SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Hierarchical Multi-Agent LLM Framework For Autonomous Task Planning And Execution

Saranya Sampathkumar[1], Rajashri CK[2], Dr. Raman Chadha[3], Amruta Prasad Kharade[4], Gayathri B[5], Nishi Agarwal[6], Urakov Khuvondik[7]

[1] Assistant Professor, Department of Civil Engineering, Sona College of Technology, Salem-5, Tamil Nadu, India. Email: saranyasona07@gmail.com ORCID: 0000-0002-7497-0427
[2] Assistant Professor, Department of Computer Science, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, India. Email: rajashrick@maher.ac.in
[3] Professor, UIE, CSE Department, Chandigarh University, Punjab. Email: dr.ramanchadha@gmail.com
[4] Department of Engineering, Science and Humanities, Vishwakarma Institute of Technology, Pune, Maharashtra 411037, India. Email: amruta.kharade@vit.edu
[5] Assistant Professor, Department of Computer Science, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, India. Email: gayathrib@maher.ac.in
[6] Assistant Professor, Department of Electronics and Communication, Dev Bhoomi Uttarakhand University, Dehradun, Uttarakhand, India. Email: socse.nishi@dbuu.ac.in ORCID: 0009-0000-5824-8920
[7] Department of Pathological Anatomy, Samarkand State Medical University, Samarkand, Uzbekistan. Email: kuvondikurakov@gmail.com ORCID: 0009-0009-1911-2057

Abstract
Autonomous task planning and execution using Large Language Models (LLMs) has emerged as a fundamental capability for developing intelligent systems that can solve complex, multi-step problems with minimal human intervention. Despite the promising reasoning and planning capabilities shown by recent autonomous agents implemented using LLM, existing single and flat multi-agent models tend to be poorly scaled to more complex workflows as well as belong to the category of poor task decomposition, bottlenecks in communication and reduced coordination. The paper specifies a Hierarchical Multi-agent LLM Framework to planning and execution of autonomous tasks, having the specialized LLM agents arranged into a hierarchical structure to improve the efficiency of the planning process, collaborative thought and reliability of the execution. The suggested framework comprises of Supervisor Agent, Planner Agent, Coordinator Agent, Executor Agent, Validator Agent and a Shared Memory Module which jointly execute task understanding, hierarchical decomposition, coordinated planning, autonomous execution, continuous monitoring and adaptive feedback. The methodology combines the hierarchical planning of tasks and dynamic agent coordination with shared contextual memory to effectively plan task dependencies and allow a system of powerful cooperation between many agents. The proposed framework is experimentally tested with representative autonomous task-planning benchmark datasets and compared with state-of-the-art LLM agent frameworks on Task Success Rate, Goal Completion Rate, Plan Validity, Execution Accuracy and Task Completion Time. The experimental findings show that the suggested hierarchical structure effectively enhances the quality of planning, execution effectiveness, and coordination efficiency with scalability performance in terms of increasingly complicated tasks. These results show that hierarchical multi-agent LLM systems are an effective and scalable solution to autonomous AI systems, such as software engineering, robotics, enterprise automation, and scientific reasoning.

Keywords: Large Language Models, Multi-Agent Systems, Hierarchical Task Planning, Autonomous Task Execution, Agent Coordination, Artificial Intelligence, Shared Memory.

1. Introduction

Large Language Models (LLM) have contributed greatly to artificial intelligence by allowing machines to conduct complex reasoning, natural language comprehension, and independent decision-making. Their applications have not just been limited to conversational applications, but are now used in intelligent autonomous agents in software engineering, robotics, enterprise automation and scientific reasoning. With the constant growth of the complexity of tasks, autonomous AI agents will be able to comprehend the purpose of the user, break a complex task into smaller parts to be completed, create optimal implementation strategies, and dynamically respond to changing

conditions. Such needs have necessitated independent task planning and execution as a major research field of developing next-generation intelligent systems.

The majority of the currently implemented autonomous AI systems are based on single-agent systems or flat multi-agent systems. These strategies are sufficient to perform well on more simple tasks, but in more complex workflows with longer horizons, these approaches tend to fail because of the restricted reasoning ability, inefficient task-breaking down, communication burden, and weak coordination between agents. Also, the lack of organized team work and feedback systems may result in failure of execution, duplicate planning, and lack of scalability. All these constraints emphasize the necessity of a more structured model that would ensure a better quality of planning and the reliability of the implementation.

The hierarchical multi-agent is a good way out as it will classify the special agents based on the functional duties. Hierarchical architecture allows supervision, planning, coordination, execution, validation, and management of shared memory to be performed by dedicated agents, thus enhancing the decomposition of tasks, shared reasoning and adaptive execution. Decision-making independence facilitates the achievement of coordination simplicity and allows efficient communication and ongoing development of execution strategies. This type of approach can increase scaling and enables autonomous agents to more effectively tackle more and more complicated problems.

