Learn as Individuals, Evolve as a Team: Multi-agent LLMs Adaptation in Embodied Environments

Xinran Li[1,2] Chenjia Bai[2,*] Zijian Li[1] Jiakun Zheng[2,3] Ting Xiao[3] Jun Zhang[1,*]
[1]The Hong Kong University of Science and Technology
[2]Institute of Artificial Intelligence (TeleAI), China Telecom
[3]East China University of Science and Technology
xinran.li@connect.ust.hk, baicj@chinatelecom.cn,
zijian.li@connect.ust.hk, {zhengjk@mail,xiaoting@}ecust.edu.cn,
eejzhang@ust.hk

Abstract

Large language models (LLMs) possess extensive knowledge bases and strong reasoning capabilities, making them promising tools for complex, multi-agent planning in embodied environments. However, despite LLMs' advanced abilities and the sophisticated modular design of agentic methods, existing LLM-based planning algorithms remain limited by weak adaptation capabilities to multi-agent embodied scenarios. We address this limitation by introducing a framework that enables LLM agents to learn and evolve both before and during test time, equipping them with environment-relevant knowledge for better planning and enhanced communication for improved cooperation. Inspired by centralized training with decentralized execution in multi-agent reinforcement learning, we propose a *Learn as Individuals, Evolve as a Team (LIET)* paradigm for multi-agent LLMs adaptation. At the individual level, LLM agents learn a local utility function from exploratory datasets to better comprehend the embodied environment, which is then queried during test time to support informed decision-making. At the team level, LLM agents collaboratively and iteratively maintain and update a shared cooperation knowledge list based on new experiences, using it to guide more effective communication. By combining individual learning with team evolution, LIET enables comprehensive and flexible adaptation for LLM agents. Our experiments on Communicative Watch-And-Help and ThreeD-World Multi-Agent Transport benchmarks demonstrate that LIET, instantiated with both LLaMA and GPT-4o, outperforms existing baselines and exhibits strong cooperative planning abilities.

1 Introduction

Multi-agent decision-making is a ubiquitous challenge in real-world applications ranging from daily tasks like completing household chores to industrial-scale problems in domains such as supply chain management [18, 7], smart cities [3, 5, 41] and robot swarms [32]. These problems are inherently challenging due to their long-horizon sequential nature and the complex interaction patterns among agents.

The emergence of large language models (LLMs) [34, 9, 1, 16, 10] has opened new opportunities for multi-agent decision-making. With their strong reasoning capabilities and extensive knowledge priors, LLMs have been employed as agent planners, proposing actions based on the current state of the environment. Recent works [38, 37, 23] have demonstrated the potential of prompting LLMs to achieve effective collaboration in long-horizon planning tasks in embodied environments. LLM-based

[*]Corresponding authors.

approaches often achieve strong zero-shot performance with their inherent comprehension, reasoning, and planning abilities. Despite these advantages, directly employing off-the-shelf LLMs as planners can be suboptimal without proper adaptation. LLMs, while trained on vast text-based data, lack the abilities to fully comprehend embodied contexts or facilitate dynamic multi-agent cooperation — both essential for effective multi-agent planning in embodied environments. This raises a key question: *can multi-agent LLMs adapt to such tasks by learning and evolving through interactions with their environment and peers, rather than relying exclusively on zero-shot abilities?*

Much like humans who adapt to new environments and teammates over time, LLM-based agents can improve by learning from experience. By interacting with the environment and other agents, they can refine their understanding of the physical world and enhance collaborative behaviors. To this end, we propose a novel LLM-based multi-agent decision-making framework: *Learn as Individuals, Evolve as a Team (LIET)*. LIET enables LLM-based agents to adapt to specific multi-agent embodied planning tasks through model fine-tuning and prompt evolution, enhancing both individual decision-making and team collaboration. LIET augments LLM-powered planners with environment-specific knowledge and collaboration skills derived from both prior experiences and test-time interactions. Inspired by the centralized training and decentralized execution (CTDE) paradigm from multi-agent reinforcement learning (MARL), our framework employs a semi-centralized approach where agents first explore individually to learn environment-specific information, and then collaborate as a team through communication and reflection to achieve better outcomes. To instantiate the *Learn as Individuals, Evolve as a Team* framework, we make the following technical contributions:

- **Utility-guided individual adaptation**: To enable LLM-powered planners to adapt to specific environmental contexts and make more informed decisions within limited time budgets, we augment their inputs with a utility function that estimates the cost of candidate plans. This utility function is implemented by finetuning a small LLM with a LoRA adapter and a value head on exploratory datasets.
- **Evolving prompting for communication**: To enhance team adaptation and collaboration, we design a dynamic prompting mechanism for constructive communication during test-time. This mechanism maintains a knowledge list of effective communication strategies that evolves over time, being iteratively refined through a reflection module at the receiver side. The module evaluates whether received messages improve planning outcomes in new situations, thereby fostering increasingly effective multi-agent communication.
- **Superior performance**: Through extensive experiments on multi-agent planning benchmarks, including Communicative Watch-And-Help (C-WAH)[27, 38] and ThreeDWorld Multi-Agent Transport (TDW-MAT)[6, 38], we demonstrate that LIET significantly outperforms existing baselines. Our results highlight its effectiveness in enabling long-horizon multi-agent collaborative planning.

2 Background

2.1 Problem Setting

We consider a fully cooperative multi-agent decision-making problem, which can be characterized as a decentralized partially observable Markov decision process (Dec-POMDP) [26]. The Dec-POMDP is formalized by a tuple $\mathcal{M} = \langle N, T, \mathcal{S}, \mathcal{O}, \mathcal{A}, P, \mathcal{G}, R \rangle$, where N is the number of agents and T is the task episode horizon. $\mathcal{S}$ represents the set of global states, and $\mathcal{O} = \mathcal{O}^{\text{env}} \times \mathcal{O}^{\text{comm}}$ is the set of local observation, consisting of partial environmental state information and messages from other agents. The action set is composed of high-level actions $\mathcal{A} = \mathcal{A}^{\text{env}} \cup \mathcal{A}^{\text{comm}}$, where $\mathcal{A}^{\text{env}}$ represents actions executed in the embodied environment and $\mathcal{A}^{\text{comm}}$ involves broadcasting natural language messages to other agents. The transition probability $P(s' \mid s, \boldsymbol{a})$ defines the likelihood of reaching a new state $s' \in \mathcal{S}$ after taking joint actions $\boldsymbol{a} = [a_1, a_2, \dots, a_N \mid a_i \in \mathcal{A}, i \in \{1, 2, \dots, N\}]$ in the current state $s \in \mathcal{S}$. $g \in \mathcal{G}$ specifies the common goal for all agents to achieve, and the reward $r = R(s)$ represents the degree of completeness of the team goal g.

In our experiments, we instantiate this problem in long-horizon embodied environments, assigning agents household tasks $g \in \mathcal{G}$ such as `Transport 2 bananas and 1 apple to the bed` or `Prepare a meal`. The agents can perform high-level actions $a \in \mathcal{A}^{\text{env}}$ such as `Grab(object)` or `GoTo(location)`, but execution failures may occur if actions are not well-situated in the current state of the environment (e.g., attempting to grab an object that is not present).

