GAS AI: A Graph-Based Asynchronous Framework for AI Agents Systems

Soham Bafana

February 15, 2024

Abstract

Recent advancements in Large Language Models (LLMs) have enabled generative artificial intelligence to help with reasoning, research, and coding. Building upon human and animal interactions, researchers have been developing multi-agent systems (MAS) that leverage the strength of collaborative artificial intelligence roles in a team. This study introduces the Graph Asynchronous Swarm ("GAS") AI framework that integrates LLMs with agents to complete tasks without human intervention. These tasks include querying the internet, coding documents, and producing graphs. Initial comparisons to existing AI tools indicate that GAS AI obtains more sources to get different perspectives, a similar rating on content, and an efficient user experience. The GAS AI Framework also has a Graphical User Interface ("GUI") that enables a more straightforward way to define tasks and roles amongst the multiple AI agents. Further development and testing of the GAS AI Framework is needed through open-source collaboration with other AI developers; with this additional investment and collaboration, GAS AI could improve productivity and reduce bias in generative AI.

1 Introduction

Large Language Models (LLMs) have marked a significant milestone in the evolution of Artificial Intelligence systems in the domain of natural language processing (NLP). These models are powered by transformer architecture [Vaswani et al., 2023] and demonstrate remarkable capabilities in understanding, generating and reasoning with natural language [Webb et al., 2023, Huang and Chang, 2023]. Recent research has found that LLMs are more valuable than their original purpose as text generators as they can perform complex reasoning tasks [Webb et al., 2023, Huang and Chang, 2023] .These tasks include inferring context, drawing conclusions, and even generating explanations for their reasoning processes [Yao et al., 2023, Xi et al., 2023]. LLM reasoning plays a critical role in the development of agents which are aimed at multi-step tasks. These can include the development and testing of codebases, large scale research, database querying with text and more. Given the complexity of these tasks, an effective strategy to enhance agent capabilities is through the deployment of multi-agent systems. Multi-agent systems are composed of multiple autonomous agents that interact with each other to achieve individual or collective goals.

Multi-agent systems offer several advantages over single-agent systems. For example, reinforcement learning in multi-agent systems shows that collaboration among agents could solve more complex tasks in less time [Wang and Cai, 2015]. Additionally, trust and coordination mechanisms of multi agent systems[Aref, 2018, Wei, 2015] can surpass the ability of single agent systems.

The development of multi-agent systems (MAS) is increasingly important for two reasons. First, multi-agent systems can help us understand multi-human systems. By modeling human behaviors as agents within a MAS, researchers can gain insights into how individuals within a group may interact, make decisions, and solve problems collectively[Cai et al., 2017]. This understanding is crucial for applications such as online disaster response, target surveillance, and social structure modeling[De Mooij et al., 2023, Sharma et al., 2018]. Second, the future of AI may possibly be large scale multi-agent systems, a form of autonomous agent swarm. Swarm intelligence refers to the collective behavior of decentralized, self-organized systems, whether natural or artificial. It is inspired by the behavior of social creatures such as birds, bees, and ants, which demonstrate that a

group can often outperform individual members when solving problems and making decisions. In the context of AI, autonomous swarm-based intelligence could lead to the development of systems where multiple AI agents work together to perform tasks more efficiently than a single agent could. This approach is being explored for various applications, including controlling unmanned vehicles, planetary mapping, and even medical applications like locating tumors.

Recent studies highlight the benefits of multi-agent systems in LLMs, including the encouragement of "thinking outside of the box" [Liang et al., 2023], the improvement of reasoning accuracy [Du et al., 2023], and task validation mechanisms (checking if a task has been completed) [Wu et al., 2023]. In light of these advantages, an important question arises: How can we construct LLM applications that harness the potential of multi-agent systems to tackle complex challenges effectively?

The introduction of the GAS AI (Graph Asynchronous Swarm, Artificial Intelligence) framework aims to answer this question. This innovative approach is designed to enhance the capabilities of AI agents by enabling them to work in unison towards achieving complex, collective goals, much like a swarm.

2 GAS AI Framework

GAS AI stands for **Graph**, **Asynchronous**, **Swarm** Artificial Intelligence.

2.1 Graph

The Graph-Based Communication system is the cornerstone of GAS AI System. A Graph is a data structure consisting of nodes and edges, where nodes are objects and edges are the pathways between those objects. Nodes can represent agents, tools, or other functional entities within the system. Each node has a `get_completion()` function that returns a set of messages that detail the node's execution. For example, a Research Agent may return their research function calls and a final message. In the communication network, connections between nodes (directed edges) represent communication pathways. Each edge starts at a particular node and ends at a particular node. Once a node has completed a `get_completion()` request, it will send its final message through the communication pathway to the recipient node. Then, that node will start a `get_completion()` request. Altogether, nodes and edges make up the communication system for GAS AI.

