Proceedings of the 44th Chinese Control Conference
July 28-30, 2025, Chongqing, China

GMATP-LLM: A General Multi-Agent Task Dynamic Planning Method using Large Language Models

Xiangkun Deng[1], Gang Tao[1], Chenxu Wen[1,3], Xi Zhang[1], Zhiyang Ju[1,2], Jianwei Gong[1,2,3]

1. Intelligent Vehicle Research Center, Beijing Institute of Technology, Beijing 100081, China
E-mail: gongjianwei@bit.edu.cn

2. Zhengzhou Research Institute, Beijing Institute of Technology, Zhengzhou 450053, China

3. Beijing Institute of Technology, Zhuhai 519088, China

Abstract: In this work, we introduce a generalized task planning innovation framework named GMATP-LLM, which is designed for multi-agent systems. This method utilizes Chain-of-Thought (CoT) prompting to enable Large Language Models (LLMs) to perform task decomposition and assignment processes, transforming high-level task instructions into a set of sub-tasks. Based on the assignment strategy, it generates a PDDL goal plan, which is solved by an intelligent planner to generate a sequence of actions. The framework introduces a multi-agent three-dimensional Spatio-Temporal Motion Corridor (STMC) to constrain and optimize the parallel motion of the agents, improving system efficiency. This method combines the reasoning capabilities of LLMs and the fast solving advantage of the intelligent PDDL planners. It has been verified through simulation and real-world experiments across various task categories, achieving favorable results in multi-agent task planning.

Key Words: multi-agent task planning, large language models, PDDL planning

1 Introduction

In recent years, with the continuous development of artificial intelligence technologies,research and applications in multi-agent task planning systems have become increasingly widespread. These include tasks such as disaster rescue operations in outdoor scenes[1], underwater detection missions[2], multi-vehicle transportation tasks[3], and indoor tasks such as household chores[4], navigation tasks[5], and human-machine interaction tasks[6]. These systems, composed of multiple agents, homogeneous or heterogeneous, can leverage the advantages of each unit and, through collaborative cooperation among agents, provide great potential and flexibility for solving complex task planning challenges. However, the differences in attributes and skills among agents add complexity to the system's task planning. The key research challenge lies in how to improve the efficiency of task execution while ensuring the correct assignment of tasks, which requires the assistance of external domain-specific knowledge related to the tasks.

Traditional multi-agent systems for task planning in specific scenarios have mature research with fixed algorithms[7].However, when dealing with diversified scenarios and complex environments, these systems face challenges[8], with weaker generalization and reasoning abilities, especially when faced with vague natural language task instructions, which require extensive prior external knowledge for assistance. The emergence of large language models, such as GPT-3.5[9], GPT-4o[10], Llama-3[11], and Qwen-2.5[12], have demonstrated excellent capabilities in natural language understanding, logical reasoning, and scene generalization, offering solutions for generalized task planning in complex scenarios. Considering the slower efficiency of LLMs when generating long context content for complex task planning, traditional symbolic planners can quickly find optimal solutions when facing well-defined tasks.

Therefore, in this paper, we propose GMATP-LLM, a generalized and dynamic multi-agent task planning system framework that combines LLMs and PDDL symbolic planners. Using CoT chain prompting, LLMs are guided to decompose complex tasks into sub-tasks. Based on task assignment strategies, LLMs assign tasks to agents and generate PDDL goal plans. The symbolic planner quickly solves the sub-task goals for each agent, while also considering the influence of dynamic environments. It calls the agent action skill library to perform actions. Through modular invocation, the task types are not limited to indoor environments. Moreover, multi-agent motion planning is considered during the execution of actions to avoid collisions, and an method for generating three-dimensional Spatio-Temporal Motion Corridors is employed to generate optimal motion control trajectories.

2 Related Works

This section first introduces the application of classical artificial intelligent planners in task planning within the state spaces of agents (such as robots, Autonomous vehicles, unmanned aerial vehicles, etc.). It then discusses the implementation approaches of traditional hierarchical multi-agent task planning, highlighting the advantages of these methods (such as quick solutions for specific scenarios) and their limitations (such as a lack of correctness and robustness guarantees). Finally, the development and key role of LLMs in agent task planning are introduced.