2.2 Related Work

LLM-based Single-agent Planning Recent advancements in large language models (LLMs) have led to the development of highly capable models, including LLaMA 3 [34, 9], GPT-4 [1, 16], and DeepSeek-R1 [10]. By leveraging their extensive general knowledge and exceptional reasoning capabilities, LLM-based agents have achieved remarkable success in embodied decision-making tasks that integrate perception, action, and interaction with physical or simulated environments [31]. Seminal works such as SayCan [2], grounded decoding [14] and Text2Motion [21] ground agent actions by using value functions to bridge language with real-world tasks. Follow-up studies have enhanced LLM planning capabilities by introducing thinking modules [15, 20] or world abstractions [25]. Others have leveraged in-context learning [30] when expert demonstrations are available. Another research direction focuses on aligning visual perception with language objectives, resulting in algorithms such as VLMaps [13], TaPA [35], mobility VLA [36], and NavGPT [40, 39]. These methods refine LLMs' ability to plan and act in environments requiring seamless integration of language and vision.

Multi-agent Collaboration in Embodied Environments Following the success of LLM-based single-agent planning, researchers have sought to extend these capabilities to multi-agent systems, aiming to improve task efficiency [33, 11]. Early approaches adopt a decentralized perspective, using modular designs [8] to facilitate cooperation by introducing communication modules [38] or teammate belief modules [37, 19]. While these decentralized approaches ensure flexibility, performance may suffer due to the lack of global consensus, particularly in tasks requiring a high degree of collaboration [4]. To address this, centralized methods such as RoCo [23] and CaPo [22] incorporate multi-round discussions among agents to enhance collaboration. However, as task complexity increases, partial observability becomes significant; or as the number of agents grows, these methods struggle to maintain consensus efficiently. Communication costs also scale rapidly, making centralized approaches less practical in real-world scenarios. Our proposed LIET strikes a balance between decentralized and centralized methods by adopting a semi-centralized framework. It incorporates an individual planning module enhanced with a utility function and facilitates cooperation through an evolving communication scheme that leverages past experiences. This approach enables agents to dynamically tailor their messages to better accommodate complex environments.

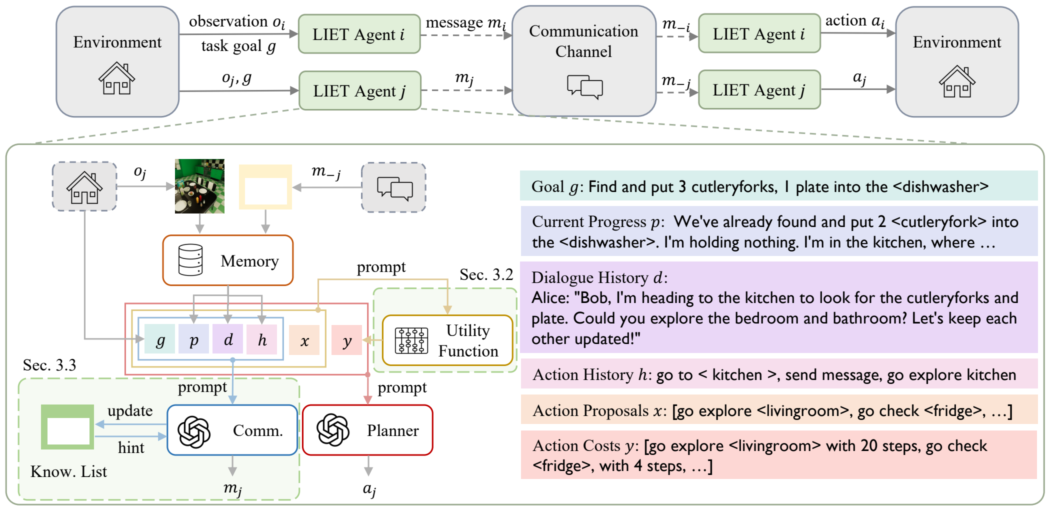
Furthermore, some recent research seeks to address the gap between simulation environments and real-world settings by considering more realistic scenarios. Notably, LLaMAR [24] restricts agent observations to imperfect knowledge from the simulator and employs a verifier to ensure the feasibility of plans, while SMART-LLM [17] converts high-level instructions into executable actions and deploys these methods on real robots. While these efforts address challenges that are orthogonal to the focus of our work, they could potentially be combined with LIET to enhance its applicability in real-world tasks.

3 Method

In this section, we propose a novel framework for multi-agent embodied planning, called *Learn as Individuals, Envolve as a Team (LIET)*. LIET adapts multi-agent LLM-based planners to embodied environments by addressing decision-making challenges from both local and global perspectives. In Section 3.1, we first examine the challenges of multi-agent embodied planning tasks and analyze how to comprehensively address these challenges by bridging local decision-making with global goals. We then introduce the overall framework of LIET, which is built around two core components: (1) a decentralized utility module that enables agents to make more informed and efficient decisions independently (*learn as individuals*), and (2) an evolving communication scheme that enhances cooperation and mutual understanding among agents through iterative prompt improvements (*evolve as a team*). In Section 3.2, we elaborate on the utility module, which is instantiated as a fine-tuned LLM that provides cost information to augment local planning. In Section 3.3, we expand on the evolving communication scheme, which leverages test-time improvements based on agents' prior interactions to enhance team collaboration.

3.1 Overall Framework of LIET

LIET is designed to enhance the capabilities of LLM-enabled planning agents by addressing two key challenges in multi-agent embodied tasks: (1) **Limited Environment-relevant Knowledge**:



Environment
observation o_i
task goal g
o_j, g
LIET Agent i
LIET Agent j
message m_i
m_j
Communication Channel
m_{-i}
m_{-j}
LIET Agent i
LIET Agent j
action a_i
a_j
Environment
o_j
m_{-j}
Memory
prompt
Sec. 3.2
g p d h x y
Utility Function
Sec. 3.3
prompt
prompt
update
hint
Comm.
Planner
Know. List
m_j
a_j
Goal g: Find and put 3 cutleryforks, 1 plate into the <dishwasher>
Current Progress p: We've already found and put 2 <cutleryfork> into the <dishwasher>. I'm holding nothing. I'm in the kitchen, where …
Dialogue History d:
Alice: "Bob, I'm heading to the kitchen to look for the cutleryforks and plate. Could you explore the bedroom and bathroom? Let's keep each other updated!"
Action History h: go to < kitchen >, send message, go explore kitchen
Action Proposals x: [go explore <livingroom>, go check <fridge>, …]
Action Costs y: [go explore <livingroom> with 20 steps, go check <fridge>, with 4 steps, …]

Figure 1: The overall framework of LIET. It adopts a semi-centralized decision making scheme with communication, where each agent plans independently and communicates with others at critical timesteps. During local planning, LIET agents utilize information from the environment, memory, and a utility function (detailed in Section 3.2) to make informed decisions. For communication, LIET agents generate messages based on the environment, memory, and an iteratively updated knowledge list (abbreviated as Know. List), which contains cooperation tips derived from prior experiences (detailed in Section 3.3). In LIET, the utility function, communication module (abbreviated as Comm.), and planner are all powered by LLMs.