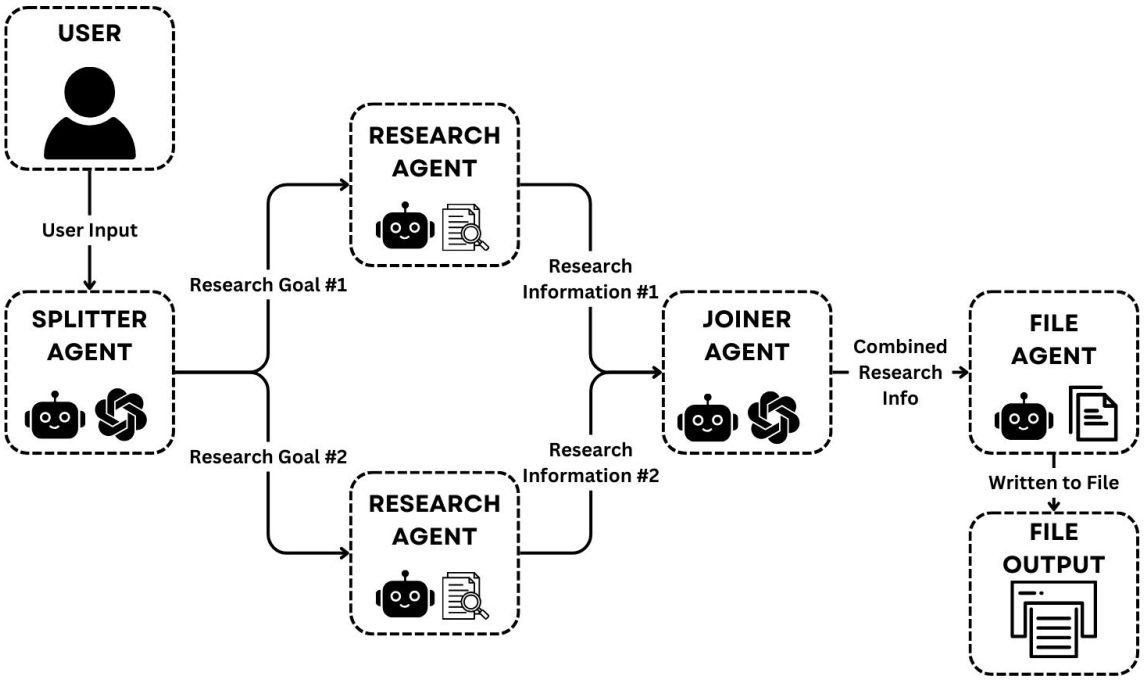


Figure 1: Basic GAS AI Agent Graph

Consider a scenario depicted in Figure 1, illustrating a basic GAS AI agent network. In this setup, a user (node) initiates communication by sending a message through a communication pathway to a splitter agent (node). This agent, in turn, delegates tasks to various research agents (nodes), which subsequently forward their findings to a joiner agent (node). The joiner agent consolidates the information and passes it to a file agent (node) responsible for compiling the research into a comprehensive report. Each agent calls the `get_completion()` function, showing the user their thought process and sending the final message to the next agent. This streamlined communication is made possible through the graph-based connections established between the user and agent nodes.

For enhanced user interaction and system monitoring, GAS AI incorporates a visual GUI. Here, users can build graphs, nodes and edges through a drag and drop solution. Then, a user can connect nodes of interest and build their own communication graph. By facilitating easy multi-agent systems through a user-friendly interface, GAS AI lowers the technical barriers for creating and managing complex agent networks.

2.2 Asynchronous

A defining feature of GAS AI is its emphasis on Asynchronous AI Agents, allowing for parallel concurrency and efficient task management within the multi-agent environment.

In GAS AI, the `get_completion()` function is designed to operate using asynchronous generators. Instead of delivering a single response upon function call, it yields an object that continually supplies new responses as they become available. This feature enables GAS AI to continuously update users with the agent's actions and reasoning processes. Additionally, it facilitates a unique approach to managing and interacting with tools, allowing for dynamic and real-time information exchange. Tools are external functionalities or services that agents can invoke to accomplish specific tasks. Within the GAS AI framework, tools extend the capabilities of agents, providing them with access to specialized processing, data analysis, or other operations necessary for completing their assignments .For example, the Splitter Agent can call the `split_tasks` tool, which takes a list of goals that need to be accomplished and sends the goals to the nodes connected to the outward facing communication pathways. Each task can be run simultaneously, managed and eventually completed.

2.3 Swarm

The ultimate goal of GAS AI is to build a swarm, or a set of autonomous agents that can work and develop themselves. Swarms are collaborative and innovative, using all the resources at their disposal to solve a problem. To model the potential of a swarm, we introduce the idea of Chats or Nodes in Nodes. Chats are specialized nodes composed of multiple agents working collaboratively. This setup allows groups of agents to engage in internal dialogues, pooling their resources and intelligence to arrive at collective decisions or outputs, thereby enhancing the system's problem-solving capabilities. A chat is a node, so it maintains a `get_completion()` method. When it is called, the Decider Agent receives the message and communicates with the other agents in the chat. The Decider Agent chooses when the other agents should speak. It conveys previous attempts and information and eventually decides when the goal has been completed. Once the goal has been completed, the completion has been finalized, and the resulting message is passed onto the next agent. Thanks to the use of asynchronous generators, one can view the inner workings of the chat module. Chats are powerful as they allow agents to specialize in roles but they maintain versatility in solving complex problems.

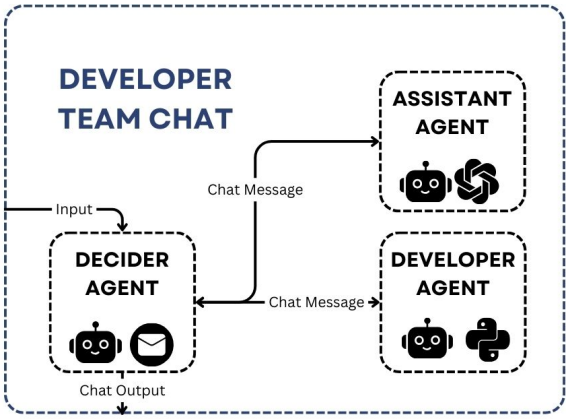


Figure 2: Example of a Developer Chat (Node in Nodes)

Figure 2 showcases the functionality of a Chat node within the system. In this setup, input is initially directed to a 'decider agent', who serves as a coordinator among the various agents. This decider agent possesses a comprehensive understanding of each agent's capabilities and how they can interact with the instructions provided. Through this coordination, the decider agent facilitates communication between the agents, overseeing the dialogue to determine if the predefined goal has been met. If the objective is achieved, the decider agent forwards the output accordingly; otherwise, the discussion among the agents persists until the desired outcome is realized, illustrating the Chat node's dynamic and goal-oriented operation.

