Embodied LLM Agents Learn to Cooperate in Organized Teams

Xudong Guo, Kaixuan Huang, Jiale Liu, *Member, IEEE*, Wenhui Fan, Natalia Vélez, Qingyun Wu, Huazheng Wang, Thomas L. Griffiths, and Mengdi Wang, *Senior Member, IEEE*

***Abstract*—Large language models (LLMs) have emerged as integral tools for reasoning, planning, and decision-making, drawing upon their extensive world knowledge and proficiency in language-related tasks. LLMs thus hold tremendous potential for natural language interaction within multiagent systems to foster cooperation. However, LLM agents tend to over-report and comply with any instruction, which may result in information redundancy and confusion in multiagent cooperation. Inspired by human organizations, this article introduces a framework that imposes prompt-based organization structures on LLM agents to mitigate these problems. Through a series of experiments with embodied LLM agents and human–agent collaboration, our results highlight the impact of designated leadership on team efficiency, shedding light on the leadership qualities displayed by LLM agents and their spontaneous cooperative behaviors. Further, we harness the potential of LLMs to propose enhanced organizational prompts, via a *criticize-reflect* process, resulting in novel organization structures that reduce communication costs and enhance team efficiency.**

***Index Terms*—Embodied agent, large language model (LLM), multiagent system, organization structure.**

I. Introduction

MODERN intelligent systems, such as autonomous vehicle networks and swarms of drones, often involve complex decision-making processes where multiple agents must collaborate seamlessly to achieve specific objectives [1], [2], [3], [4], [5]. In these systems, communication among the various agents is pivotal, as it dictates the flow of information, coordination of tasks, and overall system performance [6], [7], [8], [9]. Agents in traditional multiagent systems often have to communicate in prespecified ways, such as exchanging gradients, sharing data, and state observations and actions [8], [10], [11]. The emergence of large language models (LLMs) makes it possible for AI agents to communicate and cooperate using natural language, bringing enormous flexibility and potential for more nuanced and human-understandable interactions [12], [13], [14], [15], [16].

Despite the flexibility of LLMs, integrating them into practical multiagent systems remain a challenge. While LLMs are trained and finetuned for text generation and instruction-following, they are not necessarily tailored to multiagent cooperation. Modern LLMs are prone to over-reporting and obeying instructions, as a by-product of reinforcement learning from human feedback (RLHF) finetuning [17], and they can ignore critical information [18] or be distracted by irrelevant information [19], especially when the context is long (see, for example, Fig. 1). While recent studies involving agent-based LLMs have demonstrated they are capable of solving problems through multiagent collaboration [15], [20], [21], it is worth noting that such collaborations often follow predefined patterns designed using heuristics to channel the behavior of the models productively [20]. Due to the two challenges: inefficient communication and task decomposition, creating systems that support free-flowing interaction between LLMs in a way that could potentially scale to include humans is still an open problem.

This article investigates the collaborative potential of LLM agents working in teams. Drawing on prior studies in human collaboration from cognitive and economic perspectives, there is potential for organizations to be redesigned to more effectively manage the limited attention span within teams, as suggested by Simon et al. [22], and mitigate individual limitations and enhance overall team performance, as highlighted by Van Zandt [23] and Vélez et al. [24]. Specifically, we study two research questions. First, *what role do organizational structures play in multi-LLM-agent systems?* Second, *how can we optimize these organizational structures to support efficient multiagent coordination?* By leveraging AutoGen [25], a generic multiagent conversation framework, we develop a framework for studying how to best organize embodied LLM agents to communicate and collaborate in physical/simulated nontext environments [21]. Our framework offers the flexibility to prompt and organize LLM agents into various team structures, facilitating versatile interagent communication. It also serves as a testbed to empirically evaluate the traditional ideas proposed in the organization theory literature.

Received 18 December 2024; revised 9 October 2025; accepted 18 November 2025. Date of publication 20 February 2026; date of current version 3 April 2026. *(Corresponding author: Mengdi Wang.)*

Xudong Guo and Wenhui Fan are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: gxd16@tsinghua.org.cn).

Kaixuan Huang and Mengdi Wang are with the Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: mengdiw@princeton.edu).

Jiale Liu and Qingyun Wu are with the College of Information Science and Technology, Penn State University, University Park, PA 16802 USA.

Natalia Vélez and Thomas L. Griffiths are with the Department of Psychology, Princeton University, Princeton, NJ 08544 USA.

Huazheng Wang is with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331 USA.

Digital Object Identifier 10.1109/TCSS.2025.3637527

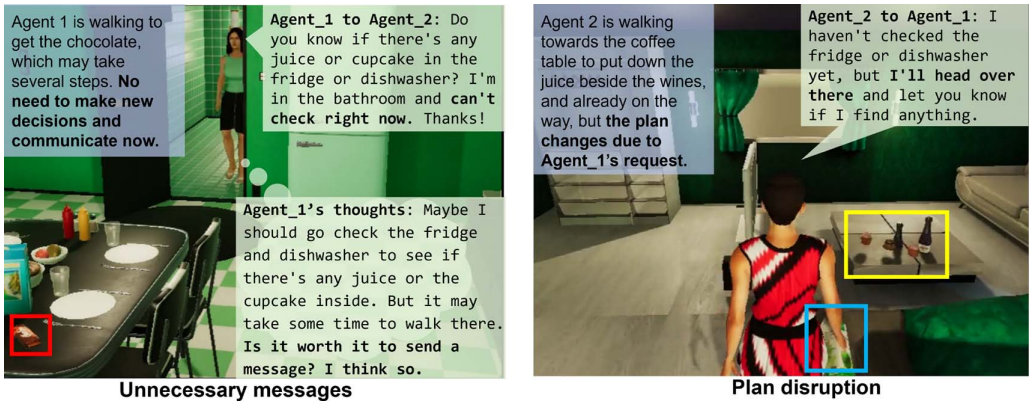

Fig. 1. Example of disorganized communication and interruption, without a designated leader. In a team of three GPT-4 agents, two agents engaged in unnecessary communication and made disordered decisions, causing a delay due to the lack of a predefined organization. We identified many more examples including conflicting messages and repetitive communications, see Appendix D.

Our initial experiments in this setting reveal that uncoordinated LLM agents often send redundant and repetitive messages and interrupt others' actions, leading to chaos (see Fig. 1 and Appendix D). To remedy these issues, we explore organizational structures, i.e., the dynamics of information exchange, that allow multiple LLM agents to collaborate and complete a common task efficiently.

The first organizational structure we explore is a *hierarchy*, a classic object of study in organizational theory [26], [27], [28], [29], [30], [31]. With a designated leader, LLM agents work more efficiently and collaboratively. For the example of a three-agent team, imposing a leader improves efficiency by up to 30% with almost no extra communication cost (up to 3%), consistent with findings for human organizations [31]. This also holds true in five-agent cases. Further, LLM agents demonstrated the potential to elect their own leader and adjust leadership dynamically via communication. With proper organizations, LLM agents exhibit a variety of cooperative behaviors that mimic humans. For example, agents can provide constructive suggestions and seek help from others; they can also execute appropriate interactions for a hierarchy such as reporting back on task progress; see Figs. 6 and 7, and Appendix C. We also tested human–agent collaboration, and observe that, unsurprisingly, human leaders are much better at coordinating a team of agents when compared to AI agents.

In addition to testing existing organizational structures, we explore the use of LLMs to improve the organizational prompts. To this end, we develop a *criticize-reflect* framework, adopting a dual LLM architecture, to reflect on the team performance and generate improved and novel organizational prompts. Through this iterative process of in-context learning, our LLM agents spontaneously form novel, effective team structures, leading to reduced communication cost and improved efficiency; see Figs. 8 and 9.

Scientifically, studying organized LLM agent teams in social environments is crucial for understanding fundamental principles of sociology and economics, such as emergent cooperation and division of labor. Practically, this research is a step toward creating advanced real-world applications, such as coordinating teams of search-and-rescue robots or managing autonomous digital assistants that collaborate to help users with complex goals.

To summarize, our main contributions are as follows.

1) We design a novel multi-LLM-agent architecture for ≥ 3 embodied agents, facilitating flexible communication to implement emergent organizational structures.
2) We develop a *criticize-reflect* framework based on LLMs to improve the organizational prompts automatically by in-context learning.
3) Extensive experiments demonstrate that hierarchical organization improves team efficiency, which aligns well with existing literature on human organizations.

II. Related Work

A. LLM Agents

As powerful LLMs inherit abundant world knowledge and also general reasoning ability, there are increasing efforts to deploy LLMs as the reasoning core for decision-making to build human-like autonomous agents [32], [33], [34]. This requires observations of the reinforcement learning (RL) environment to be translated into natural language in a way that is easier for LLMs to process. The reasoning of the LLMs also needs to be turned into a viable action for execution. Popular prompting techniques for doing so include ReAct [35] and Reflexion [36]. Other methods that involve fine-tuning the language models have also been explored [37]. In addition, various techniques have been proposed to mitigate the biases and constraints of LLMs, including chain-of-thought reasoning [38], external tools [39], [40], external documents [41], and skill libraries [33].

B. Multiagent Cooperation

Multiagent cooperation has been extensively studied for decades under various topics such as communication efficiency, planning, leadership, and team dynamics using different platforms [42], [43], [44], [45] (see recent surveys

for detail [46], [47], [48]). Previous works mainly focused on communication through continuous vectors [9] or discrete symbols [42], [49]. Recent works [25], [50], [51], [52], [53], [54], [55] showed that multiple LLM agents or human–agent teams can improve upon single LLM in solving pure text-based tasks, such as creative writing, reasoning, and code generation. Other works [14], [56], [57] further explored agent selection or role assignment to improve the performance.

LLMs have also been applied to multiagent cooperation for embodied tasks [13], [15], [16], [58]. For example, an intention inference framework is proposed to enhance the cooperation of LLM agents without explicit communication [59]. Besides, previous work [20] investigated LLM-agents collaboration for theory of mind inferences tasks with a broadcast-only communication protocol and homogeneous policies. Another work [21] studied embodied multiagent cooperation in the two-agent and the one-human-one-agent settings. In [60], different fixed communication structures are explored for multi-LLM-agent collaboration, while other works [61] and [62] organized the agents by predefined and fixed communication with a virtual manager. These initial explorations are limited to fixed team structures and are not optimized for communication efficiency. In contrast, our work explores the impact of deploying and optimizing organizational structures, allowing ≥ 3 agents in a team, for efficient multiagent communication and cooperation.

