Robot Task Optimizer with Large Language Models

Yuan Zhang, Apratim Banerjee, Shiyun Huang, Tianqi Zeng, Bike Zhang

***Abstract*— This paper explores the challenges of utilizing Large Language Models (LLMs) for task planning within the robotics domain. While LLMs demonstrate potential in reasoning and action planning, their effectiveness diminishes in describing real-world scenarios and intricate instructions. The primary challenge addressed is the gap in the effectiveness of LLMs when applied to complex, real-world scenarios, particularly in robotics, where current LLM-based solutions often fail to optimize task sequences efficiently. In response, we introduce the Large Language Multi-tasking Optimizer (*LLAMO*), which capitalizes on the capabilities of advanced LLMs, such as GPT-4, for task decomposition, models sequential dependencies, and integrates a graph optimization framework. Comprehensive testing on both simulated environments and hardware illustrates significant enhancements in planning efficiency.**

I. Introduction

Large Language Models (LLMs) pre-trained on massive datasets have shown significant advancements in many fields, including natural language processing, computer vision, image generation, etc. Such progress has also notably sparked a growing interest within the realm of robotics, particularly in the field of enhancing task understanding, motion planning, and execution capabilities. This integration aims to bridge the gap between high-level task comprehension and the nuanced execution of tasks by robots in dynamic and complex environments.

Recent work in LLM-based task planning often ignores the critical analysis and optimization of the output. Relying on the raw outputs of LLMs may not guarantee the desired levels of efficiency, repeatability, and accuracy in industrial engineering applications where reliability and long-horizon planning are critical [1]. In this paper, we explore how large language models can be used to instruct a sophisticated full-sized Digit humanoid robot [2] to perform long-horizon planning tasks correctly and efficiently. Our research builds upon recent successes in the application of LLMs, exemplified by GPT-4, for low-level control in robotics [3].

Inspired by this, we introduce the **L**arge **La**nguage **M**ulti-tasking **O**ptimizer (LLaMO), a novel mechanism designed to harness the analytical ability of LLMs within complex scenarios, subsequently refining these insights through established optimization techniques. Our approach unfolds into three key parts. First, we employ a few-shot prompting strategy with state-of-the-art LLMs (e.g., GPT-4) to decompose complex tasks into manageable subtasks, each associated with corresponding actions. Next, we identify and model the sequential dependencies between actions at different subtasks (either by a deterministic function or learned neural network such as Transformer). The final part involves mapping these relationships onto the graph, where the graph optimal solution is derived from calculating the shortest path through critical nodes. We have subjected our methodology to extensive tests in both simulated environments and real-world robots, recording notable enhancements in motion planning across various scenarios. Our contributions can be summarized as follows:

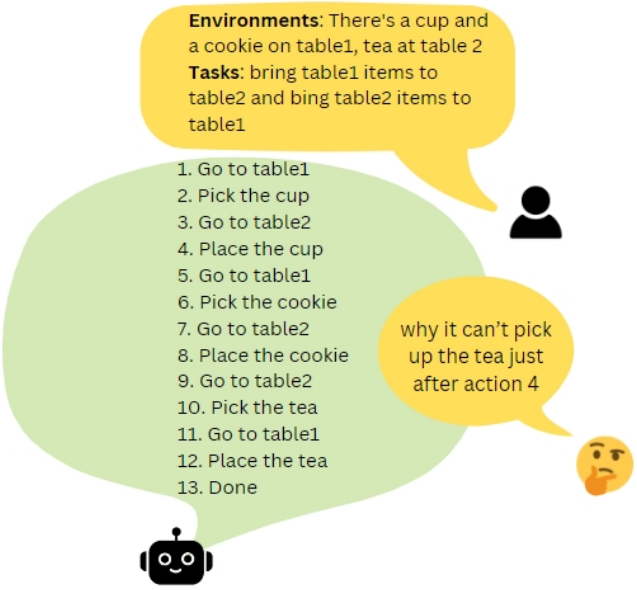


Fig. 1: An example showing direct LLMs output lacks optimal solution

1) We propose LLaMO, a mechanism that enhances environmental analysis capabilities in robotics through the integration of LLM-based reasoning with graph optimization techniques.
2) The development of a Retrieval Augmentation Generation(RAG) based information retrieval module, enabling robots to adhere more effectively to workplace rules or real-time information.
3) Demonstrated improvements in both accuracy and efficiency, as evaluated by extensive testing on simulation and experiments with actual robots.

A. Related Work

Large Language Models for Robotics: In recent years, LLMs have demonstrated significant progress in enhancing various facets of robotics, including task planning, manipulation, and control. Notable contributions in this domain highlight the versatility of LLMs in addressing complex challenges [4] [5] [5] [6] [7] [8]. Wang et al. [3] explored the application of LLMs to prompt robots with low-level