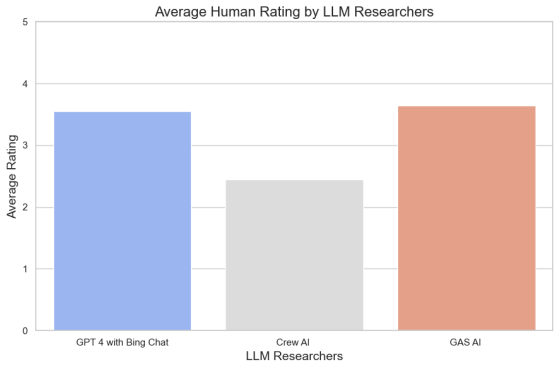
By integrating these advanced features, GAS AI not only addresses the complexities inherent in managing asynchronous operations and distributed systems but also provides a robust, scalable framework for building intelligent, highly concurrent applications.

3 GAS AI Applications

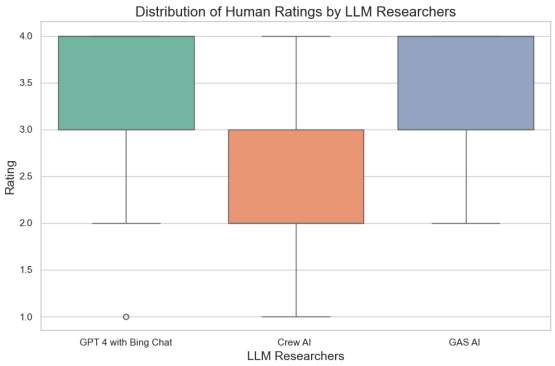
GAS AI can be used for many different purposes, ranging from research, software development, database management, email management and more. Its strength lies in its ability to integrate with any external API which makes it an effective tool for various tasks. Here are a few examples of GAS AI use cases:

3.1 Research

To gauge GAS AI's research accuracy, I employ the ANTIQUE dataset [Hashemi et al., 2019] developed from non-factoid question answering banks. I randomly sampled 15 different questions, and planted 15 of our own questions and tested GAS AI against two cutting edge LLM Researchers: GPT 4.0 with Bing Chat, and LangChain based Crew AI. For each of the 30 samples per agent, I measured the amount of sources used. I also measured an average human rating, which is a rating on a scale of 1-4 measured by 3 judges. On this scale, 1 means research question not answered, 2 means research question partially answered, 3 means research question answered, 4 means research question answered and more information provided. A simplistic prompt was used for all 3 systems: "Research the following question {Insert Question}. Provide sources."

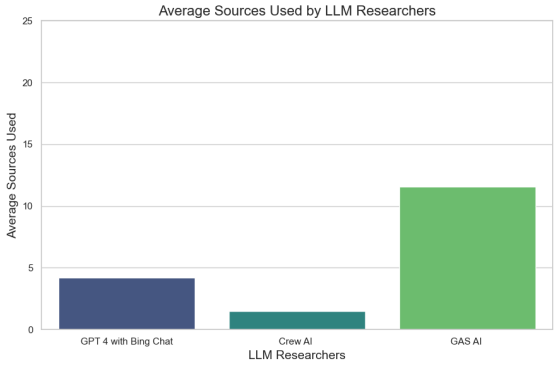


(a) Average Human Rating

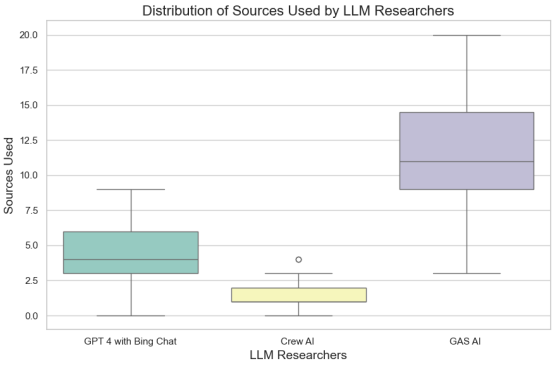


(b) Distribution of Human Ratings

Figure 3: Human Ratings of Research Tasks



(a) Average Sources Used



(b) Distribution of Sources Used

Figure 4: Source Usage of Research Tasks

GAS AI exhibits more sources used than GPT 4 with Bing Chat which could demonstrate lead to less bias in research. Gas AI exhibits an equivalent Average Human Rating as GPT4 with Bing Chat and a higher rating than Crew AI-Langchain. These metrics indicate that GAS AI achieves high accuracy and provides holistic research answers. The figure below shows a GAS AI research operation.

