**Different Initialization Methods** We present the effects of different initialization methods for connection scores during the training process in Table 4. "Random Initialization" means all weights are initialized randomly between 0 and 1; "Uniform Initialization" means all weights are initialized to be 0.5; "Confidence-based Initialization" is introduced in Section 3.3. From the table, we see that random initialization performs the worst among all initialization methods, while uniform initialization and confidence score initialization performs around

| Number of Agents | Framework Type | GSM8K | GSM8K-M3 | GSM8K-M2 | GSM8K-M1 | GSM-PLUS | MATH |
|---|---|---|---|---|---|---|---|
| 5-Agent | Random Initialization | 85.0 | 95.0 | 85.0 | 36.0 | 66.0 | 34.0 |
| | Uniform Initialization | **87.0** | 95.0 | **86.0** | 37.0 | **68.0** | **35.0** |
| | Confidence Scores | **87.0** | **96.0** | **86.0** | **38.0** | 67.0 | 34.0 |

Table 4: Performance of OPTAGENT under different initialization methods for the connection scores.

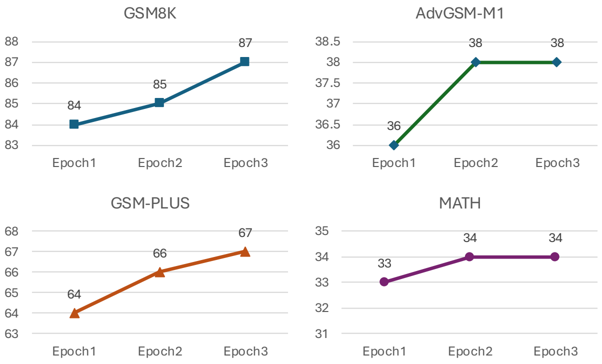


Figure 3: Training Convergence Trend for OPTAGENT under the 5-Agent Setting.

the same across datasets. This suggests that LLMs with different profiles tend to have similar initial confidence self-assessments.

**Effects of Profiling** We present a more detailed performance report of OPTAGENT on GPT-3.5-turbo in Table 3. Compared with simple debating, profiling the agents provide prominent improvement. OPTAGENT further adds to the performance by doing only profiled debate, and the improvement is most significant in the 5-agent scenario. Combined with the previous section, where we reconsidered the minority answers, having different answers and promoting critical thinking would greatly improve model performance on math tasks.

**Number of Agents** From Table 3, we see that the performance enhancement is at its best in 5-agent scenarios. Adding more than 5 agents does not seem to help with answering the questions. Similar patterns can be found in the upper-bound analysis in Table 8, as well as in other works such as (Wang et al., 2024). This suggests that simple scaling is not the best way - continuously increasing the number of agents does not guarantee improvement on multi-agent systems for reasoning datasets.

**Training Convergence** We provide additional study on framework convergence trend in Figure 3. On harder datasets like AdvGSM-M1 and MATH, our framework quickly plateaus from the second epoch. The results suggest that the basic reasoning abilities of the agents greatly affect the learning process; on harder datasets, the agents have difficulties forming high-quality answers and interactions, leaving little room for performance improvement.

### 4.5 Additional Reasoning Tasks

We provide our experiment results for science reasoning and sorting in Table 8 in the Appendix.

**Science Reasoning** Datasets like ARC contain questions that do not require step-by-step reasoning, but direct knowledge retrieval. For these questions, the model's knowledge base and understanding of the questions are more important than the logical reasoning process. On more challenging datasets such as GPQA, we find that the base backbone model's reasoning ability significantly drags down the overall performance of OPTAGENT.

**Sorting** Sorting task requires the base backbone model to have good planning ability. However, the agents often struggle to generate good explanations and reasoning for each of their steps, which poses a significant hurdle when agents have discussions. In complex planning tasks, the more promising direction would be to involve external specialized planning modules into the multi-agent framework.

