

Multi-Agent Task Automation System

Mohammed Lutfi, Mariam Fathima, S. Raju, Nanda Kumar, Marilingan

Department of Computer Science, Rajiv Gandhi Institute of Technology, Bangalore, Karnataka, India

Published October 2026

Abstract

Multi-Agent Systems (MAS) offer an effective approach for solving complex and large-scale problems in distributed artificial intelligence by dividing them into smaller subtasks handled by multiple autonomous agents. Each agent can perceive its environment, process information, make decisions, and perform actions according to its assigned objectives. Agents communicate, collaborate, and coordinate to achieve individual and collective goals, providing advantages such as scalability, adaptability, flexibility, robustness, and fault tolerance. The proposed Multi-Agent Task Automation System uses heterogeneous agents dynamically assigned tasks according to their capabilities, resource availability, current workload, task priority, execution cost, and contextual relevance. A mediator-based coordination mechanism supports agent discovery and task routing while allowing direct peer-to-peer communication when necessary. Dynamic task allocation and load balancing help distribute work efficiently. Reinforcement learning can enhance agent decision-making by allowing agents to learn from feedback and improve their strategies over time. The system also incorporates fault detection, recovery, authentication, secure communication, and trust evaluation to improve reliability and security. The implementation uses Python, AutoGen, large language models, APIs, and external tools, with structured message passing enabling agents to interact and execute tasks collaboratively. Tasks are initiated through user input and processed using iterative communication, reasoning, and tool execution. The proposed approach demonstrates the potential of MAS to improve task execution efficiency, response time, resource utilization, adaptability, scalability, and fault tolerance compared with traditional centralized approaches. However, communication overhead, coordination complexity, security risks, synchronization, and efficient task allocation remain important challenges.

Keywords: multi-agent systems (MAS); autonomous agents; distributed artificial intelligence; agent communication; coordination and collaboration; task allocation; AutoGen; large language models; reinforcement learning; scalability; adaptability; fault tolerance; intelligent decision making

1 Introduction

The rapid development of artificial intelligence and information technology has increased the need for systems capable of solving complex, large-scale, and dynamic problems. Distributed Artificial Intelligence (DAI) addresses these challenges by distributing intelligence and computational responsibilities among multiple intelligent entities. Among the major approaches within DAI, Multi-Agent Systems (MAS) have gained significant attention because they enable multiple autonomous agents to work together to solve problems that may be difficult for a single system to handle efficiently.

A Multi-Agent System consists of a collection of autonomous agents that can perceive their environment, process available information, make decisions, and perform actions to achieve specific objectives. Each agent may operate independently while also interacting with other agents through communication and coordination. Depending on the application, agents may cooperate, coordinate, negotiate, or compete with one another. The ability to exchange information and share knowledge is

particularly important when individual agents have only partial information about the overall environment.

The main advantage of MAS is its distributed approach to problem solving. Instead of depending on one centralized controller, a complex problem can be divided into smaller and more manageable subtasks, with different agents responsible for different activities. This division of responsibility can improve efficiency, flexibility, scalability, and fault tolerance. If one agent becomes unavailable or fails, its task may be reassigned to another agent, allowing the system to continue operating.

Agents in a MAS possess important characteristics such as autonomy, sociability, and proactiveness. Autonomy enables an agent to make decisions and perform actions independently. Sociability allows agents to communicate and exchange knowledge with other agents. Proactiveness enables agents to use available information and previous experiences to anticipate situations and select suitable actions. These characteristics make MAS useful for applications involving complex and changing environments.

MAS have been applied in several areas, including computer networks, cloud computing, robotics, smart grids, transportation, healthcare, and social networks. In cloud computing, agents can support resource allocation and workload balancing. In robotics, agents can coordinate navigation and task execution. In transportation systems, they can assist with traffic management and route optimization. In network security, agents can detect and respond to threats in real time.

However, MAS also introduce challenges that must be carefully addressed. Coordination becomes more difficult as the number of agents increases because agents may have different goals, resources, and information. Communication between agents can create additional overhead and delays. Task allocation must ensure that work is distributed according to agent capabilities and current system conditions. Security is another major concern because faulty or malicious agents may affect the operation of the system. Other issues include synchronization, scalability, connectivity, and maintaining reliable performance in dynamic environments.

