2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)
October 19-25, 2025. Hangzhou, China

2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3315-4393-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/IROS60139.2025.11246692

CLGA: A Collaborative LLM Framework for Dynamic Goal Assignment in Multi-Robot Systems

Xin Yu[1], Haoyuan Li[2], Yandong Wang[2], Simin Li[1], Rongye Shi[2,3], Gangzheng Ai[4], Zhiqiang Pu[5], Wenjun Wu[2,3]

***Abstract*— Goal assignment is a critical challenge in multi-robot systems. The emergence of large language models (LLMs) has enabled the use of natural language commands for tackling goal assignment problems. However, applying LLMs directly to these tasks presents two limitations: 1) limited accuracy and 2) excessive decision delays due to their autoregressive nature, hindering adaptability to unexpected changes. To address these issues, inspired by dual-process theory, we propose a framework called Collaborative LLMs for dynamic Goal Assignment (CLGA). Specifically, we leverage LLMs for pre-planning tasks and invoke an external solver to generate an initial goal assignment solution, ensuring solution accuracy. During execution, small-scale models enable real-time adjustments to respond to dynamic environmental changes. This approach integrates the strengths of slow, precise pre-planning and fast, adaptive online adjustments, allowing agents to efficiently handle real-world challenges. Additionally, we introduce a benchmark dataset for NLP-based goal assignment to advance research in this domain. Simulation and real-world experiments demonstrate that CLGA significantly enhances task execution efficiency and flexibility in multi-robot systems. The prompt, experimental videos, and datasets associated with this work are available at https://sites.google.com/view/project-clga/.**

I. INTRODUCTION

In recent years, multi-robot systems have demonstrated significant advantages in scenarios such as navigation, logistics, and communication [1], [2], [3]. Their collaborative capabilities not only improve task execution efficiency but also enhance system fault tolerance. A common challenge in many applications is how to achieve optimal one-to-one matching that minimizes global costs, such as total travel distance or task completion time. This is exactly the problem addressed by the Linear Sum Assignment Problem (LSAP) [4]. Although traditional optimization algorithms have been successful in static environments, dynamic disturbances in real-world scenarios, such as sudden robot failures (shown in Figure 1), pose significant challenges to existing methods in terms of real-time responsiveness. This dynamic uncertainty places dual demands on goal assignment algorithms: they must both establish precise mathematical models to handle unstructured information and possess adjustment capabilities to respond to unforeseen events [5].

Recent breakthroughs in LLMs in complex decision-making and logical reasoning have opened up new possibilities for developing goal assignment frameworks that combine mathematical rigor with flexibility [6]. Studies have shown that LLMs can decompose human instructions into subtasks and plan agent behaviors through natural language interfaces [7]. This ability to map semantics to actions allows systems to quickly respond to unstructured task requirements. However, directly applying LLMs to multi-robot goal assignment still faces two major technical bottlenecks. First, the decisions generated by LLMs based on probabilistic reasoning lack optimality guarantees, which may lead to suboptimal solutions [8]. Second, the autoregressive reasoning mechanism of LLMs causes decision delays, making it difficult to meet the stringent real-time planning requirements in applications like search and rescue [9].

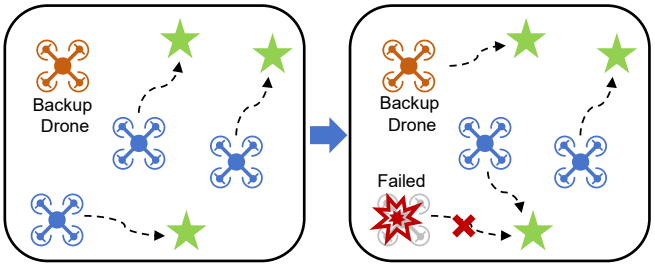


Fig. 1: The initial goal assignments shown on the left. When emergency occurs, such as a drone failure, the system re-plans to adapt to the dynamic situation by utilizing backup drones and adjusting the goal assignment accordingly.

To address these challenges, we propose a framework called Collaborative LLMs for Dynamic Goal Assignment (CLGA). CLGA can generate assignment solutions based on natural language problem descriptions of goal assignment. CLGA integrates a slow thinking module and a fast thinking module to effectively handle dynamic goal assignment. We first use the slow thinking module for task pre-planning, invoking external solvers to generate initial goal assignment solutions for the multi-robot system to ensure solution accuracy. Upon detecting an emergency, the fast thinking module generates a temporary plan, while the slow thinking module continues computing the optimized solution. Once the slow thinking module is ready, the system compares its cost with that of the fast thinking module and selects the solution with the lower cost for execution. In the slow thinking module, a multi-agent collaboration framework is employed, where each agent is assigned a specific role based on domain knowledge, and a central commander coordinates the agents' actions. We employ the Decentralized Hungarian algorithm