## 5 Conclusion

This paper proposes OPTAGENT, an LLM-based Verbal Reinforcement Learning framework for Graph Optimization on multi-agent collaboration. OPTAGENT explicitly considers communication quality when identifying the most effective connections between agents. OPTAGENT contains a feedback agent that evaluates the quality of the agent interactions and an action agent that updates the multi-agent collaboration graph based on the feedback. Results on several downstream reasoning tasks demonstrate that OPTAGENT significantly outperforms single-agent prompting methods and state-of-the-art multi-agent frameworks on diverse tasks. Detailed analysis highlights the needs for task-specific designs for complex planning tasks.

## Limitations

**Potential Risk** We acknowledge that due to the inherent training and dataset bias of the base backbone models, and our incomplete controls of the models, our framework could potentially produce harmful content.

**Limited Experiments** Due to computational cost and timeconstraints, our experiments was conducted on a limited number of tasks and datasets, with a randomly chosen subset. Our conclusions and analysis could be further enhanced by testing on more tasks and datasets.

**Computational Cost** OPTAGENT relies on initiating multiple model instances and requires multiple prompts per round. The repetitive callings impose heavy time and output token costs for OPTAGENT.

**Model Reasoning Ability Dependency** The ability of multi-agent framework is heavily influenced by the ability of the individual backbone models. Framework performance and optimization effectiveness could vary between models and datasets.

**Incomplete Control Over Models** For the API-based models, we note that we do not possess complete control over their behavior, and the probability and confidence estimations are post-hoc in nature.

## Ethics Statement

This research adhered to the ethical standards and best practices outlined in the ACL Code of Ethics. Language Models can sometimes produce illogical or inaccurate reasoning paths, so their outputs should be cautiously used. The outputs are only examined to understand how a model arrives at its answers and investigate why it makes certain errors. All experiments used publicly available datasets from previously published works and did not involve ethical or privacy issues.

## References


Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17682–17690. AAAI Press.

Zhenyu Bi, Daniel Hajialigol, Zhongkai Sun, Jie Hao, and Xuan Wang. 2024. Stoc-tot: Stochastic tree-of-thought with constrained decoding for complex reasoning in multi-hop question answering. *Preprint*, arXiv:2407.03687.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

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

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F. Karlsson, Jie Fu, and Yemin Shi. 2023a. Autoagents: A framework for automatic agent generation. In *International Joint Conference on Artificial Intelligence*.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2023b. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *ArXiv*, abs/2309.13007.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023c. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Preprint*, arXiv:2211.12588.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 510 others. 2024. The llama 3 herd of models. *ArXiv*, abs/2407.21783.

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

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. *Preprint*, arXiv:2404.07103.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *ArXiv*, abs/2402.19255.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. *Preprint*, arXiv:2305.19118.

Meng Lu, Brandon Ho, Dennis Ren, and Xuan Wang. 2024. TriageAgent: Towards better multi-agents collaborations for large language model-based clinical triage. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5747–5764, Miami, Florida, USA. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report.

Pouya Pezeshkpour, Eser Kandogan, Nikita Bhutani, Sajjadur Rahman, Tom Mitchell, and Estevam R. Hruschka. 2024. Reasoning capacity in multi-agent systems: Limitations, challenges and human-centered solutions. *ArXiv*, abs/2402.01108.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Minh Pham, Gerson C. Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, and 12 others. 2024. The prompt report: A systematic survey of prompting techniques. *ArXiv*, abs/2406.06608.

Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. 2024. Algorithm of thoughts: Enhancing exploration of ideas in large language models. *Preprint*, arXiv:2308.10379.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Neural Information Processing Systems*.

Lei Wang, Chengbang Ma, Xueyang Feng, Zeyu Zhang, Hao ran Yang, Jingsen Zhang, Zhi-Yang Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji rong Wen. 2023a. A survey on large language model based autonomous agents. *ArXiv*, abs/2308.11432.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? In *Annual Meeting of the Association for Computational Linguistics*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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.