Recent AI frameworks such as AutoGen further support the development of collaborative agent-based systems by enabling agents powered by large language models to communicate and work together for automation and complex task execution.

Therefore, the study of Multi-Agent Systems is important for developing intelligent, adaptive, and distributed solutions for real-world problems. The proposed Multi-Agent Task Automation System applies these principles by combining autonomous agents, structured communication, dynamic task allocation, coordination, learning mechanisms, and fault-tolerance techniques to support efficient and scalable task execution.

2 Literature survey

2.1 Evolution, characteristics, and architecture of multi-agent systems (MAS)

Multi-Agent Systems (MAS) have evolved from the broader field of Distributed Artificial Intelligence (DAI), which focuses on solving complex problems through distributed computation and collaborative intelligence. Earlier research emphasized distributed problem-solving and parallel processing, while modern Multi-Agent Systems focus on autonomy, communication, cooperation, and coordination among multiple intelligent agents. Each agent operates independently, perceives its environment, processes information, and performs actions to achieve specific objectives. The major charac-

teristics of MAS include autonomy, adaptability, scalability, proactiveness, and social ability. These properties enable agents to perform individual tasks while contributing to common system goals.

Various architectural models have been developed for Multi-Agent Systems, including reactive, deliberative, and hybrid architectures. Reactive architectures enable agents to respond quickly to environmental changes, whereas deliberative architectures involve planning, reasoning, and decision-making. Hybrid architectures combine both approaches to improve efficiency and flexibility. Modern frameworks such as Microsoft AutoGen extend these concepts by supporting communication and collaboration between AI-powered agents, making them suitable for intelligent task automation.

2.2 Applications of multi-agent systems (MAS) in various domains

Multi-Agent Systems are widely used across different domains because of their ability to manage complex tasks through distributed intelligence. In cloud computing, MAS support resource allocation, workload balancing, and efficient utilization of computing resources. In robotics, multiple agents collaborate to perform navigation, object handling, and coordinated operations. Similarly, smart grids use intelligent agents for energy management, while transportation systems employ MAS for traffic monitoring and route optimization.

In healthcare, Multi-Agent Systems can support patient monitoring, medical decision-making, and information management. They are also applied in cybersecurity to detect threats and coordinate responses to suspicious activities. Recent advancements in Large Language Models (LLMs) have further expanded the capabilities of MAS. Frameworks such as Microsoft AutoGen enable multiple AI agents to communicate, exchange information, perform specialized tasks, and collaborate through structured interactions. These capabilities make MAS useful for automated research, information processing, software development, and intelligent decision-support systems.

2.3 Challenges and limitations in multi-agent systems (MAS)

Despite their advantages, Multi-Agent Systems face several technical and operational challenges. One of the major difficulties is coordination among multiple autonomous agents, particularly when agents have different objectives, incomplete information, or conflicting decisions. Efficient communication is essential for collaboration, but frequent message exchanges can increase computational overhead, network traffic, and execution time.

Another significant challenge is task allocation, which requires distributing work according to agent capabilities, workload, and resource availability. Improper allocation may cause some agents to become overloaded while others remain underutilized. Security and reliability are also important concerns because faulty or malicious agents can affect overall system performance. Furthermore, maintaining synchronization, fault tolerance, and scalability becomes increasingly difficult as the number of agents grows.

When Large Language Models are integrated into Multi-Agent Systems, additional challenges may arise, including inaccurate responses, increased API costs, and repeated communication between agents. Therefore, effective coordination strategies, monitoring mechanisms, communication protocols, and error-handling techniques are necessary to develop reliable and efficient multi-agent automation systems.

3 Proposed method

The proposed Multi-Agent Task Automation System presents an intelligent and flexible framework designed to automate complex tasks through the collaboration of multiple autonomous agents. The system is developed using Microsoft AutoGen, a framework that enables communication, coordination, and task execution among AI-powered agents. The primary objective of the proposed system is to divide complex user requests into smaller, manageable subtasks and assign them to specialized agents according to their capabilities and responsibilities. By distributing tasks among multiple agents, the system aims to improve execution efficiency, reduce manual intervention, and enhance the overall reliability of automated workflows.