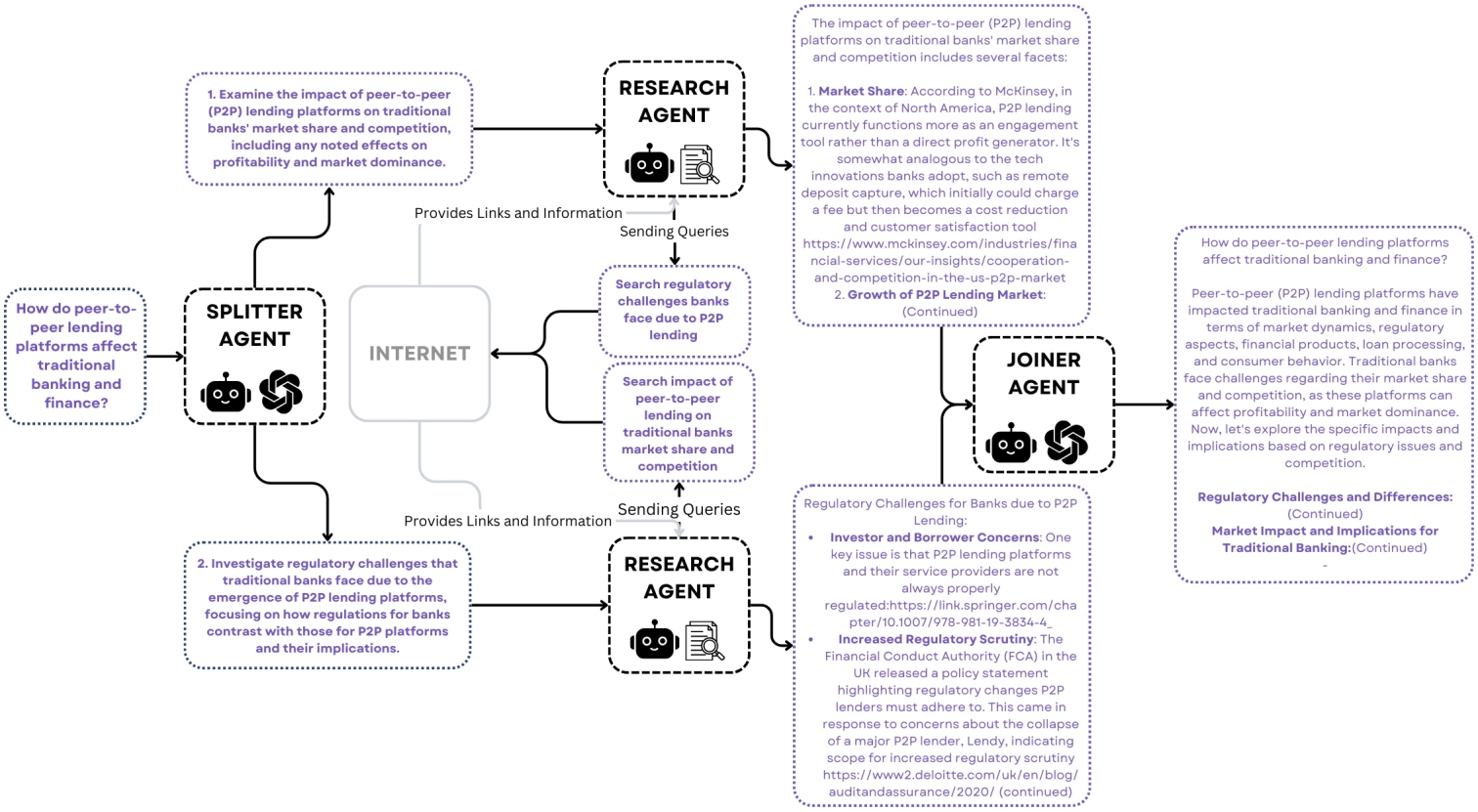


Figure 5: Example of a Research Chat in the Research Comparison Task

Figure 5 showcases the operational mechanism of the GAS AI system, illustrating its process from initiation to completion. Initially, the system employs a 'splitter' function to deconstruct a user's query into distinct subgoals. This feature allows for the engagement of multiple agents, beyond the two illustrated here, to tackle separate subgoals simultaneously. The strength of this approach lies in its ability to delegate specific research tasks to different agents, each dedicated to exploring a part of the question. Upon completing their investigations, these agents compile their findings into reports, which are then forwarded to a 'joiner' agent. The role of the joiner agent is to synthesize these individual reports into a cohesive and highly detailed summary. This method enhances the research process by ensuring a thorough examination of each subtopic, thereby enriching the overall understanding of the main topic through detailed exploration of its components.

3.2 Research and Development Team

GAS AI's unique parallel concurrency allows for development and research to run at the same time. To demonstrate, I provide a case study of stock analysis and research. I use the following GAS AI infrastructure.