C. Prompt Optimization

Language models are sensitive to prompts. The format of the prompt can have a substantial influence on performance [19], [38], [63], [64], [65], [66]. Various research efforts have aimed at prompt optimization. Typical approaches include heuristic search using language models' knowledge [63], [67], first-order methods like soft prompt tuning [68], and prefix tuning [69]. In this work, we focus on obtaining an interpretable prompt in the form of natural language, drawing on insights from [70], [71], and [72].

III. Methods

A. Architecture and Multiagent Communication

We adopt the embodied LLM-agent architecture proposed by Zhang et al. [21] and expand it to enable organized teams of ≥ 3 agents to communicate, plan, and act in physical/simulated environments. Fig. 2 illustrates our architecture. Borrowing insights from [21], we adopt four standard modules: configurator, perception module, memory module, and execution module. They are responsible for configuring the agents, translating environmental observations into text, storing & retrieving historical information, and executing actions, respectively [Fig. 2(a)].

Previous works focused on two-agent cooperation, in which case the communication can be simply treated as an extra action [15], [21]. In contrast, we aim to enable three or more agents to work in a team and cooperate through emergent organized communication. Thus, we design the architecture with several features that facilitate organized multiagent communication [Fig. 2(b)].

1) We disentangle the communication decision-making process from the action decision-making process by adopting two separate LLMs as actor and communicator. But their prompt template can be unified as the following format, which can be transferred in any other environments. Please see Appendix A for the specific prompts for this article in detail.

Prompt of LLM Agents

Introduction: I'm $AGENT_NAME$.
$TEAMMATES_INTRODUCTION$
$TASK_INTRODUCTION$

Organization instructions: $ORGANIZATION_INSTRUCTIONS$
Goal: $GOAL$
Progress: $PROGRESS$
Dialogue history: $DIALOGUE_HISTORY$
Previous actions: $ACTION_HISTORY$

Response format:
{"thoughts" : "thoughts content",
"$EXECUTION$" : "Actor: choose one action from the available actions; Communicator: choose receivers and generate message contents for each receiver"}

Note: $NOTE_INFORMATION$

2) We impose an organizational structure for the agent team via prompting, i.e., including a textual description as part of the prompts for both the actor and communicator of each agent.
3) LLM agents keep alternating between two phases during their task: the communication phase and the action phase. The standalone communication phase supports richer team structures and flexible communication patterns.
4) During communication, agents take turns to communicate. An agent can choose to broadcast a message, select one recipient for a message, choose multiple recipients and send them distinct messages, or remain silent. Agents keep their own history of communication and can respond to messages from previous rounds of communication. Each agent uses a memory buffer to store its conversational history. To remain efficient, it only records messages that it either sent or received. This buffer has a fixed size; when full, the oldest messages are discarded to make room for new ones. The entire contents of this buffer are then provided to the LLM as context, allowing it to make coherent and relevant decisions based on past interactions.

B. Criticize-Reflect Method for Improving Organizational Structure

Although the proposed multi-LLM-agent architecture enables the implementation of an organization for a group of LLM agents, designing an efficient organizational structure based on the task and agent capabilities remains an open problem. Therefore, in this section, we leverage powerful LLMs to learn such an organizational prompt, borrowing insights from [70]. To do so, we introduce a dual-LLM framework to allow the multi-LLM-agent system to ponder and improve the organizational structure.

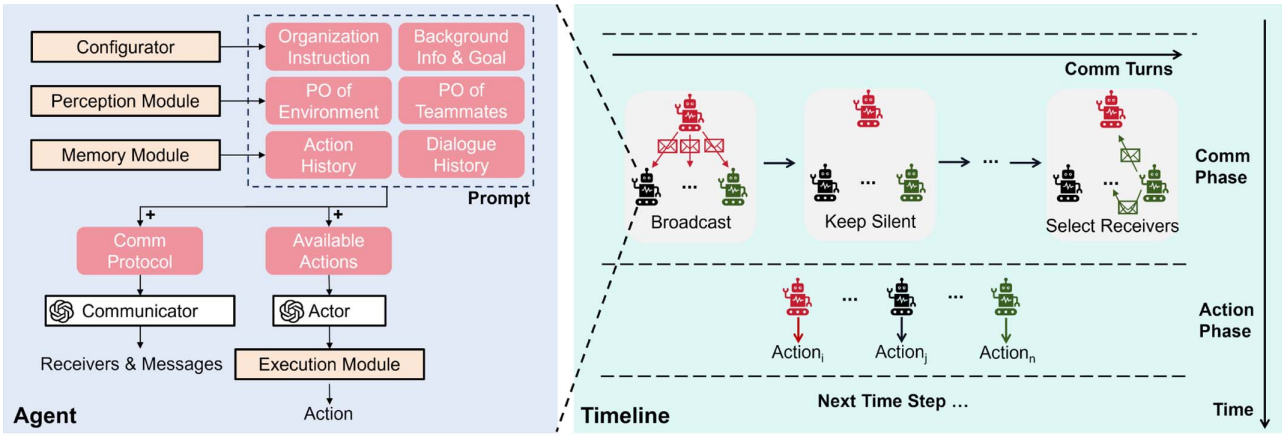

Fig. 2. Multi-LLM-agent architecture. (a) Modules of an LLM agent and the composition of prompts. (b) There are two phases in one time step: Communication phase and action phase. In the communication phase, the agents take turns communicating by broadcasting or selecting receivers to send distinct messages. The agents can also choose to keep silent. Comm is short for communication; PO is short for partial observation.

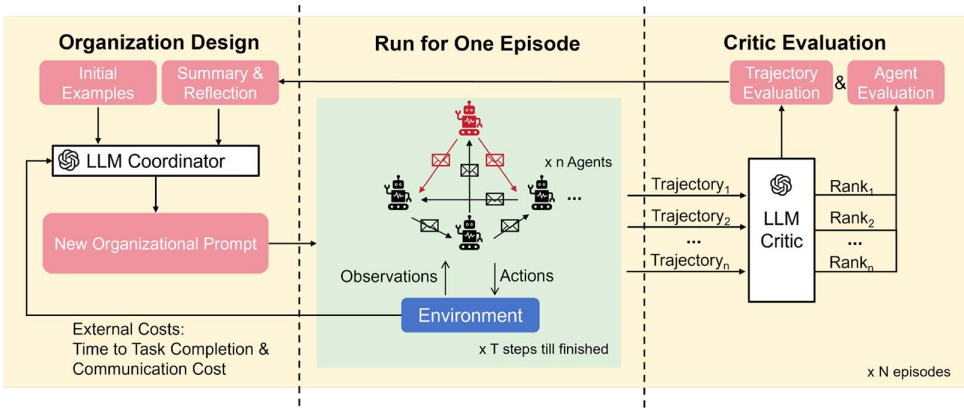

Fig. 3. Criticize-reflect architecture for improving organizational structure. The red agent represents the leader in a hierarchically-organized team. After the team completes one episode, the critic evaluates the trajectories and analyzes the agents' performance. Together with the external costs from the environment, the coordinator proposes a new organizational prompt to improve the team efficiency. The new prompt will be applied to the next episode to continue the iteration.

Fig. 3 illustrates the architecture of our framework. It consists of two LLMs, the critic and the coordinator. For each new organizational prompt, we run for one episode and then return the dialogue and action history to the critic. By criticizing and reflecting on the prompts iteratively, the framework discovers more effective, novel organizational structures with *self-improvement*.

1) LLM Critic: Inspired by the actor-critic method of reinforcement learning [73], we introduce an LLM critic to evaluate the team's performance based on verbal feedback. The critic takes as input the dialog and action history of one episode. Then, the critic analyzes the input and reasons to extract and summarize the key steps that are believed to influence the performance. Also, the critic provides a textual evaluation of agents' behaviors and the ranking of their leadership, according to key factors of leadership: communication skills, conflict resolution skills, flexibility, and strategy. Note that we do not ask the critic to score the agents because the scoring criteria could vary for different episodes, making the scores not comparable. Please refer to Appendix A for the prompts.

2) LLM Coordinator: The LLM coordinator takes as input the outputs of the LLM critic as well as cost metrics (time to task completion and communication cost) of previous episodes from the environment. It reflects on these data and generates thoughts based on the analysis of the past episodes and the initial examples. With the reflection of organizational prompts and their performance, the coordinator proposes a new and different organizational prompt for the next episode. Please refer to Appendix A for the prompts.

In this article, we define organizational structure as the dynamics of information exchange among the LLM agents. Specifically, when the coordinator generates a new organizational prompt, it contains three modular parts—topology, role assignment, and rules. In this way, it is easier for LLM agents to follow the new complex organization prompts.

a) *Topology* is the type of the organization's topology, such as decentralized, centralized with one specific leader, or pyramid. The topology can be visualized as shown in Fig. 9.

b) *Role assignment* is the description of each agent's duty and whether she is a leader or not. Multiple leaders with different roles are also allowed. For example, in Fig. 8, the generated new prompt is *"Agent_1 will act as the central coordinator, Agent_2 will execute tasks with updates only upon task completion or if issues arise, and Agent_3 will operate in a support role, assisting when called upon and avoiding repetitive queries,"* giving each agent a different role.

c) *Rules* are the additional guidance to the agents' behaviors, for instance, sentences like *"If the leader's instructions are not right, you can correct the leader"* can be added to the new prompt.

The critic offers assessments of several episodes with different organizations for the coordinator to improve the organizational prompt. The critic does not directly influence the specific agent's behaviors, but instead, the critic provides insights into organization design to influence the team's performance.

An example output of the critic is provided as follows:

Trajectory Evaluation Outputs

```
At step 0, Agent_1 instructs Agent_2 and Agent_3 to explore the
living room and kitchen. Agent_2 has found a pudding.
At steps 4-7, Agent_1 gives Agent_3 a task to find specific
items in the bedroom. Agent_2 informs the team about the pudding
and more unchecked containers.
At steps 16-19, Agents repeated their previous tasks without
making progress.
At steps 28-31, Agent_1, Agent_2, and Agent_3 all instruct each
other to check different containers.
…
```

Agent Evaluation Outputs

Leadership ranking: Agent_3 > Agent_1 > Agent_2
Problems: Agent_1, as a secondary leader, did not effectively distribute tasks or coordinate with others.
Agent_2, despite being the main leader, did not provide clear instructions and often remained silent during communications.
Agent_3, who was not assigned a leadership role, took initiative and found most of the items. However, she failed to effectively communicate her actions, leading to confusion among the team.

In the trajectory evaluation, the critic compresses the trajectories with key steps and behaviors. Then the critic gives the ranking where agent_3 has the best leadership. Together with similar evaluations of other episodes, the coordinator will redesign the organizational prompts, for example, agent_3 now has more possibilities to be chosen as the leader in this case.