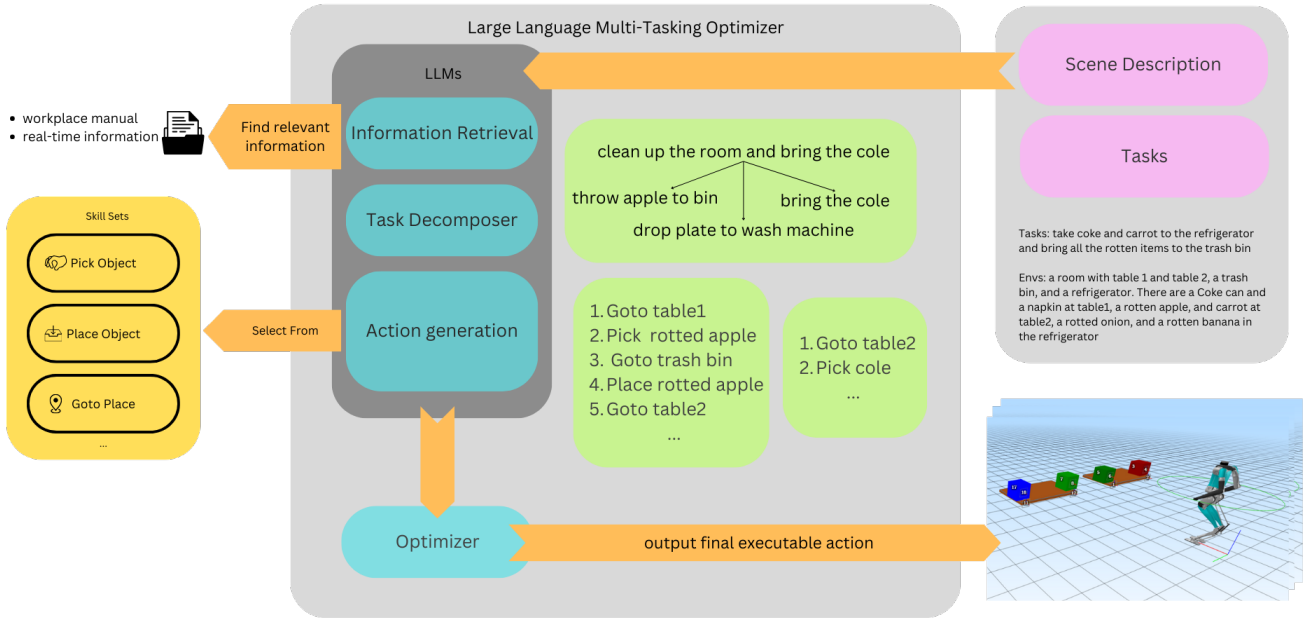


Fig. 2: Overview of LLaMO workflow. The framework processes user queries by retrieving necessary information and breaking down the main tasks into smaller, manageable ones for the LLM. It selects suitable skills to complete these subtasks, which are then optimized and transformed into executable robot commands.

controls, specifically focusing on the task of walking. The development of RoboTool by Xu et al. [9] introduces a flexible and user-friendly system for long-term planning and creative utilization of robots. This tool showcases the potential of LLMs in enabling robots to engage in creative and extended planning processes. Brohan et al. [10] proposed the Robotics Transformer, a significant advancement that promises high-capacity architectures capable of absorbing diverse robotic data while executing real-world tasks. This transformative model class holds the potential to address challenges in generalization and adaptability within robotic applications. In the realm of human-robot interaction, Ahn et al. [11] introduced task-planning user interfaces that enable the execution of complex tasks specified by humans. This demonstrates the practicality of integrating LLMs into user-friendly interfaces for seamless human-robot collaboration. The scalability of robotic deployment is exemplified in the works of Ahn et al. [12], and Wei et al. [13]. Ahn et al. leverage vision-language models to enable robots to navigate unseen environments effectively. Wei et al. showcase the progress achieved through massive datasets, emphasizing the use of foundation models to prompt reasoning tasks and action plans. Furthermore, the integration of LLMs in robotics has sparked a growing interest in human-robot interaction [14]. Zhang et al. emphasize the importance of understanding and addressing challenges in this domain for the successful deployment of robots in real-world scenarios [15].

LLM-based task planning: There has been significant effort to address the gap of optimization and accuracy in task planning for robotics [16] [17] [18] [19] [20]. To address the gap between high-level plans and low-level execution, Guo et al. [21] introduce DoReMi, which uses LLMs in a dual role, aiding in high-level planning and generating constraints that indicate potential misalignments during execution. Vision Language Models (VLMs) are used as feedback to recover from misalignments. Gramopadhye et al. [22] propose an approach that incorporates environmental objects and object relations as additional inputs into LLM action plan generation. Another novel approach, Robotic Vision-Language Planning (ViLa) [23], directly integrates perceptual data into its reasoning and planning process. This includes spatial layouts and object attributes, enabling a more comprehensive grasp of the physical environment. We also have seen failure and feedback mechanisms to improve task planning. REFLECT [24], a framework that queries LLMs for failure reasoning based on a hierarchical summary of a robot's past experiences generated from multi-sensory observations in order to correct the failures. There has also been work in grounding these plans in expansive, multi-floor, and multi-room environments. SayPlan [25], a scalable approach to LLM-based, large-scale task planning for robotics using 3D scene graph (3DSG) representations.