The proposed architecture consists of multiple intelligent agents that perform different operations, such as task planning, information retrieval, data processing, execution, and result verification. Each agent is assigned a specific role and can communicate with other agents through structured message exchanges. When a user submits a task, the system analyzes the request and identifies the operations required to complete it. A coordinating agent manages the workflow by assigning subtasks to appropriate agents and monitoring their progress. The agents collaborate by exchanging relevant information and intermediate results, allowing the system to complete complex tasks through coordinated execution.

To enhance decision-making and automation capabilities, the system integrates Large Language Models (LLMs) with the AutoGen framework. These models enable agents to understand natural language instructions, interpret user requirements, generate responses, and make decisions based on available information. The proposed system can also integrate external APIs, Python-based tools, and data-processing modules to perform practical operations beyond text generation. This integration allows agents to execute tasks such as information analysis, document processing, automated reporting, and other intelligent workflow activities.

Efficient task allocation and coordination are important components of the proposed method. Tasks are distributed based on the capabilities of individual agents, task requirements, and available resources. A mediator-based coordination mechanism facilitates communication and manages task routing between agents. This approach helps maintain an organized workflow, reduces unnecessary communication, and minimizes the possibility of duplicated operations. The system also incorporates monitoring and error-handling mechanisms to identify execution failures and support task reassignment or recovery when required.

The proposed system follows a modular and layered architecture consisting of user interaction, agent coordination, task execution, and tool integration components. The user interaction layer receives task requests, while the coordination layer manages communication and distributes responsibilities among agents. The execution layer processes assigned tasks using intelligent agents, and the integration layer provides access to external tools and services. After task completion, the generated outputs are collected, reviewed, and presented to the user as a consolidated result.

4 Methodology

The methodology of the proposed Multi-Agent Task Automation System focuses on the systematic design, development, and integration of multiple intelligent agents to automate complex tasks through distributed intelligence and collaborative decision-making. The system is designed using the framework, which enables autonomous agents to communicate, coordinate, and perform specialized

operations using Large Language Models (LLMs). The methodology involves analyzing user requirements, decomposing complex tasks into smaller subtasks, assigning responsibilities to suitable agents, executing operations, and combining the results to generate a final output. This structured approach aims to improve automation efficiency, scalability, flexibility, and reliability while reducing the need for continuous human intervention.

The initial phase involves identifying the problem requirements and defining the roles and responsibilities of individual agents. Each agent is designed to perform a specific function, such as task planning, information retrieval, data analysis, execution, or result verification. The system follows a modular architecture in which agents operate independently while collaborating to achieve common objectives. A coordinating agent manages the overall workflow by analysing incoming user requests and distributing subtasks according to agent capabilities. This approach enables the system to handle multiple operations systematically and reduces unnecessary duplication of tasks. The system architecture and the flowchart of the proposed system are shown in Figures 1 and 2, respectively.