Classical AI Planning for Agent. Traditional intelligent planning has been extensively researched and is well-established in the field of artificial intelligence. One mainstream intelligent planning method is PDDL (Planning Domain Definition Language) [13] and intelligent planners. PDDL excels in terms of correctness, completeness, and efficiency for planning single-agent tasks[14]. It divides the model of a planning problem into two parts: domain description and problem description. The planner solves the problem by generating an ordered sequence of actions between the initial and goal states[15]. However, it can only solve non-hierarchical planning problems. Based on this, the improved HDDL[16] method was proposed, which can

express the requirements of hierarchical task planning. HTN (Hierarchical Task Network) planning is another approach for handling hierarchical task planning[17].

Multi-Agent Task Planning. Typically, the traditional multi-agent task planning process consists of three distinct stages: complex task decomposition, task assignment, and task execution[8].

In the task decomposition stage, complex tasks that cannot be directly executed are divided into executable sub-tasks or atomic tasks[18]. The most straightforward method is rule-based task decomposition using a knowledge base[19]. Methods such as PADIA, which focuses on multi-timeline, non-linear narrative structures, construct hierarchical task networks to achieve effective decomposition[20]. However, such methods require a high-quality knowledge base and have low generalization for specific task decomposition methods[21]. Another intuitive approach is to use natural language descriptions and pre-trained language models to decompose tasks and predict the temporal sequence between sub-tasks[22]. In this paper, we use pre-prompting LLM to decompose complex tasks into sub-tasks corresponding to agent skills, facilitating the next steps of task assignment and execution.

In the task assignment stage, the sequence of sub-tasks from task decomposition needs to be assigned to multiple agents. Researchers have used various methods, including auction-based[23],contract-net-based[24],game-theory-based[25], and reinforcement-learning-based[26] approaches. Although these methods are reliable, they are typically customized and optimized for specific final goals and application scenarios, with limited portability across different scenes. In this paper, we leverage the inherent generalization and common-sense reasoning abilities of LLMs to divide multiple agents based on skill sets and perform task assignment for sub-task sequences.

In the task execution stage, the execution of each task action is usually implemented by the agent's control module[27]. This corresponds to the verbs in PDDL planning. Dai et al. constructed an action space for six agent skills[6] to facilitate direct invocation by LLMs. We further improve this concept by proposing a generalized, modular skill space to accommodate the skill requirements of different environments and agents.

Multi-agent Prompting. Prompt Engineering is an emerging discipline focused on the development and optimization of prompts, which can be used to enhance the ability of LLMs to handle complex task scenarios. Compared to free-form prompts, specific prompt patterns can generate better results[28]. Chain of Thought (CoT) involves prompting the model to decompose problems in a human-like manner, transforming the solution to a complex problem into a series of logically clear and progressively reasoned steps, thereby improving the model's accuracy and interpretability[29]. In this paper, we apply prompt engineering in multi-agent systems, optimizing the collaboration of multiple agents with different attributes to accomplish tasks.

Task Planning with LLMs. With the rapid development of artificial intelligence technology, the applications of LLMs have been continually expanding in recent years[30]. LLMs excel in generalization and common-sense reasoning, enabling them to learn methods through few-shot or zero-shot learning[9], and they have been widely applied in task planning for agents. However, LLMs still have limitations in handling multi-agent systems. To address this, researchers have proposed unified approaches for task decomposition, team formation, assignment, and execution using LLMs[31]. These approaches divide agents into specialized functional units[32], but the generation of specific execution sequences tends to be time-consuming. In response to LLMs' insufficient ability for long-term reasoning and planning for complex tasks, scholars have suggested combining classical planning methods with LLMs[33]. However, these methods still lack sufficient generalization and adaptability for multi-agent task planning systems and do not focus on the underlying trajectory planning and obstacle avoidance for individual agents.

Therefore, we propose a hierarchical multi-agent task planning architecture for general scenarios, which uses CoT chain prompting to guide LLM in decomposing complex tasks into sub-tasks. Based on task assignment strategies, LLM assigns tasks to individual agents and generates PDDL plans. A symbolic planner quickly solves the sub-task goals for each agent. By modularly invoking a generalized agent action skill library, the task types are not limited to indoor environments. Additionally, we consider the collaborative motion planning problem of multiple agents during action execution and use a multi-agent three-dimensional STMC to generate optimal motion control trajectories to avoid collisions.