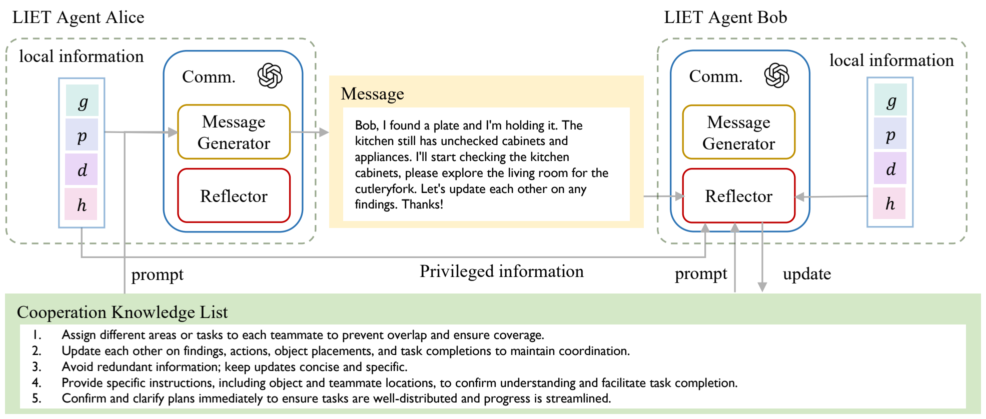
LLMs lack inherent knowledge of the physical world, as they are primarily trained on text corpora rather than embodied tasks [31]. (2) **Collaboration Gaps**: LLMs are not inherently equipped with the experience or mechanisms to collaborate effectively with other agents. As a result, LLM-based planners often generate plausible yet inefficient or inapplicable plans for embodied environments.

To address these challenges, LIET enhances the prompts of LLM-based planners by incorporating task-relevant knowledge and mechanisms specifically designed for multi-agent embodied environments. The overall framework of LIET, as illustrated in Figure 1, introduces two key innovations that distinguish it from existing methods. At the individual level, LIET equips each agent with a pretrained utility function that infers the cost of actions, enabling more efficient use of the effective time management and improving the agent's ability to generate practical and effective plans as a decentralized decision-maker. At the team level, LIET fosters seamless collaboration by maintaining a shared knowledge list containing cooperation tips, which serves as a dynamic resource for guiding constructive communication among agents. This knowledge list evolves iteratively through reflections on past coomunication experiences, allowing agents to continuously refine their coordination strategies over time. For the ease of illustration, we describe the framework using a two-agent setting. However, LIET is not limited to this configuration and can be readily applied to scenarios involving more agents, as demonstrated empirically in Section 4.2.

3.2 Learn as Individuals

In this subsection, we describe how LIET agents learn as individuals to understand the embodied environment and construct a utility function that assists in decentralized decision-making. The utility function plays a critical role in guiding agents by providing task-relevant information. Importantly, the utility function should meet two key criteria: it must (1) provide environment-related insights that aid decision-making, and (2) remain goal-agnostic to accommodate a wide variety of tasks.

Utility Function to Estimate Action Costs Inspired by the affordance function proposed by Brohan et al. [2], we adopt a value-based utility function to estimate the action costs in terms of environmental steps. This cost estimation is crucial for long-horizon embodied planning tasks, as high-level actions often have widely varying execution costs. For instance, traveling to another room may require significantly more steps than placing an object on a tray. By incorporating this information, LIET empowers



Bob, I found a plate and I'm holding it. The kitchen still has unchecked cabinets and appliances. I'll start checking the kitchen cabinets, please explore the living room for the cutleryfork. Let's update each other on any findings. Thanks!

1. Assign different areas or tasks to each teammate to prevent overlap and ensure coverage.
2. Update each other on findings, actions, object placements, and task completions to maintain coordination.
3. Avoid redundant information; keep updates concise and specific.
4. Provide specific instructions, including object and teammate locations, to confirm understanding and facilitate task completion.
5. Confirm and clarify plans immediately to ensure tasks are well-distributed and progress is streamlined.

Figure 2: The scheme for LIET communication. In LIET, communication happens at critical timesteps in a broadcast fashion. The key idea is that the LIET agents maintain a common cooperation knowledge list collaboratively based on reflection on previous conversation and use the knowledge to guide future message generation. At a decision timestep, agent Alice generates a message based on local information and the current cooperation knowledge list and sends it to agent Bob. Upon receiving the message, Bob uses it for its local decision making. In the meanwhile, Bob optionally leverages a reflector module and analyzes how the message can be further improved from the receiver's perspective. The insights from such reflection will be merged to the cooperation knowledge list. The example message and the cooperation knowledge list are derived from the CWAH task experiments. The local information contains goal, current progress, dialogue history and action history, as illustrated in Figure 1.

planners to make more efficient and context-aware decisions. Additionally, the utility function enhances coordination in multi-agent settings. Since agents operate simultaneously in the environment, understanding the cost of their actions helps them allocate efforts more effectively, improving overall task execution. A key advantage of this approach is that collecting cost information does not require task-specific expertise or human expert labeling. Instead, LIET leverages agents' ability in exploring the environment to autonomously gather the necessary data, making it a scalable and efficient solution.

In LIET, the utility function, denoted as $f(\ell_{o_i}, \ell_a)$, takes the text description of the current local observation o_i and the text description of the action candidate a as input and outputs a scalar value $c = f(\ell_{o_i}, \ell_a)$ representing the estimated number of environmental steps required to execute the action. During test time, LIET agents query the utility function to evaluate the costs of their current action plans, enabling them to make better-informed decisions, as illustrated in Figure 1.

Finetuning an LLM as the Utility Function Given the LLMs' strong abilities to comprehend the text context, we instantiate the utility function as a finetuned LLM augmented with an additional value head to produce scalar outputs. This design transforms the task into a regression problem, where the objective is to minimize the prediction error on action cost estimation. Specifically, we employ the Mean Squared Error (MSE) loss as the training objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\ell_{o_i}, \ell_a, c^{\mathrm{GT}} \sim \mathcal{D}}[(c^{\mathrm{GT}} - f(\ell_{o_i}, \ell_a; \theta))^2], \tag{1}$$

where $\mathcal{D}$ is a pre-collect exploratory dataset consist of description-cost pairs, c^{GT} represents the ground-truth cost, and θ includes the parameters of the LoRA adapter [12] and the value head, which is composed of MLP layers.

3.3 Evolve as a Team

While individual agents are equipped with environment-relevant information for more grounded and effective planning, the next challenge lies in adapting LLMs to function as part of a team. Compared to the individual learning scheme, we adopt a more versatile approach for team cooperation, enabling LIET agents to continuously improve [29] their communication by leveraging incoming experiences during task execution. The core mechanism of this team-level adaptation is an evolving prompting framework that enhances multi-agent communication. Specifically, agents exchange feedback on the messages they receive and collaboratively summarize these insights into a shared cooperation knowledge list. This knowledge list serves as a guide for generating more constructive messages in future interactions, allowing agents to refine their communication strategies over time. We term this process "evolve as a team" because agents collectively contribute to building and improving the shared knowledge during test time. The overall workflow of this approach is illustrated in Figure 2.

Message Generation Based on Experience To enable test-time improvement in multi-agent planning tasks, LIET incorporates prior experience into the message generation process through the shared knowledge list. This list is iteratively updated and used as part of the prompt for message generation, structured as follows:

Message Generator Prompting

Prompt: `<Task Description>` + `<Local Observation>` + `<Knowledge List>`. Please help me generate a message ...
LLM: `<Message>`.

Here, the placehoder `<Task Description>` refers to the goal for the task g, `<Local Observation>` contains sender agent's local information such as current progress, action history and dialogue history, `<Knowledge List>` is the hints maintained by all the agents iteratively on how to generate more constructive messages to teammates, which we will elaborate on in the following paragraph.