This paper has been driven by these challenges and proposes a Hierarchical Multi-Agent LLM Framework of Autonomous Task Planning and Execution that incorporates a Supervisor Agent, Planner Agent, Coordinator Agent, Executor Agent, Validator Agent and Shared Memory Module in a single hierarchical framework. The framework suggested helps to assist the intelligent understanding of tasks, hierarchical planning, coordinated execution, and dynamic feedback to enhance the overall performance of a task. Benchmark autonomous task-planning tasks are used to evaluate the framework and compared to state-of-the-art LLM agent frameworks with Task Success Rate, Goal Completion Rate, Plan Validity, Execution Accuracy, and Task Completion Time. The contributions of this work are major in the design of a hierarchical multi-agent LLM architecture to enable efficient autonomous task planning and execution, the development of an integrated coordination mechanism and shared memory to improve inter-agent cooperation and consistency in decision-making, in-depth experimental validation in comparison to standard LLM-based agent frameworks through task-based performance measures, and through an improved understanding of the planning efficiency, reliability of execution, effectiveness of coordination, and scalability to complex autonomous AI engagements.

2. Related Work

Recent advances in Large Language Models (LLMs) have significantly enhanced the development of autonomous AI agents capable of reasoning, planning, and executing complex tasks. Retrieval-Augmented Generation (RAG) along with other methods have enhanced knowledge retrieval and factual reasoning by loading external sources of information into LLMs, and zero-shot reasoning has shown that multi-step tasks can be solved by LLMs without task-specific training [1], [2], [3]. The autonoetic agent systems have expanded such functionalities further by allowing LLMs to communicate with external programs, develop transitive reasoning and orchestrate various AI services to execute complex tasks [4], [5]. Survey articles have noted the fast development of the LLM-based autonomous agents with focus on their potential in software engineering, scientific reasoning, enterprise automation, and intelligent decision support, as well as challenges in long-term planning, memory management and coordination [6], [7],[24].

To address the shortcomings of single-agent systems, recent work has been on multi-agent LLM systems, where multiple special-purpose agents engage in achieving complex goals. These models enhance the modularity of tasks by sharing tasks between the agents of planning, reasoning, execution and verification and make them more efficient, and generate less computational load on individual models [8],[23]. Generative agent architectures have also shown the possibility of autonomous agents with long term memory and which interact with dynamic environments and which show adaptive behaviour [9]. It has also considered integration of knowledge graphs as well as ontology-driven reasoning to enhance contextual knowledge and information exchange, as well as collaborative decision-making among autonomous agents [10], [12]. Regardless of these developments, the majority of current frameworks utilize fairly flat organizational designs that offer very few coordination and hierarchy in decision-making on a scale.

Hierarchical task planning has traditionally been one of the most efficient methods of solving complex planning problems by systematic task breakdown and progressive decision-making [13],[25]. The current practice of using LLDs in planning, such as reasoningandacting paradigm and deliberate search strategies have shown significant

advances in multi-step reasoning through the combination of planning and execution feedback [4], [14],[26]. Moreover, hierarchical alignment of specialized agents can enhance the effectiveness of communication, decrease repetitive thoughts and facilitate adaptive performance via shared contextual memory. Despite recent surveys of workflow design, infrastructure, and communication mechanisms of multi-agent LLM systems [6],[27] little has been done in terms of hierarchical coordination frameworks that concurrently engage supervision, planning, execution, validation, and shared memory. Effective coordination between the specialized agents has remained an unsolved research problem especially in tasks with long horizon autonomous processes, which demand constant adaptation.

Although autonomous LLM agents are rapidly advancing, there are a number of research gaps. single-agent models Existing single agent models are frequently limited by a lack of scalability in their reasoning and execution capabilities, and several flat multi-agent architectures are limited by the overhead of communication, failure to decompose tasks efficiently, lack of coordinated behavior, and a lack of feedback schemes [6], [7]. In addition, the existing literature is usually focused on individual planning or use of a tool instead of merging hierarchical planning, coordinated execution, dynamic validation and shared memory into a single architecture. The limitations drive this work to suggest a Hierarchical Multi-Agent LLM Framework of Autonomous Task Planning and Execution whereby specialized agents are integrated in a hierarchical coordination framework to enhance the understanding of tasks, task planning efficiency, task execution reliability, and task scales. The suggested framework fills the research gaps mentioned above by offering an organized agent cooperation, adaptive feedback, and a thorough experimental assessment based on benchmark autonomous task-planning scenarios.