3 Methodology

In our multi-agent system, we assume that the complex high-level task instructions T provided by humans are known. Within an environment $E = \{E_1, E_2, \cdots, E_K\}$ that contains K types of objective entities and objects, there are N heterogeneous agents $R = \{R_1, R_2, \cdots, R_N\}$. Each agent possesses M different skill sets $S_i = \{S_i^1, S_i^2, \cdots, S_i^M\}$, where $i \in 1, 2, \cdots N$. Skills are modularly invoked to effect changes in the environment's state. For example, the skill "OpenObject" can change the state of an entity, such as a "refrigerator", from "closed" to "open". Additionally, we use a pre-defined planning domain description file, which conforms to the PDDL 1.2 specification standard. The predicate actions in this file correspond to the skill module library.

As shown in Figure 1, this is the overall framework of our system. Our method leverages LLMs to understand user needs and subsequently performs task decomposition, assignment, and action sequence planning for complex task instructions. The action sequences are then executed by the respective agents. We solve for the three-dimensional STMC and the optimal control sequence for multiple agents, maximizing resource utilization and optimizing trajectories to achieve the user's desired state.

3.1 Task Decomposition and Allocation

In the task decomposition stage, we use LLM to receive task instructions T and environment information E. The task instructions are first decomposed to generate a time-ordered sub-task sequence :

$$T_{sub} = \{T_{sub1}, T_{sub2}, \cdots, T_{subH}\},$$

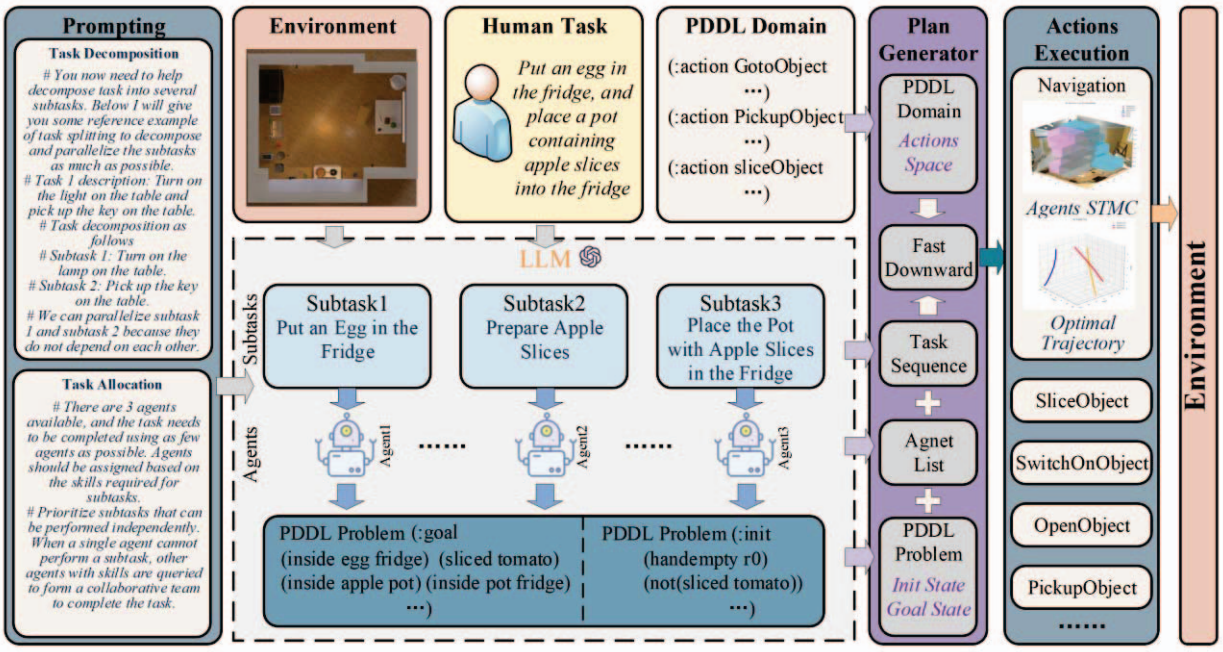

Fig. 1: **The overall architecture of GMATP-LLM.** The main execution process consists of task decomposition, task allocation, plan generator, and action execution, including a three-dimensional spatio-temporal motion corridor.

where consisting of H sub-tasks. Each T_{subi} is defined as a quintuple:

$$T_{subi} = \{name, skills, aim, starttime, endtime\},$$

where *name* represents the name of the sub-task; *skills* refers to the skills required to execute the sub-task; *aim* indicates the object to be executed in the sub-task; *starttime* and *endtime* represent the start and end times of the sub-task. The relationship between sub-tasks is defined by O, which includes both sequential and parallel relationships.