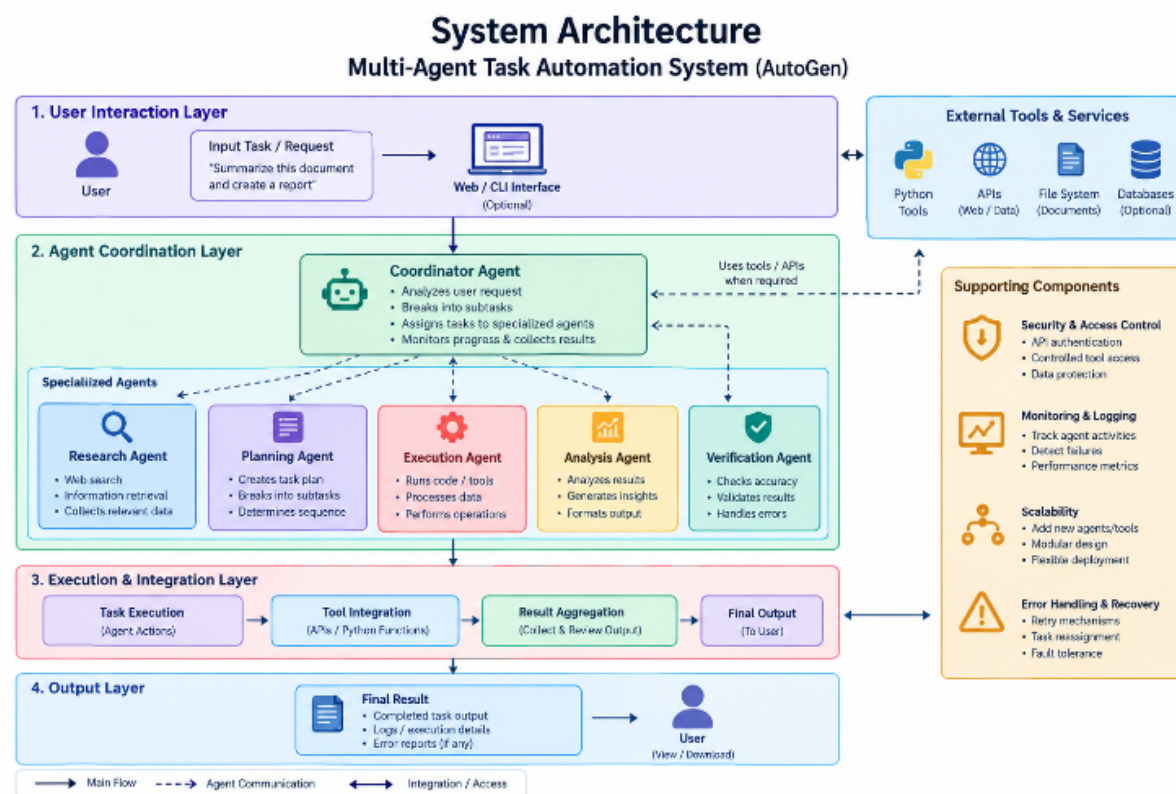


Figure 1. System architecture of the proposed multi-agent task automation system.

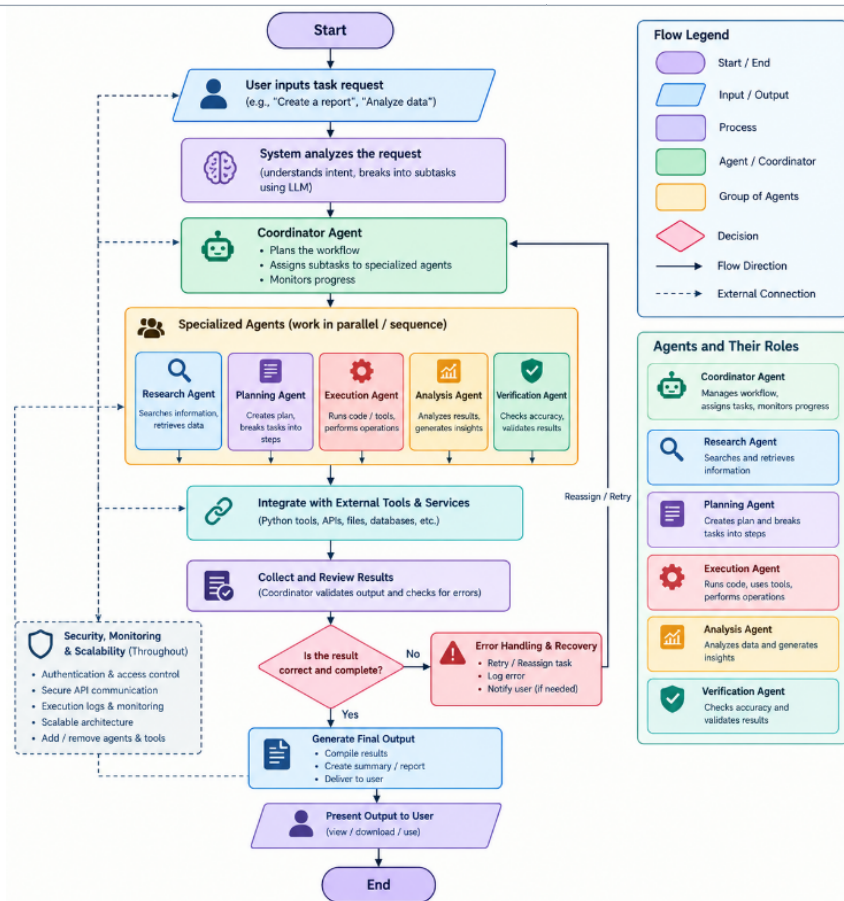


Figure 2. Flowchart of the proposed system.