Long-Term Planning and Optimization: Task planning utilizing LLMs aims to enhance model generalization, yet they encounter challenges in dynamically capturing the initial

state of planning problems [26] [27] [28] [29]. Addressing this issue, Birr et al. [30] proposes AutoGPT+P, a novel system that amalgamates an affordance-based scene representation with a planning system. AutoGPT+P not only achieves symbolic planning with arbitrary objects but also executes plans specified by users in natural language. This approach automatically corrects semantic and syntactic errors, surpassing the state-of-the-art LLM-based planning method SayCan, with an impressive success rate of 98% on the SayCan instruction set [11]. In addition to AutoGPT+P, traditional robot task planning methods grapple with challenges in unstructured environments and complex tasks. A notable contribution by Zhen et al. [31] introduces the LLM prompt template, Think_Net_Prompt. This method adopts a progressive task decomposition strategy, generating a task tree to effectively reduce planning volume for each task. While exhibiting commendable performance in handling specified code formats, understanding task relationships, and extracting parameters from text descriptions, there are acknowledged challenges. These include limitations in handling complex task logic and ambiguity in precise assembly locations. Text2Motion [1] extends this paradigm by actively resolving geometric dependencies across sequences of skills so that the generated plans are not only semantically meaningful but also feasible in the physical world.

II. Method

To facilitate the generation of optimal action plans in response to complicated user commands, LLaMO employs a hierarchical strategy for task decomposition and integration. Upon receiving the command, the system evaluates the current environmental state and consults available knowledge relevant to these tasks. Utilizing LLMs' capabilities and gathered information, the tasks are partitioned into discrete and manageable subtasks. Then, it generates specific actions while concurrently analyzing the information of these actions. All of these decompositions and analyses result in the later optimization process, which constructs the network of interdependent actions across different subtasks. The final part of this process is to utilize graph-theoretical optimization techniques to ascertain the most efficient sequence of actions, thereby determining the optimal solution path. An overview of our approach is shown in Fig. 2.

A. Problem Formulation

The task planning problem can be formally defined within the framework of a given scene description S, the prior knowledge, and user-specified command C. The objective is to generate an optimal plan P^* that translates command C into a coherent sequence of robotic actions $A = \{a_1, a_2, \ldots, a_n\}$. An action $a_i \in A$ is characterized as the execution of a capability by the agent π, parameterized by arguments $\rho = (\rho_1, \ldots, \rho_n)$, where the argument ρ is subject to objects in S. The set of capabilities C_π for each agent, which encapsulates the programs for low-level action execution on the robot, would be dynamically determined at runtime based on the S.

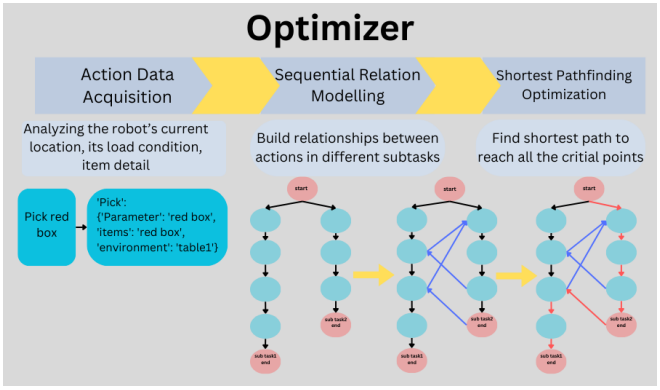


Fig. 3: Components of the Optimizer and example illustration

We denote $T = \{T_1, T_2, \ldots, T_m\}$ as the set of subtasks generated from command C via few-shot prompting techniques with LLMs. Each subtask T_i corresponds to a specific subset of actions $A_i \subseteq A$, formulated to address the subtask effectively. For each T_i, the task decomposition system identifies a set of viable actions A_i and assesses their applicability and impact, according to the current scene S and the external knowledge K. This process is represented by the function:

$$A_i = f(T_i, K, S)$$

where f represents the analyzing operation performed by the LLMs to propose and evaluate potential actions.

For the further optimization step, we construct a directed graph $G = (V, E)$, with vertices V symbolizing the actions and states and edges E delineating the transition between actions, factoring in availability which are derived from criteria such as efficiency, task rationality, and predetermined workplace regulations. The ultimate aim is to determine the most cost-effective path P^* within graph G, minimizing the total cost $Cost(P)$ while adhering to the constraints dictated by the environmental state S and the task specifications. Mathematically, the optimal path is obtained through the optimization:

$$P^* = \arg\min_{P \in \mathcal{P}} Cost(P) \wedge T \in P^*$$

B. LLaMO Framework

We propose LLaMO, a mechanism designed to address complex and multi-task planning challenges through the integration of LLMs in a few-shot learning context combined with graph optimization methodologies. The ultimate goal of LLaMO is to interpret natural language instructions and convert them into executable commands that are aligned with the robot's capabilities and specified parameters. The architecture of LLaMO is hierarchical, comprising four key components: an *information retrieval system*, a *task decomposition module*, an *action generation mechanism*, and an *optimization framework*. This structure enables LLaMO to efficiently process and act upon instructions, facilitating enhanced performance in robotic motion planning and task execution.

1) Information Retrieval System: In the context of the LLaMO framework, the Information Retrieval component acts as the pivotal interface for integrating predefined work environment regulations, current robotic operational contexts, and user instruction, thereby providing up-to-date data to augment decision-making processes. For instance, when a robot detects a radiation warning symbol, this module ensures subsequent navigational strategies are recalibrated to eschew the marked area in adherence to the safety guidelines. This functionality leverages the Retrieval Augmentation Generation (RAG) architecture, which combines the strengths of traditional information retrieval techniques with the generative capabilities of state-of-the-art wording embedding, specifically utilizing OpenAI's text-embedding-ada-002 for this study.