We use pre-defined Chain of Thought prompts to guide LLM in executing the task decomposition. We referenced the G-PlanET dataset[34] to set up the task decomposition prompts, including examples of simple tasks, complex tasks, and ambiguous tasks. Instead of directly using natural language prompts, we structured them with a code-like style, complete with comments and standardized templates for LLM to understand and reason through. This approach ensures concise token usage while providing detailed explanations of the task decomposition requirements, enhancing LLM's understanding and reasoning ability[35].

In the task assignment stage, we established prompts for the task assignment strategy. These prompts analyze the list of skills required for each sub-task T_{subi} and the relationships between single-agent skills. They describe how to assign the sub-task sequence to each agent, guiding LLM to reason using the decomposed sub-task sequence T_{sub}, the set of agents R, and the set of agent skills S. This process generates the task assignment result—each agent's sub-task sequence to be executed, denoted as:

$$T_{sub}^{j} = \{T_{sub1}^{j}, T_{sub2}^{j}, \cdots, T_{subi}^{j}\},\ j \in 1,2,\cdots N.$$

Our task assignment strategy aims to optimize resource allocation, enhance collaboration, and maximize efficiency by considering multiple factors such as the skills required for task execution, the skills of team members, and task priorities[36]. The assignment strategy is as follows:

(1)Local Task Priority: Each agent selects sub-tasks it can independently execute based on its own status and the task requirements, prioritizing tasks according to their urgency.

(2)Global Task Allocation Balance: In some cases, a single agent cannot execute a sub-task. Through Pareto front allocation, we query other agents with the necessary skills, forming collaborative teams to complete the task. This ensures the overall balance of task distribution among the agent team, optimizing the utilization of task resources.

After task assignment, we use prompts to guide LLM in translating each agent's pending sub-task sequence into a problem.pddl file that conforms to the PDDL 1.2 standard. For example, the sub-task "cutting tomatoes and placing them on the table" is translated into a PDDL problem, with the goal state defined as: (:goal (exists (?o - tomato ?o - table))), meaning that the goal state is that the tomatoes are on the table and have been sliced, preparing for subsequent rapid action planning.

3.2 Task Plan Generator

In the task sequence planning phase, we use a classical planner to receive each agent's pending sub-task sequence T_{sub}^{i} and the corresponding PDDL problem file. Based on the pre-defined PDDL domain description file, the planner solves for the action sequence $A^{j} = \{A_{1}^{j}, A_{2}^{j}, \cdots, A_{N}^{j}\}$ for each agent, where $j \in 1,2,\cdots N$.

As shown in Table 1, we first define the action space of the agents. This corresponds to the predicates in the classical planning domain, as well as the lower-level discretized skill actions. This action space is modular and can be expanded. Each action has preconditions, expected effects, and potential scenarios. We assume that scenarios are generated only during the execution of actions and can only be observed as a whole before and after the action is executed.

Table 1: Discretized action space

Action	Precondition	Expected Outcome	Possible Situations
GoTo	1. The agent and object are in the same environment.	1. The agent reaches the object.	1. The agent reaches the object 2. The agent has not reached the object
Pickup	1. The object is not in the hands of the agent 2. The agent's hands are empty	1. The object is picked up	1. The object is picked up 2. The object has not been picked up
Put	1. The object is in the hands of the agent 2. The agent is next to the placed object	1. The object is put down	1. The object is placed at the specified location 2. The object is put down and is not at the designated location 3. The object has not been put down and is still in the hands of the agent
SwitchOn	1. The object switch is not turned on	1. The object switch is turned on	1. The object switch is turned on 2. The object switch is not turned on
SwitchOff	1. The object switch is not turned off	1. The object switch is turned off	1. The object switch is turned off 2. The object switch is not turned off
Open	1. The object is not opened	1. The object is opened	1. The object is opened 2. The object is not opened
Close	1. The object is not closed	1. The object is closed	1. The object is closed 2. The object is not closed
Slice	1. The object is not sliced 2. The agent has a knife in its hand	1. The object is sliced	1. The object is sliced and the knife is in the hands of the agent 2. The object is sliced, but the knife is not in the hands of the agent 3. The object has not been sliced, and the knife is in the hands of the agent 4. The object has not been sliced, and the knife is not in the hands of the agent

After the task assignment module constructs different PDDL problem files, we use the Fast Downward solver[37] to solve them and generate action plans. A plan consists of a series of actions, where each action, when correctly executed by the corresponding agent, allows the environment state to transition from the initial state to the goal state. The use of classical planning ensures the rapid and accurate resolution of actions.