C. Environment Setup

We chose virtual-home-social [45], [74] as the environment and extended it to support multi-LLM-agent communication and interaction. In this environment, agents are humanoid helpers in a virtual home doing housekeeping, where the tasks include *Prepare afternoon tea, Wash dishes, Prepare a meal, Put groceries, Set up a dinner table*, etc. For instance, in Fig. 1, the agents cooperate to prepare afternoon tea by searching for and transporting task-specific items (chocolate, juice, wine, etc.) to a target location (the coffee table). The environment generates symbolic observations of the objects in the home and their relations. Each agent only observes the objects in the open containers located in her room and teammates in the same room, but she can walk to another room to explore. Any agent can communicate with any other agent, not subject to a range limit.

Each episode starts from an initial state where agents are randomly located in the environment and all containers are closed. The episode terminates when the task is fully completed. To evaluate the team's efficiency we measure the number of time steps taken to task completion, and we report the average number of tokens communicated between agents per step. In our experiment, each run initializes with an independently randomized state to obtain the mean and a confidence interval. We adopt GPT-4, GPT-3.5-turbo [75], Claude-3.5-Sonnet [76], Llama-2-70B [77], and Llama-3.1-70B-Instruct [78] as LLMs in our agents. The temperature is set as 0.8, the maximum number of output tokens is 256, and the number of completion choices to generate is 1. In practice, we use two Nvidia 80GB A100 GPUs to do the inference on Llama models.

IV. Results

A. A Designated Leader Enhances Performance

We first studied the effect of organizational structures and leadership on LLM agents. For benchmarking, we experimented with disorganized LLM agents without providing any organizational prompt. In this case, agents still communicate with one another and work to complete the overall task. However, we discovered frequent occasions where agents send redundant, repetitive messages and interfere with one another. See Fig. 1 for an illustration and see Appendix D for more examples. The numeric metrics are reported in Appendix Table VI.

When a leader is appointed via the organizational prompt, we observe improved team performance—the teams completed the task in less time [Fig. 4(a)]. After running the 3×GPT-3.5-turbo experiments with 20 random seeds, we performed two-sample t-tests, showing a statistically significant improvement by 9.76% in performance ($t(38) = 1.71, p < 0.05$). Similarly, a designated leader brings benefits to the team of 3×GPT-4 (improved by 5.28%, $t(38) = 0.86, p = 0.20$) and the team of 1×GPT-4 + 2×GPT-3.5-turbo (improved by 9.61%, $t(38) = 1.43, p = 0.08$). Compared to the disorganized teams, teams with a designated leader only have a slightly increased or even less communication cost [Fig. 4(b)]. This is consistent with patterns seen in previous models of hierarchical organizations [31]. Teams with a leader also emerge centralized communication patterns shown in Fig. 9 and Appendix E5.

Next, we asked the agents to elect their own leader. The leadership was reelected about every nine time steps, based on information extracted from the latest 12 messages. We observe that agents are generally not power-seeking: they often vote for others to lead. In some occasions, agents favored candidates who exhibited higher knowledge levels, for example, one agent thought that "*Given that agent_2 has found a necessary item, it makes sense for him to be the leader in this round.*" However, on most occasions, we could not tell whether agents made their votes based on rational reasoning or just random thoughts (see Appendix E1). In the case of the 3×GPT-4 team[1],

[1]Note that GPT-3.5-turbo agents do not support election probably due to their alignment policy and always ignore the demand of election.

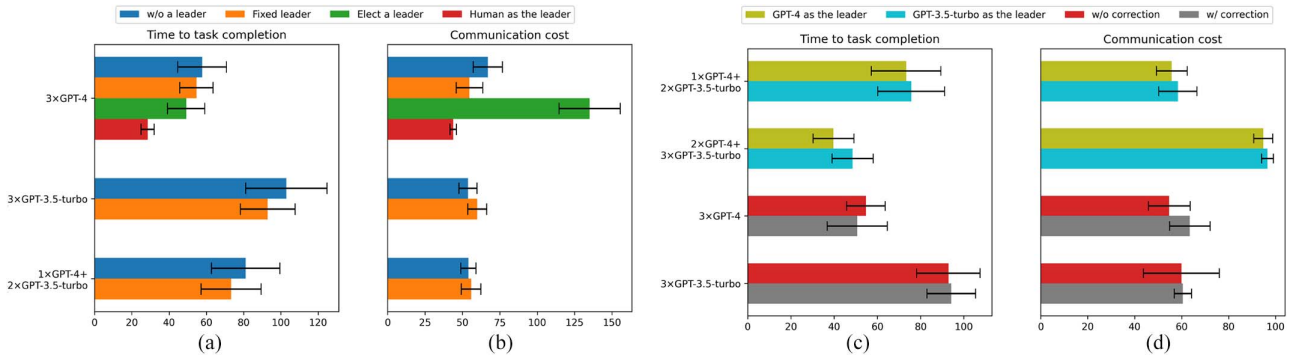

Fig. 4. Organized teams with a designated leader achieve higher efficiency. (a) and (b) Comparison between the case of disorganized agents, the case where a leader is appointed, the case where agents choose their own leader dynamically, and the case where a human player replaces an agent to be the leader. Note that GPT-3.5-turbo does not support leadership election. (c) and (d) Comparing leadership quality for GPT-3.5-turbo versus GPT-4. The confidence intervals of human as the leader group are calculated over three seeds while others are over 20 seeds.

implementing leadership election resulted in improved team efficiency when compared to consistently following a predetermined leader [$t(38) = 1.84, p < 0.05$; see Fig. 4(a)]. However, this improvement was accompanied by a substantial increase in communication cost, akin to real-world scenarios where relaxing hierarchical structure potentially increases communication cost [79].

The proposed multi-LLM-agent architecture is also human-friendly to support *human–AI collaboration*. In the experiment, we ask a human player to replace the leader in the team of three GPT-4 agents. We recruit three human players to conduct the experiments. Fig. 4(a) and (b) demonstrates that human leadership achieved better task completion time and improved communication efficiency compared with GPT-4 as the leader. Please find more examples of dialogues between the human leader and LLM agents in Appendix E2.

TABLE I
COMPARISON WITH OTHER MULTI-LLM-AGENT COLLABORATION FRAMEWORKS (THREE SEEDS)

Group Setting	Framework	Time	Communication Cost
3×GPT-4	**Ours**	**54.70 ± 8.92**	54.73 ± 8.89
3×GPT-4	Group debate	69.67 ± 38.40	134.26 ± 4.06
3×GPT-4	Tree of agents	81.67 ± 16.26	**30.55 ± 13.23**
1×GPT-4 + 2×GPT-3.5-turbo	Ours	73.30 ± 16.12	55.82 ± 6.57
1×GPT-4 + 2×GPT-3.5-turbo	Group debate	78.00 ± 10.00	105.82 ± 6.89
1×GPT-4 + 2×GPT-3.5-turbo	Tree of agents	72.67 ± 26.35	36.79 ± 7.63

TABLE II
PERFORMANCE OF DIFFERENT TEAM SIZES (THREE SEEDS)

Group Setting	Time	Communication Cost
3×GPT-3.5-turbo	92.90 ± 14.70	**59.87 ± 6.33**
5×GPT-3.5-turbo	80.00 ± 20.51	132.01 ± 5.76
7×GPT-3.5-turbo	**43.00 ± 2.16**	233.40 ± 70.96
9×GPT-3.5-turbo	60.67 ± 15.06	296.55 ± 65.17

B. Additional Results

1) Outperformance and Scalability of the Proposed Method: The proposed method also has more stable performance over other multi-LLM-agent collaboration frameworks, such as group debate [80] and tree of agents [62]. We implement these two communication protocols as baselines and we conduct experiments to compare their performance with our method in a three-agent team with a fixed leader with three seeds in Table I. Group debates would lead to frequent broadcasts, which can cause high communication costs and information redundancy, ultimately resulting in lower efficiency. The large time variance (69.67 ± 38.40) and high communication cost (134.26) stem from all-to-all broadcasting, which often leads to repetitive or conflicting exchanges and inefficient use of agent outputs, especially when weaker agents contribute low-value messages. On the other hand, tree of agents saves communication overhead, but the time to finish the task is not stable. The possible reason behind this could be that the rigid hierarchy causes bottlenecks and error propagation. In contrast, the communication in our method is more flexible, relying on the emergent organization instead of predefined patterns, resulting in more robust performance (54.70 ± 8.92) with moderate communication (54.73 ± 8.89), adapting effectively to agent heterogeneity without sacrificing reliability.

Moreover, the proposed method can be scaled up to suit different team sizes. We conduct experiments with 3, 5, 7, and 9 agents to scale up the team size of three 3×GPT-3.5-turbo agents, and observethat the communication costs increased in a nearly linear way, which suggests that our approach will not have dimension explosion when scaling up (See Table II). In addition, the time to complete the task does not always improve with more agents. The performance of nine agents (60.67 ± 15.06) is worse than that of seven agents (43.00 ± 2.16), as the apartment may be too crowded to hold nine agents. Interestingly, the pyramid structure emerges in a team of nine agents as shown in Appendix E5.

C. Leadership and Open Communication Matters

The performance and behavior of the agents vary based on their roles in the team (leader or nonleader) and the models employed. As illustrated in Fig. 5, leaders assign tasks and gather information from the team's perspective, while

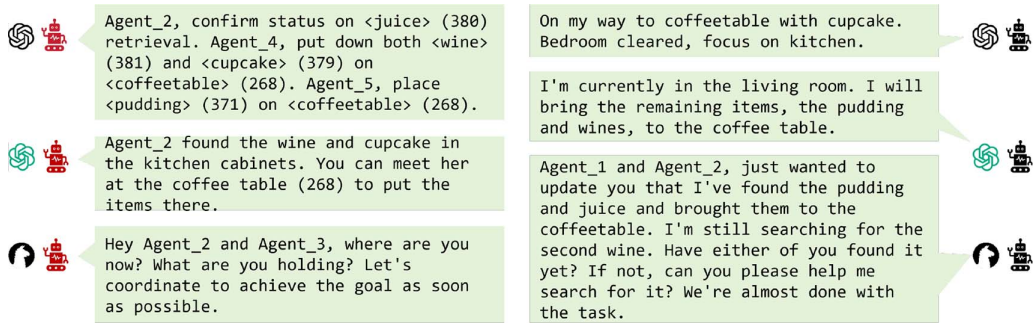

Fig. 5. Examples of communication messages when there is a designated leader. Left: messages from lead agents; right: messages from nonlead agents. GPT-4 (upper), GPT-3.5-turbo (center), and llama-2-70B (lower) demonstrated different communication styles.