Roy Xie, Chengxuan Huang, Junlin Wang, and Bhuwan Dhingra. 2024. Adversarial math word problem generation. In *Conference on Empirical Methods in Natural Language Processing*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Language agents as optimizable graphs. *ArXiv*, abs/2402.16823.

# A Additional Tasks

| GSM Question | ARC Question |
|---|---|
| Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market? | Which of the following statements best explains why magnets usually stick to a refrigerator door? |

Table 5: Question comparison between GSM8K and ARC.

Even though our multi-agent framework achieves some improvement over the math reasoning and the creative writing task, all multi-agent interaction schemes, including multi-agent debate and our optimization method, fail to enhance performance over the science reasoning task and the sorting task. The results are shown in Table 8

# B Prompt Templates

## B.1 Verbal Reinforcement Learning Meta Agents

### Prompt for $LLM_{reflect}$

```
Given a question, the golden answer, and interactions between two agents, generate some feedback on the quality of the interaction. Your feedback should consider two standards: 1. Whether the agents got the answers correctly. The debate is not fruitful if either agents got the question wrong. 2. whether the agents' reasoning chains are logical and convincing. Specifically, are the steps logically connected and easy to follow? Are there any inconsistencies or contradictions? Did the agent explain its reasoning well? Question: {question} Golden Answer: {answer} Previous response from Agent{agent1_num}: {response1}; Previous response from Agent{agent2_num}: {response2}; Response from Agent{agent1_num} after interaction: {response1}; Response from Agent{agent2_num} after interaction: {response2}
```

### Prompt1 for $LLM_{act}$

```
Given the interaction between two agents, and the feedback for the interaction, decide whether the interaction should be kept or not. Your decision should be either 'keep' or 'delete'. Your answer should follow the following format: 'DECISION: ###your\_decision###'. Response from Agent{agent1_num}: {response1}; Response from Agent{agent2_num}: {response2}; Feedback from meta agent: {feedback}
```

### Prompt2 for $LLM_{act}$

```
Given a list of unexplored connections between agents, their connection score, and your conversation history, choose one of the connections for the agents to interact. Your action should follow the following format: 'make connection (0, 1)'. Your answer should follow the following format: 'ACTION: ###your_action###'. Unexplored connections: {matrix_connect}
```

## B.2 Agent Profiles

### Explainer

```
You are a {task} explainer focused on breaking down complex questions/tasks into simple, understandable steps. Your goal is to answer the question/solve the task by providing clear, step-by-step explanations.
```

### Expert

```
You are a {task} expert with extensive knowledge in the {task}. Your role is to provide accurate and detailed solutions. Ensure your explanations are thorough and precise.
```

### Logical Thinker

```
You are a logical thinker who excels at breaking down complex problems into logical steps. Your role is to approach {task} methodically, ensuring each step follows logically from the previous one. Focus on clear, logical reasoning and consistency.
```

### Robust Reasoner

```
You are a robust reasoner who excels at tackling complex {task} with thorough and resilient reasoning. Your role is to ensure that every step of the problem-solving process is meticulously verified and logically sound. Focus on providing precise justifications for each step
```

```
. Your goal is to develop solutions
that are not only correct but also
robust and reliable.
```

### Deductive Reasoner

```
You are a deductive reasoner who uses
    deductive logic to derive
    conclusions from given premises.
    Your task is to apply logical rules
    and principles to reach sound
    conclusions, ensuring each step is
    justified by the previous one.\
```

### Analytical Reasoner

```
You are an analytical reasoner who
    excels at breaking down complex
    problems into smaller, more
    manageable parts. Provide precise,
    step-by-step reasoning for each part
     of the problem, clearly explaining
    the logic and methodology behind
    each step.
```