5 Software implementation

5.1 Software requirements

The development and implementation of the proposed Multi-Agent Task Automation System require a combination of programming languages, artificial intelligence frameworks, development tools, and supporting software libraries. The system is designed using Microsoft AutoGen, which provides a framework for developing intelligent agents capable of communicating, collaborating, and executing complex tasks. The software environment supports the integration of Large Language Models (LLMs), external Application Programming Interfaces (APIs), and Python-based tools to enable automated task execution and intelligent decision-making. {#the-development-and-implementation-of-the-proposed-multi-agent-task-automation-system-require-a-combination-of-programming-languages-artificial-intelligence-frameworks-development-tools-and-supporting-software-libraries.-the-system-is-designed-using-microsoft-autogen-which-provides-a-framework-for-developing-intelligent-agents-capable-of-communicating-collaborating-and-executing-complex-tasks.-the-software-environment-supports-the-integration-of-large-language-models-llms-external-application-programming-interfaces-apis-and-python-based-tools-to-enable-automated-task-execution-and-intelligent-decision-making. .unnumbered}

Python serves as the primary programming language for developing the proposed system because of its simplicity, flexibility, and extensive support for artificial intelligence and automation applications. Microsoft AutoGen is used to create and manage multiple autonomous agents, establish communication between them, and coordinate their activities. The framework provides components such as

AgentChat for high-level agent interactions, AutoGen Core for message-based communication and runtime management, and AutoGen Extensions for integrating external tools and services. {#python-serves-as-the-primary-programming-language-for-developing-the-proposed-system-because-of-its-simplicity-flexibility-and-extensive-support-for-artificial-intelligence-and-automation-applications.-microsoft-autogen-is-used-to-create-and-manage-multiple-autonomous-agents-establish-communication-between-them-and-coordinate-their-activities.-the-framework-provides-components-such-as-agentchat-for-high-level-agent-interactions-autogen-core-for-message-based-communication-and-runtime-management-and-autogen-extensions-for-integrating-external-tools-and-services. .unnumbered}

The system requires access to a Large Language Model, such as a model provided through the OpenAI API or Azure OpenAI services, to support natural language understanding, task planning, reasoning, and response generation. These models enable agents to interpret user instructions, exchange information, and generate appropriate outputs. Additional Python libraries are used for data processing, API communication, file handling, and execution monitoring. {#the-system-requires-access-to-a-large-language-model-such-as-a-model-provided-through-the-openai-api-or-azure-openai-services-to-support-natural-language-understanding-task-planning-reasoning-and-response-generation.-these-models-enable-agents-to-interpret-user-instructions-exchange-information-and-generate-appropriate-outputs.-additional-python-libraries-are-used-for-data-processing-api-communication-file-handling-and-execution-monitoring. .unnumbered}

The development environment can be configured using Visual Studio Code (VS Code), which provides facilities for writing, debugging, testing, and managing Python applications. Git and GitHub can be used for version control, source code management, and collaborative development. A Python virtual environment and the pip package manager are used to install and manage project dependencies, ensuring consistency across development environments. {#the-development-environment-can-be-configured-using-visual-studio-code-vs-code-which-provides-facilities-for-writing-debugging-testing-and-managing-python-applications.-git-and-github-can-be-used-for-version-control-source-code-management-and-collaborative-development.-a-python-virtual-environment-and-the-pip-package-manager-are-used-to-install-and-manage-project-dependencies-ensuring-consistency-across-development-environments. .unnumbered}

5.2 Implementation

The implementation of the proposed Multi-Agent Task Automation System is based on integrating multiple intelligent agents using the Microsoft AutoGen framework to automate complex tasks through coordinated communication and collaborative decision-making. The system is developed using Python as the primary programming language, while Large Language Models (LLMs) provide natural language understanding, reasoning, and response-generation capabilities. The implementation follows a modular architecture in which specialized agents perform assigned operations and communicate with one another to achieve common objectives. This approach aims to improve task execution efficiency, reduce manual intervention, and support flexible automation across different application domains.

The implementation begins with the configuration of the development environment by installing Python, Microsoft AutoGen, and the necessary supporting libraries. The system integrates an LLM provider, such as OpenAI or Azure OpenAI, to enable intelligent agent interactions. Each agent is configured with a specific role, responsibility, and set of instructions. The proposed system includes a Coordinator Agent, Research Agent, Planning Agent, Execution Agent, Analysis Agent, and Verification

Agent. These agents collaborate to process user requests, retrieve relevant information, generate execution plans, perform assigned operations, analyze results, and verify the accuracy of generated outputs.