[1]School of Computer Science, Beihang University, China. [2]School of Artificial Intelligence, Beihang University, China. [3]Hangzhou International Innovation Institute, Beihang University, China. [4]School of Astronautics, Beihang University, China. [5]Institute of Automation, Chinese Academy of Sciences, China. Corresponding author: Wenjun Wu.

as the fast thinking module on the edge to make real-time adjustments in response to dynamic environmental changes (other decentralized models can be used as needed) [10]. This approach combines the powerful capabilities of LLMs with the flexibility of real-time decision-making, improving the accuracy and flexibility of multi-robot goal assignment. We also introduce a benchmark dataset that covers a series of typical multi-robot goal assignment problems. Each sample in this dataset consists of a textual description of the goal assignment problem and its corresponding ground truth, enabling the development and evaluation of LLM-based goal assignment methods. We conducted extensive evaluations, including simulations and real-world experiments, and the results demonstrate that CLGA outperforms existing LLM-based methods in task execution efficiency and flexibility. The main contributions of this paper are as follows:

- A hybrid framework integrating slow and fast thinking modules for dynamic goal assignment.
- An LLM-based multi-agent collaboration system with role-specific agents and centralized coordination for modeling and solving optimization problems.
- A benchmark dataset designed for evaluating the multi-agent goal assignment problem, consisting of different textual descriptions with varying numbers of agents.
- Validation of the CLGA in both simulations and real-world experiments, demonstrating improvements.

II. Related Works

A. LLMs for Planning.

Planning involves organizing actions to solve given problems, typically by generating a sequence of high-level symbolic operators, and then executing them using low-level controllers [11]. LLM-based task planners can generate solutions without strict symbol definitions [12], whereas traditional task planners require predefining operators using domain knowledge about available actions and constraints [13], [14]. Most LLM-based planning approaches adopt a static planning strategy, which typically includes zero-shot or few-shot prediction methods. Zero-shot methods generate solutions based solely on input commands, while few-shot methods leverage learning from a limited set of similar examples [15], [16]. However, due to limited reasoning capabilities, LLMs often provide suboptimal solutions in task planning [17], [18]. An alternative approach, adaptive planning, enables robots to modify their solutions or actions based on feedback. This can be done either by generating new solutions according to environmental observations [19], [20], or by detecting faults and making corresponding adjustments [21]. However, adaptive planning suffers from significant decision delays. In the field of multi-agent systems, progress has been made in enhancing agent cooperation using LLMs [22], [23]. Unlike previous approaches, our method integrates the strengths of LLMs and decentralized small-scale models to address goal assignment in multi-robot systems. This hybrid strategy overcomes the accuracy and latency challenges of LLM-only methods, while avoiding the need for each robot to have its own LLM [22], [23].

B. Goal Assignment

Goal assignment plays a crucial role in various multi-robot system applications [24], [25], [26]. Existing approaches can be broadly categorized into optimization-based, market-based, and learning-based paradigms. Optimization-based methods typically achieve optimal solutions through decentralized adaptations of algorithms such as the Hungarian algorithm. To improve efficiency, heuristic approaches, such as greedy algorithms [10] guaranteeing a 2-approximation to the optimal solution. Market-based methods are mainly based on auction mechanisms for goal assignment [27], such as the Consensus-Based Auction Algorithm [28]. Learning-based goal assignment has leveraged Graph Neural Networks to model agent–task relations as bipartite graphs and predict assignments via supervision [29]. These methods require users to have a certain level of experience and cannot be invoked using natural language. Additionally, these methods can be incorporated into the CLGA framework as the fast thinking module.

III. Problem Formulation

We have N robots and N goals, denoted by $\mathscr{R} = \{r_1, r_2, \ldots, r_N\}$ for the set of robots and $\mathscr{G} = \{g_1, g_2, \ldots, g_N\}$ for the set of goals. Initially, each robot is assigned exactly one goal. The assignment is denoted by $s_{ij}^{(t)}$, where $s_{ij}^{(t)} = 1$ indicates that robot r_i is assigned to goal g_j at step t, and $s_{ij}^{(t)} = 0$ otherwise. The cost of assigning robot r_i to goal g_j at time t is denoted by $c_{ij}^{(t)}$, which is specific to the application. At time $t = T$, two types of emergency may occur:

- **Emergency Situation 1 (Faulty Robots).** A group of M robots has malfunctioned and can no longer perform the assigned tasks. M new robots need to be dispatched to continue completing the tasks.
- **Emergency Situation 2 (New Goals).** M new targets have been added, requiring more robots to be assigned to these new targets. M new robots need to be dispatched to continue completing the tasks.