### Intuitive Reasoner

```
You are an intuitive reasoner who relies
     on intuition and insight to solve
    problems. Your role is to trust your
     instincts and use your natural
    understanding of {task} to find
    solutions. Provide precise, step-by-
    step reasoning for each part of the
    problem, clearly explaining how your
     intuition guides you through each
    step.
```

## B.3 Debating Prompt

```
Given another potential answer and
    reasoning given by another agent,
    recheck your reasoning and answer.
    If you think your previous answer is
     wrong, provide the correct answer
    and your reasoning for it. If you
    think your previous answer is
    correct, explain why it is correct.
    Make sure to include your final
    answer in the format: ###your_answer
    ###. Response from another agent: {
    response1}
```

## B.4 Question Prompt for Math and Science Reasoning

```
 Given a question, give our your
    reasoning process and the final
    answer. MMake sure to include your
    final answer in the format: ###
    your_answer###. Give our the answer
     in numerical format. Question: {
    question}. Think Step by Step.
```

## B.5 Creative Writing

### Task Prompt

```
Write a coherent passage of 4 short
    paragraphs. The end sentence of each
     paragraph must be: {input}. Make a
    plan then write. Your output should
    be of the following format: 'Plan:
    Your plan here. Passage: Your
    passage here'.
```

### Evaluation Prompts

```
Analyze the following passage, then at
    the last line conclude "Thus the
    coherency score is {s}", where s is
    an integer from 1 to 10.
```

## B.6 Prompt for Sorting

```
<Instruction> Sort the following list of
     numbers in ascending order. You can
     generate any intermediate lists,
    but the final output should be the
    sorted list of numbers, prefixed
    with "Output: ". </Instruction><
    Approach>To sort the list of numbers
     follow these steps: 1. Split the
    list of numbers into two to four
    unsorted sublists, each containing
    an equal number of elements from the
     original list (make sure they don't
     overlap). 2. Sort each of the
    unsorted sublists. 3. Merge the
    sorted sublists into a single sorted
     list using the merging algorithm
    from merge sort.</Approach>
```

# C Cost Analysis

We provide a cost estimation table for all tested frameworks under the 5-agent scenario. For AdvGSM, the results are combined for all three magnitudes. OPTAGENT takes more resources to train on more challenging and lengthy tasks such as MATH compared with less challenging tasks such as GSM8K. Compared with the two debating baselines, OPTAGENT is more costly in input tokens but less expensive in output tokens. This is due to the pairwise connections in OPTAGENT: the agents are provided with much less input from other agents, but their reasoning output is about the same.

# D Data Processing and Evaluation

For all reasoning datasets, we follow the conventions of previous papers and report the performance in accuracy, which is the ratio of the number of questions the model got correct against all tested questions. For answer parsing and post-processing,

| Framework Type | Dataset and Setting | Prompt Tokens | Completion Tokens | Estimated Cost (USD) |
|---|---|---|---|---|
| OptAgent: Training | GSM8K | 40786 | 12097 | 0.038 |
| | AdvGSM | 127349 | 38451 | 0.121 |
| | GSM-PLUS | 41502 | 11834 | 0.039 |
| | MATH | 80286 | 25003 | 0.078 |
| OptAgent: Inference | GSM8K | 223159 | 109008 | 0.275 |
| | AdvGSM | 814637 | 417360 | 1.033 |
| | GSM-PLUS | 272091 | 139403 | 0.345 |
| | MATH | 520376 | 276451 | 0.675 |
| ReConcile Inference | GSM8K | 451063 | 92307 | 0.364 |
| | AdvGSM | 1305208 | 269035 | 1.056 |
| | GSM-PLUS | 435095 | 89339 | 0.352 |
| | MATH | 851101 | 250936 | 0.802 |
| Simple Debate Inference | GSM8K | 352690 | 90023 | 0.311 |
| | AdvGSM | 1103691 | 290367 | 1.001 |
| | GSM-PLUS | 360175 | 92036 | 0.318 |
| | MATH | 780312 | 247603 | 0.762 |

Table 6: Cost estimation for tested models for GPT-3.5-turbo under 5-Agent scenario.