Reflection on the Receiver Side One of the key challenges in developing a test-time evolving scheme for communication lies in identifying effective message exchanges. Communication in multi-agent tasks contributes indirectly to task performance by providing decision-makers (i.e., message receivers) with additional information. However, it is often difficult to assess the relevance or quality of a message during execution. To address this, LIET introduces a reflection [28] mechanism that prompts the message receiver to evaluate the usefulness of received messages for decision-making. Since receivers inherently possess more context about what information is helpful for planning, they are better positioned to assess communication effectiveness. After reflecting on the message, the receiver generates insights on how future messages could be improved. These insights are then incorporated into the shared knowledge list, allowing the entire team to benefit from the reflection process. The prompt used for the reflector is as follows:

Reflector Prompting

Prompt: `<Task Description>` + `<Local Observation>` + `<Privileged Information>` + `<Received Message>` + `<Knowledge List>`. Please reflect ...
LLM: `<Updated Knowledge List>`

Here, the placeholder `<Privileged Information>` maintains additional details from the sender, such as their current plans, and the `<Updated Knowledge List>` replaces the existing `<Knowledge List>` in the next round of message generation and reflection, ensuring iterative improvement in message quality.

4 Experimental Results

In this section, we instantiate LIET with different LLMs to test its effectiveness on embodied household cooperative planning tasks. Specifically, we first compare LIET against several baselines to demonstrate its superior performance. Then, we demonstrate the intelligent and cooperative behaviors and analyze the mechanisms behind LIET through visualization and ablation studies. Furthermore, we extend LIET to scenarios with varying numbers of agents, demonstrating the flexibility of its semi-centralized framework.

4.1 Environmental Setups

Benchmark Descriptions Following previous works [38, 22], we adopt two long-horizon multi-agent planning benchmarks to evaluate LIET: Communicative Watch-And-Help (C-WAH) [27, 38]

Table 1: Average steps (↓) to complete tasks on the C-WAH benchmark under visual (Vis. Obs.) and symbolic (Sym. Obs.) perception. The best-performing method in each setting is highlighted.

	MHP*	LLaMA-3.1 Agents				GPT-4o Agents			
		CoELA	ProAgent	RoCo	LIET	CoELA	ProAgent	RoCo	LIET
Vis. Obs.	103	98.5	103.7	113.9	88.8	93.5	86.2	107.4	85.0
Sym. Obs.	75	58.8	64.4	73.9	50.4	48.4	61.0	64.7	40.3

* MHP results are adopted from the original CoELA paper [38].

Table 2: Transport rate (%) comparison on the TDW-MAT benchmark tasks with and without oracle perception. Tasks are categorized into "food" and "stuff" based on the object types. The best-performing method in each setting is highlighted.

	RHP*	LLaMA-3.1 Agents				GPT-4o Agents			
		CoELA	ProAgent	RoCo	LIET	CoELA	ProAgent	RoCo	LIET
w/o Oracle Perception									
Food (↑)	67	77.5	76.7	75.0	79.2	80.0	82.5	79.2	83.3
Stuff (↑)	54	65.0	69.2	69.2	69.2	68.3	70.8	76.7	75.8
Avg. (↑)	61	71.3	72.9	72.1	74.2	74.2	76.7	77.9	79.6
w/ Oracle Perception									
Food (↑)	76	84.2	82.5	80.8	82.5	85.0	79.1	85.8	90.0
Stuff (↑)	74	75.8	76.7	80.0	78.3	84.2	85.0	85.8	84.2
Avg. (↑)	75	80.0	79.6	80.4	80.4	84.6	82.0	85.8	87.1

* RHP results are adopted from the original CoELA paper [38].

and ThreeDWorld Multi-Agent Transport (TDW-MAT) [6, 38] (detailed in Appendix A.2). Both benchmarks feature cooperative tasks within realistic household simulators, incorporating both language and visual interfaces. Agents can navigate the environment and interact with objects to achieve task goals.

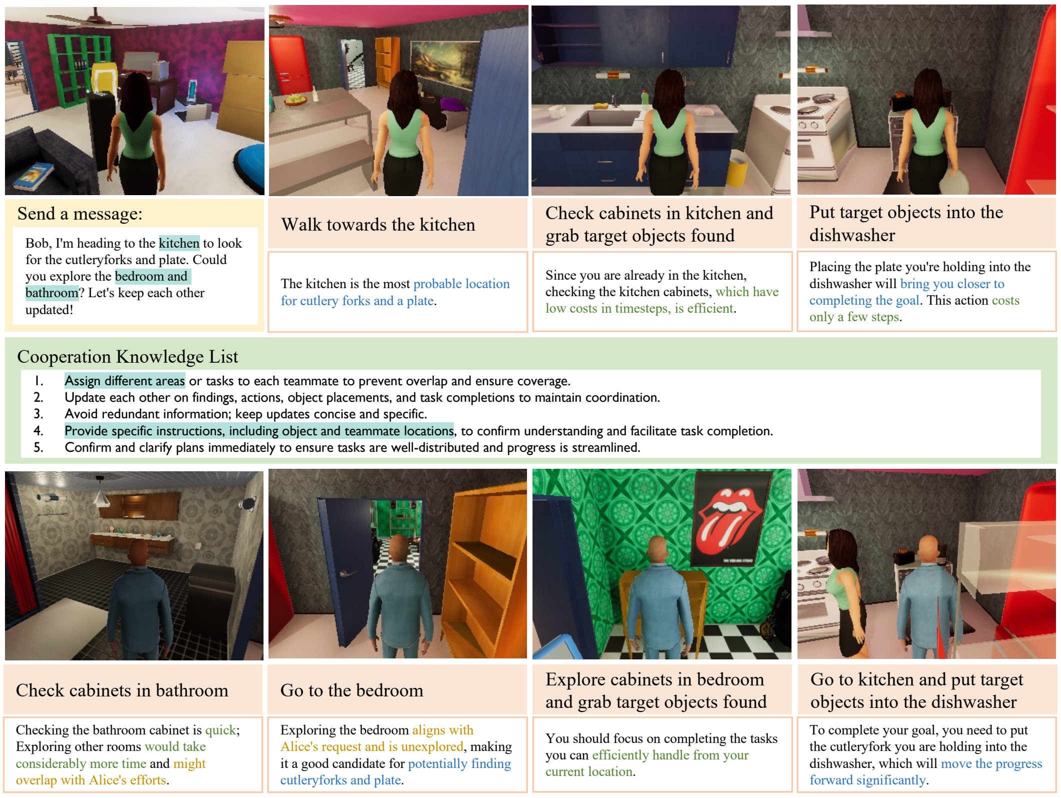
Built on the VirtualHome-Social simulator, C-WAH extends the Watch-And-Help Challenge [27] to feature multi-agent communication. Agents are required to collaboratively complete household chores by coordinating efforts through communication. The evaluation set consists of 10 episodes spanning five different tasks, each with a horizon of 250 timesteps. Performance is measured as the average timesteps required to complete all tasks.

Based on the ThreeDWorld Transport Challenge [6], TDW-MAT extends the task to multi-agent settings. Agents must transport objects scattered across different rooms to designated goal locations, using containers to assist in transportation. The evaluation set includes 24 episodes divided into "food" and "stuff" tasks, with each episode spanning up to 3000 time frames. Performance is evaluated using the transport rate, defined as the percentage of successfully transported objects relative to the total tasked objects within the time budget.

Baselines To evaluate LIET, we benchmark it against two categories of methods: traditional agents and LLM-based agents.

The traditional agents include: (i) *MCTS-based Hierarchical Planner (MHP)* [38]: A hierarchical planning approach designed for the original Watch-And-Help Challenge. It features a Monte Carlo Tree Search (MCTS)-based high-level planner and a regression-based low-level planner. (ii) *Rule-based Hierarchical Planner (RHP)* [38]: A heuristic-based hierarchical planning approach designed for the original ThreeDWorld Transport Challenge. It uses a rule-based high-level planner combined with an A-start-based low-level planner for navigation.