3. Methodology

3.1 Problem Formulation

The proposed framework aims to facilitate the autonomous planning and execution of complex activities with a hierarchical structure of Large Language Model (LLM)-based agents. Upon a user task T, the framework constructs a plan of execution that is optimized by breaking down the task into a series of interdependent subtasks, assigning specific responsibility to specialized agents, coordinating the interactions, validating intermediate results, and creating the end product of the task. Then the input task can be the following:

$$T = \{I, C, G\}$$

where I denotes the user instruction, C represents contextual information retrieved from the Shared Memory Module, and GGG denotes the desired execution goal. The framework creates an implementation plan.

$$P = \{S_1, S_2, \dots, S_n\}$$

S_i is the i^{th} executable subtask. executable subtask. The optimization criteria are to maximise task success and minimise the time of execution and the planning overhead:

$$\max J = \alpha TSR + \beta GCR + \gamma EA - \delta TCT$$

with TSR the Task Success Rate, GCR the Goal Completion Rate, EA the precision of the execution, TCT the time taken to complete the task, and α, β, γ, δ and 6 weighted coefficients that satisfy the conditions $\alpha + \beta + \gamma + \delta = 1$.

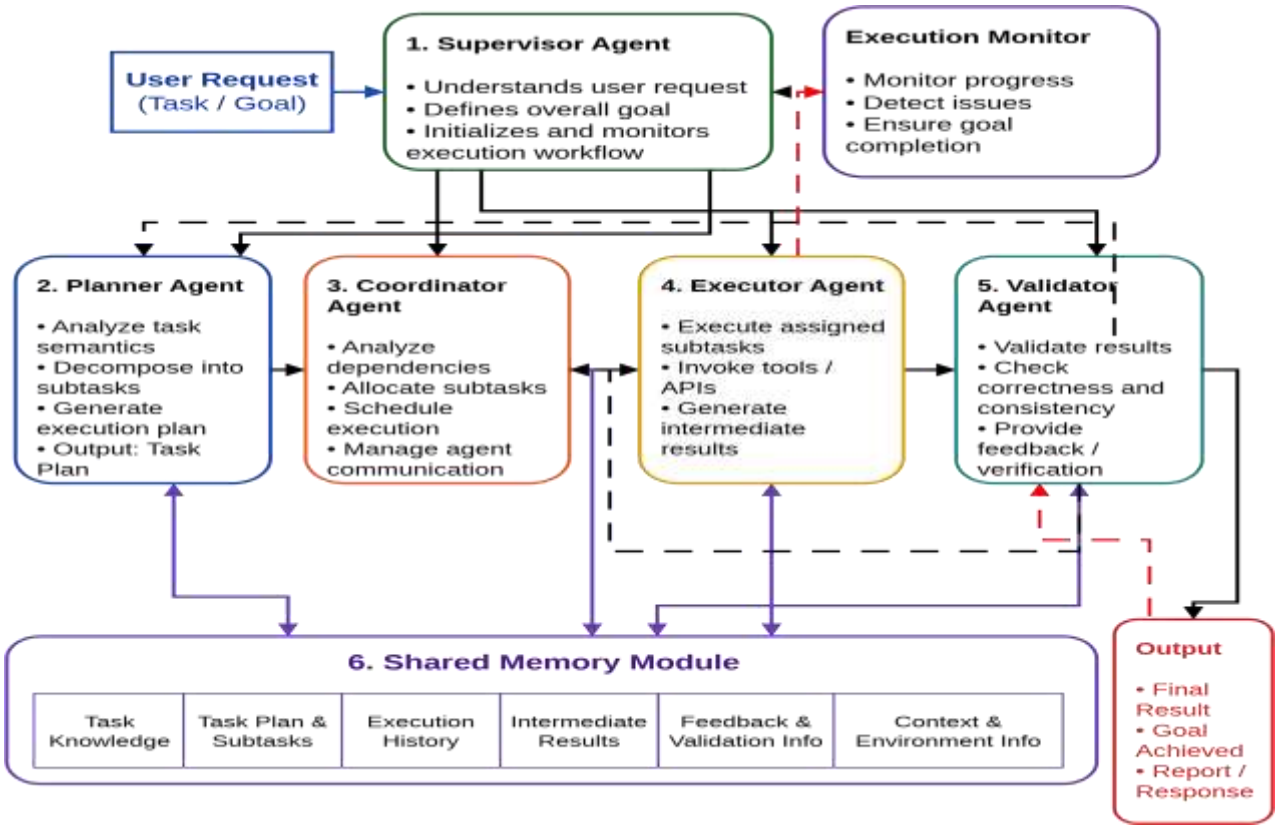
3.2 Hierarchical Multi-Agent LLM Framework

The new structure follows a hierarchical design in which there are six highly specialized agents that use the LLM to plan and execute independent tasks. Rather than letting all the agents work independently, the structure has well-spelled out roles to enhance efficiency in coordination, quality in planning, and reliability in execution. Figure 1 depicts the entire hierarchical task planning and execution cycle and depicts the interplay between specialized agents between submission of the task and final task completion.