We need to optimize the sum of the initial cost and the cost after the emergency situation, ensuring that both assignment schemes satisfy the one-to-one constraint while minimizing the total cost. This involves:

$$
\begin{aligned}
\min_{s_{ij}^{(t)}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}^{(0)} s_{ij}^{(0)} + \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}^{(T)} s_{ij}^{(T)} \\
\text{s.t.} \quad & \sum_{j=1}^{N} s_{ij}^{(t)} = 1, \quad \forall i \in \{1,\ldots,N\}, \\
& \sum_{i=1}^{N} s_{ij}^{(t)} = 1, \quad \forall j \in \{1,\ldots,N\}, \\
& s_{ij}^{(t)} \in \{0,1\}, \quad \forall i,j \in \{1,\ldots,N\}, \quad t \in \{0,T\}
\end{aligned}
$$

The first cost term, $\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}^{(0)} s_{ij}^{(0)}$, represents the total cost of the initial assignment. The second cost term, $\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}^{(T)} s_{ij}^{(T)}$, represents the cost of the assignment after the emergency occurs.

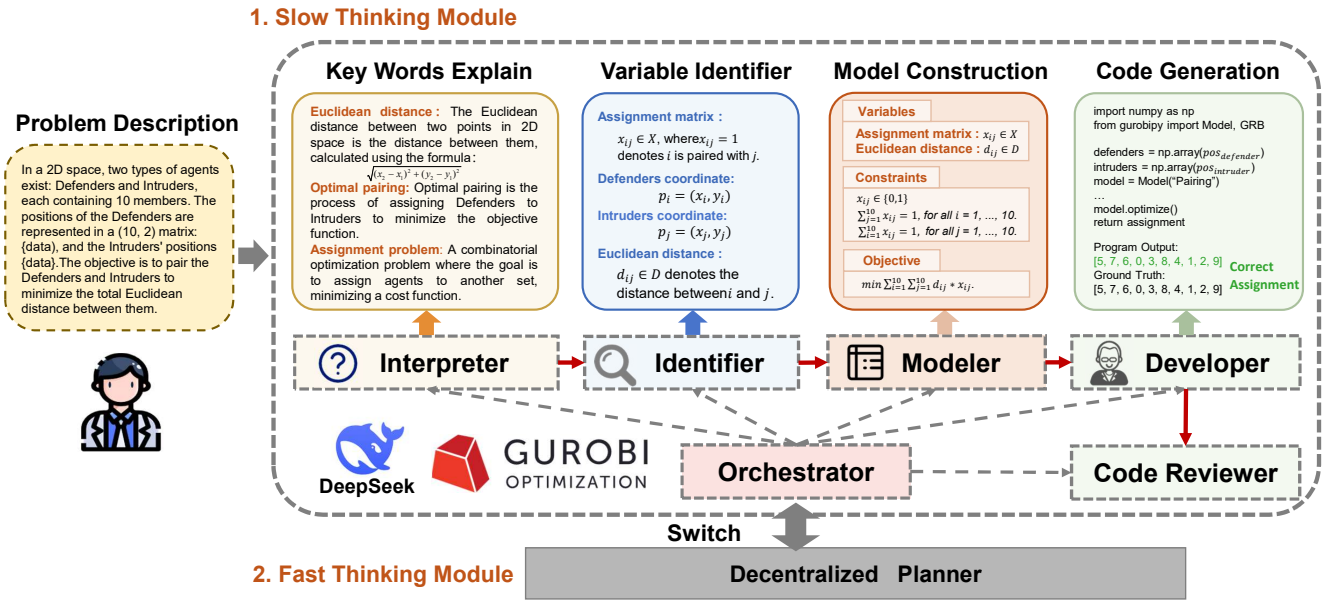


Fig. 2: CLGA integrates a slow thinking module and a fast thinking module to handle dynamic goal assignment. 1) The slow thinking module utilizes an Orchestrator to coordinate specialized agents—Interpreter, Identifier, Modeler, Developer, and Code Reviewer—to process NLP based goal assignment. 2) The fast thinking module, through a decentralized planner, enables an switching mechanism to respond to emergency situation. Upon detecting emergency, the decentralized planner generates a temporary plan, while the slow thinking module continues computing an solution. Once the slow thinking module is ready, CLGA compares its cost with that of the fast thinking module and selects the solution with the lower cost.

IV. Methodology

This section describes the design of the CLGA framework. As shown in Figure 2, CLGA comprises two main modules: the slow thinking module and the fast thinking module. The slow thinking module handles NLP-based goal assignment; it takes task descriptions as input and generates an initial solution. The fast thinking module is responsible for handling emergencies. When an emergency is detected, the system activates a decentralized planner to regenerate the solution.

A. Slow Thinking Module: Multi-LLM Collaboration

The slow thinking module primarily comprises multiple agents, each performing distinct functions, which are coordinated by the Orchestrator. The Orchestrator is responsible for selecting the appropriate agents to address the problem at hand. The agents include the Interpreter, the Identifier, the Modeler, the Developer, and the Code Reviewer. The problem-solving process is typically divided into two parts. The first part involves problem modeling, where the natural language description of the issue is transformed into a structured operations research model. This step includes explaining key words, identifying variables, and constructing the model (as depicted in Figure 2). The second part entails writing code based on this model to implement a solution. Each agent's role and function will be discussed sequentially.