The LLM-based baselines include: (iii) *CoELA* [38]: A decentralized LLM-based planning framework with modular components for perception, communication, planning, and execution. (iv) *ProAgent* [37]: A decentralized LLM-based planning method that explicitly models teammate intentions. (v) *RoCo* [23]: A centralized LLM-based planning scheme that incorporates multi-turn communication for collaborative decision-making.



Send a message:

Bob, I'm heading to the kitchen to look for the cutleryforks and plate. Could you explore the bedroom and bathroom? Let's keep each other updated!

Walk towards the kitchen

The kitchen is the most probable location for cutlery forks and a plate.

Check cabinets in kitchen and grab target objects found

Since you are already in the kitchen, checking the kitchen cabinets, which have low costs in timesteps, is efficient.

Put target objects into the dishwasher

Placing the plate you're holding into the dishwasher will bring you closer to completing the goal. This action costs only a few steps.

Cooperation Knowledge List

1. Assign different areas or tasks to each teammate to prevent overlap and ensure coverage.
2. Update each other on findings, actions, object placements, and task completions to maintain coordination.
3. Avoid redundant information; keep updates concise and specific.
4. Provide specific instructions, including object and teammate locations, to confirm understanding and facilitate task completion.
5. Confirm and clarify plans immediately to ensure tasks are well-distributed and progress is streamlined.

Check cabinets in bathroom

Checking the bathroom cabinet is quick; Exploring other rooms would take considerably more time and might overlap with Alice's efforts.

Go to the bedroom

Exploring the bedroom aligns with Alice's request and is unexplored, making it a good candidate for potentially finding cutleryforks and plate.

Explore cabinets in bedroom and grab target objects found

You should focus on completing the tasks you can efficiently handle from your current location.

Go to kitchen and put target objects into the dishwasher

To complete your goal, you need to put the cutleryfork you are holding into the dishwasher, which will move the progress forward significantly.

Figure 3: Snapshots of an example episode from the C-WAH benchmark with the task: "Find and put 3 cutleryforks, 1 plate into the <dishwasher>". Each agent's visualization includes annotations of their plans and thoughts. The current cooperation knowledge list is also displayed. The message in the yellow dialogue box is directly generated by the LIET agent, while the thoughts are compact excerpts from the agent's Chain of Thought (CoT) process.

Implementation Details To evaluate LIET across different underlying LLMs, we instantiate the LLM-based planner in LIET and other LLM-based baselines using two state-of-the-art models: LLaMA 3.1-70B[9], an open-source model deployed on the TogetherAI platform, and ChatGPT-4o[16], a closed-source model accessed via the OpenAI API. We use temperature = 0.7, top-p = 1, and a maximum token limit of 256 for both models. Unless otherwise stated, all experiments involve two agents on both benchmarks. Further details regarding LIET implementation are provided in Appendix A.1.

4.2 Results

Performance We summarize the performance of LIET and baseline methods equipped with LLaMA 3.1-70B and ChatGPT-4o on the C-WAH and TDW-MAT benchmarks in Table 1 and Table 2, respectively. Overall, LIET outperforms the baselines, demonstrating the effectiveness of its semi-centralized framework. By enabling agents to adapt through individual exploration and collaborate via team interactions, LIET significantly enhances performance across both benchmarks.

Visualization To better understand how LIET operates in practice, we visualize an episode of the C-WAH task in Figure 3. Overall, LIET agents exhibit intelligent and cooperative behaviors. Analysis of the agents' CoT reasoning reveals that their decision-making process comprehensively considers three key factors: (1) task progress, analyzed by the LLMs, (2) efficiency, measured through costs estimated by the utility function and (3) dialogue history, generated by the evolving communicative prompting. These considerations enable LIET agents to make informed and considerate decisions. Moreover, the visualization highlights the direct correspondence between the cooperative knowledge list and generated messages, demonstrating how team adaptation guides effective communication. In the visualized episode, agent Alice (in green) initiates the task by proposing a division of labor. Agent

Table 4: Average steps (↓) to complete tasks on the C-WAH benchmark with symbolic perception, using different numbers of agents. All agents are powered by GPT-4o. The best performing method in each setting is highlighted.

	CoELA	ProAgent	RoCo	LIET
2 agents	48.4	61.0	64.7	40.3
3 agents	38.9	50.9	44.4	34.9
4 agents	36.5	46.6	53.9	33.1

Bob (in blue) evaluates Alice's suggestions and selects the more efficient subtask. The two agents then proceed with their respective subtasks, dynamically balancing action costs with the progress made toward task completion. This cooperative strategy exemplifies LIET's ability to foster efficient collaboration. Additional visualizations, including scenarios where the agents' exploration areas overlap and more intense discussions are required, can be found in Appendix B, further demonstrating the communication patterns induced by LIET.

Ablation Studies We conduct ablation studies on the C-WAH benchmark with symbolic perception to evaluate the impact of key design components in LIET. The results, summarized in Table 3, compare the full LIET framework with three ablated versions: (i) *LIET w/o individual learning*, in which we remove the utility function instantiated by a finetuned LLM described in Section 3.2, eliminating individual-level adaptation. (ii) *LIET w/ prompted costs estimation*, in which we replace the finetuned utility function with an explicit prompt requesting the planner to estimate action costs, simulating a non-adapted cost assessment mechanism. (iii) *LIET w/o team evolving*, in which we omit the teammate reflection process during test time (detailed in Section 3.3) and prompt the communication with a fixed template, removing team-level adaptation. The performance degradation across all ablations demonstrates that both individual learning and the team evolving process are critical to LIET's effectiveness. These results highlight that successful multi-agent planning with LLMs requires integrated adaptation at both individual and team levels to achieve optimal outcomes.

Table 3: Average steps (↓) to complete tasks on the C-WAH benchmark in ablation studies. All agents are powered by GPT-4o.

LIET Ablations	
LIET	40.3
LIET w/o individual learning	52.7
LIET w/ prompted costs estimation	57.6
LIET w/o team evolving	49.3

LIET with More Agents We further examined the performance of LIET under varying numbers of agents, using GPT-4o as the LLM planner on the C-WAH benchmark. Results in Table 4 demonstrate that LIET consistently outperforms baselines across different agent counts. This is primarily due to LIET's flexible framework, which integrates local costs estimation (via the pretrained individual utility function) with global cooperation (via team-evolving communication prompts). Notably, the utility functions used across all multi-agent settings were finetuned exclusively on datasets collected from two-agent scenarios. This demonstrates LIET's strong generalization capabilities, showing how adaptation learned from simpler contexts can effectively transfer to more complex multi-agent environments.

5 Conclusions

In this work, we explored multi-agent planning tasks in embodied environments and proposed the *Learn as Individuals, Evolve as a Team (LIET)* framework as an adaptation scheme for multi-agent LLMs. LIET significantly enhances planning performance by enabling LLM agents to adapt to specific environments through both individual learning and team evolution. Our semi-centralized approach implements adaptation at two critical levels: individually, agents learn utility functions during exploration that inform decision-making at test time; as a team, agents employ an evolving prompting scheme that continuously refines communication strategies through mutual feedback. Experiments on multi-agent embodied planning benchmarks demonstrated LIET's superior performance and robustness when extended to additional agents. These results highlighted the importance of adaptation mechanisms over relying solely on LLMs' zero-shot capabilities in complex embodied scenarios.

While effective, LIET shares limitations with previous approaches regarding multi-modal information processing, relying on pretrained vision modules to convert visual information into text for LLM planning. The comprehensive integration of multi-modal information in embodied environments remains an important direction for future research.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. *Gpt-4 technical report*. Technical Report. OpenAI.