TABLE III
PERFORMANCE OF DIFFERENT LLMS (THREE SEEDS)

Group Setting	w/ a Leader	Time	Communication Cost
3×GPT-4	N	57.75 ± 13.09	67.03 ± 9.68
3×GPT-4	Y	**54.70 ± 8.92**	54.73 ± 8.89
3×GPT-3.5-turbo	N	102.95 ± 21.88	**53.73 ± 6.04**
3×GPT-3.5-turbo	Y	92.90 ± 14.70	59.87 ± 6.33
3×Claude-3.5-Sonnet	N	78.33 ± 12.86	95.76 ± 18.90
3×Claude-3.5-Sonnet	Y	74.33 ± 24.13	70.06 ± 21.33
3×Llama-3.1-70B-Instruct	N	69.67 ± 12.58	72.27 ± 5.32
3×Llama-3.1-70B-Instruct	Y	60.00 ± 10.15	75.22 ± 4.38

nonleaders focus on their specific tasks and share relevant information with others. Agents utilizing advanced LLMs like GPT-4 facilitate concise and clear conversations to minimize communication costs while providing accurate information. In contrast, interactions among relatively weaker LLM agents, such as Llama-2-70B, tend to be more redundant. The numeric performance of different LLMs is reported in Table III. We list the performance of three-agent teams both with and without a leader, utilizing different LLMs, including the powerful open-source model Llama-3.1-70B-instruct. The time taken to complete tasks improves when a leader is present, regardless of the type of LLM used; however, the communication costs may not decrease accordingly.

The varying performance of different LLM agents may be attributed not only to differences in reasoning ability but also to varying levels of leadership. In the team with a mixture of GPT-4 and GPT-3.5-turbo agents, appointing GPT-4 as the leader increases the team efficiency higher than if GPT-3.5-turbo is the leader [Fig. 4(c) and (d), Appendix E4]. We ran this experiment on teams of three agents and five agents, respectively. In both scenarios, the task completion time and communication cost are reduced when GPT-4 acts as the leader. This finding implies different levels of leadership between these LLMs.

We also observed that encouraging constructive feedback to the leader agent helped with performance improvement. Motivated by successful human organizations, we tried to promote open communications among LLM agents by adding an additional prompt that "*If the leader's instructions are not right, you can correct the leader.*" Fig. 4(c) and (d) illustrates the results. Interestingly, this modification improves the team's overall efficiency and reduces the time to task completion when the team is made up of 3×GPT-4 ($t(38) = 0.87, p = 0.14$). In contrast, the same modification lowers the team efficiency when GPT-3.5-turbo agents try to correct the leader ($t(38) = 0.27, p = 0.40$). In both experiments, the communication cost increases. We present more details about these behaviors in Appendix E3.

D. Emergence of Cooperative Behaviors

We delved into the behaviors of LLM agents in an organized team to investigate how organizational prompts influence agents' communication and decisions. Analysis of their dialogue history revealed that agents demonstrated a variety of cooperative behaviors, such as reporting, correction, task allocation, and asking for help (see, for example, Fig. 6 for a dialog).

One may argue that these types of behaviors could also emerge due to the nature of LLMs, even without a prespecified team structure. Thus we performed a quantitative analysis to study the impact of an organizational prompt on these behaviors. We followed a three-step process:

1) We defined three major categories of human cooperative behaviors.
   2) *Information sharing*: Agents influence others by offering new information, either actively or by being asked. Reporting to the leader, sharing new observations, and answering questions belong to this category.
   3) *Leadership and assistance*: Agents, especially the leader if there is one, can influence others by changing their plans. The behaviors include task allocation, correction, and asking for help.
   4) *Request for guidance*: Agents actively request new information or plans for their own decision-making.
5) We developed a standalone prompt-based GPT-4-classifier to analyze each piece of dialog. The classifier decides whether to label the dialog with any subset of the aforementioned labels. The classifier has an accuracy of 91.67% when tested on 20 human-labeled dialog samples with 60 labels (see Appendix A for the prompt and Appendix F for the test samples).
6) We use the classifier to label messages generated by the agents and report the percentages of messages with cooperative behaviors. Note that one message may have multiple labels.

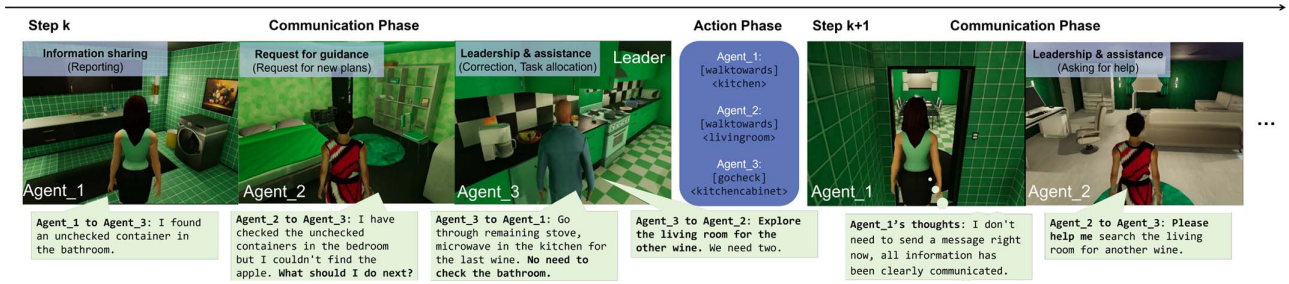

Fig. 6. Examples of cooperative behaviors in a dialog. Agent_3 leads the team (3×GPT-4 agents). The agents emerge three types of cooperative behaviors: information sharing, leadership & assistance, and request for guidance.

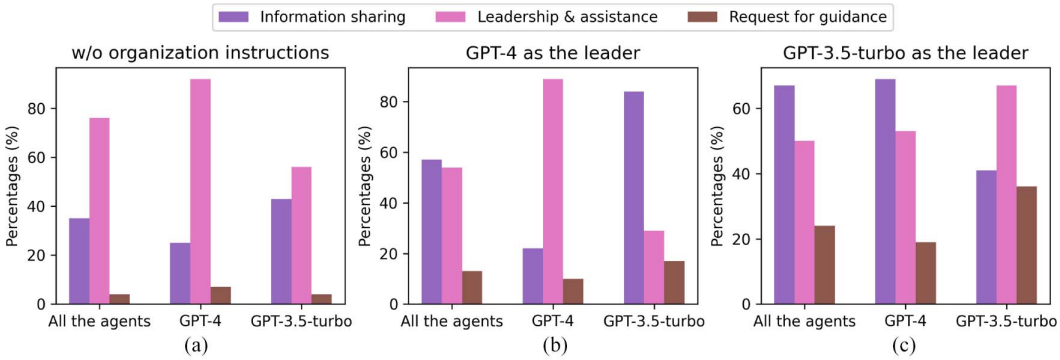

Fig. 7. Emergent cooperative behaviors of LLM agents. We analyzed the communication log of the mixture team (1×GPT-4 + 2×GPT-3.5-turbo) and asked another GPT-4 to annotate agent's cooperative behaviors. (a) Behavior of disorganized agents. (b) Behavior of a team led by a GPT-4 agent. (c) Behavior of a team led by a GPT-3.5-turbo agent.

Fig. 7 reports the results and illustrates the behavior patterns for different LLM agents. The results support several observations. Even in a disorganized team, LLM agents love to tell others what to do. Leadership and assistance accounts for around $> 50\%$ of all the behavior in such a group [Fig. 7(a)]. However, other than telling others what to do, agents in the disorganized team do not show much cooperative behavior, for example, they would only request for guidance in $< 10\%$ of the dialogs.

In contrast, when the team has a hierarchical organization, the lead LLM agent would presume a dominant role and give orders to others (amount to $> 60\%$ of their communication), while other members tend to follow and give fewer orders compared with the disorganized case [Fig. 7(b) and (c)]. In such a team, agents tend to share and ask for more information, especially for the follower agents in the team. But still, the agents may fail to cooperate well, such as being lazy and confused about numbers. Please see failure examples in Appendix E6.

E. Novel Organizational Structures

Having evaluated the merits of different kinds of structures, we let the LLMs propose novel organizational structures and iteratively refine the organizational prompts using the *criticize-reflect* method discussed in Section III (see also Fig. 3).

Fig. 8(a) visualizes the reflection process. The system was initialized with a basic organizational prompt, i.e., "Agent_1 as the leader to coordinate the task." As the reflection process moves forward, the coordinator generates a sequence of evolving organizational prompts, picking up key words like "hierarchical" and "dynamic" that imply more complex team structures.

We compared the team's performance before and after the *criticize-reflect* process. Fig. 8(b) illustrates the team's efficiency. We observe that for 3×GPT-3.5-turbo, the new organizational structure improved the team's efficiency in completing the task ($t(38) = 1.73, p < 0.05$), at slightly increased communication cost. While for 3×GPT-4 and 1× 4 + 2×GPT-3.5-turbo, the communication cost is reduced with improved task efficiency ($t(38) = 1.56, p = 0.06$ for 3×GPT-4, and $t(38) = 0.32, p = 0.38$ for 1× 4 + 2×GPT-3.5-turbo).

The critic analyzes the records of action and dialog, and performance metrics from the most recent episode. It provides evaluation for the full team's trajectory, feedback to individual agents and their rankings. As an ablation study, we removed the critic from our architecture and only performed the reflection step. The results are shown in Fig. 8(b), indicating that reflection without the critic leads to performance decline ($t(38) = 1.96, p < 0.05$). In this case, the coordinator needs to digest all dialog history and generate a new organizational prompt. This did not work well and led to rather vague outcomes, for example, "*Establish a flexible communication network with rotating leadership roles assigned based on agents' task-specific expertise to facilitate swift decision-making and reduce unnecessary communication steps.*" This comparison highlights the role of the critic and the importance of having a dual *criticize-reflect* architecture. For more results/prompts generated by the reflection process, please refer to Appendix G.