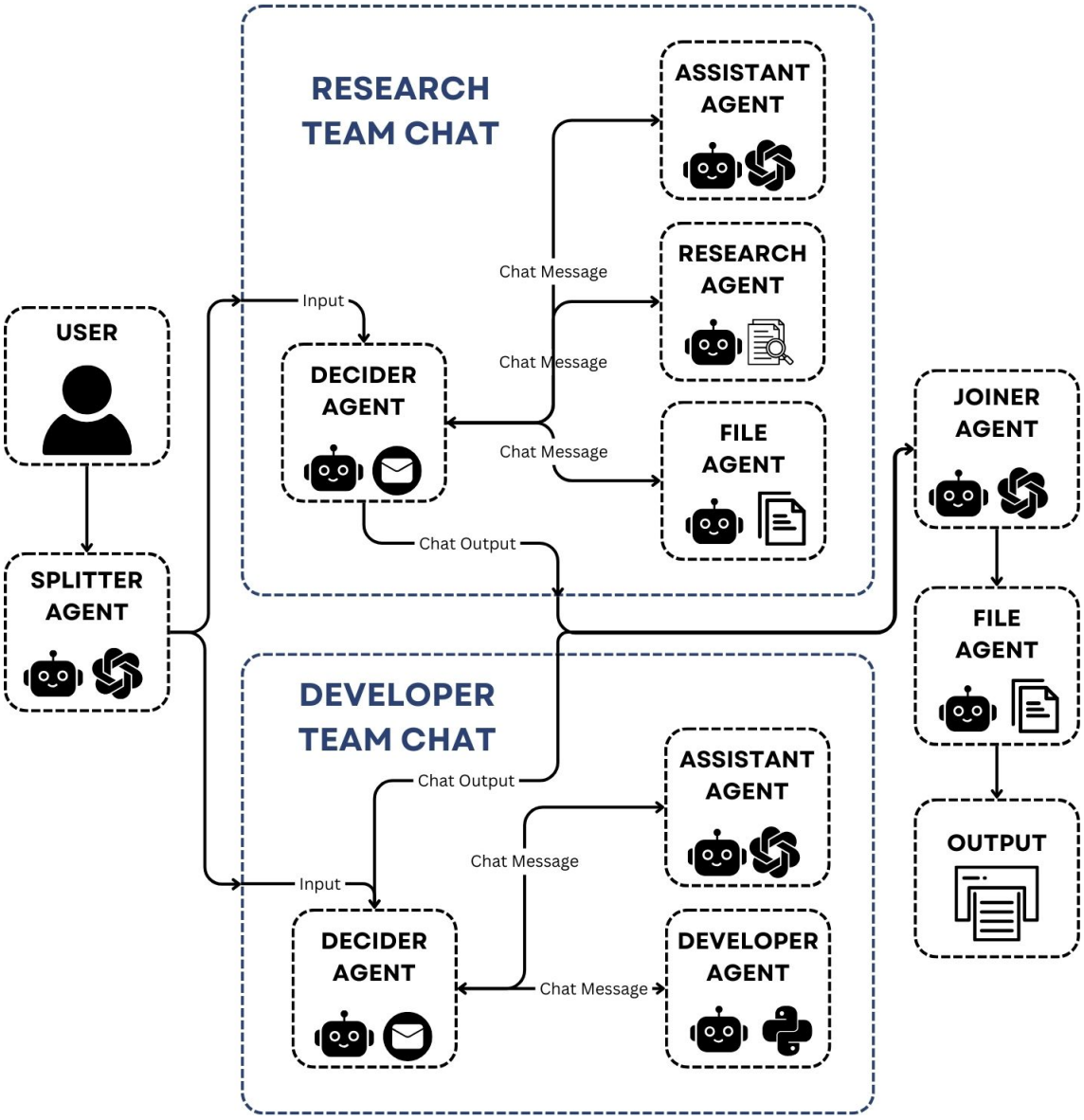


Figure 6: Research and Development Demo Setup

Figure 6 demonstrates a combined research and development team, where all the research tasks are delegated to the research team and the programming is delegated to the development team. To test this paradigm, I provide the prompt: "Provide me with information on META's stock price increase since Jan 1st 2024, and provide me with the reason that that price increase occurred. Write a report detailing your findings". Below is a table demonstrating the chat history of the schema as it creates a solution to the problem. I, the user, provided no insight or aid to the program. Each action and decision are entirely its own.

Speaker	Message
User	Provide me with information on META's stock price increase since Jan 1st 2024 and analyze the numbers, and provide me with the reason that that price increase occurred. Write a report detailing your findings

Speaker	Message
Splitter	**The task has been successfully divided between the Research Team and the Dev Team:** • **Research Team:** Tasked with providing detailed information on META's stock price increase since January 1st, 2024. They are to perform an analysis of the numbers and research the reasons behind the price increase. Furthermore, they are responsible for compiling a comprehensive report detailing the findings, which must include links to sources and references to support the data presented. • **Dev Team:** Role is to develop the necessary tools for extracting and analyzing META's stock price data since January 1st, 2024. The team needs to ensure that the tools can visualize the data to aid in better understanding and presentation within the compiled report. Additionally, they must make sure that the tools are compatible with any existing code provided by the user, following any specified code structures and requirements. The outputs from both teams will come together to result in a full report for the user.
Research Team Decider Agent	**Inner Thoughts:** First, we need to research the current and past stock prices of META to understand the increase since January 1st, 2024. After getting the numbers, an analysis of the stock price increase must be conducted. Finally, researching the reasons behind the stock price increase will allow us to compile a comprehensive report with findings that include links and sources. **Message to Research Agent:**Provide detailed information on METAś stock price increase since January 1st, 2024, along with an analysis of the numbers. Research the reasons behind the price increase and compile a comprehensive report detailing your findings, including links and sources.
Dev Team Decider Agent	**Inner Thoughts:** To start with the development of tools for extracting, analyzing, and visualizing META's stock price data, the Developer Agent should be engaged first. After the development is complete, the Test Agent will be responsible for testing the new tools. **Message to Developer Agent:**Develop tools necessary for extracting and analyzing META's stock price data since January 1st, 2024. Ensure the tools can visualize the data for better understanding and presentation in the report
Research Agent	`Function(arguments='{"phrase":"META stock price since January 1st, 2024"}', name='SearchWebDDGS'): [{'title': 'Meta Platforms, Inc. (META) Stock Historical Prices and Data - Yahoo Finance', 'href': 'https://finance.yahoo.com/quote/META/history/', 'body': 'Nasdaq ...'}]`
Research Agent	`Function(arguments='{"phrase":"reasons behind META stock price increase 2024"}', name='SearchWebDDGS'): [{'title': 'Why Meta Stock Tops Amazon As A 2024 Generative AI Play - Forbes', 'href': 'https://www.forbes.com/sites/petercohan/2024/02/02/why-meta-stock-tops-amazons-as-a-2024-generative-ai-play/', 'body': 'Amazon: +77,..."}]`

Speaker	Message
Research Agent	The information I found indicates that Meta Platforms, Inc. (META) experienced a significant stock price increase since January 1st, 2024. • According to Yahoo Finance, as of the date of the search, the most recent data shows that Nasdaq, where META is listed, has had a positive performance Yahoo Finance. • A forecast from Stockscan.io suggested an average analyst price target of $481.42 for META's stock, indicating a potential increase of +21.95% from a then-current price of $394.78. They also note the highest analyst price target at $628.47 and the lowest at $334.37. This indicates that analysts were generally optimistic about the stock's prospects Stockscan.io. • Macrotrends detailed how META's all-time high closing price was $401.02 on January 29, 2024. At the time of their reporting, the 52-week high stock price was $406.36, which is 2.9% above their current share price then. The 52-week low was much lower at $167.66, showing a strong recovery and uptrend over the previous year Macrotrends. For the reasons behind the stock price increase, several pieces of information were found: • A Forbes article highlighted META as a leading generative AI play over Amazon in 2024, suggesting that the company's focus on AI has contributed to its strong performance in the stock market Forbes. • Reuters reported that META shares jumped significantly after the company announced its first-ever dividend plan and that its 'Year of Efficiency' paid off, with headcount and costs dropping. This efficiency drive seems to have been well-received by investors, leading to a surge in stock market value Reuters. • The New York Times covers a surge in Meta's stock following a jump in profits, suggesting that the company's financial performance was a key driver behind the price increase The New York Times.
File Agent	I have created the requested markdown file titled Meta Stock Performance 2024 with the overview of Meta's stock performance and it is located in the Markdown Files directory. If you need the content of this file or any other assistance, feel free to let me know. (Function call was excluded for simplicity sake)
Research Team Decider Agent	The comprehensive report detailing META's stock price increase since January 1st, 2024, has been successfully compiled and is located in a markdown file titled `"META_Stock_Performance_2024.md"` within the `"Markdown Files"` directory. If you need to view the content of this file or require further assistance, please let me know.
Dev Team Decider Agent	**Thought Process:** Since the user has not provided any specific files, the Developer Agent will have to create new tools from scratch while following best practices in Python programming. This will increase the likelihood that the tools will be compatible with most Python codebases. **Message to Developer Agent**: Develop tools necessary for extracting and analyzing METAś stock price data since January 1st, 2024. Ensure the tools can visualize the data for better understanding and presentation in the report. The tools should be compatible with any general Python codebase, considering no specific files were provided by the user.