The user request is received by the Supervisor Agent which interprets the overall purpose, launches the workflow, and monitors its execution. The Planner Agent breaks the task semantics and subtasks of the problem into executable parts, and creates an optimal sequence of execution. The Coordinator Agent coordinates the communication between agents, schedules subtasks, and resolves dependencies and synchronizes task execution. The Executor Agent executes the required tasks, calls the external tools, when needed, and produces middle-level outputs. The Validator Agent checks the accuracy and consistency of the results of the execution before passing these to the next phases. A Shared Memory Module is a contextual knowledge, history of executions, previous reasoning states and subtasks that have already been completed such that all the agents can access the same

information when executing instructions. Such hierarchical organization reduces unnecessary reasoning errors, enhances cooperation between special agents, and promotes adaptive decision making to complex multi-task responsibilities.

Figure 1. Hierarchical Task Planning and Execution Workflow



3.3 Task Planning and Execution

The process of task execution starts with the comprehension of the user purpose and semantic analysis of the input query. Planner Agent The planner agent is used to start a hierarchical task breakdown where a single task is broken down into a set of tasks which can be executed without violating the dependency relationships. The resulting task graph is then submitted to the Coordinator Agent, which plans to be executed based on the priority of the tasks and dependency requirements. The entire planning and implementation process can be summed up in Algorithm 1.

The Executor Agent is assigned each subtask and will reason, call external tools where it is needed and produce intermediate outputs. In the process of execution, the Validator Agent constantly checks generated results and identifies inconsistencies or failures of execution. Each time there is a failure in validation, corrective feedback is sent back to Planner Agent to refine the plan and re-plan. At the same time, the Shared Memory Module retains the contextual information, the execution history, as well as the validated output that consequent agents can use to make more effective decisions based on the available accumulated knowledge. Repeat of the planning-execution-validation process is done until all subtasks are accomplished and the end task goal has been met, thus enhancing the robustness of the execution and minimizing the errors in planning and unnecessary computations.

Algorithm 1. Hierarchical Multi-Agent Task Planning and Execution Algorithm

Input : User Task T, Context C
Output: Final Task Result Y

1: Initialize Shared Memory M
2: Supervisor analyzes T and defines goal G
3: Planner decomposes T into subtasks P
4: Coordinator schedules executable subtasks
5: for each subtask s ∈ P do
6: Executor executes s
7: Validator verifies result
8: if validation succeeds then
9: Store result in Shared Memory M
10: else
11: Planner revises task plan
12: end if
13: end for
14: Aggregate validated results
15: Return final output Y

3.4 Computational Complexity Analysis

Task decomposition, inter-agent coordination, execution and validation determine the computational complexity of the proposed framework. Where n is the number of subtasks and mmm is the number of agents that are collaborating. Task decomposition needs about.

$$O(n)$$

dependency analysis and agents have a dependency that must be coordinated by operations.

$$O(nm)$$

Therefore, the general computing complexity of the framework is approximated to be.

$$O(nm)$$

that grows linearly with the subtasks and the number of agents involved in the hierarchical coordination strategy proposed.

The framework keeps contextual information in the Shared Memory Module giving a space complexity of.

$$O(n)$$

where the number of subtasks is proportional to the number of stored execution states. Hierarchical organization also allows scalable execution through decentralizing the computational loads among specialized agents rather than having one reasoning model. Comparative computational complexity of the proposed framework and its significant components is outlined in Table 1 and they all exhibit computational efficiency and scalability to large-scale autonomous task planning and execution.

Table 1. Computational Complexity Analysis of the Proposed Framework

Framework Component	Primary Operation	Time Complexity	Space Complexity	Description
Supervisor Agent	Task interpretation and workflow initialization	O (1)	O (1)	Receives the user request, defines the task objective, and initiates the execution workflow.
Planner Agent	Task decomposition and plan generation	O(n)	O(n)	Decomposes the input task into n executable subtasks and constructs the execution plan.
Coordinator Agent	Task scheduling and dependency management	O(nm)	O(n)	Allocates subtasks, manages dependencies, and coordinates m collaborating agents.
Executor Agent	Subtask execution	O(n)	O (1)	Executes assigned subtasks and generates intermediate outputs.