The operational mechanism is composed as follows: preliminary to task execution, a preprocessing phase is applied to the external information, segmenting its content into discrete chunks. Each chunk is then encoded into vector form utilizing the aforementioned text embedding technology and stored within a database. Upon receiving a natural language instruction, alongside the assessments of the robot's current operational environment, these inputs are processed through the same embedding technique. Subsequent retrieval is metricized by cosine similarity measures, with a selection of the Top-K highest-ranking documents. This information subsequently enriches the LLMs' output, providing a richer understanding of the task's condition and enabling more accurate decomposition and planning.

2) Task Decomposition: Using the information gained from the previous section, LLaMO then systematically decomposes the complicated and multi-task instructions into a series of subtasks T_i. Each subtask T_i represents a portion of the work T that contributes towards achieving the final goals. This decomposition is guided by the LLM's understanding of the task domain, the logical sequence of actions required, and the robot's capabilities. Because some subtasks may have dependencies or need to be executed in a specific order, LLaMO also models the sequential and conditional dependencies between actions. This step ensures that the execution plan is coherent and logically structured, preventing conflicts and inefficiencies.

3) Action Generation: The action generation module within LLaMO operates by analyzing subtasks into executable commands that are tailored to both the robot's capabilities and the environmental conditions. This module informs itself of the robot's abilities through in-context learning, and a series of examples are presented in the prompts to illustrate the robot's skill sets. Subsequently, it maps the identified subtasks T_i to the robot's skill set, ensuring that each action is feasible and aligned with the robot's capabilities. Finally, it specifies the parameters of the chosen skill, which may be the target positions, object identifiers, etc. Specifically, it assigns each subtask T_i a critical node at the end of the action sequence, and reaching this critical usually indicates the finish of this subtask T_i. All of these critical nodes play a crucial role in the later shortest path optimization step.

4) Optimizer: Inspired by the hierarchical structure of modern computer systems, which consists of components for information analysis, optimization, and final code generation, our proposed LLaMO adopts a similar framework. This hierarchical organization ensures efficient information gathering and utilization, leading to optimized solutions. Specifically, our optimizer consists of three key components: action data acquisition, sequential relation modeling, and shortest pathfinding optimization. This structure enables the systematic decomposition of complex tasks, modeling of dependencies between subtasks, and derivation of optimal solutions through graph optimization techniques.

Action Data Acquisition: In this module, all the actions of each subtask T_i are forwarded to the LLMs with the aim of predicting useful information during the execution of each subtask T_i, which involves monitoring the robot's current location, its load condition, and the arrangement of items within the operational environment. This data collection serves as the foundation for the intricacies of task interdependencies that are thoroughly captured and exploited.

Sequential Relation Modelling: This phase focuses on developing sequential relationships, identifying actions that can be executed consecutively across different sub-tasks T_i. The actual implementation for establishing these links varies, supporting deterministic rules or adaptive algorithms learned from data. This approach changes the transition of action representations from a hierarchical structure to a more versatile graph-based structure, enhancing the representation of inter-task relationships and preparing for later optimization techniques.

Shortest Pathfinding Optimization: After constructing a comprehensive network comprising action nodes and their interconnections, our subsequent objective is to identify a feasible solution that minimizes cost while still adhering to certain task-specific requirements and constraints. To solve this, we implement a graph shortest path algorithm to ascertain the most efficient route through the critical nodes within the action graph. Specifically, in our subsequent experiments, we opt for Dijkstra's algorithm to determine the shortest paths. The ultimate outcome is an optimized sequence of actions that efficiently accomplishes the intended tasks. The state transition of Dijkstra's algorithm can be expressed by the following expression:

$$d(v, T') = \min_{(u,v)\in E} \begin{cases} d(u, T') + Cost(u, v), & v \notin T' \\ d(u, T' \setminus \{v\}) + Cost(u, v), & v \in T' \end{cases}$$

, where $d(v, T')$ denotes the shortest path distance from the source to vertex v, given the current node v and the set of completed subtasks is T'. The optimal solution is $\min_{u\in T} d(u, T)$. The function $Cost(u, v)$ represents the weight of the edge connecting nodes u and v. The goal is to find the minimum value of $d(u, T'), \forall u \in T$, which corresponds to the optimal solution.

III. Results

We conduct experiments to answer the following questions:

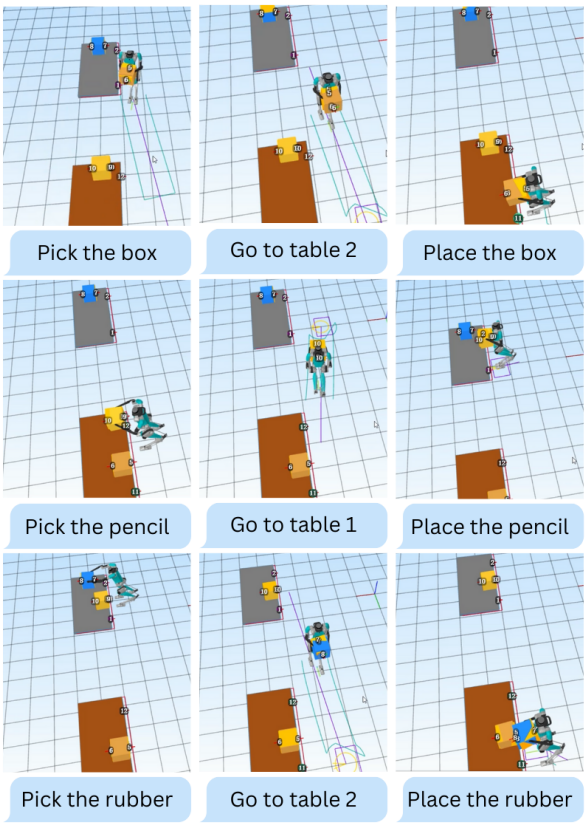


Fig. 4: Example experiment under Agility simulator

1) Does LLaMO work for the real-world scenario?
2) How does its performance compare with other LLM-based task planning strategies?
3) To what extent the information retrieval part helps better understand the working environment?
4) Does the multi-task optimizer support dynamic optimization during operation?
5) Can LLaMO help robots plan with LLMs more efficiently on real robots?

A. Simulation Experiment Setup

In our study, we leverage the GPT-4 model provided by OpenAI for all the experiments. To ensure the consistency and reliability of the results, we configured the model's temperature parameter to zero and only used the top probability response. For our simulation environment, we selected the Agility humanoid robot and constrained the robot's skill set to three fundamental operations: "Pick", "Place", and "Go To". These operations were selected for their critical importance of humanoid robot working environments, like warehouses and logistics. We conducted validations of the robot's output actions within the Agility simulator by designing appropriate environments. A series of prompts were formulated to implement the few-shot learning strategy, enabling GPT-4 to comprehend the robot's operational context and its limited skill set efficiently.

Results Comparison	
GPT4	LLaMO
Go to table 1	Pick the box
Pick the box	Go to table 2
Go to table 2	Place the box
Place the box	Pick the pencil
Go back to table 1	Go to table 1
Pick the rubber	Place the pencil
Go to table 2	Pick the rubber
Place the rubber	Go to table 2
Pick the pencil	Place the rubber
Go to table 1	
Place the pencil	

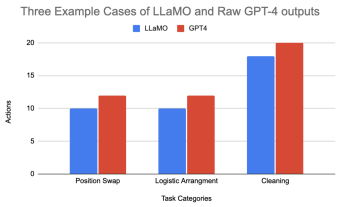


Fig. 5: The right panel displays comparative results across three distinct categories, while the left panel details the outcomes of the position swap experiment

B. Optimize Action Sequences

To more comprehensively evaluate the performance of our method in contrast to GPT-4 outputs, we conducted comparative experiments across three distinct categories. Furthermore, we present a detailed analysis of the outcomes in a specific case involving position swapping. The scenario is "a box and a rubber are located at table 1, and a pencil is at table 2", with the task being to "swap the items' positions between table 1 and table 2". All the details are presented in Fig.5. GPT-4's suggested processes included unnecessary steps, such as returning to table 1 from table 2 without transporting any items, a clear indication of process redundancy on the other side, LLaMO generates optimized procedures by eliminating such inefficiencies.

C. Competitive Analysis

To further validate our method against other existing methods, we devised 10 test cases covering diverse aspects of robot planning, each with a different environment and task descriptions. We tested the accuracy and optimizing ability of LLaMO with raw GPT-4 output, LLM+P and Tree of Thought (ToT) methods. Each test case was tested 10 times on the same solver as LLM results might differ despite being given the same instructions. We tabulated the accuracy of each method tested on each test case.

An output is considered "correct" when the sequence of subtasks enables the robot to reach the goal specified in the query, the efficiency of the output action sequence is not taken into consideration in this experiment. If there is any mistake, for example, misplacing an item or trying to pick an item when the robot's hands are not free, the output will be deemed as "wrong". Table I below shows our result.

Method	**Average Success Rate**
Raw GPT-4	80%
LLM+P	70%
Tree of Thought	90%
LLaMO	**94**%

TABLE I: Average success rate of LLM models over 10 test cases.

With LLaMO framework, our method achieved the highest success rate in solving various test cases, including those with higher complexity which requires long-term planning and a good understanding of the dependencies of actions.

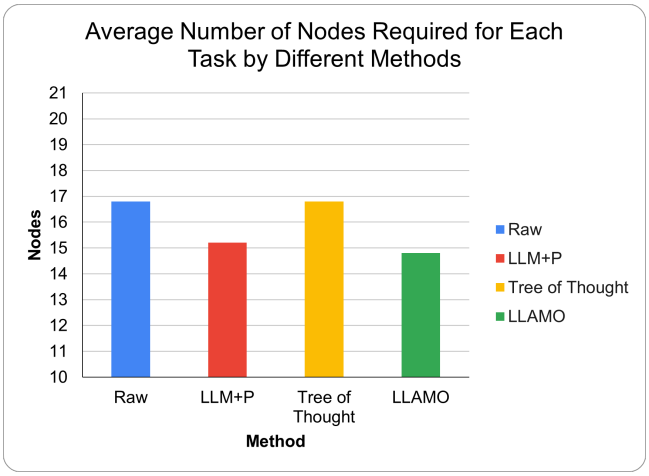


Fig. 6: Average number of action nodes from raw GPT-4, PDDL, ToT and LLaMO

Results Comparison	
LLaMO with retrieval	LLaMO without retrieval
Go to desk	Go to desk
Pick beef soup	Pick curry
Go to table 3	Go to table1
Place beef soup	Place curry
Go to desk	Go to desk
Pick noodle	Pick noodle
Go to table 2	Go to table 2
Place the noodle	Place the noodle
Go to desk	Go to desk
Pick curry	Pick beef soup
Go to table1	Go to table3
Place curry	Place beef soup

TABLE II: Output result comparison between GPT-4 and LLaMO.