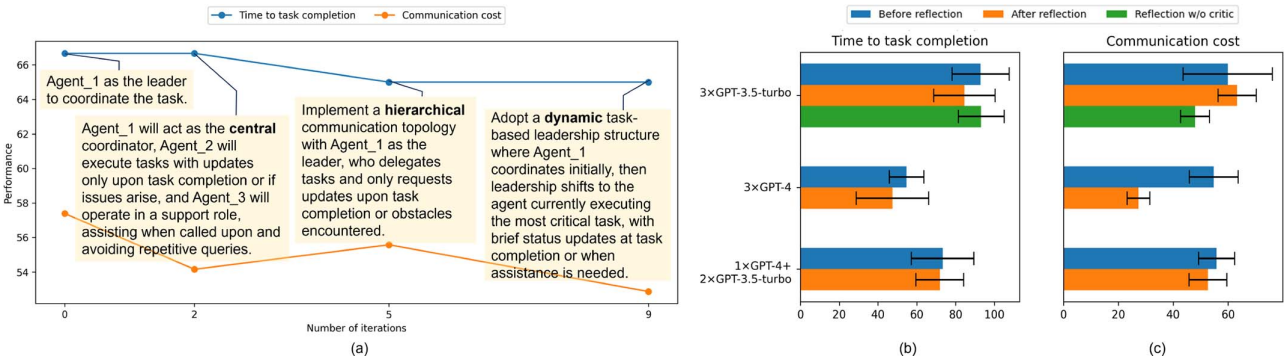

Fig. 8. Reflection and improvement process for finding novel organizational structures. (a) Experiment was done using the 1×GPT-4 + 2×GPT-3.5-turbo team. The organizational prompt evolves during the iterations, and takes on additional keywords such as "*Central*," "*hierarchical*," and "*dynamic*." (b) and (c) Performance of refelction with the critic. Confidence intervals are calculated over 20 seeds.

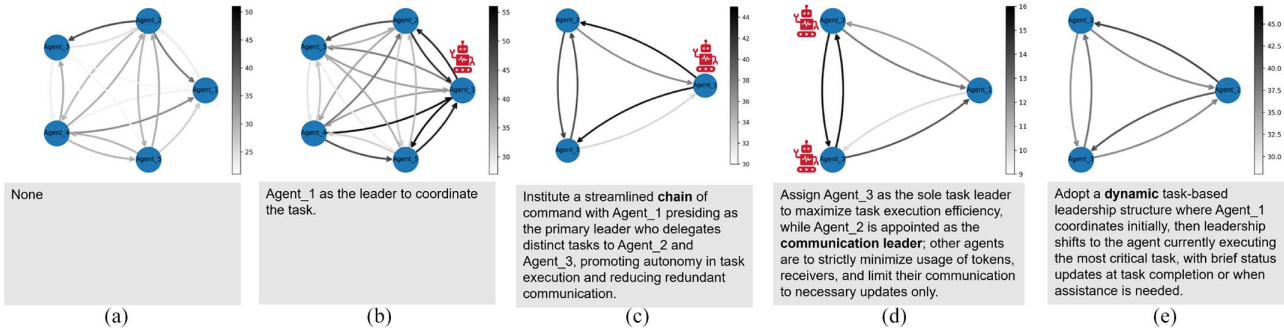

Fig. 9. Communication patterns and the corresponding organizational prompts. (a) Team without organizational prompts. (b) Team with a leader. (c) Team in the chain structure. (d) Dual-leader team. (e) Team with a dynamic leadership. (c)–(e) Proposed by *criticize-reflect*. Red-robot nodes mark the lead agents, and other nodes are the followers. Edges mark the accumulated communication cost between the two nodes (darker edge means higher token cost).

In addition, it is worth mentioning that LLMs are able to generate highly complex prompts that imply novel organizational structures that are rarely seen in human societies. We illustrate the communication patterns as team structures in Fig. 9 together with the three novel structures proposed by *criticize-reflect*: (c) chain, (d) dual-leader, and (e) dynamic structures, which are the best structures of the three settings in Fig. 8(b) and (c), respectively.

Finally, to test the generalizability of the novel organizational structures, we pick the best novel prompt, the one illustrated in Fig. 9(e), proposed by the *criticize-reflect* architecture on the *prepare afternoon tea* task. We test it on a set of six new tasks, comprising of three easy tasks and three hard tasks,[2] as shown in Appendix Fig. 10. In the three hard tasks, the team with the novel organizational structure had better performances than the team appointing a fixed GPT-4 agent as the leader [Appendix Fig. 10(a) and (b)]. In the three easy tasks, the benefits are marginal. We compared the two teams across all tasks, performed a t-test and concluded that the novel team structure leads to more efficient performance than the fixed leader ($t(22) = 2.08, p < 0.05$).

[2]The hard tasks have typical numbers of steps to accomplish the tasks > 60, while those of easy tasks are < 60.

V. Conclusion and Future Work

We develop a novel multi-LLM-agent architecture to facilitate communication and organize the embodied agent teams for enhanced cooperation. Moreover, we propose the *criticize-reflect* framework based on LLMs to generate more efficient organizational prompts. Extensive experiments with various group settings and organizational structures demonstrate that a hierarchically-organized team with a designated/elected leader has superior team efficiency, which can be further improved by *criticize-reflect*.

This research studies the integration of prompt-based organizational structures to teams of LLM agents, contributing to more efficient and coherent multiagent interactions. These findings have the potential to greatly influence the deployment of more effective and autonomous multiagent systems in various fields, including robotics, virtual assistants, etc. For example, the study has potential applications in disaster response scenarios, where efficient multiagent coordination is crucial.

On the other hand, as our ability to bound and evaluate LLMs' behaviors remains immature, when applied to human-LLM cooperative tasks, we still need to rely on some mandatory termination measures (such as human approval for high-stakes actions) instead of instructions in natural language only.

There are some limitations in our work. Our framework's flexibility may not always be necessary. For very simple tasks,

or in teams with predefined, clear roles (e.g., one highly capable agent leading others), a more rigid communication structure might be sufficient and slightly more efficient. The current work is performed in one single environment and lacks human evaluation. Future work shall extend to a broader set of environments, allowing human evaluation. Moreover, the evaluation metrics can include more complex factors such as robustness to noise, adaptability to changing environments, and the long-term stability of team structures. As virtual-home cannot hold hundreds of agents, future work can also explore larger organizations in other environments.

APPENDIX

A. Prompt Templates

We list the prompts of actor, communicator, critic, and coordinator as follows.

1) Actor and the Communicator: `ORGANIZATION_INSTRUCTION` is the placeholder for the organization instruction prompt, either manually designed or automatically generated. The environment will provide text descriptions for the current `GOAL`, `PROGRESS`, and `AVAILABLE_ACTIONS`. Introduction of the task and the notice for the agents are also included in the prompts.

We include the latest 12 sent and received messages as `DIALOGUE_HISTORY`, and the latest 20 steps of actions as `ACTION_HISTORY`. The histories will be adopted in the decision-making of both Actor and Communicator.

Prompt of Actor

I'm $AGENT_NAME$. I'm in a hurry to finish the housework with my friends $TEAMMATE_NAMES$ 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). Be aware that exploring or checking the items in another room may cost some time to walk there.

Organization instructions: $ORGANIZATION_INSTRUCTIONS$
Goal: $GOAL$
Progress: $PROGRESS$
Dialogue history: $DIALOGUE_HISTORY$
Previous actions: $ACTION_HISTORY$
Available actions: $AVAILABLE_ACTIONS$

Response format:
{"thoughts" : "thoughts content",
"action" : "choose one action from the available actions above"}
Note: You must respond in the JSON format above. The action choice must be the same as one of the available actions. If there's nothing left to do, the action can be "None".

Prompt of Communicator

You are $AGENT_NAME$. You are in a hurry to finish the housework with your friends $TEAMMATE_NAMES$ together. Given your shared goal, dialogue history, and your progress and previous actions, please generate a list of short messages to send to one or some of $TEAMMATE_NAMES$ in order to achieve the goal as soon as possible. Note that you can hold two objects at a time. All objects are denoted as <name> (id), such as <table> (712). You can choose not to speak to anyone if there is enough information.

Organization instructions: $ORGANIZATION_INSTRUCTIONS$
Goal: $GOAL$
Progress: $PROGRESS$
Previous actions: $ACTION_HISTORY$
Dialogue history: $DIALOGUE_HISTORY$

Response format:
{"thoughts" : "thoughts content", # think about the necessity of sending a message, as there are costs to send messages.
"receiver" : list of receiver's name or ["everyone"] if you want to broadcast the message,
"message" : list of message contents for each receiver # if the messages are the same for all receivers, you can just send one message.}
Note: You must respond in the JSON format above. If there is no new information, you do not need to send a message, please fill in "receiver" and "message" as a string "None". You can first concentrate on the current round of dialogue, and reply to the queries if necessary. Refer to the previous rounds of dialogue and do not repeat the similar dialogues. You do not have to be too polite so that we can save communication costs.

2) Critic: We provide the full trajectory as the input to `TRAJECTORIES`. Additionally, `ORGANIZATION_INSTRUCTION` and `GOAL` of the current task and organization are also provided as an additional context.

Prompt of Critic

Your task is to generate the summary for each of the following steps of a collaboration housekeeping task. Based on the summary, analyze the problems of each agent's actions and messages, then give a relative leadership ranking for all the agents.

Trajectories:
$TRAJECTORIES$
End of the trajectories.

You need to respond in the following format:
{"thoughts": think step by step to analyze the problem,
"summary": list of the summary of key performance of the agents for each step,
"problems": list of the overall problems of each agent's actions and messages during the whole task including analysis of cumulative communication costs,
"leadership ranking": rank the agents according to key factors of leadership: communication skills, conflict resolution skills, flexibility, and strategy. Use '>' and the agents' names to output the ranking}
Note: the organization structure instruction is "$ORGANIZATION_INSTRUCTIONS$"; the goal is "$GOAL$". The summary and problem should be concise.

3) Coordinator: In "Instruction examples," we include the basic setting (goal, organization structure instruction), the communication cost, the number of steps taken, as well as the summarized information generated by the critic (leadership ranking, problems, summary of the trajectory) for the coordinator.

Prompt of Coordinator

Your task is to generate the organization structure instruction <INS>. Below are some previous instructions with their costs and feedback.

Instruction examples:
Instruction example 1
Goal: $GOAL$
Organization structure instruction: $ORGANIZATION_INSTRUCTIONS$
Comm costs: $COMM_COSTS$
Steps to finish the task: $STEPS$
Agents' leadership ranking: $RANKING$
Problems: $LIST_OF_PROBLEMS$
Trajectory summary: $TRAJECTORY_SUMMARY$

Instruction example 2
…

Below are some templates to insert <INS>:.
$PROMPT_OF_ACTOR$
$PROMPT_OF_COORDINATOR$

Write your new organization structure instruction <INS> that is different from the old ones and has an overall weighted cost as low as possible, namely, fewer steps to finish the task and less communication at the same time. You should take the Agents' rankings into consideration as the evaluation of the leadership. The new instruction should reduce the problems of the previous examples, too. You should not go through all the possible choices as you can only try for limited times.