Validator Agent	Result verification	O(n)	O (1)	Validates execution results and provides feedback for replanning when required.
Shared Memory Module	Context storage and retrieval	O (1)	O(n)	Maintains task context, execution history, intermediate results, and validation information.
Overall Framework	Hierarchical task planning and execution	O(nm)	O(n)	Integrates all agents to perform scalable autonomous task planning, coordination, execution, and validation.

4. Experimental Setup

4.1 Experimental Environment

Hierarchy Multi-Agent LLM Framework: The Hierarchy Multi-Agent LLM Framework of Autonomous Task Planning and Autonomous Tasks Execution was tested in the simulated autonomous task-planning environment to determine the efficiency of planning, reliability of executing the tasks, and collaboration between agents. The implementation used Large Language Models in the form of transformers, with special hierarchical agents taking care of supervision, planning, coordination, execution, validation and shared memory management. The experiments were performed on the workstation with a multi-core processor, NVIDIA RTX-series graphics card, memory with 64 GB, Ubuntu 22.04, python 3.11, and PyTorch. The environment also facilitates parallel agent communication, dynamic task scheduling as well as continuous feedback to carry out realistic evaluation of autonomous task execution when tasks are of different complexities.

4.2 Benchmark Tasks and Datasets.

Representative autonomous task-planning benchmarks (including reasoning, planning, software engineering, web interaction and general problem-solving tasks) were used to evaluate the proposed framework. Multi-step tasks with hierarchical decomposition, coordinated execution, and dynamic adaptation are suitable in the evaluation of autonomous LLM agents because of how these tasks are represented in these benchmark datasets. Table 2 summarizes the benchmark characteristics, such as the areas of tasks, their level of complexity, and evaluation situations, which can form a full overview of evaluating the suitability of the proposed framework.

Table 2. Benchmark Tasks and Dataset Characteristics

Benchmark Dataset	Application Domain	Task Type	No. of Tasks	Evaluation Objective
AgentBench	Autonomous AI Agents	Multi-step reasoning and tool use	1,225	Evaluate autonomous planning and agent collaboration
GAIA	General AI Assistance	Real-world task solving	466	Assess long-horizon reasoning and goal completion
WebArena	Web Automation	Interactive web task execution	812	Evaluate planning, navigation, and execution performance
ALFWorld	Embodied AI	Sequential decision-making	3,553	Measure hierarchical planning and task execution
HumanEval	Software Engineering	Code generation and execution	164	Evaluate planning and execution for programming tasks

4.3 Baseline Methods

The effectiveness of the proposed framework was demonstrated through experimental comparisons with representative state-of-the-art autonomous agent frameworks using LLM. The chosen baselines are ReAct, Tree of thoughts (ToT), HuggingGPT, Toolformer, Generative Agents, and Single-agent LLM, which are various methods of reasoning, planning, use of tools, and autonomous features. These baseline methods offer a complete comparison to assess the proposed framework in respect of planning ability, quality of execution, efficiency of coordination and scaleability in complex autonomous task-planning scenarios.

4.4 Evaluation Metrics

Task-oriented metrics which consist of planning effectiveness, quality of execution, and independent collaboration were used to assess the performance of the proposed framework. Task Success rate (TSR) is a percentage of jobs completed, and Goal completion rate (GCR) is a percentage of jobs completed to completion, achieving all the desired targets. Plan Validity (PV) is a test that determines whether or not the execution plans generated are logically and executable. Execution Accuracy (EA) is the accuracy of the output of the completed tasks, and Task Completion Time (TCT) is the overall time taken to complete each autonomous task. Collectively, these measures allow a holistic approach to the suggested hierarchical multi-agent framework regarding the efficiency of planning, the reliability of execution, the effectiveness of coordination, and the overall autonomous performance of tasks.