3.3 Action Execution

In the action execution phase, as shown in Table 1, we define a unified action space A. By modularly calling the underlying skill APIs, we control each agent, ensuring the system's generality and the efficiency of task execution. Before each action is executed, the system extracts knowledge about the action's preconditions from the planner's domain description. The action is executed only if these preconditions are satisfied.

For example, the action SliceObject(?r - robot ?o - tomato ?k - knife) has the following preconditions: (beside ?r ?o), (inhand ?k ?r), (not(inhand ?o ?r)), and (not(sliced ?o)). This means that in order to slice the tomato, the agent must be holding the knife, be next to the tomato, and the tomato must be in an unsliced state. The expected effect of executing the SliceObject action is that the tomato is sliced. Possible scenarios that may arise from this action are:(1) The tomato is sliced, and the knife is in the agent's hand.(2) The tomato is sliced, and the knife falls on the table. (3) The tomato is not sliced, and the knife is in the agent's hand. (4) The tomato is not sliced, and the knife falls on the table.

After the action is executed, the system updates the state representation based on the actual scenario that occurs. We provide the updated environment state to the classical planner, using it as the new initial state to regenerate the plan. The process continues until the higher-level complex task instruction is completed. This method allows the agents to adapt to environmental changes and unexpected outcomes of actions.

3.4 Collaborative Motion Planning

For the low-level motion planning in the multi-agent system, we consider the parallelism of task execution and the need for obstacle avoidance among multiple agents. We have innovatively used a multi-agent collaborative motion planning approach based on hierarchical three-dimensional Spatio-Temporal Motion Corridors. The system takes environmental map information, the initial states of the agents, and their target states as inputs, and produces safe and smooth motion trajectories for each agent as outputs.

As shown in Figure 2, we adopt a hierarchical multi-agent motion planning approach. For the upper-level reference path layer, we use the A* algorithm to search for a rough reference path for each agent in the environment. By applying a priority determination method[38], we allow each agent to determine its movement reference path.

For the middle layer, which involves generating the Spatio-Temporal Corridors for multi-agent interactions, it is a multi-objective collaborative optimization problem. We consider collision constraints between each agent and environmental obstacles to generate the three-dimensional motion corridor parameters for each agent. To solve this, we

construct a Mixed-Integer Quadratic Programming (MIQP) model[39] and use the Gurobi solver[40] to compute the solution, providing the passable three-dimensional motion corridor information for each agent based on their reference paths.

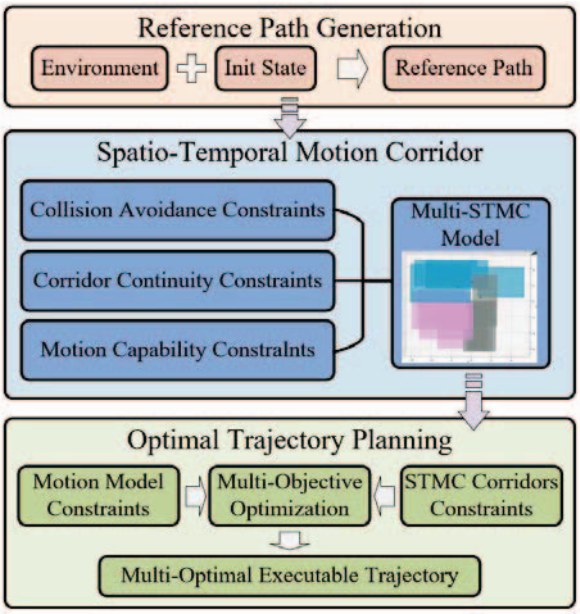

Fig. 2: Hierarchical Collaborative Motion Planning Architecture

In the lower-level optimal trajectory generation layer, we use the three-dimensional spatio-temporal motion corridor information from the middle layer as constraints. The agent's kinematic model is used as equality constraints, and the dynamic model serves as inequality constraints. The objective is to optimize trajectory smoothness and deviation from the reference point based on the agents' initial states. We construct a nonlinear optimization model and solve it using the Gurobi solver to generate a collision-free trajectory sequence with parameters related to the motion of each agent: $[x_t^i, y_t^i, \theta_t^i, \delta_t^i, v_t^i, a_t^i]^{\mathrm{T}}$.

This method allows us to solve the passable corridors and Spatio-Temporal optimized trajectories for multiple agents' movement actions, ensuring that the resulting paths are continuous, smooth curves and capable of achieving optimized obstacle avoidance in the environment.