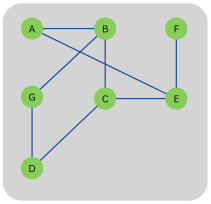
Techniques such as the Tree of Thought have notably enhanced accuracy compared to the baseline GPT outputs. It is important to highlight that the LLM+P model exhibited the lowest success rate, indicating current LLM's limitations in accurately modeling situations and queries to the PDDL (Planning Domain Definition Language) framework.

For test cases that were solved by all four methods with 100% accuracy, we counted the number of action nodes required to correctly perform the query to observe the optimizing capabilities of LLaMO. Fig. 6 depicts the average number of actions required for various methods to solve tasks. It is apparent that our method maintains superior performance compared to others. Notably, LLM+P also achieves commendable scores in this experiment, highlighting the significant potential of integrating traditional optimization techniques to enhance LLM outputs.

D. Information Retrieval Experiment

Furthermore, we study the effectiveness of the information retrieval module on the appropriate decision-making capabilities. We conducted a series of experiments, testing the output actions' difference between LLaMO with retrieval module and without. In our experiment, we start by collecting some external information that is designed for our experimental requirements. these would be then chunked as 200-size pieces, each converted into embeddings utilizing OpenAI's text embedding API. When conducting a task query, both the content of the query and the scene description are converted into embeddings via the same API. We then identify the top 5 results based on cosine similarity and incorporate this information as the additional prompts provided to GPT-4. The core objective of our experiment is to ensure whether LLaMO could accurately identify the appropriate segment of text and subsequently generate the right actions based on this information. Table II is a typical example to demonstrate the output difference by applying the retrieval module. In this case, our task is to serve the meal to the customer and the scene description about what meal has been prepared and what each customer is waiting for. The additional information contains the waiting time for each customer, table 3's customer with the longest waiting time, and table1's get order recently. It can be seen from the output result that LLaMO with the retrieval module prioritizes the meal that has the longer waiting time, which better accommodates real-world scenarios.

E. Simulation Experiment on Go1



LLaMO vs Raw GPT-4					
Framework	C,F	B,G,F	C,E,F,B	C,E,G,D,B	G,D,B,C,E,F
LLaMO	**4**	**6**	**4**	**6**	**8**
GPT-4	6	8	6	9	9
Nodes Reduced	2	2	2	3	1

TABLE IV: Go1 on Maze travelling

Fig. 7: The left panel presents the visual layout of the maze, while the right panel shows a comparative analysis of action outcomes for navigating to specific maze locations.

To demonstrate the adaptability of LLaMO across a broad spectrum of robotic platforms, test robots with different capabilities. We set up an experiment with a quadruped robot, Go1, which represents a significant deviation from our initial models, with its primary function limited to locomotion. The experimental setup and results are depicted in Figure 6. When provided with coordinates for target locations, LLaMO demonstrates its capability to devise optimal navigation strategies.

F. Hardware Testing on Go1

Furthermore, we test the system's capability to optimize tasks dynamically, which incorporates the additional tasks during its operational phase. The specifics of this experiment are illustrated in Fig. 7. The initial task is to let the Go1 robot navigate a predetermined loop ($A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$). In the middle of its operation, we introduce a supplementary task: directing the robot to point E. Initially, following the raw GPT-4 suggested action sequence, the robot would complete the original loop, returning to point A and then proceeding to point E. This approach is straightforward but also is far from efficient.

Leveraging LLaMO, we observe a significant optimization in the action sequence. The revised path directed the robot from point A to B, then B to C, diverging to point E, and subsequently returning from E to C. From there, it continued from C to D, and ultimately, from D back to A. This optimized route proved substantially more efficient than the initial strategy, underscoring LLaMO's potential to enhance operational efficiency through intelligent task sequencing.

IV. Discussion

After validating our approach with these experiments, we next provide a discussion on what we learned and the limitations of the current approach.

A. Leveraging Hierarchical LLMs in Task Planning

It is interesting to note that hierarchical LLMs analysis plays a crucial role in robot task planning, we utilize the LLM to decompose task, generate action and analysis action information. All of these steps systematically leverage LLMs to generate in-depth information for to aid decision-making. Traditional LLM outputs, which typically adhere to an end-to-end format, often fall to deliver accurate and optimized sequences of actions. More than that, they usually fail to cooperate the environment with workplace rules and external information. However, by adopting a thoughtfully designed hierarchical structure, LLMs can substantially improve their success rates in executing complex task planning. This shift not only enhances the efficiency of LLMs in practical applications but also opens new avenues for exploring sophisticated models of decision-making and problem-solving. While we provide an example of the hierarchical task analysis , the structure for LLMs to do the task analysis is still under-explored.

B. Post-optimization Leads to Efficient LLM Output

Our experiment illustrates that the application of a post-optimization method significantly reduces unnecessary actions and redundancies in tasks. Unlike other optimization approaches that often overlook the potential interconnections between subtasks, our method uniquely identifies and leverages these links. By employing graph-theoretical optimization techniques, we effectively establish a connection between theoretical models and their real-world applicability, bridging the gap and enhancing overall task efficiency.