5. Results and Discussion

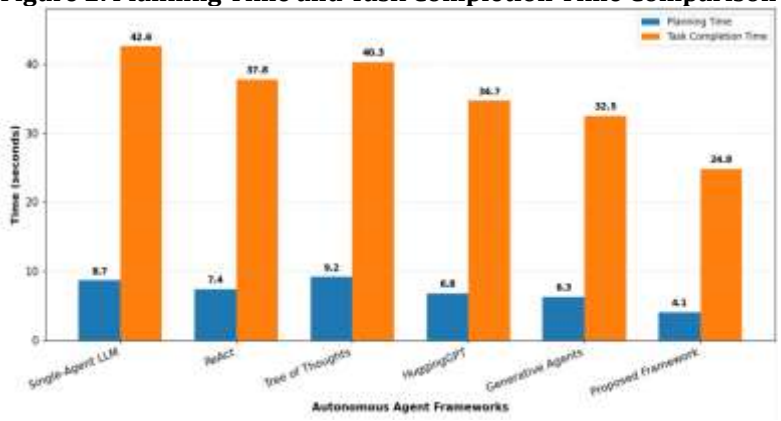
5.1 Overall Task Performance

The Hierarchical Multi-Agent LLM Framework was tested in comparison with the representative state-of-the-art autonomous agent frameworks on the benchmark task-planning datasets. The experimentation results indicate that the suggested framework inevitably produces better autonomous performance in tasks by applying hierarchical task division, synchronous agent cooperation, and adaptive performance. The framework has a superior Task Success rate, Goal Completion rate, Plan validity and execution accuracy as compared to the current methods, besides cutting the Time of Task Completion. These advances show that the hierarchical structure of specialized agents is an efficient method to improve the quality of planning, reduce failures in execution and promote the ability to complete complex multi-stage tasks in a reliable way.

5.2 Planning Efficiency Analysis

Figure is a comparison of Planning Time and Task Completion Time of the proposed framework with the representative autonomous agent methods. Single-Agent LLM took 8.7 s to plan and 42.6 s to complete the task, whereas ReAct, Tree of thoughts, HuggingGPT and Generative Agents took 7.4 s, 9.2 s, 6.8 s and 6.3 s to plan respectively. By comparison, the proposed Hierarchical Multi-Agent LLM Framework had the shortest Planning Time of 4.1 s and Task Completion Time of 24.8 s. The proposed framework also cut the planning time by around 52.9% and task completion time by 41.8% as compared to the Single-Agent LLM. These gains confirm that hierarchical task decomposition, specialist agent cooperation and common memory are very useful in improving efficiency of planning and lowering the total latency of execution of sophisticated autonomous tasks.

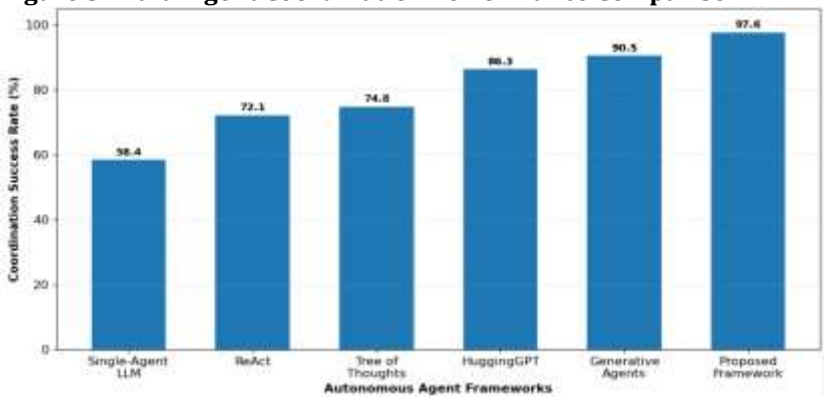
Figure 2. Planning Time and Task Completion Time Comparison



5.3 Multi-Agent Coordination Performance

Figure 3 is a comparison of the coordination performance of exemplary autonomous agent architectures. Single-Agent LLM (58.4%), ReAct (72.1%), Tree of Thoughts (74.8%), HuggingGPT (86.3%), and Generative Agents (90.5%) had the lowest Coordination Success Rate. The proposed Hierarchical Multi-Agent LLM Framework achieved the highest coordination success rate of 97.6%, outperforming all baseline methods. These findings indicate that the hierarchical structure of the Supervisor, Planner, Coordinator, Executor, Validator and Shared Memory Module enhances the degree of synchronization amongst the agents, minimizes the communication overhead and increases the degree of collaborative execution of complex autonomous functions.

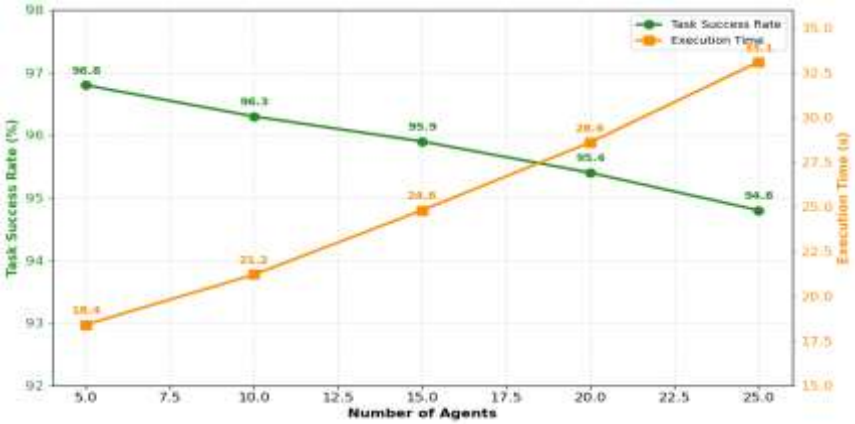
Figure 3. Multi-Agent Coordination Performance Comparison