4 Experiments

4.1 Experiment Setup

We conducted extensive experiments to evaluate the performance of GMATP-LLM. The simulation validation of our method was primarily carried out in the AI2-THOR[41] simulation environment, using the benchmark dataset G-PlanET[34] for rigorous evaluation and comparison with baseline methods. It is important to note that the prompt task examples, task data, and simulated room scenarios we used for testing were different, meaning that all tasks in the test dataset were treated as unknown during the testing phase.

We extracted 15 task instructions from the benchmark dataset, encompassing three task types:

Simple Tasks: These tasks cannot be directly executed and require task decomposition. They are tasks that can be executed by a single agent without involving collaboration between heterogeneous robots.

Complex Tasks: These tasks also cannot be directly executed and require both task decomposition and task assignment. The assignment strategy is more flexible (including sequential, parallel, and other logical relationships) and considers the collaborative relationships and specific sub-task execution between heterogeneous multi-robots.

Ambiguous Tasks: These are special cases of complex tasks that can introduce ambiguity into the system. For example, "Put two bars of soap in the sink." The system needs to determine whether one robot should execute the task or multiple robots should collaborate to complete it, as the assignment outcome affects the execution efficiency.

The test set includes 5 simple tasks, 5 complex tasks, and 5 ambiguous tasks. For each task instruction, we selected the corresponding simulation room environment and set up a collection of robots available for task execution, including their skill sets, to provide the system with environmental information. For the same task instruction, we used multiple LLMs to evaluate the performance of our GMATP-LLM method, including GPT-4o, GPT-3.5, and Llama-3-70B.

For the PDDL planning method, we used the Fast Downward solver[42] and set a search time limit of 100 seconds. To verify the plan and the changes in task state conditions, we used the VAL2 tool[43]. Additionally, we compared the task planning execution time of our method with the baseline method, which solely relies on LLMs to generate the agents' actions. This comparison demonstrates that the use of PDDL planning significantly improves the overall task planning speed while still retaining the reasoning capabilities of LLMs, thus overcoming the limitations of traditional planning algorithms.

In the action execution layer, we support the modular invocation of robot skills through API calls. Furthermore, we developed a multi-agent underlying navigation and motion planning framework, which determines collision-free, passable Spatio-Temporal Motion Corridors for each robot. This framework ensures obstacle avoidance during both parallel and serial execution of actions.

4.2 Reults

We used different LLM and provided five task decomposition examples as prompts. We then used task instructions from the benchmark dataset, representing three different task types, as inputs. We recorded the output from the LM and compared it with the benchmark dataset, measuring the task decomposition accuracy. Each task was executed three times, and the final average was taken.

Table 2 shows the experimental results of our method using different LLMs and task categories. It is evident that our method yields favorable results across various LLMs, especially with GPT-4o and Llama-3-70B. Even for smaller parameter language models, success was achieved, likely due to our clear prompt structure. However, our method faced challenges with GPT-3.5 during task decomposition and assignment, which is likely due to its weaker reasoning capabilities.

When handling complex tasks, LLM-generated PDDL problems sometimes presented inaccuracies, leading to

lower accuracy in TCSR compared to TDA. Nevertheless, thanks to our prompt design, the LLM tends to prioritize multi-threading and parallel execution when dealing with ambiguous tasks, which enhances efficiency.

Table 2: Evaluation experiments with large language models using different task types

Method GMATP-LLM	Simple Tasks		Complex Tasks		Vague Tasks	
	TDA	TCSR	TDA	TCSR	TDA	TCSR
GPT-4o	1.00	1.00	0.92	0.87	1.00	1.00
GPT-3.5	0.83	0.83	0.62	0.55	0.75	0.70
Llama-3	1.00	1.00	0.85	0.82	1.00	1.00

As shown in Table 3, we also compared our method with the baseline approach where the entire task plan is generated solely by LLMs. For the three types of typical tasks, we tested the planning time and success rate in the action sequence planning layer. The results showed that generating the PDDL problem with LLMs and solving it using a classical planner was faster and more accurate. This approach combines the advantages of classical planning and LLM planning, making it more suitable for deployment in practical multi-agent systems.