C. Limitation

The current framework exhibits several limitations. Firstly, it comes out when handling tasks with constraints. Specifically, our framework lacks a clear mechanism to determine if tasks are redundant or if they are left unaddressed due to constraints within the tasks. An illustrative example would be the tasks requiring the swapping of two tables' positions, compounded by a constraint that restricts the maximum number of items permissible on a table. Sometimes, it may be necessary to move items to another table to mitigate this constraint. However, from our experiments, such actions might be treated as redundant tasks.

Another major limitation is the speed of inference in hierarchical LLMs, due to the extra steps compared with direct LLM outputs. Typically, robots spend some time formulating subsequent plans prior to action execution, which is generally acceptable. However, given that our framework supports task optimization during operational phases, the delay in awaiting further action outputs from the robot could present operational challenges.

V. Conclusion

Undoubtedly, the ChatGPT4 Large Language Model (LLM) stands as a potent tool, offering immense potential for analyzing and resolving complex problems that traditionally demand significant human effort. However, during our extensive testing and experimentation, we discovered certain limitations in ChatGPT4's ability to effectively guide robots through tasks. These limitations often manifested as redundancies and unnecessary complexities, hindering the efficiency and fluidity of task completion. Developing a solution that could harness the power of LLMs like GPT-4 while integrating advanced graph optimization methodologies seemed fruitful as our efforts to build LLaMO (Large Language Multi-tasking Optimizer) helped solve these challenges.

The LLaMO system comprises four integral components: Retrieval Augmentation Generation (RAG), Task Decomposition, Action Generation, and Optimizer. These components effectively integrate regulations and user instructions, break down tasks into manageable sub-tasks, analyze subtask information, generate optimized action plans to boost efficiency, and derive optimal solutions for task planning.

In our experiments, we showcased LLaMO's superiority over other LLM-based task-planning methodologies, gauging its precision and efficiency across various test scenarios. Notably, we underscored LLaMO's adeptness in environmental analysis and dynamic resource allocation, reinforcing its robustness and versatility through extensive testing on both agile humanoid robots and real quadrupedal robots.

The implications of our research extend far beyond the confines of the laboratory, promising to revolutionize long-term horizon task planning across a myriad of industries. By harnessing the computational prowess of LLMs like GPT-4, coupled with sophisticated optimization techniques, LLaMO offers unparalleled efficiency, repeatability, and accuracy in tasks ranging from logistics and distribution to manufacturing and beyond. As we continue to refine and expand upon the capabilities of LLaMO, we envision a future where robots seamlessly integrate into various facets of human life, augmenting our capabilities and unlocking new frontiers of productivity and innovation.

References

[1] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *arXiv preprint arXiv:2303.12153*, 2023, submitted on 21 Mar 2023 (v1), last revised 26 Nov 2023 (this version, v5). [Online]. Available: https://arxiv.org/abs/2303.12153

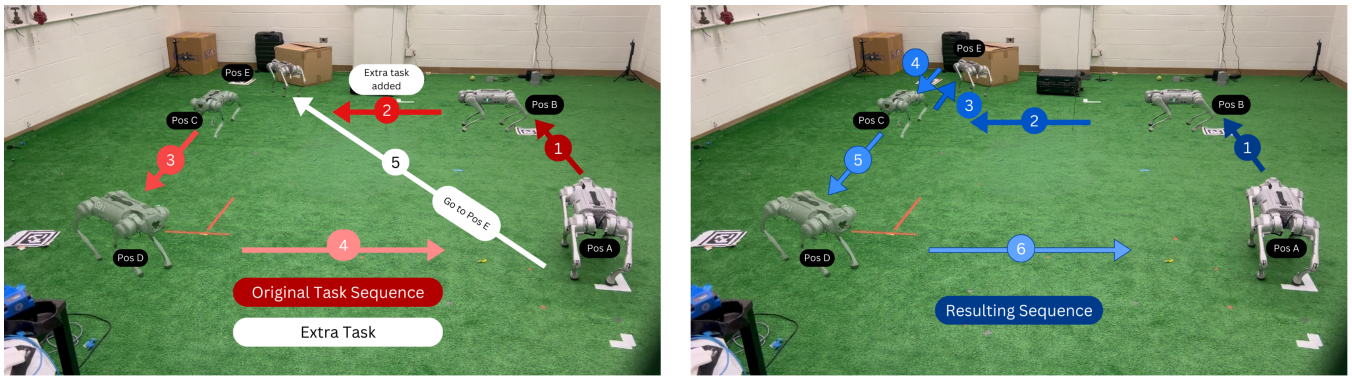


Fig. 8: Dynamic allocation of extra task during execution, without and with LLaMO

[2] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Learning humanoid locomotion with transformers," *arXiv preprint arXiv:2303.03381*, 2023.

[3] Y.-J. Wang, B. Zhang, J. Chen, and K. Sreenath, "Prompt a robot to walk with large language models," *arXiv preprint arXiv:2309.09969*, 2023.

[4] H. Zhen, X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong, and C. Gan, "3d-vla: A 3d vision-language-action generative world model," *arXiv preprint arXiv:2403.09631*, 2024.

[5] F. Xie and S. Schwertfeger, "Empowering robotics with large language models: osmag map comprehension with llms," *arXiv preprint arXiv:2403.08228*, 2024, submitted on 13 Mar 2024.