[2] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the Conference on Robot Learning*. PMLR, 287–318.

[3] Jie Cai, Donghun Kim, Rita Jaramillo, James E Braun, and Jianghai Hu. 2016. A general multi-agent control approach for building energy system optimization. *Energy and Buildings* 127 (2016), 337–351.

[4] Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. 2024. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?. In *2024 IEEE International Conference on Robotics and Automation*. IEEE, 4311–4317.

[5] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. 2019. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE transactions on intelligent transportation systems* 21, 3 (2019), 1086–1095.

[6] Chuang Gan, Siyuan Zhou, Jeremy Schwartz, Seth Alter, Abhishek Bhandwaldar, Dan Gutfreund, Daniel LK Yamins, James J DiCarlo, Josh McDermott, Antonio Torralba, et al. 2022. The threedworld transport challenge: A visually guided task-and-motion planning benchmark towards physically realistic embodied ai. In *2022 International Conference on Robotics and Automation*. IEEE, 8847–8854.

[7] Mihalis Giannakis and Michalis Louis. 2011. A multi-agent based framework for supply chain risk management. *Journal of Purchasing and Supply Management* 17, 1 (2011), 23–31.

[8] Ran Gong, Qiuyuan Huang, Xiaojian Ma, Yusuke Noda, Zane Durante, Zilong Zheng, Demetri Terzopoulos, Li Fei-Fei, Jianfeng Gao, and Hoi Vo. 2024. MindAgent: Emergent Gaming Interaction. In *Findings of the Association for Computational Linguistics*. 3154–3183.

[9] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. *The llama 3 herd of models*. Technical Report. Meta AI.

[10] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. *Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning*. Technical Report. DeepSeek-AI.

[11] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence*.

[12] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *Proceedings of the 10th International Conference on Learning Representations*.

[13] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. 2023. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation*. IEEE, 10608–10615.

[14] Wenlong Huang, Fei Xia, Dhruv Shah, Danny Driess, Andy Zeng, Yao Lu, Pete Florence, Igor Mordatch, Sergey Levine, Karol Hausman, et al. 2024. Grounded decoding: Guiding text generation with grounded models for embodied agents. In *Advances in Neural Information Processing Systems*, Vol. 36.

[15] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2023. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *Proceedings of the Conference on Robot Learning*. PMLR, 1769–1782.

[16] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. *Gpt-4o system card*. Technical Report. OpenAI.

[17] Shyam Sundar Kannan, Vishnunandan LN Venkatesh, and Byung-Cheol Min. 2024. Smart-llm: Smart multi-agent robot task planning using large language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 12140–12147.

[18] J-H Lee and C-O Kim. 2008. Multi-agent systems applications in manufacturing systems and supply chain management: a review paper. *International Journal of Production Research* 46, 1 (2008), 233–265.

[19] Huao Li, Yu Chong, Simon Stepputtis, Joseph P Campbell, Dana Hughes, Charles Lewis, and Katia Sycara. 2023. Theory of Mind for Multi-Agent Collaboration via Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 180–192.

[20] Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. 2024. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *Advances in Neural Information Processing Systems* 36 (2024).

[21] Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. 2023. Text2motion: From natural language instructions to feasible plans. *Autonomous Robots* 47, 8 (2023), 1345–1365.

[22] Jie Liu, Pan Zhou, Yingjun Du, Ah-Hwee Tan, Cees GM Snoek, Jan-Jakob Sonke, and Efstratios Gavves. 2024. CaPo: Cooperative Plan Optimization for Efficient Embodied Multi-Agent Cooperation. *arXiv preprint arXiv:2411.04679* (2024).

[23] Zhao Mandi, Shreeya Jain, and Shuran Song. 2024. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation*. IEEE, 286–299.

[24] Siddharth Nayak, Adelmo Morrison Orozco, Marina Ten Have, Jackson Zhang, Vittal Thirumalai, Darren Chen, Aditya Kapoor, Eric Robinson, Karthik Gopalakrishnan, James Harrison, et al. 2024. Long-Horizon Planning for Multi-Agent Robots in Partially Observable Environments. In *Advances in Neural Information Processing Systems*.

[25] Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. 2023. Do embodied agents dream of pixelated sheep: Embodied decision making using language guided world modelling. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 26311–26325.

[26] Frans A Oliehoek and Christopher Amato. 2016. *A concise introduction to decentralized POMDPs*. Springer.

[27] Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B Tenenbaum, Sanja Fidler, and Antonio Torralba. 2021. Watch-And-Help: A Challenge for Social Perception and Human-AI Collaboration. In *Proceedings of the 9th International Conference on Learning Representations*.

[28] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, Vol. 36. 8634–8652.

[29] David Silver and Richard S Sutton. 2025. Welcome to the era of experience.

[30] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2998–3009.

[31] Jiankai Sun, Chuanyang Zheng, Enze Xie, Zhengying Liu, Ruihang Chu, Jianing Qiu, Jiaqi Xu, Mingyu Ding, Hongyang Li, Mengzhe Geng, et al. 2023. A Survey of Reasoning with Foundation Models: Concepts, Methodologies, and Outlook. *Comput. Surveys* (2023).

[32] Gokul Swamy, Siddharth Reddy, Sergey Levine, and Anca D Dragan. 2020. Scaled autonomy: Enabling human operators to control robot fleets. In *2020 IEEE International Conference on Robotics and Automation*. IEEE, 5942–5948.

[33] Alejandro Torreno, Eva Onaindia, Antonín Komenda, and Michal Štolba. 2017. Cooperative multi-agent planning: A survey. *ACM Computing Surveys (CSUR)* 50, 6 (2017), 1–32.

[34] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. *Llama: Open and efficient foundation language models*. Technical Report. Meta AI.

[35] Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. 2023. Embodied task planning with large language models. *arXiv preprint arXiv:2307.01848* (2023).

[36] Zhuo Xu, Hao-Tien Lewis Chiang, Zipeng Fu, Mithun George Jacob, Tingnan Zhang, Tsang-Wei Edward Lee, Wenhao Yu, Connor Schenck, David Rendleman, Dhruv Shah, et al. 2024. Mobility VLA: Multimodal Instruction Navigation with Long-Context VLMs and Topological Graphs. In *Proceedings of the 8th Conference on Robot Learning*.

[37] Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, et al. 2024. ProAgent: building proactive cooperative agents with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17591–17599.

[38] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. 2024. Building Cooperative Embodied Agents Modularly with Large Language Models. In *Proceedings of the 12th International Conference on Learning Representations*.

[39] Gengze Zhou, Yicong Hong, Zun Wang, Xin Eric Wang, and Qi Wu. 2024. Navgpt-2: Unleashing navigational reasoning capability for large vision-language models. In *Proceedings of the 18th European Conference on Computer Vision*. Springer, 260–278.

[40] Gengze Zhou, Yicong Hong, and Qi Wu. 2024. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 7641–7649.

[41] Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, et al. 2021. Smarts: An open-source scalable multi-agent rl training school for autonomous driving. In *Proceedings of the Conference on Robot Learning*. PMLR, 264–285.

A Experimental Details

A.1 Implementation Details on LIET

Utility Functions Finetuning Details For the exploratory dataset, we collected 10 episodes for C-WAH benchmark and 24 episodes for TDW-MAT benchmark. We used 80% of this data as the training set to finetune a LLaMA 3.2-1B model with LoRA adapters and MLP layers as the value head. The architectural and training hyperparameters are listed in Table 5.

Table 5: Hyperparameters for utility functions.