5.4 Scalability Analysis

Figure 4 shows how the proposed framework can be scaled with the addition of more collaborating agents (5 to 25). The Task Success Rate remained consistently high, decreasing only slightly from 96.8% with 5 agents to 96.3%, 95.9%, 95.4%, and 94.8% with 10, 15, 20, and 25 agents, respectively. At the same time, the Execution Time was gradually growing to 18.4 s to 21.2 s to 24.8 s to 33.1 s with the increase in workload and coordination requirements. Although the framework took longer time to execute, it also ensured a task success rate of over 94% at all the levels of the scalability, which is a strong indication that it can operate efficiently with an increment of the tasks requested. These findings show that hierarchical task breakdown, specialized agent interaction, and shared memory mechanism are an effective way to allocate computational loads and reduce the number of coordination bottlenecks and ensure efficient and scalable autonomous planning and execution of tasks.

Figure 4. Scalability Analysis Under Increasing Task Complexity and Number of Agents



5.5 Ablation Study

The ablation study that has assessed the value of every significant element within the proposed framework is presented in Table 3. The complete framework achieved the best performance with a Task Success Rate of 96.8%, Goal Completion Rate of 95.9%, Plan Validity of 97.2%, Execution Accuracy of 96.4%, and Task Completion Time of 24.8 s. The Task Success Rate decreased to 91.7% and the completion time was 29.6 s when the Supervisor Agent was removed. Excluding the Planner Agent had the greatest impact, decreasing the Task Success Rate to 87.9%, Plan Validity to 84.8%, and increasing the Task Completion Time to 34.7 s. The removal of the Coordinator Agent decreased the Task Success Rate to 89.6 and the removal of the Validator Agent decreased the accuracy of the Execution to 86.7 meaning it had more execution errors. In a similar context, when the Shared Memory Module was removed, the Task Success Rate declined to 88.4% and the completion time was raised to 33.5 s as a result of slower information exchange among agents. These findings confirm that every component contributes to achieving a high level of planning quality, efficiency of coordination, reliability of execution and general scalability of the hierarchical multi-agent LLM framework proposed.

Table 3. Ablation Study of the Proposed Hierarchical Multi-Agent LLM Framework

Framework Configuration	Task Success Rate (%)	Goal Completion Rate (%)	Plan Validity (%)	Execution Accuracy (%)	Task Completion Time (s)
Full Proposed Framework	96.8	95.9	97.2	96.4	24.8
Without Supervisor Agent	91.7	90.4	92.1	91.3	29.6
Without Planner Agent	87.9	86.5	84.8	88.1	34.7
Without Coordinator Agent	89.6	88.7	90.2	89.4	32.8
Without Validator Agent	90.8	89.9	91.4	86.7	28.9
Without Shared Memory Module	88.4	87.3	89.1	88.0	33.5

6. Conclusion

The Hierarchical Multi-Agent LLM Framework of Autonomous Task Planning and Execution that was introduced in this paper incorporates a Supervisor Agent, Planner Agent, Coordinator Agent, Executor Agent, Validator Agent, and Shared Memory Module within a single hierarchical framework. The suggested structure enhances the autonomous task comprehension, hierarchy, coordinated action and adaptive feedback, which allows managing the complex multi-step tasks effectively. Hierarchical agent collaboration was proven to be effective with experimental assessment on benchmark autonomous task-planning datasets showing higher performance relative to representative LLM-based agent frameworks in terms of Task Success Rate, Goal Completion Rate, Plan Validity, Execution Accuracy and Task Completion Time. The framework enhances autonomous multi-agent LLM systems with structured coordination, the improved reliability of executing tasks, scalable task management, and efficient inter-agent communication. The proposed approach can be used in software engineering, automation of enterprises, robotics, scientific reasoning, and other intelligent decision-support applications due to these capabilities. Future studies will incorporate adaptive learning measures, long-term memory efficiency, multi-modal thinking and real-world applications to further improve the scalability, strength and autonomy of hierarchical multi-agent LLM systems.