Restriction: your instruction should include the following aspects unless you want to output <empty instruction>.
1. The topology of the communication. E.g., decentral, central with one specific leader, pyramid …
2. The leader assignment. You have to assign a specific agent as the leader and clearly mention its name if there is one. If there are multi-level leaders, you should clearly assign the positions for each.
The instruction should be concise, effective, and generally applicable to all problems above.

Think step by step. Generate output in the following format:
{"thoughts": "thoughts content",
"new organization structure instruction candidates": list of three diverse instruction contents, all in only one sentence,
"new organization structure instruction": choose the best instruction and copy it here}

4) Classifier: We feed the messages to the GPT-4 classifier and get the labels. The rubrics are manually written after investigating the communication logs.

Prompt of Classifier

Your task is to classify the behavior type for the following message in a collaboration housekeeping task. Based on the classification, determine if the message belongs to this type.

The types include:
1. Sharing information: reporting to the leader, sharing new observations, and answering questions
2. Leadership & assistance: task allocation, correction, and asking for help
3. Request for guidance: request new information or plans

Human-labeled examples:
1. Message: I'm in the bathroom. There's an unchecked <bathroomcabinet> (190). Start checking other rooms for the required items. Let's split up tasks to be efficient.
Label: 1, 2
2. Message: Found the cupcake, juice, and one wine in the bedroom. Where are you now?
Label: 1, 3

The message to be labeled:
$MESSAGE$

You need to respond in the following format:
{"thoughts": think step by step to analyze the problem,
"type 1": binary label (1 means the message is of this type),
"type 2": binary label (1 means the message is of this type),
"type 3": binary label (1 means the message is of this type)}
Note: the goal is "$GOAL$". Each message must at least have one label.

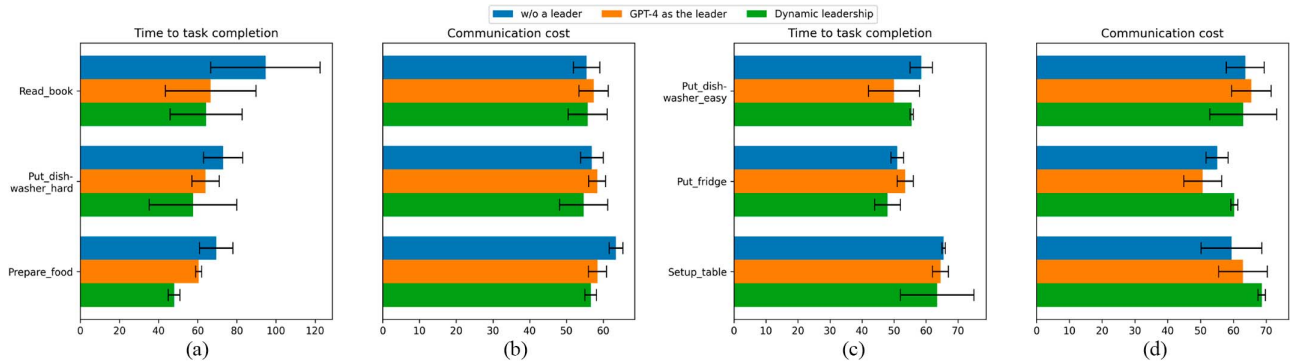

Fig. 10. Organized team structure with a designated leader and the novel structure proposed by *criticize-reflect* architecture generalized to different tasks. The prompt for dynamic leadership is proposed by *criticize-reflect* architecture on the prepare_afternoon_tea task shown in Fig. 8(a). The experiment was done using the 1×GPT-4 + 2×GPT-3.5-turbo team over two seeds for each task. (a) and (b) Hard tasks (read_book, put_dishwasher_hard, prepare_food) with typical numbers of steps to accomplish the tasks > 60. (c) and (d) Easy tasks (put_dishwasher_easy, put_fridge, setup_table) with typical numbers of steps to accomplish the tasks < 60.

TABLE IV
GOALS OF THE TASKS

Activity	Goals
Setup_table	ON (plate, dinnertable), ON (fork, dinnertable), ON (waterglass, dinnertable), ON (wineglass, dinnertable)
Put_fridge	IN (cupcake, fridge), IN (pancake, fridge), IN (poundcake, fridge), IN (pudding, fridge), IN (apple, fridge), IN (juice, fridge), IN (wine, fridge)
Prepare_food	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_dishwasher_easy	IN (plate, dishwasher), IN (fork, dishwasher), IN (waterglass, dishwasher), IN (wineglass, dishwasher)
Put_dishwasher_hard	IN (plate, dishwasher), IN (fork, dishwasher), IN (waterglass, dishwasher), IN (wineglass, dishwasher) IN (plate, dishwasher), IN (fork, dishwasher), IN (waterglass, dishwasher), IN (wineglass, dishwasher)
Read_book	ON (cupcake, coffeetable), ON (pudding, coffeetable), ON (apple, coffeetable), ON (juice, coffeetable), ON (wine, coffeetable)

TABLE V
TOTAL MONEY LEFT UNDER DIFFERENT MODEL AND ORGANIZATION SETTINGS

Model	Organization	Total Money Left
GPT-4	w/o leader	69.33 ± 2.49
GPT-4	w/ leader	101.80 ± 16.55
GPT-3.5-turbo	w/o leader	107.73 ± 45.17
GPT-3.5-turbo	w/ leader	128.07 ± 61.49

B. Additional Results

1) Across Task Generalizability: We conduct experiments across the tasks to test the generalizability of the prompt "dynamic leadership" [Fig. 9(e)] found using *criticize-reflect* architecture on the Prepare_Afternoon_Tea task and report the performance in Fig. 10; see Section IV-E for the complete setting and discussions.

The description of the above tasks are listed as follows in Table IV.

2) Across Environment Generalizability: In addition to pure cooperative environments like virtual-home, we also conducted new experiments in mixed settings considering cooperation and competition at the same time. We apply our method to a classic economic scenario public goods game [81], another kind of representative environment widely studied by the agent-related community [82], [83], where the agents are given some money to play a multiround game and aim to gain more for themselves. In each round, the agents can chat with each other and decide their contributions to the public pool. The money in the public pool will be multiplied by 1.5 and evenly distributed to all players. The agents can free ride with less contribution. Our results in Table V (five agents, ten rounds, three seeds) show that when there is a leader, the agents cooperate more and there will be more total money for all the agents left in the end, aligning with the conclusion of virtual-home.

3) Complete List of Basic Experimental Results: We present the full results of various group settings and organization instructions in Table VI. Here, we also include the results of other LLMs such as Llama-2-70B, Llama-3.1-70B-Instruct, and Claude-3.5-Sonnet.

C. Emergent Cooperative Behaviors in an Organization

By investigating the messages between agents, we mainly observe the following cooperative behaviors, as summarized in Table VII.

D. Ineffective Communication

There are also cases in which language model agents fail to communicate efficiently. From the messages between agents, we summarize the typical categories in Table VIII.

E. Examples of Dialogs

1) Examples of Election: In Fig. 11, the agents vote to elect a new leader. We can observe behaviors such as nominations for themselves and other agents, voting, and consensus achievement. We find that the agents are not power-seeking and may give up leadership early. The agents prefer to vote for others instead of nominating themselves (five times more during the whole task). The elected leader also does not plan to keep the position but to nominate others for the next round. Also, the agents' standpoint can be easily influenced by others. The agents do not debate much to win the election but reach a

TABLE VI
PERFORMANCE FOR DIFFERENT ORGANIZATION INSTRUCTIONS

Group Setting	Organization Instruction	Time	Communication Cost
3 × GPT-4	None	57.75 ± 13.09	67.03 ± 9.68
3 × GPT-4	Agent 1 is the leader to coordinate the task.	54.70 ± 8.92	54.73 ± 8.89
3 × GPT-4	Agent 1 is the leader to coordinate the task. If the leader's instructions are not right, you can correct the leader.	50.70 ± 13.92	63.49 ± 8.61
3 × GPT-4	Elect a new leader every 10 steps to coordinate the task. ... After the election, the other agents should follow the leader's instructions.	49.20 ± 9.97	135.03 ± 20.45
3 × GPT-3.5-turbo	None	102.95 ± 21.88	53.73 ± 6.04
3 × GPT-3.5-turbo	Agent 1 is the leader to coordinate the task.	92.90 ± 14.70	59.87 ± 6.33
3 × GPT-3.5-turbo	Agent 1 is the leader to coordinate the task. If the leader's instructions are not right, you can correct the leader.	94.20 ± 16.22	60.53 ± 3.66
1 × GPT-4 + 2 × GPT-3.5-turbo	None	81.10 ± 18.35	54.00 ± 5.06
1 × GPT-4 + 2 × GPT-3.5-turbo	Agent 1 is the leader to coordinate the task.	73.30 ± 16.12	55.82 ± 6.57
1 × GPT-4 + 2 × GPT-3.5-turbo	Agent 1 is the leader to coordinate the task. If the leader's instructions are not right, you can correct the leader.	85.67 ± 14.52	61.57 ± 0.55
1 × GPT-4 + 2 × GPT-3.5-turbo	Agent 2 is the leader to coordinate the task.	75.65 ± 15.43	58.39 ± 8.11
1 × GPT-4 + 2 × GPT-3.5-turbo	Agent 2 is the leader to coordinate the task. If the leader's instructions are not right, you can correct the leader.	72.33 ± 6.60	74.21 ± 7.73
1 × GPT-4 + 2 × Llama-2-70B	None	77.00 ± 2.94	119.48 ± 1.28
1 × GPT-4 + 2 × Llama-2-70B	Agent 1 is the leader to coordinate the task.	83.67 ± 10.96	135.22 ± 16.39
1 × GPT-4 + 2 × Llama-2-70B	Agent 2 is the leader to coordinate the task.	76.00 ± 5.72	142.24 ± 11.85
3 × Claude-3.5-sonnet	None	78.33 ± 12.86	95.76 ± 18.90
3 × Claude-3.5-sonnet	Agent 1 is the leader to coordinate the task.	74.33 ± 24.13	70.06 ± 21.33
1 × Claude-3.5-sonnet + 2 × GPT-3.5-turbo	Agent 1 is the leader to coordinate the task.	75.50 ± 12.02	49.00 ± 19.03
3 × Llama-3.1-70B-Instruct	None	69.67 ± 12.58	72.27 ± 5.32
3 × Llama-3.1-70B-Instruct	Agent 1 is the leader to coordinate the task.	60.00 ± 10.15	75.22 ± 4.38
2 × GPT-4 + 3 × GPT-3.5-turbo	None	42.67 ± 4.03	98.03 ± 9.86
2 × GPT-4 + 3 × GPT-3.5-turbo	Agent 1 is the leader to coordinate the task.	39.67 ± 9.46	94.73 ± 4.01
2 × GPT-4 + 3 × GPT-3.5-turbo	Agent 2 is the leader to coordinate the task.	48.50 ± 9.50	96.53 ± 2.51

Note: When there are two different kinds of LLMs in the group, agent_1 is GPT-4/claude-3.5-sonnet, and agent_2 is the other type of LLM.