Orchestrator. At each step, the Orchestrator reviews the ongoing conversation and selects the next agent whether it be the Interpreter, Identifier, Modeler, Developer or Code Reviewer. The Orchestrator is responsible for generating and assigning tasks to the selected agents, such as reviewing and fixing formulas. The Orchestrator plays a central role in ensuring smooth communication and task flow among the agents. The Orchestrator sequentially selects agents to contribute their expertise. Let the task description be I, and the set of agents as $\mathscr{A} = \{A_1, A_2, \ldots, A_k\}$, where k is the total number of agents. At each step t, the Orchestrator selects an agent based on the current state S_t, where $S_t = (I, C_t)$. The C_t is the set of output of previous agents ($C_0 = \emptyset$). The selection policy is modeled as:

$$A^t = \text{Orchestrator}(S_t)$$

A^t is the agent selected in step t. Once selected, the agent contributes an output c^t :

$$c_t = A^t(I, C_t)$$
$$C_{t+1} = C_t \cup \{c_t\}$$

The process continues for K steps, and the final outcome is produced after all the outputs are aggregated.

Interpreter. The Interpreter provides domain-specific knowledge that enhances problem understanding and formulation. The Interpreter contributes by providing background knowledge and ensuring that the problem is clearly understood, enabling the proper identification of variables and the development of an appropriate solution framework. Specifically, it explains the key terms that appear in the task description.

Identifier. The Identifier focuses on identifying and extracting the relevant variables from the problem description. These variables represent the decision points in the optimization problem. Leveraging expertise in the domain, this

agent ensures that all critical variables are accurately defined and extracted, setting the stage for the subsequent modeling process. The extracted variables are essential for building an accurate mathematical model of the problem.

Modeler. The Modeler is responsible for constructing the mathematical optimization model based on the information extracted by the other agents. This agent utilizes the operational research knowledge inherent in the LLMs itself to assist in problem solving. Specifically, the Modeler will output the optimization objectives and constraints in LaTeX format, which will then be used for code writing.

Developer. The Developer is tasked with writing the code for the solver. When assigned a task, the Developer begins by reading the provided task description and the previous comments. The agent will generate code based on the formulas. If the task is to fix errors identified by the Orchestrator, the Developer will review and correct the marked expressions. In our experiments, the programmer uses Python as the programming language and Gurobi as the solver [30]. CLGA can support other solvers and programming languages.

Code Reviewer. The Code Reviewer is responsible for executing the generated code on the data and identifying any runtime errors that occur. If an error is encountered, the Code Reviewer flags the variable or clause causing the issue and provides an error explanation to the Orchestrator. This information is then used by other agents to fix formulas and debug the code, ensuring that the final solution is correct.

During the operation of the CLGA, each agent uses a Chain of Thought for reasoning [31]. Our framework is not limited to dynamic goal assignment but is versatile enough to be applied to a broader spectrum of robotic planning tasks.

B. Integration of Fast and Slow Thinking Module

We propose an adaptive switching mechanism for dynamic goal assignment tasks, integrating a slow thinking module and a fast thinking module to address emergency situation. The slow thinking module, based on LLMs, leverages external solvers to generate high-precision global assignment solutions. The fast thinking module can quickly generate adjustment solutions in response to emergency, ensuring rapid reactions. The CLGA adopts the Decentralized Hungarian in its implementation [32]. It is worth noting that the fast thinking module is designed to be interchangeable with other suitable algorithms, depending on different application scenarios and requirements.

Initially, the system operates the slow thinking module to generate and execute an assignment plan P_{slow}. Upon detecting an emergency, the fast thinking module is activated to produce and implement a temporary plan P_{fast}, while the slow thinking module concurrently computes an optimized plan P'_{slow}. Once P'_{slow} is ready, the system compares its cost $C(P'_{\text{slow}})$ against the cost of the fast thinking module's current plan $C(P_{\text{fast}})$ using a unified cost function. It then selects the plan with the lower cost for continued execution. If $C(P'_{\text{slow}}) < C(P_{\text{fast}})$, the system smoothly transitions to P'_{slow}. Otherwise, it maintains P_{fast}. If P_{fast} has partially completed targets, only the unexecuted portions are optimized.

V. Experiments

Reinforcement learning (RL) has been widely applied to motion control tasks [33], [34], [35], [36], [37]. We adopt an RL-based controller to govern the agents' movements [38].

A. Simulation Experiments

The experimental validation in this paper focuses on two tasks: perimeter defense game and navigation, evaluated in both two-dimensional (2D) and three-dimensional (3D) environments. In the perimeter defense game (hereafter referred to as 'Defense'), defenders aim to intercept intruders, while the intruders attempt to reach the perimeter without being captured [39]. The navigation task requires multiple agents to reach a designated target while avoiding obstacles. Figure 3 illustrates these tasks. We assign intruders to defenders and goals to navigation agents. To evaluate the effectiveness of our approach, we simulate the following emergency scenarios during task execution. In both navigation and defense, there is the concept of a goal.