Hyperparameter	Value
Number of MLP layers	2
MLP hidden dimension	32
LoRA target modules	[q_proj, v_proj]
LoRA rank (r)	8
LoRA alpha (α)	32
LoRA dropout rate	0.1
Batch size	64
Weight decay	0.1
Training epochs	20

Prompt Templates We provide example prompt templates for the LLM-based modules used in our framework for the C-WAH benchmark in the following.

Prompt for the planner module in C-WAH

I'm $AGENT_NAME$. I'm in a hurry to finish the housework with my teammate $OPPO_NAME$ together. Given our shared goal, dialogue history, and my progress and previous actions, please help me choose the best available action to achieve the goal as soon as possible.
Note that I can hold two objects at a time and there are no costs for holding objects. All objects are denoted as <name> (id), such as <table> (712).
Goal: $GOAL$
Progress: $PROGRESS$
Dialogue history:
Alice: "Hi, I'll let you know if I find any goal objects and finish any subgoals, and ask for your help when necessary."
Bob: "Thanks! I'll let you know if I find any goal objects and finish any subgoals, and ask for your help when necessary."
$DIALOGUE_HISTORY$
Previous actions: $ACTION_HISTORY$
Available actions: $AVAILABLE_ACTIONS$
Answer:

Prompt for the message generator module in C-WAH

I'm $AGENT_NAME$. I'm in a hurry to finish the housework with my friend $OPPO_NAME$ together. Given our shared goal, dialogue history, and my progress and previous actions, please help me generate a short message to send to $OPPO_NAME$ to help us achieve the goal as soon as possible.
Note that I can hold two objects at a time and there are no costs for holding objects. All objects are denoted as <name> (id), such as <table> (712).
Goal: $GOAL$
Progress: $PROGRESS$
Previous actions: $ACTION_HISTORY$
Dialogue history:

Alice: "Hi, I'll let you know if I find any goal objects and finish any subgoals, and ask for your help when necessary."
Bob: "Thanks! I'll let you know if I find any goal objects and finish any subgoals, and ask for your help when necessary."
$DIALOGUE_HISTORY$
Here are some hints to help you generate more useful messages based on previous experiences:
$KNOWLEDGE_LIST$
Note: The generated message should be accurate, helpful and brief. Do not generate repetitive messages and please output the message to send only.
Message:

Prompt for the reflector module in C-WAH

I'm $AGENT_NAME$. I'm in a hurry to finish the housework with my teammate $OPPO_NAME$ together.
Note that I can hold two objects at a time and there are no costs for holding objects. All objects are denoted as <name> (id), such as <table> (712).
Given a new piece of decision making experience based on our shared goal, my progress, previous actions, and our current plans, please reflect on the dialogue history and update the knowledge list to better guide future effective communication in cooperative planning.
Goal: $GOAL$
Progress: $PROGRESS$
Dialogue history:
Alice: "Hi, I'll let you know if I find any goal objects and finish any subgoals, and ask for your help when necessary."
Bob: "Thanks! I'll let you know if I find any goal objects and finish any subgoals, and ask for your help when necessary."
$DIALOGUE_HISTORY$
Previous actions: $ACTION_HISTORY$
Current plans: $CURRENT_PLANS$
Knowledge list:

——————————-

$KNOWLEDGE_LIST$

——————————-

Note: Please help update the knowledge list within the '——'. The updated list should integrate insights from both new experiences and previously accumulated knowledge, aiming to provide hints to enhance future communication message exchange between teammates. This knowledge list is shared among all teammates to enable effective communication to improve decision-making during our decentralized planning. Keep the list concise and informative, not exceeding 100 words.

Codebase We conduct our experiments using the following codebase:

- CoELA [38]: `https://github.com/UMass-Embodied-AGI/CoELA`

We implement the baselines following:

- ProAgent [37]: `https://github.com/PKU-Alignment/ProAgent`
- RoCo [23]: `https://github.com/MandiZhao/robot-collab`

The code for this work will be released publicly after the review process.

A.2 Additional Details on Environments

We follow the evaluation schemes established in previous work [38, 22]. For completeness, we include details of the experimental environments below.

A.2.1 TDW-MAT

Tasks TDW-MAT features two distinct tasks: the *food-transporting* and *stuff-transporting* challenges. Each task involves different target objects and container types. The agents' objective is to transport maximum target objects to designated goal positions, utilizing containers as transportation tools. Each container can accommodate up to three objects simultaneously, whereas agents without containers are limited to carrying only two objects at once. Success is measured by the number of objects successfully transported within a 3000-frame time limit.

Layouts The environment comprises 4 floorplans, each with 3 layout variations—two floorplans serve as training environments while the remaining two are reserved for testing.

The food-transporting task requires agents to collect 6 types of edible items (apple, banana, orange, bread, loaf bread, and burger) using 3 available containers (bowl, plate, and tea tray). Meanwhile, the stuff-transporting task involves gathering 6 types of personal items (calculator, mouse, pen, lighter, purse, and iPhone) using different carriers (plastic basket, wood basket, and wicker basket). Each task scenario contains 10 target objects and between 2-5 containers distributed throughout the environment.

The virtual space consists of four room types—living room, office, kitchen, and bedroom—with objects placed according to real-world expectations. For instance, food items are predominantly found in kitchen areas, while stuff items are typically located in office spaces.

Observation Space Agents in TDW-MAT receive these observations:

- **RGB image**: the egocentric image comes from the camera facing forward, with screen size 512 × 512 and field of view 90;
- **Depth image**: the depth image has the same camera intrinsic parameters as the RGB image
- **Oracle Perception** (optional): an image where each object id is mapped to a color and the camera intrinsic parameters are the same as the RGB image;
- **Agent position and rotation**: the agent's position and rotation in the simulation world;
- **Messages**: the messages sent by all the agents.

Action Space There are 7 types of actions for agents to interact with the environment or communicate with each other. Each action takes several frames and the detailed action space is listed here:

- `Move forward`: move forward 0.5m;
- `Turn left`: turn left by 15 degrees;
- `Turn right`: turn right by 15 degrees;
- `Grasp`: grasp an object, only the agent is close to the object can he perform the action successfully. The object can be either a target or a container;
- `Put in`: put the target into the container, only the agent is holding a target in one hand and a container in another hand can he perform the action.
- `Drop`: drop the objects held in hand;
- `Send message`: Send a message to other agents. In each frame, no more than 500 characters can be sent.

A.2.2 C-WAH

Tasks C-WAH features five household task scenarios: Prepare afternoon tea, Wash dishes, Prepare a meal, Put groceries, and Set up a dinner table. Each task represents a different domestic activity and consists of 3-5 subgoals expressed as predicates in the format "`ON/IN`(x, y)" (meaning "`Put` x `ON/IN` y'). Table 6 provides detailed descriptions of each task. Agents must satisfy all specified subgoals within a 250-step time limit to successfully complete a task.

Table 6: Tasks in C-WAH.