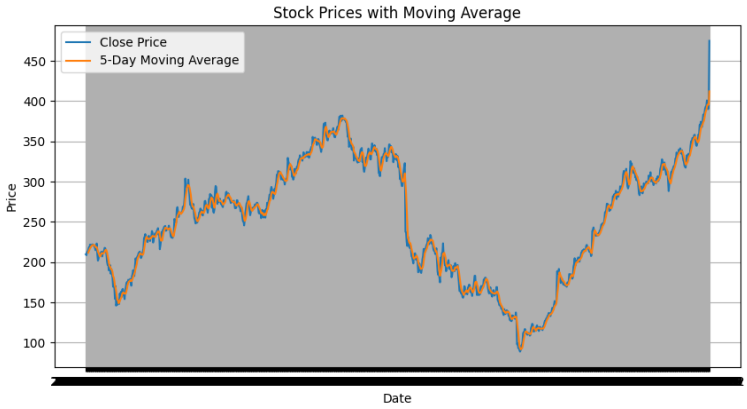
Speaker	Message
Developer Agent	Function(arguments='{"file_name": "workspace/fetch_data.py", "body": "import yfinance as yf\\nimport pandas as pd\\n\\n\\ndef fetch_stock_data(ticker, start_date):\\n # Fetch data from Yahoo Finance API\\n data = yf.download(ticker, start=start_date)\\n return data\\n\\n\\ndef save_to_csv(data, file_name):\\n # Save data to a csv file\\n data.to_csv(file_name)\\n\\n\\ndef save_to_excel(data, file_name):\\n # Save data to an excel file\\n data.to_excel(file_name)\\n"}', name='CreateFile')
Developer Agent	Function(arguments='{"file_name": "workspace/analyze_data.py", "body": "import pandas as pd\\n\\n\\ndef load_data(file_name):\\n # Load data from csv or excel file\\n return pd.read_csv(file_name) if file_name.endswith(\'.csv\') else pd.read_excel(file_name)\\n\\n\\ndef calculate_daily_change(data):\\n # Calculate daily price change\\n data[\'Daily_Change\'] = data[\'Close\'].diff()\\n return data\\n\\n\\ndef calculate_moving_average(data, window_size=5):\\n # Calculate moving average\\n data[\'Moving_Average\'] = data[\'Close\'].rolling(window=window_size).mean()\\n return data\\n"}', name='CreateFile')
Developer Agent	{Function(arguments='{"file_name": "workspace/visualize_data.py", "body": "import matplotlib.pyplot as plt\\n\\n\\ndef plot_stock_prices(data, title=\'Stock Prices Over Time\'):\\n # Plot stock prices\\n plt.figure(figsize=(10, 5))\\n plt.plot(data[\'Date\'], data[\'Close\'], label=\'Close Price\')\\n plt.title(title)\\n plt.xlabel(\'Date\')\\n plt.ylabel(\'Price\')\\n plt.legend()\\n plt.grid(True)\\n plt.show()\\n\\n\\ndef plot_moving_average(data, window_size=5):\\n # Plot moving average\\n plt.figure(figsize=(10, 5))\\n plt.plot(data[\'Date\'], data[\'Close\'], label=\'Close Price\')\\n plt.plot(data[\'Date\'], data[\'Moving_Average\'], label=f\'{window_size}-Day Moving Average\')\\n plt.title(\'Stock Prices with Moving Average\')\\n plt.xlabel(\'Date\')\\n plt.ylabel(\'Price\')\\n plt.legend()\\n plt.grid(True)\\n plt.show()\\n"}', name='CreateFile') : File written to workspace/visualize_data.py}

Speaker	Message
Developer Agent	The tools necessary for extracting and analyzing META's stock price data since January 1st, 2024, have been structured into three Python scripts: 1. `fetch_data.py`: This script will use the `yfinance` library to fetch stock price data for META from Yahoo Finance starting from the specified date (January 1st, 2024) and includes functions to save the data to CSV or Excel format. 2. `analyze_data.py`: This script contains functions to load the data from CSV or Excel files and perform basic analysis like calculating daily price changes and moving averages using pandas. 3. `visualize_data.py`: This script provides functions for visualizing the stock price data using matplotlib, including plotting stock prices over time and displaying moving averages. These scripts have been saved in the workspace directory and are designed to be compatible with a general Python codebase. Each script is modular, making it easy to integrate into other code or to run as standalone tools.You can now utilize these tools by calling functions from the respective scripts to extract, analyze, and visualize META's stock price data. Would you like me to proceed with a test run of these tools to fetch, analyze, and visualize the stock data for the specified period?
Dev Team Decider Agent	**Thought Process:** The Developer Agent has created the necessary scripts for extracting, analyzing, and visualizing stock data. To verify that the goal has been achieved, the next logical step is to test these tools. Therefore, we should engage the Test Agent to perform a test run of the tools which will fetch, analyze, and visualize METAś stock price data since January 1st, 2024, to ensure they work as expected. **Sent to Test Agent:** The Developer Agent has developed the necessary tools for extracting, analyzing, and visualizing stock price data. Please perform a test run to verify that the tools work correctly and that the data is accurately fetched, analyzed, and visualized according to the requirements.