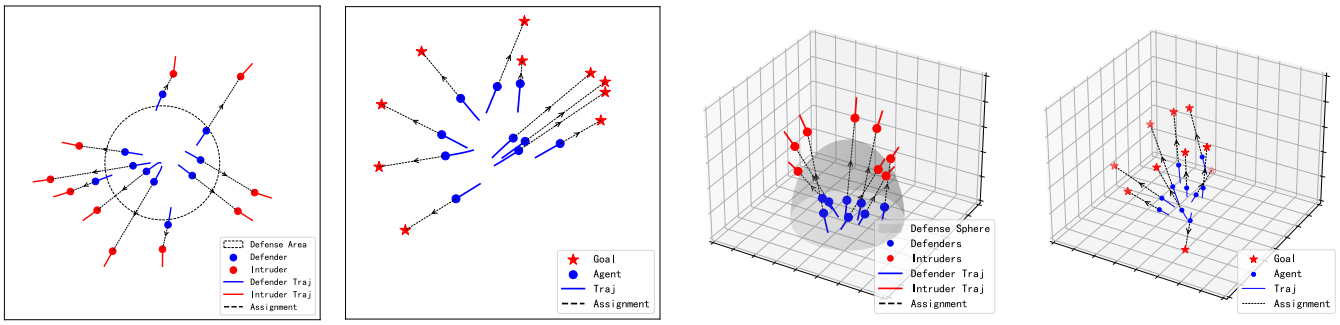
- Faulty Robots: When a group of M robots malfunctions and can no longer perform their assigned tasks, M replacement robots are deployed to assume the responsibilities of the malfunctioning robots.
- New Goals: When M new goals are introduced, M additional robots are dispatched and assigned to these newly defined objectives.

In both emergency scenarios, the robot-to-goal assignments must be dynamically updated to accommodate the changes. Following the occurrence of an emergency at time t=T, the CLGA transitions to a local planner, which utilizes the decentralized Hungarian algorithm to manage goal assignments.

VI. Results and Discussion

We evaluate the effectiveness of CLGA from two perspectives. The first perspective focuses on assessing the overall performance of the CLGA in response to emergency situations. The second perspective focuses on evaluating the accuracy of the initial solutions produced by the CLGA. To facilitate this assessment, we have developed a benchmark dataset specifically tailored for evaluating NLP-based goal assignment methods. We use DeepSeekV2.5 as the default LLMs and perform five runs.

Benchmark Dataset. The dataset is designed to encompass a variety of task types, ensuring a comprehensive evaluation across different dimensions and complexities. The dataset comprises 40 instances, evenly distributed across four categories: 2D Navigation, 2D Defense, 3D Navigation, and 3D Defense, each containing 10 instances. Every instance is accompanied by a task description and its corresponding ground truth assignment. To enhance the diversity and naturalness of these descriptions, we refined them using chatgpt, ensuring variability in phrasing while maintaining clarity and relevance to the task. Within each instance, the number of agents, their initial positions, and target positions (where applicable) vary, introducing diversity and increasing the complexity of the scenarios.



(a) 2D Defense (b) 2D Navigation (c) 3D Defense (d) 3D Navigation

Fig. 3: Illustration of Defense and Navigation Tasks in 2D and 3D Environments.

TABLE I: Comparison of task completion steps between Standard and CLGA, with percentage reduction in steps.

Method	Scenario 1		Scenario 2	
	Defense	Navigation	Defense	Navigation
Standard (2D)	46	58	44	54
CLGA (2D)	35	49	39	44
%Reduction ↓	-23.9%	-15.5%	-11.4%	-18.5%
Standard (3D)	47	49	40	53
CLGA (3D)	40	44	35	42
%Reduction ↓	-14.9%	-10.2%	-12.5%	-20.8%

TABLE II: Comparison of cost between Standard and CLGA, with percentage reduction in cost.

Method	Scenario 1		Scenario 2	
	Defense	Navigation	Defense	Navigation
Standard (2D)	75.5	104.1	110.1	126.7
CLGA (2D)	58.9	87.5	97.9	111.0
%Reduction ↓	-22.0%	-16.0%	-11.0%	-12.4%
Standard (3D)	96.3	91.8	105.4	137.3
CLGA (3D)	76.7	79.2	93.8	119.3
%Reduction ↓	-20.3%	-13.8%	-11.0%	-13.1%

A. *Overall Performance*

Firstly, we compared the effects of using CLGA for replanning versus not replanning (Standard) in response to emergency situations. This evaluation employs two primary metrics: cost and the number of steps required to complete the tasks. For the defense task, cost is defined as the sum of Euclidean distances between each defender and its assigned intruder. For the navigation task, cost represents the sum of distances from each agent to its designated goal.