[6] J. Liu, Y. Yuan, J. Hao, F. Ni, L. Fu, Y. Chen, and Y. Zheng, "Enhancing robotic manipulation with ai feedback from multimodal large language models," *arXiv preprint arXiv:2402.14245*, 2024.

[7] W. Liang, G. Sun, Q. He, Y. Ren, J. Dong, and Y. Cong, "Never-ending embodied robot learning," *arXiv preprint arXiv:2403.00336*, 2024.

[8] H. Zhao, C. Ma, G. Wang, J. Su, L. Kong, J. Xu, Z.-H. Deng, and H. Yang, "Empowering large language model agents through action learning," *arXiv preprint arXiv:2402.15809*, 2024.

[9] M. Xu, P. Huang, W. Yu, S. Liu, X. Zhang, Y. Niu, T. Zhang, F. Xia, J. Tan, and D. Zhao, "Creative robot tool use with large language models," *arXiv preprint arXiv:2310.13065*, 2023.

[10] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2023.

[11] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. Jauregui Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2023.

[12] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, B. Ichter, A. Irpan, N. Joshi, R. Julian, S. Kirmani, I. Leal, E. Lee, S. Levine, Y. Lu, S. Maddineni, K. Rao, D. Sadigh, P. Sanketi, P. Sermanet, Q. Vuong, S. Welker, F. Xia, T. Xiao, P. Xu, S. Xu, and Z. Xu, "Autort: Embodied foundation models for large scale orchestration of robotic agents," *arXiv preprint arXiv:2401.12963*, 2023.

[13] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.02810v6*, 2023.

[14] B. Zhang and H. Soh, "Large language models as zero-shot human models for human-robot interaction," *arXiv preprint arXiv:2303.03548*, 2023.

[15] F. Zeng, W. Gan, Y. Wang, N. Liu, and P. S. Yu, "Large language models for robotics: A survey," *arXiv preprint arXiv:2311.07226*, 2023.

[16] T. Kagaya, T. J. Yuan, Y. Lou, J. Karlekar, S. Pranata, A. Kinose, K. Oguri, F. Wick, and Y. You, "Rap: Retrieval-augmented planning with contextual memory for multimodal llm agents," *arXiv preprint arXiv:2402.03610*, 2024.

[17] V. Bhat, A. U. Kaypak, P. Krishnamurthy, R. Karri, and F. Khorrami, "Grounding llms for robot task planning using closed-loop state feedback," *arXiv preprint arXiv:2402.08546*, 2024.

[18] K. Shirai, C. C. Beltran-Hernandez, M. Hamaya, A. Hashimoto, S. Tanaka, K. Kawaharazuka, K. Tanaka, Y. Ushiku, and S. Mori, "Vision-language interpreter for robot task planning," *arXiv preprint arXiv:2311.00967*, 2023.

[19] J. Brawer, K. Bishop, B. Hayes, and A. Roncone, "Towards a natural language interface for flexible multi-agent task assignment," *arXiv preprint arXiv:2311.00153*, 2023.

[20] M. A. Graule and V. Isler, "Gg-llm: Geometrically grounding large language models for zero-shot human activity forecasting in human-aware task planning," *arXiv preprint arXiv:yyyy.xxxxx*, 2024.

[21] Y. Guo, Y.-J. Wang, L. Zha, Z. Jiang, and J. Chen, "Doremi: Grounding language model by detecting and recovering from plan-execution misalignment," *arXiv preprint arXiv:2307.00329*, 2023.

[22] M. Gramopadhye and D. Szafir, "Generating executable action plans with environmentally-aware language models," *arXiv preprint arXiv:2210.04964v2*, 2023, submitted on 10 Oct 2022 (v1), last revised 2 May 2023 (this version, v2).

[23] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao, "Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning," *arXiv preprint arXiv:2311.17842v2*, 2023.

[24] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," *arXiv preprint arXiv:2306.15724v4*, 2023.

[25] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning," *arXiv preprint arXiv:2307.06135v2*, 2023.

[26] Y. Wei, S. Fu, W. Jiang, J. T. Kwok, and Y. Zhang, "Rendering graphs for graph reasoning in multimodal large language models," *arXiv preprint arXiv:2402.02130*, 2024.

[27] A. Uzunoglu, A. R. Safa, and G. G. Şahin, "Paradise: Evaluating implicit planning skills of language models with procedural warnings and tips dataset," *arXiv preprint arXiv:2403.03167*, 2024.

[28] Y. Wang, Z. Wu, J. Yao, and J. Su, "Tdag: A multi-agent framework based on dynamic task decomposition and agent generation," *arXiv preprint arXiv:2402.10178*, 2024.

[29] T. Huang, Z. Sun, Z. Jin, G. Li, and C. Lyu, "Knowledge-aware code generation with large language models," *arXiv preprint arXiv:2401.15940*, 2024.

[30] T. Birr, C. Pohl, A. Younes, and T. Asfour, "Autogpt+p: Affordance-based task planning with large language models," *arXiv preprint arXiv:2402.10778*, 2024.

[31] Y. Zhen, S. Bi, L. Xing-tong, P. Wei-qin, S. Hai-peng, C. Zi-rui, and F. Yi-shu, "Robot task planning based on large language model representing knowledge with directed graph structures," *arXiv preprint arXiv:2306.05171*, 2023.