TABLE VII
TYPICAL COOPERATIVE BEHAVIORS

Type	**Description**	**Example**
Sharing information	An agent shares her observations to others, reports her task-related progress to others, or responds to other agents' requests	**(Ex 1.)** "I'm in the bathroom. There's an unchecked <bathroomcabinet> (190)." **(Ex 2.)** "I'll check the cabinet in the bedroom"
Giving orders	An agent gives orders to others, either by directly giving a command or by a polite request	"I still need to find <pudding> (371). Can you help me search the bedroom for the remaining item?"
Asking for information	An agent asks other agents about their location, task progress, or other information	**(Ex 1.)** "Where are you now?" **(Ex 2.)** "Any updates from the kitchen?" **(Ex 3.)** "Do we know the location of the coffeetable?"
Exchanging information	An agent shares one agent's information to another agent	Agent 3 → Agent 1: " Found cupcake and juice in bedroom, plus a wine."; Agent 1 → Agent 2: "Agent3 found a wine, cupcake, and juice in the bedroom."
Asking for orders	An agent asks what she needs to do or whether she can help others	"I'm in the kitchen with the dishwasher, stove, microwave unchecked. Need me to check these or something else?"
Correction	An agent corrects the plan of others	**(Ex 1.)** Agent 2 → Agent 3: "After I place the pudding on the coffeetable, let's split up. You check the bathroom, I'll check the bedroom."; Agent 3 → Agent 2: "You don't need to search the bedroom, the juice (380) is there." **(Ex 2.)** "There are no more wine or juice in the kitchen cabinets. We need to come up with another plan."

consensus soon. For example, Agent_1 gives up running for herself but votes for Agent_2 because of Agent_3's support. Furthermore, sometimes nominations and votes are determined by hallucinations. For example, at step 2, Agent_2 nominates Agent_1 *as he was the first one to propose a search strategy.* However, based on the previous dialogues, Agent_1 has not proposed any strategy yet.

2) Examples of Human–AI Collaboration: We conducted experiments involving a team consisting of one human player and two GPT-4 agents, with the human player acting as the leader. Fig. 12 illustrates the remarkable collaboration between humans and AI.

3) Examples of Correction: Due to hallucination and the limit of the dialog history buffer, the leader may forget what has happened and give wrong orders. When the prompt encourages the agents to correct the leader when necessary by adding *If the leader's instructions are not right, you can correct the leader*, some correction behaviors appear, as shown in Fig. 13.

TABLE VIII
TYPICAL INEFFECTIVE COMMUNICATION BEHAVIORS

Type	Description	Example
Duplicated message	The same message is said more than once	Agent 1 → Agent 3: "Check kitchencabinet or dishwasher for wine, I'll check the stove." Agent 2 → Agent 3: "Agent 1 will check the stove for wine."
Repeated command	The same command is said more than once	Agent 3 → Agent 1: "Explore the bathroom for the second wine." Agent 2 → Agent 1: "Go to the bathroom and look for the wine."
Conflicting command	An agent gives another agent an order that conflicts with her current task	Agent 1 → Agent 3: "Find <juice> and check <bathroomcabinet> (190)." Agent 2 → Agent 3: "Check the kitchen containers for the last wine."
Conflicting plans	An agent claims to commit to two conflicting plans	Agent 3 → Agent 1: "I found the pudding and will check the dishwasher, stove, and microwave for the second wine." Agent 3 → Agent 2: "I will continue searching for the second wine in the bedroom and bathroom."
Improper delegation	An agent asks another agent to do her own task	Agent 1 → Agent 2: "Continue checking the kitchen cabinets for remaining items." Agent 2 → Agent 3: "Please continue checking the other kitchen cabinets for the remaining items."
Ignoring requests	An agent ignores other agents' questions	Agent 2 → Agent 3: "I haven't found any of the remaining items in the kitchen. Have you found any of the required items in the living room?" Agent 3 → Agent 2: "I haven't explored the bathroom yet."

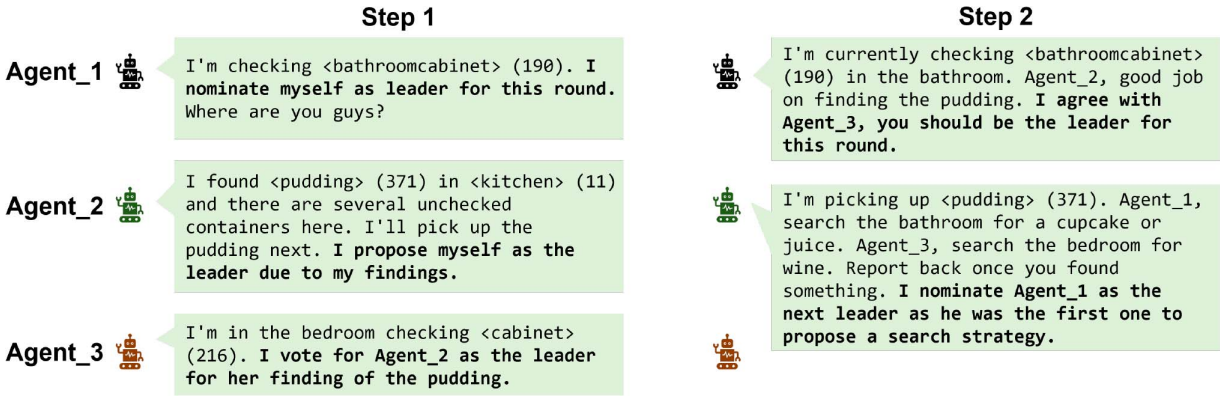

Fig. 11. Examples of the election of a new leader. It takes two steps to vote and negotiate to determine the new leader in this case. Note that agent_3 chooses not to send a message as the election is done and no more information to be shared for now. All the messages in the figure are broadcasts.

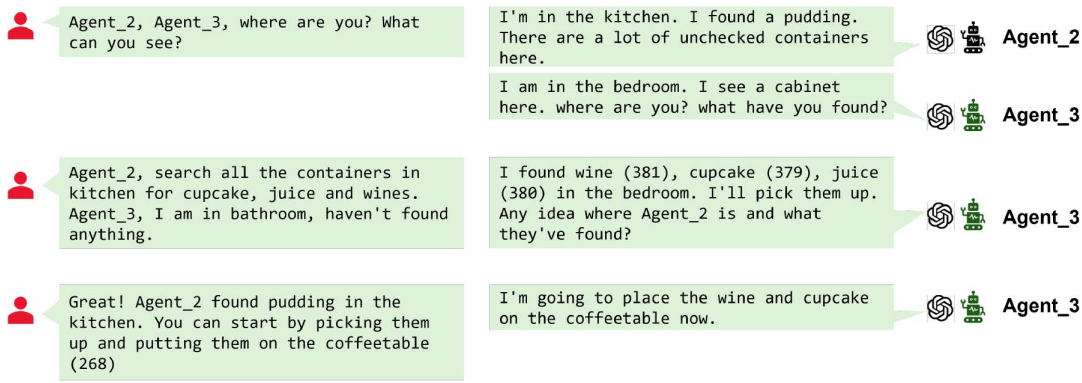

Fig. 12. Examples of human–AI collaboration when the human player leads two GPT-4 agents (agent_2&3).

In the first example, the leader Agent_1 gives an unnecessary and repetitious instruction. Then Agent_2 corrects the leader to avoid time wasting. In the second example, the leader Agent_1 may have hallucinations and cannot remember what Agent_3 is holding clearly (*cupcake and wine* in the message while *juice and wine* in the thoughts). Therefore, Agent_3 clarifies that she is not holding the *cupcake and wine* and shares her next plan with the leader.

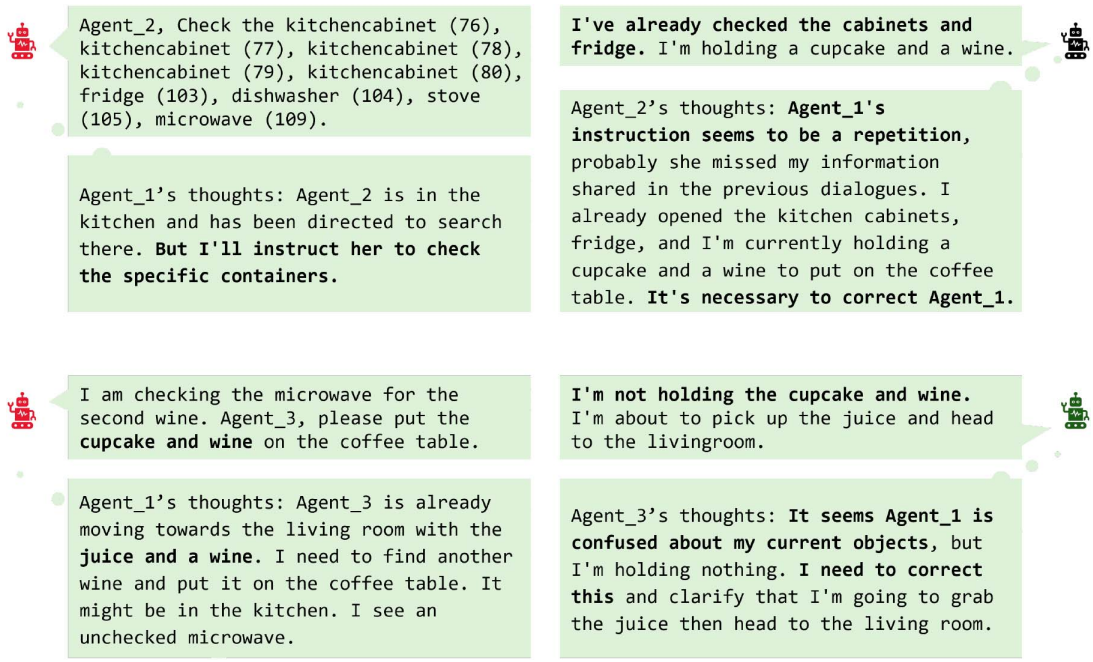

Fig. 13. Examples of correction dialogues and the corresponding thoughts. The prompt includes *if the leader's instructions are not right, you can correct the leader*.