As shown in Table I, CLGA consistently outperforms the standard method in terms of task completion steps, resulting in significant step reductions across different tasks. As detailed in Table II, CLGA reduces the cost compared to the standard method. These results highlight the effectiveness of the CLGA framework in optimizing task completion and cost under emergency scenarios, confirming its enhanced performance in both 2D and 3D environments.

TABLE III: AC, CE, and RE of different methods.

Method	2D			3D		
	AC	CE	RE	AC	CE	RE
CLGA	95%	0%	5%	95%	0%	5%
Classic	40%	0%	60%	40%	15%	45%
PHP	45%	50%	5%	40%	30%	30%
REFF	50%	15%	35%	60%	30%	10%
CoT	45%	10%	45%	55%	0%	45%

B. *Evaluation of the Slow Thinking Module*

We evaluate the slow thinking module's effectiveness in NLP-based goal assignment by comparing its performance against various reasoning methods, including PHP [40], REFF [41], CoT [31], and Classic. The Classic refers to having the LLMs generate and execute code based on the description. We employ an automated code evaluation process where we generate programming code for each instance in our dataset. If the generated assignment matches the ground truth of the instance, we consider the problem to be successfully solved. The success rate is expressed in terms of accuracy. We measure the compilation error rate (CE), which captures the percentage of programs generated that fail to compile. We also measure the run-time error rate (RE), which captures errors encountered during execution. We do not distinguish between navigation and defense tasks, as they are similar from a modeling perspective. However, there are differences in dataset format and distance calculations between 3D and 2D.

As shown in Table III, the CLGA consistently outperforms the other methods in terms of accuracy, achieving 95% AC in 2D and 3D environments. In comparison, the PHP method performs relatively poorly, with a 45% AC in 2D. The REFF method shows 50% AC in 2D, while the CoT method achieves 45% accuracy with low compilation errors (10% CE). It appears that advanced reasoning frameworks such as REFF, PHP can improve the effectiveness of LLMs in goal assignment. However, frameworks specifically tailored for particular tasks, such as CLGA, demonstrate enhanced performance. The sum of AC, CE, and RE does not equal 100%. This discrepancy arises because the code may execute correctly yet yield incorrect assignment outcomes.

TABLE IV: Ablation study of the CLGA framework.

Method	2D			3D		
	AC	CE	RE	AC	CE	RE
CLGA (Full)	95%	0%	5%	95%	0%	5%
w/o Comment	90%	0%	5%	85%	0%	15%
w/o CoT	90%	0%	5%	85%	5%	5%
w/o Orchestrator	75%	10%	10%	60%	5%	20%

TABLE V: CLGA performance across LLMs.

Method	2D			3D		
	AC	CE	RE	AC	CE	RE
DeepseekV2.5	95%	0%	5%	95%	0%	5%
DeepseekV3	100%	0%	0%	100%	0%	0%
GPT-4o-mini	80%	0%	20%	80%	0%	20%
Gemini2.0-pro-exp	90%	0%	0%	85%	5%	0%
Claude3.5-sonnet	100%	0%	0%	95%	0%	0%

Different LLM base model. We tested our slow thinking module using different LLMs. The comparative results between various models (DeepSeekV2.5, DeepSeekV3, GPT-4o-mini, Genmini2.0-pro-exp, Claude3.5-sonnet) are listed in Table V. The evaluation metrics include AC, CE, and RE. From the results, we observed that DeepSeekV2.5 achieved an accuracy of 95% in both 2D and 3D environments. DeepSeekV3 and Claude3.5-sonnet both reached an accuracy of 100% in 2D and 3D tasks, with no compilation or runtime errors. GPT-4o-mini and Genmini2.0-pro-exp showed acceptable performance. These outcomes suggest that using better base models can improve the results of the CLGA.

C. Ablation Study

We remove the Orchestrator, Comment, and CoT components from the CLGA for an ablation study. Then we can assess the contributions of individual components to system performance while maintaining CLGA's core functionality. The modifications were structured as follows:

- w/o Orchestrator: In the CLGA, the Orchestrator is responsible for coordinating the order of agent execution. Removing the Orchestrator results in CLGA executing agents randomly, passing the output from one agent to the next until the final code is generated.
- w/o Comment: In the complete CLGA, each agent receives the output from the previous agent. Removing the comment means each agent will independently generate results based on the task description, and the developer generates code based on all outputs.
- w/o CoT: In the CLGA, each agent employs a reasoning method called CoT. The removal of CoT here implies that all agents within the CLGA will no longer use this reasoning method. The CoT mentioned here differs from that referred to in Table III, where it represents a single agent using CoT to generate allocation schemes.