Task Name	Predicate Set
Prepare afternoon tea	`ON`(cupcake, coffeetable), `ON`(pudding, coffeetable), `ON`(apple, coffeetable), `ON`(juice, coffeetable), `ON`(wine, coffeetable)
Wash dishes	`IN`(plate, dishwasher), `IN`(fork, dishwasher)
Prepare a meal	`ON`(coffeepot, dinnertable), `ON`(cupcake, dinnertable), `ON`(pancake, dinnertable), `ON`(poundcake, dinnertable), `ON`(pudding, dinnertable), `ON`(apple, dinnertable), `ON`(juice, dinnertable), `ON`(wine, dinnertable)
Put groceries	`IN`(cupcake, fridge), `IN`(pancake, fridge), `IN`(poundcake, fridge), `IN`(pudding, fridge), `IN`(apple, fridge), `IN`(juice, fridge), `IN`(wine, fridge)
Set up a dinner table	`ON`(plate, dinnertable), `ON`(fork, dinnertable)

Observation Space Agents in C-WAH receive these observations:

- **RGB image:** Forward-facing egocentric view with 256×512 resolution and $60°$ field of view;
- **Depth image**: Depth information with identical camera parameters as the RGB image;
- **Oracle Perception** (optional): Color-coded object segmentation map where each object ID corresponds to a unique color, using the same camera configuration;
- **Agent position**: Coordinates indicating the agent's location in the simulation environment;
- **Messages**: Communication messages received from all agents in the environment.

Action Space There are 8 types of actions for agents to interact with the environment or communicate with each other. Each action takes several steps and the detailed action space is listed here:

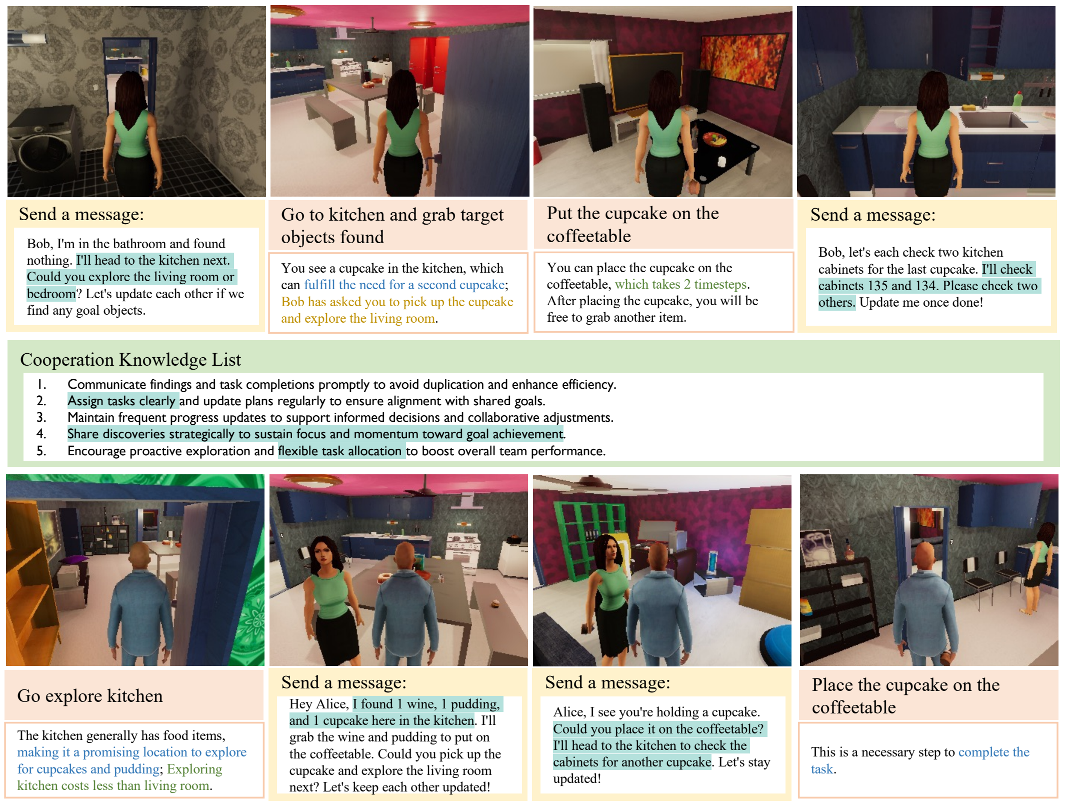
- `Walk towards`: Navigate to an object within the agent's current room or move to another room;
- `Turn left`: Rotate counterclockwise by $30°$;
- `Turn right`: Rotate clockwise by $30°$;
- `Grasp`: Pick up an object, executable only when the agent is nearby;
- `Open`: Open a closed container, executable only when the agent is nearby;
- `Close`: Close an open container, executable only when the agent is nearby;
- `Put`: Place held objects into an open container or onto a surface, executable only when the agent is close to the target location;
- `Send message`: Transmit a communication (maximum 500 characters) to other agents.

A.3 Experimental Infrastructure

The experiments were conducted using NVIDIA A100 GPUs. For LLaMA 3.1-70B APIs, we used the service provided by the TogetherAI platform. For ChatGPT-4o APIs, we used the service provided by OpenAI. Each experimental run required less than 2 days to complete.

B More Experimental Results

In this section, we provide additional visualization results of the proposed LIET policies in Figure 4. We observe that the decision process in this visualized episode comprehensively integrates task progress, efficiency, and dialogue history, further validating the conclusions presented in Section 4.2. In this particular episode, since both agents are concentrated in the kitchen and living room (where most target objects are located) during the early stages, we observe more intense discussions regarding labor division and progress updates. Notably, the knowledge list hints that agents should “assign tasks clearly” and “share discoveries strategically to sustain focus and momentum toward goal achievement”. This guidance leads to explicit task division proposals during communication. At the beginning of the episode, Alice suggests division at the room level, but as the episode progresses, the proposed division becomes increasingly specific—ultimately reaching the object level by the end, in alignment with the knowledge list’s recommendation for “flexible task allocation”. This cooperative strategy exemplifies LIET’s ability to foster effective collaboration through adaptive communication patterns.


Send a message:

Bob, I'm in the bathroom and found nothing. I'll head to the kitchen next. Could you explore the living room or bedroom? Let's update each other if we find any goal objects.

Go to kitchen and grab target objects found

You see a cupcake in the kitchen, which can fulfill the need for a second cupcake; Bob has asked you to pick up the cupcake and explore the living room.

Put the cupcake on the coffeetable

You can place the cupcake on the coffeetable, which takes 2 timesteps. After placing the cupcake, you will be free to grab another item.

Send a message:

Bob, let's each check two kitchen cabinets for the last cupcake. I'll check cabinets 135 and 134. Please check two others. Update me once done!

Cooperation Knowledge List

1. Communicate findings and task completions promptly to avoid duplication and enhance efficiency.
2. Assign tasks clearly and update plans regularly to ensure alignment with shared goals.
3. Maintain frequent progress updates to support informed decisions and collaborative adjustments.
4. Share discoveries strategically to sustain focus and momentum toward goal achievement.
5. Encourage proactive exploration and flexible task allocation to boost overall team performance.

Go explore kitchen

The kitchen generally has food items, making it a promising location to explore for cupcakes and pudding; Exploring kitchen costs less than living room.

Send a message:

Hey Alice, I found 1 wine, 1 pudding, and 1 cupcake here in the kitchen. I'll grab the wine and pudding to put on the coffeetable. Could you pick up the cupcake and explore the living room next? Let's keep each other updated!

Send a message:

Alice, I see you're holding a cupcake. Could you place it on the coffeetable? I'll head to the kitchen to check the cabinets for another cupcake. Let's stay updated!

Place the cupcake on the coffeetable

This is a necessary step to complete the task.

Figure 4: Snapshots of an example episode from the C-WAH benchmark with the task: “Find and put 1 wine, 2 cupcakes, 1 pudding onto the <coffeetable>”. Each agent’s visualization includes annotations of their plans and thoughts. The current cooperation knowledge list is also displayed. The message in the yellow dialogue box is directly generated by the LIET agent, while the thoughts are compact excerpts from the agent’s Chain of Thought (CoT) process.