Table 3: Comparative experiments with baseline methods

Method	Simple Tasks		Complex Tasks		Vague Tasks	
	PT	ASGA	PT	ASGA	PT	ASGA
DXK(GPT-4o)	1.09s	1.00	2.12s	0.87	1.87s	1.00
Baseline(GPT-4o)	5.39s	1.00	11.08s	0.72	8.11s	0.85

4.3 Real-Vehicle Experiment

We conducted patrol and transportation task tests using three YUHESEN FR-07 Pro autonomous vehicles in different areas. It is assumed that these vehicles possess the capabilities for Navigate and Patrol actions. Our method successfully generates task plans, as shown in Figure 3, where each autonomous vehicle is assigned to a different area to perform navigation patrols. The three-dimensional Spatio-Temporal Motion Corridor used successfully avoids collisions, effectively bridging the gap between simulation and real-world applications.

Fig. 3: **Real-Vehicle Experiment:** Two of the vehicles perform obstacle avoidance strategies while performing patrol tasks at the same time.

4.4 Evaluation Metrics

In this experiment, we used the following metrics to evaluate performance:

Task Decomposition Accuracy (TDA): This measures how closely the task decomposition results match the tasks in the benchmark dataset. The higher the accuracy, the better the performance.

Task Completion Success Rate (TCSR): This measures whether the final state after task decomposition, assignment, and action execution via PDDL planning matches the expected goal state. A higher success rate indicates better performance.

Plan Time (PT): This refers to the time taken by the PDDL planner to explore the complete state space and generate a plan. The shorter the time, the higher the efficiency.

Action Sequence Generation Accuracy (ASGA): This measures the accuracy with which the action sequence is generated for each agent based on their tasks. The higher the accuracy, the better the result in terms of action planning.

5 Conclusion and Future Work

In our research, we combine the advantages of LLMs and classical PDDL planning to explore their potential in heterogeneous multi-agent task planning systems. We have innovatively introduced three-dimensional Spatio-Temporal Motion Corridors at the foundational level to ensure the parallelism and obstacle avoidance of agents during task execution, significantly improving the efficiency of the task planning system.

In the experiments, our method demonstrated clear adaptability, able to process unknown tasks of varying complexity through reasoning without relying on specific environments or agent entities. It can seamlessly extend to various fields and scenarios, bridging the gap between simulation and real-world applications. In the future, we plan to explore the application of Vision-Language Models (VLMs) in multi-agent task planning systems.

References

[1] Rizk Y, Awad M, Tunstel E W. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, 2019, 52(2): 1-31.

[2] Zhou Z, Liu J, Yu J. A survey of underwater multi-robot systems. *IEEE/CAA Journal of Automatica Sinica*, 2021, 9(1): 1-18.

[3] Madridano Á, Al-Kaff A, Martín D, et al. Trajectory planning for multi-robot systems: Methods and applications. *Expert Systems with Applications*, 2021, 173: 114660.

[4] P. Benavidez, M. Kumar, S. Agaian, and M. Jamshidi, Design of a Home Multi-robot System for the Elderly and Disabled, in *2015 10th System of Systems Engineering Conference (SoSE)*, 2015.

[5] Gao Z, Prorok A. Environment optimization for multi-agent navigation. *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023: 3440-3446.

[6] Dai Y, Peng R, Li S, et al. Think, act, and ask: Open-world interactive personalized robot navigation, in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024: 3296-3303.

[7] Chakraa H, Guérin F, Leclercq E, et al. Optimization techniques for Multi-Robot Task Allocation problems: Review on the state-of-the-art. *Robotics and Autonomous Systems*, 2023: 104492.

[8] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, et al., "Integrated task and motion planning", *Annual review of control robotics and autonomous systems*, vol. 4, pp. 265-293, 2021.

[9] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 2020, 33: 1877-1901.

[10] Islam R, Moushi O M. Gpt-4o: The cutting-edge advancement in multimodal llm. *Authorea Preprints*, 2024.

[11] Dubey A, Jauhri A, Pandey A, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[12] Yang A, Yang B, Zhang B, et al. Qwen2. 5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024.

[13] Aeronautiques C, Howe A, Knoblock C, et al. Pddl| the planning domain definition language. *Technical Report*, Tech. Rep., 1998.

[14] Ghallab M, Nau D, Traverso P. Automated planning and acting. *Cambridge University Press*, 2016.

[15] Jiang, Yq., Zhang, Sq., Khandelwal, P. et al. Task planning in robotics: an empirical comparison of PDDL- and ASP-based systems. *Frontiers Inf Technol Electronic Eng 20*, 363–373 (2019).

[16] Höller D, Behnke G, Bercher P, et al. HDDL: An extension to PDDL for expressing hierarchical planning problems, in *Proceedings of the AAAI conference on artificial intelligence*. 2020, 34(06): 9883-9891.