As shown in Table IV, the complete CLGA framework achieved the highest performance, with an accuracy rate of 95%. When CoT reasoning and comments are removed, there is a slight decrease in accuracy. The removal of the Orchestrator led to a more significant performance decline, with accuracy dropping to 75% in 2D and 60% in 3D environments. Additionally, the CE increased to 10% in 2D, and the RE rose to 20% in 3D. The notable impact of removing the Orchestrator is largely due to its role in coordinating the actions of other agents. These results highlight the importance of each component within the CLGA framework.

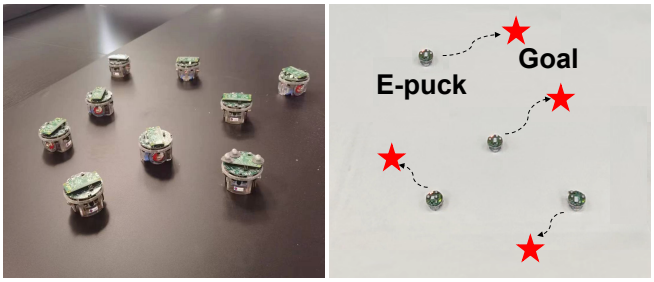


Fig. 4: Display of E-puck robots in the physical environment.

D. Real-Robot Experiments

We used e-puck robots to evaluate the performance of the CLGA framework, including its ability to handle emergency [42]. The robots were deployed in a 2D physical environment measuring $2m \times 2m$, with each robot tasked with achieving a specific objective based on natural language commands. We simulated two types of emergency situations previously mentioned. During execution, the CLGA framework effectively combined the slow yet precise pre-planning of goal assignment using LLMs with the fast, adaptive adjustments handled by decentralized smaller models. These real-time adjustments enabled the system to dynamically respond to emergency, proving the framework's robustness in dealing with real-world challenges.

VII. CONCLUSIONS

In conclusion, the CLGA framework provides a viable approach to dynamic multi-robot goal assignment. The integration of LLMs enables task pre-planning and facilitates seamless human-machine interaction, while the real-time adjustments made by smaller models provide the capability to handle unforeseen events. This paper validates the effectiveness of the proposed CLGA framework through both simulations and real-world experiments. Furthermore, the introduction of a multi-robot goal assignment benchmark dataset provides a valuable resource for advancing research in this field. Additionally, our framework is not confined to dynamic goal assignment but can be applied to a broader range of robotic planning tasks. This work highlights the potential of the LLMs to address the challenges faced by multi-robot systems in dynamic environments and offers directions for future applications.

ACKNOWLEDGMENTS

This work was supported by National Science and Technology Major Project (No. 2022ZD0117402).

REFERENCES

[1] W. Wang, L. Wang, J. Wu, X. Tao, and H. Wu, "Oracle-guided deep reinforcement learning for large-scale multi-uavs flocking and navigation," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 10, pp. 10 280–10 292, 2022.

[2] A. Farinelli, E. Zanotto, E. Pagello *et al.*, "Advanced approaches for multi-robot coordination in logistic scenarios," *Robotics and Autonomous Systems*, vol. 90, pp. 34–44, 2017.

[3] R. Shi, X. Yu, Y. Wang, Y. Tian, Z. Liu, W. Wu, X.-P. Zhang, and M. M. Veloso, "Symmetry-informed marl: A decentralized and cooperative uav swarm control approach for communication coverage," *IEEE Transactions on Mobile Computing*, 2025.

[4] M. Goarin and G. Loianno, "Graph neural network for decentralized multi-robot goal assignment," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4051–4058, 2024.

[5] W. Chen, S. Koenig, and B. Dilkina, "Why solving multi-agent path finding with large language model has not succeeded yet," *arXiv preprint arXiv:2401.03630*, 2024.

[6] J. Huang and K. C.-C. Chang, "Towards reasoning in large language models: A survey," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 1049–1065.

[7] S. S. Kannan, V. L. Venkatesh, and B.-C. Min, "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, pp. 12 140–12 147.

[8] S. Kambhampati, K. Valmeekam, L. Guan, M. Verma, K. Stechly, S. Bhambri, L. P. Saldyt, and A. B. Murthy, "Position: Llms can't plan, but can help planning in llm-modulo frameworks," in *Forty-first International Conference on Machine Learning*, 2024.

[9] W. Niu, Z. Kong, G. Yuan, W. Jiang, J. Guan, C. Ding, P. Zhao, S. Liu, B. Ren, and Y. Wang, "Real-time execution of large-scale language models on mobile," *arXiv preprint arXiv:2009.06823*, 2020.

[10] Y. Sung, A. K. Budhiraja, R. K. Williams, and P. Tokekar, "Distributed assignment with limited communication for multi-robot multi-target tracking," *Autonomous Robots*, vol. 44, no. 1, pp. 57–73, 2020.

[11] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 440–448.

[12] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International conference on machine learning*. PMLR, 2022, pp. 9118–9147.