Test Agent	`Function(arguments='{"command":"pip install yfinance pandas matplotlib "}', name='ExecuteCommand'):`
	`Installing collected packages: yfinance`
	`Successfully installed yfinance-0.2.36`

Test Agent	`Function(arguments='{"file_name":"workspace/test_data/test_fetch_data.py","body":"from fetch_data import fetch_stock_data, save_to_csv, save_to_excel\\n\\n# Define the stock ticker and start date\\nstock_ticker = \'META\'\\nstart_date = \'2020-01-01\'\\n\\n# Fetch stock data\\nstock_data = fetch_stock_data(stock_ticker, start_date)\\n\\n# Save fetched data to CSV and Excel\\nsave_to_csv(stock_data, \'workspace/test_data/stock_data.csv\')\\nsave_to_excel(stock_data, \'workspace/test_data/stock_data.xlsx\')\\n"}', name='CreateFile')`

Speaker	Message
Test Agent	Function(arguments='{"command":"python workspace/test_data/ ↪ test_fetch_data.py"}: ModuleNotFoundError: No module named 'openpyxl'', name='ExecuteCommand ↪ ')
Test Agent	Function(arguments='{"command":"pip install openpyxl"}', name=' ↪ ExecuteCommand': Installing collected packages: et-xmlfile openpyxl Successfully installed et-xmlfile-1.1.0 openpyxl-3.1.2)
Test Agent	Function(arguments='{"file_name":"workspace/test_data/ ↪ test_analyze_data.py","body":"import sys\\nsys.path.append(\' ↪ workspace\')\\n\\nfrom analyze_data import load_data, ↪ calculate_daily_change, calculate_moving_average\\n\\n# Load ↪ the stock data from the CSV file\\nstock_data = load_data(\' ↪ workspace/test_data/stock_data.csv\')\\n\\n# Calculate daily ↪ price change\\nstock_data = calculate_daily_change(stock_data) ↪ \\n\\n# Calculate moving average with a window size of 5\\ ↪ nstock_data = calculate_moving_average(stock_data, window_size ↪ =5)\\n\\n# Save the analyzed data to a new CSV file\\ ↪ nstock_data.to_csv(\'workspace/test_data/analyzed_stock_data. ↪ csv\', index=False)\\n"}', name='CreateFile') : File written to workspace/test_data/test_analyze_data.py
Test Agent	Function(arguments='{"command":"python workspace/test_data/ ↪ test_analyze_data.py"}', name='ExecuteCommand'): Executed ↪ Successfully
Test Agent	Function(arguments='{"file_name":"workspace/test_data/ ↪ test_visualize_data.py","body":"import sys\\nsys.path.append(\' ↪ workspace\')\\nfrom visualize_data import plot_stock_prices, ↪ plot_moving_average\\nfrom analyze_data import load_data\\n\\n# ↪ Load the analyzed stock data from the CSV file\\ ↪ nanalyzed_stock_data = load_data(\'workspace/test_data/ ↪ analyzed_stock_data.csv\')\\n\\n# Plot stock prices\\ ↪ nplot_stock_prices(analyzed_stock_data)\\n\\n# Plot moving ↪ average\\nplot_moving_average(analyzed_stock_data, window_size ↪ =5)\\n"}', name='CreateFile') : File written to workspace/test_data/test_visualize_data.py

Speaker	Message
Test Agent	`Function(arguments='{"command":"python workspace/test_data/test_visualize_data.py"}', name='ExecuteCommand'): Executed Successfully`  
Dev Team Decider Agent	The goal of developing tools necessary for extracting and analyzing META's stock price data since January 1st, 2024, including visualization capabilities, has been successfully achieved. If there's anything more I can assist you with or any further questions you have, feel free to ask. Otherwise, the project vision has been fulfilled, and this concludes our chat. Thank you for working with us!
Joiner	The tasks required for creating a report on META's stock price increase since January 1st, 2024, have been divided and completed by the Research Team and the Dev Team as follows: 1. Research Team: • They have been tasked with providing detailed information on META's stock price increase since January 1st, 2024. • They were to perform an analysis of the numbers and research the reasons behind the price increase. • They are responsible for compiling a comprehensive report detailing the findings, which must include links to sources and references to support the data presented.

Speaker	Message
	2 Dev Team: • They developed the necessary tools for extracting and analyzing META's stock price data since January 1st, 2024. • The tools have visualization capabilities to aid in better understanding and presentation within the compiled report. • They ensured that the tools are compatible with any existing code provided by the user, following any specified code structures and requirements. 3 Development of Tools: • The Dev Team Decider Agent confirmed the tools' development for extracting and analyzing META's stock price data. • The Developer Agent created new tools from scratch using Python programming. The developed tools include the following Python scripts, saved in the `workspace` directory: – `fetch_data.py`: Uses `yfinance` to fetch data from Yahoo Finance API and includes functions to save data to CSV or Excel format. – `analyze_data.py`: Contains functions to load data from CSV or Excel files and performs calculations like daily price changes and moving averages using `pandas`. – `visualize_data.py`: Provides functions for visualizing stock price data with the help of 'matplotlib', including plotting stock prices and moving averages. • Each Python script is designed to be compatible with a general Python codebase and is modular for easy integration or standalone use. The developer agent inquired about proceeding with a test run of these tools to demonstrate their functionality.
File Agent	I have created a summary report detailing META's stock price increase since January 1st, 2024. This document includes the analysis conducted using the developer team's Python tools and outlines the key factors contributing to the stock's performance. The report, titled `META Stock Price Increase Since January 1st, 2024`, is saved as `Report_META_Stock_Increase_Summary.md` in the `Markdown Files` directory. It includes visual representations of data, such as stock price trends and moving averages, and concludes with the reasons behind META's stock increase. It should serve as a valuable resource for stakeholders to better understand the company's financial trajectory.

The agent swarm's analysis of META's stock price increase since January 1st, 2024 works through a collaborative division of tasks between specialized Research and Development Teams. This methodology ensured a streamlined process where data gathering, analysis, and tool development occurred concurrently. The Research Team collected and synthesized data from multiple reputable sources, including Yahoo Finance, Forbes, Reuters, Stockscan.io and The New York Times, and provided a robust foundation for understanding the factors driving META's stock performance. Concurrently, the Development Team crafted custom Python scripts for fetching, analyzing, and visualizing stock data. It incorporated functionalities for data extraction from Yahoo Finance, analysis of daily price changes and moving averages, and graphical representation of stock performance trends. These scripts were designed for modularity and ease of integration, demonstrating a commitment to quality, precision, and scalability. The testing phase further ensured the reliability and accuracy of these tools. In addition, the team was able to work together

to troubleshoot errors like missing dependencies and solved those errors through command line code. Both teams did not operate forever. Instead, their respective decider agents made the call to consider their work completing the goal.