References

1. Alhassan, A. B., Zhang, X., Shen, H., & Xu, H. (2020). Power transmission line inspection robots: A review, trends and challenges for future research. International Journal of Electrical Power & Energy Systems, 118, 105862.
2. Ananda, M. P., Nagarathna, R., Krishna, L., & Rajeswari, P. (2017, February). Implementation of low cost protocol converter using common media device. In 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA) (pp. 780-783). IEEE.
3. Ali, H., Safdar, R., Ding, W., Zhou, Y., Yao, Y., Yao, L., & Gao, F. (2025). Intelligent machine learning-based multi-model fusion monitoring: application to industrial physio-chemical systems. Control Engineering Practice, 162, 106361.
4. Ali, H., Safdar, R., Rasool, M. H., Anjum, H., Zhou, Y., Yao, Y., ... & Gao, F. (2024). Advance industrial monitoring of physio-chemical processes using novel integrated machine learning approach. Journal of Industrial Information Integration, 42, 100709.
5. Ali, H., Safdar, R., Zhou, Y., Yao, Y., Yao, L., Zhang, Z., ... & Gao, F. (2024). Robust statistical industrial fault monitoring: A machine learning-based distributed CCA and low frequency control charts. Chemical Engineering Science, 299, 120460.
6. Arslan, M., Ghanem, H., Munawar, S., & Cruz, C. (2024). A Survey on RAG with LLMs. Procedia computer science, 246, 3781-3790.
7. Bahr, L., Wehner, C., Wewerka, J., Bittencourt, J., Schmid, U., & Daub, R. (2025). Knowledge graph enhanced retrieval-augmented generation for failure mode and effects analysis. Journal of Industrial Information Integration, 45, 100807.

8. Chand, R., Sharma, B., & Kumar, S. A. (2025). Systematic review of mobile robots applications in smart cities with future directions. Journal of Industrial Information Integration, 45, 100821.
9. Emilia Koskinen. (2026). Adaptive Embedded IoT Platform for Intelligent Data Processing and Communication in Distributed Cyber-Physical Systems. Archives of Embedded and IoT Systems Engineering, 35–41.
10. Erol, K., Hendler, J., & Nau, D. S. (1994, July). HTN planning: Complexity and expressivity. In AAAI (Vol. 94, pp. 1123-1128).
11. Failla, L., Rossoni, M., Quirini, M., & Colombo, G. (2025). Managing lifecycle of product information with an ontology-based knowledge framework. Journal of Industrial Information Integration, 45, 100820.
12. Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. Advances in neural information processing systems, 35, 22199-22213.
13. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems, 33, 9459-9474.
14. Li, X., Wang, S., Zeng, S., Wu, Y., & Yang, Y. (2024). A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. Vicinagearth, 1(1), 9.
15. Li, Y., & Starly, B. (2024). Building a knowledge graph to enrich ChatGPT responses in manufacturing service discovery. Journal of Industrial Information Integration, 40, 100612.
16. Maidanov, K., & Sungho, J. (2025). AI-Integrated System-on-Chip (SoC) Architectures for High-Performance Edge Computing: Design Trends and Optimization Challenges. Journal of Integrated VLSI, Embedded and Computing Technologies, 2(3), 47-55.
17. Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023, October). Generative agents: Interactive simulacra of human behavior. In Proceedings of the 36th annual acm symposium on user interface software and technology (pp. 1-22).
18. Punam, S. R. (2025). Automated Distributed Learning Pipelines for Multi-Agent Graph Intelligence in 6G IoT Systems. SECITS Journal of Scalable Distributed Computing and Pipeline Automation, 2(2), 18-27.
19. Rajeswari, P., & Sasi, S. (2024). Efficient k-way partitioning of very-large-scale integration circuits with evolutionary computation algorithms. Bulletin of Electrical Engineering and Informatics, 13(6), 4002-4007.
20. Raja, M. S., Muniyandy, E., Ambika, K. S., Rajeswari, P., Gadam, H., Mondal, A., & Kumar, M. (2026). Secure Routing and Attack Detection in VANETs Using BCS-DMCO Optimization. SN Computer Science, 7(5), 563.
21. Sindhu, S. (2026). Design of an AI-Driven Communication Framework for Conflict Mediation in Multi-Agent and Multi-Species Environments. Journal of Intelligent Assistive Communication Technologies, 9-16.
22. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., ... & Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. Advances in neural information processing systems, 36, 68539-68551.
23. Shen, Y., Song, K., Tan, X., Li, D., Lu, W., & Zhuang, Y. (2023). Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. Advances in Neural Information Processing Systems, 36, 38154-38180.
24. Uvarajan, K. P. (2025). Secure Digital-Twin-Assisted Multi-Agent Learning Architectures for Reliable Energy-Oriented Decision-Making in Distributed Vehicular Services. Transactions on Internet Security, Cloud Services, and Distributed Applications, 2(3), 1-6.
25. Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., ... & Wen, J. (2024). A survey on large language model based autonomous agents. Frontiers of Computer Science, 18(6), 186345.
26. Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., & Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. Advances in neural information processing systems, 36, 11809-11822.
27. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). React: Synergizing reasoning and acting in language models.