[17] Ilche Georgievski, Marco Aiello,HTN planning: Overview, comparison, and beyond, *Artificial Intelligence*, Volume 222,2015,Pages 124-156.

[18] Dorri A, Kanhere S S, Jurdak R. Multi-agent systems: A survey. *Ieee Access*, 2018, 6: 28573-28593.7

[19] Kattepur A, Dey S, Balamuralidhar P, Knowledge Based Hierarchical Decomposition of Industry 4.0 Robotic Automation Tasks, in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, Washington, DC, USA, 2018, pp. 7.

[20] Padia K, Bandara K H, Healey C G. A system for generating storyline visualizations using hierarchical task network planning[J]. *Computers & Graphics*, 2019, 78: 64-75.

[21] J. Motes, R. Sandstrom, H. Lee, S. Thomas, and N. M. Amato, Multi-Robot Task and Motion Planning with Subtask Dependencies, *IEEE Robotics and Automation Letters*, 2020.8.

[22] P. A. Jansen, Visually-Grounded Planning without Vision: Language Models Infer Detailed Plans from High-Level Instructions, *arXiv preprint arXiv:2009.14259*, 2020.8

[23] M. Braquet and E. Bakolas, Greedy Decentralized Auction-based Task Allocation for Multi-Agent Systems, *IFAC-PapersOnLine*, 2021.8.

[24] Zhang J, Wang G, Song Y. Task Assignment of the Improved Contract Net Protocol under a Multi-Agent System. *Algorithms*. 2019; 12(4):70.

[25] C. Zhang, Q. Li, Y. Zhu and J. Zhang, Dynamics of Task Allocation Based on Game Theory in Multi-Agent Systems, in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 6, pp. 1068-1072, June 2019.

[26] Park B, Kang C, Choi J. Cooperative Multi-Robot Task Allocation with Reinforcement Learning. *Applied Sciences*. 2022; 12(1):272.

[27] Fei Chen and Wei Ren (2019), On the Control of Multi-Agent Systems: A Survey, *Foundations and Trends in Systems and Control*,Vol. 6: No. 4, pp 339-499.

[28] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang. Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts, in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–21, 2023.

[29] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le,D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*,35:24824–24837, 2022.

[30] Shen Y, Song K, Tan X, et al. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 2024, 36.

[31] Kannan S S, Venkatesh V L N, Min B C. Smart-llm: Smart multi-agent robot task planning using large language models, in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024: 12140-12147.

[32] Chang M, Chhablani G, Clegg A, et al. PARTNR: A Benchmark for Planning and Reasoning in Embodied Multi-agent Tasks. *arXiv preprint arXiv:2411.00081*, 2024.

[33] Silver T, Hariprasad V, Shuttleworth R S, et al. PDDL planning with pretrained large language models, in *NeurIPS 2022 foundation models for decision making workshop*. 2022.

[34] Lin B Y, Huang C, Liu Q, et al. On grounded planning for embodied tasks with language models, in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2023, 37(11): 13192-13200.

[35] Wei J, Wang X, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 2022, 35: 24824-24837.

[36] Notomista G, Mayya S, Hutchinson S, et al. An optimal task allocation strategy for heterogeneous multi-robot systems, in *2019 18th European control conference (ECC)*. IEEE, 2019: 2071-2076.

[37] Helmert M. The fast downward planning system. *Journal of Artificial Intelligence Research*, 2006, 26: 191-246.

[38] Dewangan R, Shukla A, Godfrey W W. Survey on prioritized multi robot path planning, in *2017 IEEE international conference on smart technologies and management for computing, communication, controls, energy and materials (ICSTM)*. IEEE, 2017: 423-428.

[39] Zang Z, Zhang X, Song J, et al. A Coordinated Behavior Planning and Trajectory Planning Framework for Multi-UGVs in Unstructured Narrow Interaction Scenarios. *IEEE Transactions on Intelligent Vehicles*, 2024.

[40] Lapucci M, Pucci D. Mixed-integer quadratic programming reformulations of multi-task learning models. *Mathematics in Engineering*, 2022: 0-0.

[41] Kolve E, Mottaghi R, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[42] Helmert M. The fast downward planning system. *Journal of Artificial Intelligence Research*, 2006, 26: 191-246.

[43] Carbonell J, Etzioni O, Gil Y, et al. Prodigy: An integrated architecture for planning and learning. *ACM SIGART Bulletin*, 1991, 2(4): 51-55.