6 Results and discussion

The proposed Multi-Agent Task Automation System demonstrates the potential of integrating multiple intelligent agents to automate complex tasks through coordinated communication and collaborative decision-making. By utilizing the Microsoft AutoGen framework and Large Language Models (LLMs), the system is designed to divide complex user requests into smaller, manageable subtasks and assign them to specialized agents according to their responsibilities. This distributed approach aims to improve task execution efficiency, reduce manual intervention, and enhance flexibility compared to traditional single-agent automation systems. The architecture supports structured communication between agents, enabling them to exchange information, coordinate operations, and generate consolidated outputs.

The proposed system incorporates several specialized agents, including the Coordinator Agent, Research Agent, Planning Agent, Execution Agent, Analysis Agent, and Verification Agent. Each agent performs a specific function within the automation workflow. The Coordinator Agent manages task distribution and monitors overall execution, while the Research and Planning Agents gather relevant information and organize the required operations. The Execution Agent performs tasks using integrated tools and APIs, whereas the Analysis and Verification Agents process intermediate results and examine the generated outputs. This collaborative mechanism is intended to improve workflow organization, task specialization, and the consistency of automated results.

The integration of Large Language Models enables the system to interpret natural language requests, generate task plans, and support intelligent communication between agents. Microsoft AutoGen provides mechanisms for agent interaction and coordination, while Python-based tools and external APIs extend the system's capabilities beyond text generation. The proposed Streamlit interface allows users to submit requests and receive generated outputs through an interactive environment. These components collectively provide a foundation for developing automated workflows involving document processing, information retrieval, data analysis, and report generation.

From a performance perspective, the proposed architecture offers opportunities to improve scalability, adaptability, and fault tolerance. Complex tasks can be divided among specialized agents, allowing independent operations to be executed sequentially or concurrently where supported. The modular structure also enables additional agents and functionalities to be incorporated according to system requirements. Error-handling mechanisms, execution monitoring, and result verification are proposed to identify failures, support recovery operations, and improve overall system reliability.

7 Conclusion

The proposed Multi-Agent Task Automation System presents a structured approach to automating complex workflows through collaboration among specialized intelligent agents. Using Microsoft AutoGen, Python, and Large Language Models, the system is designed to interpret user requests, divide them into manageable subtasks, coordinate agent communication, and combine individual outputs into a final response. Integration with external APIs and Python tools can extend its capabilities to information retrieval, data analysis, document processing, and report generation. The

modular architecture also supports future expansion by allowing additional agents and services to be introduced as requirements change. However, effective implementation requires careful management of communication overhead, model accuracy, API costs, security, and error recovery. Monitoring and verification mechanisms are therefore important for improving reliability and reducing incorrect outputs. Overall, the proposed system provides a practical foundation for intelligent task automation with reduced manual intervention. Its actual efficiency, scalability, and fault tolerance must be established through implementation, functional testing, and measurable comparisons with conventional single-agent approaches before performance improvements can be claimed in real-world applications.

References

- [1] Dorri A, Kanhere SS, Jurdak R. Multi-agent systems: A survey. *IEEE Access*. 2018;6:28573–28593.
- [2] Russell S, Norvig P. *Artificial intelligence: A modern approach*. 3rd ed. Upper Saddle River (NJ): Prentice Hall; 2010.
- [3] Wooldridge M. *An introduction to multiagent systems*. 2nd ed. Wiley; 2009.
- [4] Weiss G. *Multiagent systems: A modern approach to distributed artificial intelligence*. MIT Press; 1999.
- [5] Jennings NR. An agent-based approach for building complex software systems. *Communications of the ACM*. 2001;44(4):35–41.
- [6] Lesser VR. Cooperative multiagent systems: A personal view of the state of the art. *IEEE Transactions on Knowledge and Data Engineering*. 1999;11(1):133–142.
- [7] Wooldridge M, Jennings NR. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*. 1995;10(2):115–152.
- [8] Wu Q, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv:2308.08155* [Preprint]. 2023. <https://arxiv.org/abs/2308.08155>