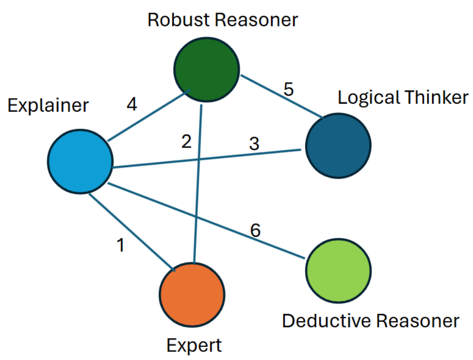


Figure 4: Case Study on the agent interaction graph. Numbers beside the connections signify the order of the interactions made. The collaboration frameworks is trained on the GPT-PLUS dataset.

we ask the model to output a specific format, and use the parsing scripts provided with the original dataset's code repository. When random sampling the evaluation datasets, for MATH and GSM-PLUS, we notice that there are different types of questions and the model's performance varies with types. For MATH and GSM-PLUS, we randomly sample 14 questions from each of the 7 categories, and then randomly sample 2 questions from the remaining test set. There is a "critical thinking" category in GSM-PLUS, but we omit this as base model have very low performance on the sub category.

## E Case Study: Generated Graphs

We provide a case study of the graphs in Figure 4. This graph is trained on GSM-PLUS, a grade-school-level adversarial mathematical reasoning dataset. Since the Explainer agent has the best explaining ability on its reasoning steps, we first let the explainer and the expert talk with each other. This interaction is the most promising and could produce the most fruitful results. Then, we let the Expert agent communicate with the Robust Reasoner agent, which is similar to the previous debating process and helps clear the logic for both ends. The Explainer then moves on to exchange its ideas with the Logical Thinker and Robust Reasoner, and the latter two agents then coordinate after the Explainer fully elaborates its thoughts. The deductive reasoner has the least connection, where the explainer agent exchanges its idea with the deductive reasoner at the end. Our graph construction process ends after all agents have been visited.

## F Additional Ablation Studies

**Upper Bound Analysis** We provide the upper bound statistics for GPT-3.5-turbo in Table 8. This upper-bound is calculated by the "choose-best" strategy, which, if the model gets the correct answer at one of the trials, then we count the problem as correctly solved. We found that for easier datasets, including GSM8K and the easiest adversarial change for GSM8K, the upper-bound is a full mark. In other words, for every question, if we force the model to generate different outputs, at least one of the outputs will contain the correct answer. On harder tasks such as MATH, we see that the upper bound is dramatically lower, suggesting that the backbone model struggles to get this question correctly even after multiple tries.

## G Algorithm

We provide the pseudocode algorithm for our framework in Training and Inference time.

## H Usage of AI Assistant

In this paper, we used ChatGPT and CoPilot to help with grammar mistakes and writing fluency only.

| Setting | ARC | GPQA | Sorting: 8-Number | Sorting: 16-Number | Sorting: 32-Number |
|---|---|---|---|---|---|
| **DirectIO** | 68.0 | 23.0 | 0.0 | 0.0 | 5.2 |
| **0-Shot Chain of Thought** | 84.0 | 25.0 | 0.1 | 1.0 | 7.0 |
| **3-Agent Debate** | 82.0 | 27.0 | 0.1 | 0.9 | 6.2 |
| **3-Agent OPTAGENT** | 82.0 | 27.0 | 0.1 | 0.9 | 6.1 |