[13] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[14] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. Sri, A. Barrett, D. Christianson *et al.*, "Pddl— the planning domain definition language," *Technical Report, Tech. Rep.*, 1998.

[15] Y. Cao and C. Lee, "Robot behavior-tree-based task generation with large language models," in *CEUR workshop proceedings*, vol. 3433. RWTH Aachen University, 2023.

[16] N. Di Palo, A. Byravan, L. Hasenclever, M. Wulfmeier, N. Heess, and M. Riedmiller, "Towards a unified agent with foundation models," in *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*.

[17] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "Llm+ p: Empowering large language models with optimal planning proficiency," *arXiv preprint arXiv:2304.11477*, 2023.

[18] K. Valmeekam, M. Marquez, S. Sreedharan, and S. Kambhampati, "On the planning abilities of large language models-a critical investigation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 75 993–76 005, 2023.

[19] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, "Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?" in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4311–4317.

[20] S. H. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, "Chatgpt for robotics: Design principles and model abilities," *Ieee Access*, 2024.

[21] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," in *Conference on Robot Learning*. PMLR, 2023, pp. 1769–1782.

[22] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin *et al.*, "Metagpt: Meta programming for a multi-agent collaborative framework," in *The Twelfth International Conference on Learning Representations*.

[23] Z. Liu, W. Yao, J. Zhang, L. Xue, S. Heinecke, R. Murthy, Y. Feng, Z. Chen, J. C. Niebles, D. Arpit *et al.*, "Bolaa: Benchmarking and orchestrating llm-augmented autonomous agents," *arXiv preprint arXiv:2308.05960*, 2023.

[24] H. Aziz, A. Pal, A. Pourmiri, F. Ramezani, and B. Sims, "Task allocation using a team of robots," *Current Robotics Reports*, vol. 3, no. 4, pp. 227–238, 2022.

[25] L. Antonyshyn, J. Silveira, S. Givigi, and J. Marshall, "Multiple mobile robot task and motion planning: A survey," *ACM Computing Surveys*, vol. 55, no. 10, pp. 1–35, 2023.

[26] J. Faigl, M. Kulich, and L. Přeučil, "Goal assignment using distance cost in multi-robot exploration," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 3741–3746.

[27] P. C. Lusk, X. Cai, S. Wadhwania, A. Paris, K. Fathian, and J. P. How, "A distributed pipeline for scalable, deconflicted formation flying," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5213–5220, 2020.

[28] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE transactions on robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[29] H. Liu, T. Wang, C. Lang, S. Feng, Y. Jin, and Y. Li, "Glan: A graph-based linear assignment network," *Pattern Recognition*, vol. 155, p. 110694, 2024.

[30] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com

[31] 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*, vol. 35, pp. 24 824–24 837, 2022.

[32] S. Ismail and L. Sun, "Decentralized hungarian-based approach for fast and scalable task allocation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 23–28.

[33] X. Yu, R. Shi, P. Feng, Y. Tian, S. Li, S. Liao, and W. Wu, "Leveraging partial symmetry for multi-agent reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, 2024, pp. 17 583–17 590.

[34] X. Yu, Y. Tian, L. Wang, P. Feng, W. Wu, and R. Shi, "Adaptaug: Adaptive data augmentation framework for multi-agent reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 10 814–10 820.

[35] X. Yu, R. Shi, P. Feng, Y. Tian, J. Luo, and W. Wu, "Esp: Exploiting symmetry prior for multi-agent reinforcement learning," in *ECAI 2023*. IOS Press, 2023, pp. 2946–2953.

[36] Y. Cao, R. Zhao, Y. Wang, B. Xiang, and G. Sartoretti, "Deep reinforcement learning-based large-scale robot exploration," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4631–4638, 2024.

[37] Y. Cao, J. Lew, J. Liang, J. Cheng, and G. Sartoretti, "Dare: Diffusion policy for autonomous robot exploration," *arXiv preprint arXiv:2410.16687*, 2024.

[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[39] D. Shishika and V. Kumar, "A review of multi agent perimeter defense games," in *Decision and Game Theory for Security: 11th International Conference, GameSec 2020, College Park, MD, USA, October 28–30, 2020, Proceedings 11*. Springer, 2020, pp. 472–485.

[40] C. Zheng, Z. Liu, E. Xie, Z. Li, and Y. Li, "Progressive-hint prompting improves reasoning in large language models," in *AI for Math Workshop@ ICML 2024*.

[41] J. Yao, H. Huang, Z. Liu, H. Wen, W. Su, B. Qian, and Y. Guo, "Reff: Reinforcing format faithfulness in language models across varied tasks," *arXiv preprint arXiv:2412.09173*, 2024.

[42] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proceedings of the 9th conference on autonomous robot systems and competitions*, vol. 1, no. 1. IPCB: Instituto Politécnico de Castelo Branco, 2009, pp. 59–65.