Through this process, the team generated a comprehensive report in Markdown along with combined visual data representations, and strategic insights into the reasons for META's stock value rise. This collaborative, automated, and technically sophisticated approach not only streamlined the analysis process but also offered scalable and efficient methodologies for broader research and development tasks.

4 Discussion

The exploration and implementation of the GAS AI framework represent a significant advancement in the field of artificial intelligence, particularly in the utilization and coordination of multi-agent systems (MAS). The framework's innovative approach, which integrates graph-based communication with asynchronous AI agents, sets a new standard for the development and execution of complex, distributed AI systems. This paper has highlighted the framework's core components, its operational mechanisms, its application in research tasks using the ANTIQUE dataset, and its application in development tasks which demonstrate GAS AI's superior performance compared to existing AI tools.

The GAS AI framework's design philosophy underscores the importance of flexibility, scalability, and efficiency in managing the interactions of multiple AI agents. By adopting a graph-based approach to communication, GAS AI enables a dynamic and adaptable networking of agents, facilitating more intuitive and efficient task delegation and information exchange. Furthermore, the incorporation of asynchronous AI agents enhances the system's concurrency, allowing for simultaneous processing of tasks and leading to significant improvements in throughput and response times.

The application of GAS AI in research tasks and development tasks showcases the framework's capability to produce high-quality outputs. In the research testing, the empirical data presented underscores the advantages of GAS AI in terms of accuracy and the use of sources. It highlights its potential to significantly enhance the efficiency and effectiveness of research processes. The system's ability to split complex queries into subgoals and to recombine the findings into cohesive outputs exemplifies the power of leveraging multi-agent systems for comprehensive research tasks. In addition, the system's ability to correct errors through agent collaboration is extremely important. In the development tasks, communication between the developer agent, decider agent and testing agent were key for creating real stock visualizations. Altogether, the true power of GAS AI is the ability to use all these agents together and build communicative applications that can actually solve problems.

Despite these promising results, it is acknowledged that the GAS AI framework is still in the early stages of experimental validation. The current findings represent a preliminary exploration into the framework's capabilities and potential applications. As such, there is a vast landscape of research opportunities and future directions to be explored. These include further refining the framework's components, expanding its application to other domains beyond research and development tasks, and addressing any scalability or performance issues that may arise as the system is tested in more demanding environments.

Future work will also focus on global collaboration and communication mechanisms among agents, improving decision-making processes within the framework, and integrating additional tools and services to expand the system's capabilities. Additionally, the exploration of real-world applications, such as in disaster response, healthcare, and environmental monitoring, will be crucial in demonstrating the practical utility and impact of GAS AI in addressing complex challenges.

In conclusion, the GAS AI framework represents a promising advancement in the development of intelligent, highly concurrent applications using multi-agent systems. Its innovative approach to agent communication and task management has the potential to revolutionize how complex problems are approached and solved within the domain of artificial intelligence. As the framework continues to evolve, it is anticipated that GAS AI will play a pivotal role in enabling more sophisticated, efficient, and effective AI systems capable of tackling the multifaceted challenges of the modern world.

5 ORCID

Soham Bafana: https://orcid.org/0009-0005-3681-1088

References

[Aref, 2018] Aref, A. (2018). Trust evaluation and establishment for multi-agent systems.

[Cai et al., 2017] Cai, L., Liu, B., Yu, J., and Zhang, J. (2017). Human behaviors modeling in multi-agent virtual environment. *Multimedia Tools and Applications*, 76(4):5851–5871.

[De Mooij et al., 2023] De Mooij, J., Bhattacharya, P., Dell'Anna, D., Dastani, M., Logan, B., and Swarup, S. (2023). A framework for modeling human behavior in large-scale agent-based epidemic simulations. *SIMULATION*, 99(12):1183–1211.

[Du et al., 2023] Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., and Mordatch, I. (2023). Improving Factuality and Reasoning in Language Models through Multiagent Debate.

[Hashemi et al., 2019] Hashemi, H., Aliannejadi, M., Zamani, H., and Croft, W. B. (2019). ANTIQUE: A Non-Factoid Question Answering Benchmark.

[Huang and Chang, 2023] Huang, J. and Chang, K. C.-C. (2023). Towards Reasoning in Large Language Models: A Survey.

[Liang et al., 2023] Liang, T., He, Z., Jiao, W., Wang, X., Wang, Y., Wang, R., Yang, Y., Tu, Z., and Shi, S. (2023). Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate.

[Sharma et al., 2018] Sharma, S., Ogunlana, K., Scribner, D., and Grynovicki, J. (2018). Modeling human behavior during emergency evacuation using intelligent agents: A multi-agent simulation approach. *Information Systems Frontiers*, 20(4):741–757.

[Vaswani et al., 2023] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention Is All You Need.

[Wang and Cai, 2015] Wang, Z. and Cai, M. (2015). Reinforcement learning applied to single neuron. *ArXiv*, abs/1505.04150.

[Webb et al., 2023] Webb, T., Holyoak, K. J., and Lu, H. (2023). Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7(9):1526–1541.

[Wei, 2015] Wei, C. (2015). Cognitive coordination for cooperative multi-robot teamwork.

[Wu et al., 2023] Wu, Y., Jia, F., Zhang, S., Li, H., Zhu, E., Wang, Y., Lee, Y. T., Peng, R., Wu, Q., and Wang, C. (2023). An Empirical Study on Challenging Math Problem Solving with GPT-4.

[Xi et al., 2023] Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu, X., Yin, Z., Dou, S., Weng, R., Cheng, W., Zhang, Q., Qin, W., Zheng, Y., Qiu, X., Huang, X., and Gui, T. (2023). The Rise and Potential of Large Language Model Based Agents: A Survey.

[Yao et al., 2023] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). ReAct: Synergizing Reasoning and Acting in Language Models.