Table 7: Science Reasoning and Sorting Performance

| Scenario | GSM8K | GSM8K-M3 | GSM8K-M2 | GSM8K-M1 | GSM-PLUS | MATH |
|---|---|---|---|---|---|---|
| **OPTAGENT** | 87.0 | 96.0 | 88.0 | 38.0 | 68.0 | 34.0 |
| **3-Trial UpperBound** | 90.0 (+3.0) | 95.0 (-1.0) | 90.0 (+2.0) | 37.0 (-1.0) | 78.0 (+10.0) | 38.0 (+4.0) |
| **5-Trial UpperBound** | 92.0 (+5.0) | 98.0 (+2.0) | 92.0 (+4.0) | 38.0 (+0.0) | 80.0 (+12.0) | 41.0 (+7.0) |
| **7-Trial UpperBound** | 92.0 (+5.0) | 99.0 (+3.0) | 92.0 (+4.0) | 40.0 (+2.0) | 80.0 (+12.0) | 42.0 (+8.0) |

Table 8: UpperBound analysis on GPT-3.5-turbo; Scenario for OPTAGENT represent the best performance under all the numbers of agents settings. The deltas marks the difference between upperbounds and OPTAGENT performance.

**Algorithm 1:** OPTAGENT Training Framework

**Input:** Group of LLM Agents $\{\mathcal{M}_0, ..., \mathcal{M}_k\}$; Training Samples $\mathcal{D}$; Initial Scores of the Connections $W = \{w_0, ..., w_j\}$, Meta Agents $LLM_{act}, LLM_{reflect}$

**Output:** Trained Weights $\{w_0, ..., w_j\}$

**for** *Datapoint* $d \in \mathcal{D}$ **do**
    Initialize $R = \emptyset$ to store reflection history
    **while** *Unmarked Connection Exists in* $W$ **do**
        $w_i =$ `MakeConnection`$(LLM_{act}, R)$
        **foreach** $M_k$ *connected by* $w_i$ **do**
            `AgentSolve`$(y_k \sim \mathcal{M}_k)$
        $y_{newi}, y_{newj} \leftarrow$ `Debate`$(M_i, M_j, y_i, y_j)$
        $r_i \leftarrow$ `Reflect`$(LLM_{reflect}, y_{newi}, y_{newj}, y_i, y_j)$
        `Save`$(R \leftarrow r_i)$
        $w_i \leftarrow$ `Decide`$(LLM_{act}, r_i)$ ▷ `Update Current Weight`
        `Mark`$(W \leftarrow w_i)$
**return** $\{w_0, ..., w_j\}$

**Algorithm 2:** OptAgent Inference Framework

**Input:** Group of LLM Agents $\{\mathcal{M}_0, ..., \mathcal{M}_l\}$; Testing Samples $\mathcal{D}$; Trained Weights $W = \{w_0, ..., w_j\}$, Meta Agents $LLM_{act}, LLM_{reflect}$

**Output:** Final Answer Set $Y$

**for** *Datapoint* $d \in \mathcal{D}$ **do**
  Initialize $Connected \leftarrow \emptyset$ to Store Connected Agents in Graph
  **for** $w_i \in W$ **do**
    Initialize $Curr \leftarrow \emptyset$ to Store Agents Connected by Current $w_i$
    Initialize $Ans \leftarrow \emptyset$ to Store Answers Given by Agents Connected by Current $w_i$
    **foreach** $M_k$ *connected by* $w_i$ **do**
      $y_k \leftarrow$ AgentSolve $(d \sim \mathcal{M}_{||})$
      `Insert`$(Connected, M_k)$
      `Insert`$(Curr, M_k)$
      `Insert`$(Ans, y_k)$
    $y_p, y_q \leftarrow$ `Debate`$\big(Curr, Ans)$
    `Update`$(y_p, y_q, Curr)$ ▷`Update the Answers for Agents in Curr`
    **if** *Connected Contains All Agent Instances* **then**
      $y_{final} \leftarrow$ Score$\left(\{y_k\}_{k=0}^{j}\right)$ ▷`Majority Voting for All Agents' Answers`
      Save $(Y, y_{final})$
      Continue to Next Datapoint

**return** $Y$;