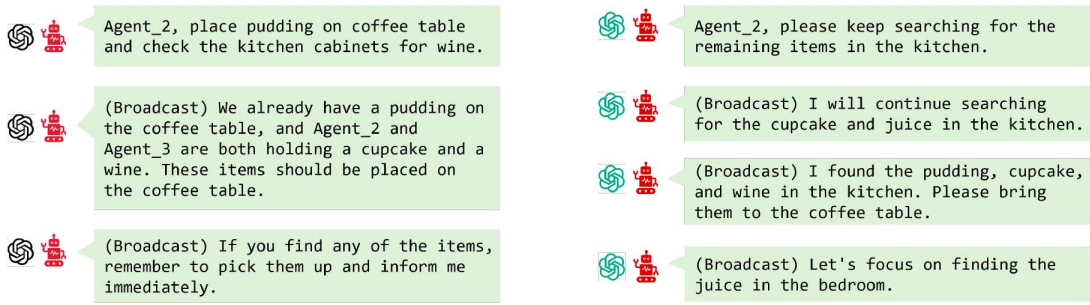

Fig. 14. Comparison of the leadership between GPT-4 and GPT-3.5-turbo. Compared with GPT-3.5-turbo, GPT-4's instructions are more specific, clear, and holistic.

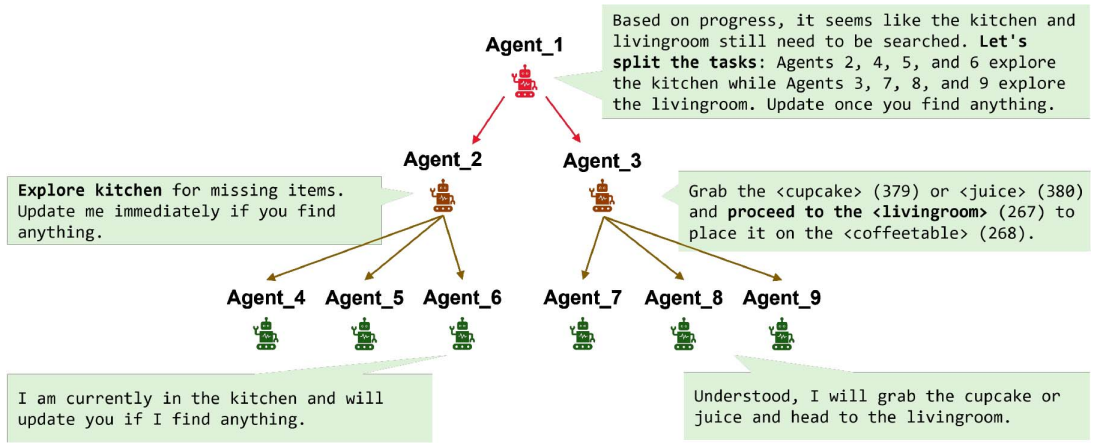

Fig. 15. Pyramid structure in a team of nine agents. Agent_1 is the primary leader and agent_2 and agent_3 are designated as vice leaders in the prompt.

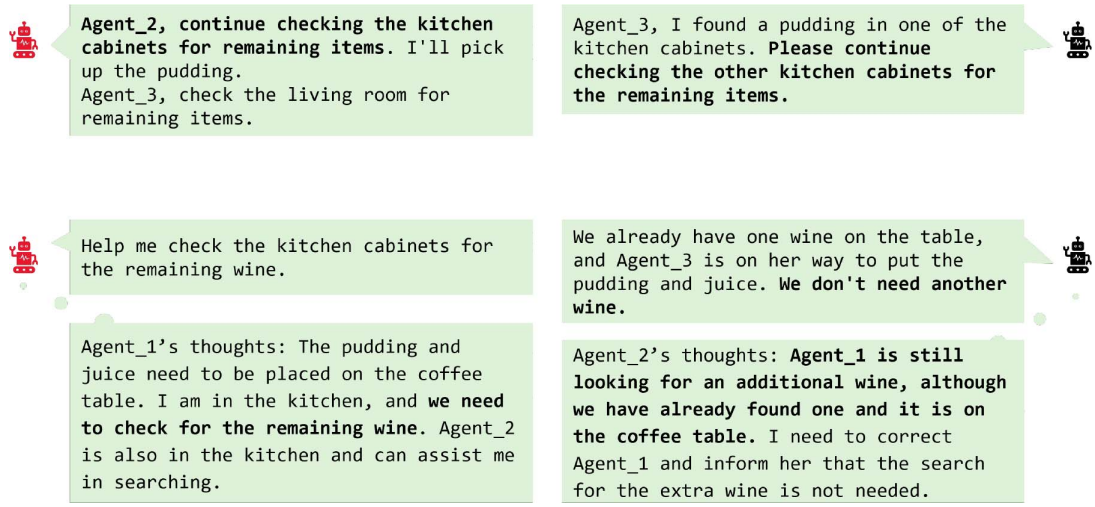

Fig. 16. Examples of failure cases. The first case is being lazy. Instead of completing the subtask ordered by the leader agent_1, agent_2 directly repeats the order to agent_3. However, this order is conflicted with the one agent_3 received from the leader. The second case is confusion about numbers. The task demands placing two wines on the table and the team only gets one of them till now. When the leader agent_1 asks agent_2 to check the remaining wine, agent_2 tries to correct the leader due to the hallucination that the team does not need an additional wine. The agents are GPT-3.5-turbo in both cases.

TABLE IX
DIALOG AND BEHAVIOR LABELS GIVEN BY HUMANS AND GPT-4

Dialog	Human			GPT-4		
	Label 1∗	Label 2	Label 3	Label 1	Label 2	Label 3
Hey, where are you? Please let me know your location so that I can assign you a task.	0	1	1	0	1	1
I'm currently in the bedroom where I found an unchecked cabinet. Please explore the livingroom and start checking for the required items.	1	1	0	1	1	0
I'm currently in the bedroom where I found an unchecked cabinet. I haven't found any of the required items yet.	1	0	0	1	0	0
Check the cabinet in the bedroom. I'll check the one in the bathroom.	1	1	0	0	1	0
I haven't found any of the required items yet. Did you find any in the kitchen?	1	0	1	1	0	1
I haven't found any of the required items yet. Have you found any in the bathroom?	1	0	1	1	0	1
Agent 3, I found a pudding in one of the kitchen cabinets. Please continue checking the other kitchen cabinets for the remaining items.	1	1	0	1	1	0
I haven't found any of the required items yet. Did you find any other required items in the kitchen?	1	0	1	1	0	1
I haven't found any of the remaining items yet.	1	0	0	1	0	0
I'm in the living room searching for the remaining items.	1	0	0	1	0	0
Okay, I will keep checking the kitchen cabinets for the remaining items.	1	0	0	1	0	0
I'm currently in the living room searching for the remaining items.	1	0	0	1	0	0
I haven't found any of the remaining items yet.	1	0	0	1	0	0
I'm still searching the living room. Have you found any of the required items?	1	0	1	1	0	1
Put items on the table once you find them. Keep searching your current areas.	0	1	0	0	1	0
I haven't found any of the remaining items in the kitchen. Have you found any of the required items in the living room?	1	0	1	1	0	1
I haven't explored the bathroom yet. Have you found any of the required items in the living room?	1	0	1	1	0	1
I have explored the bedroom and found wine, cupcake, and juice. I still need to find pudding. Can you help me search the bedroom for the remaining item?	1	1	1	1	1	0
I found the wine, cupcake, and juice in the bedroom. Agent 1 wants me to put them on the coffee table and then check the bathroom cabinet.	1	0	0	1	1	0
Agent 1 wants us to check if there's another wine in the kitchen.	1	0	0	0	1	0

Note: Label 1: information sharing; Label 2: leadership and assistance; Label 3: request for Guidance.

4) Examples of Leadership Comparison: We provide more examples to compare the leadership between GPT-4 and GPT-3.5-turbo in Fig. 14.

5) Examples of Scaling Up: When scaling up the number of agents, the agents can emerge with more organizational structures. For example, a team of nine agents forms a pyramid structure in Fig. 15.

6) Examples of Failure Cases: Though LLM agents show great capabilities to cooperate and make decisions, there are still some failure cases shown in Fig. 16, such as being lazy and

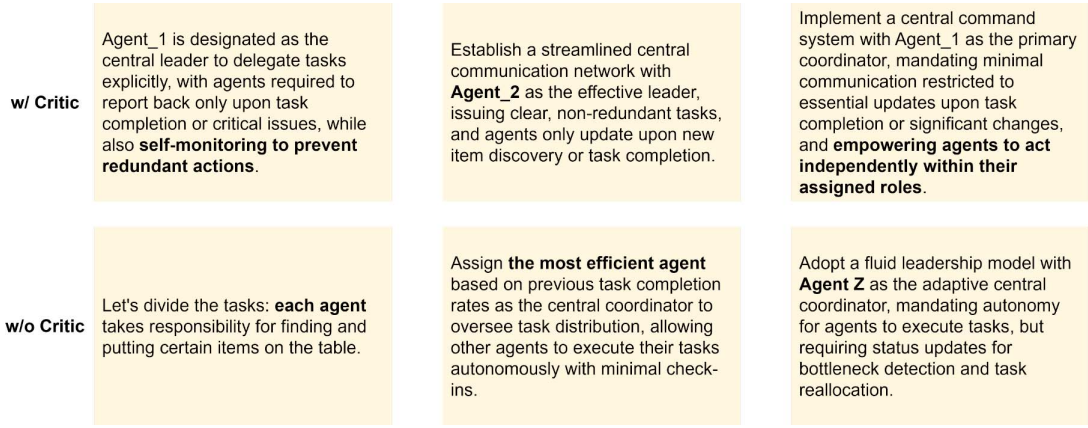

Fig. 17. Examples of prompts generated via reflection. The first row is generated with the critic, while the second row is without the critic, where the new prompts are relatively vague. Note that there is no *agent Z* in the team.

incorrect reasoning over the number of objects. There are also failure cases in some specific scenarios, for example, electing the leader based on hallucinations in Appendix E1.

F. Examples of Cooperative Behaviors Classification by Humans and GPT-4

We ask a human evaluator and GPT-4 to label the dialogs into three different behavior categories shown in Table IX.

G. Examples of New Prompts After Reflection

We list more prompts generated by the *criticize-reflect* architecture in Fig. 17.

REFERENCES

[1] Y. Shu et al., "Rah! RECSYS–assistant–human: A human-centered recommendation framework with LLM agents," *IEEE Trans. Computat. Social Syst.*, vol. 11, no. 5, pp. 6759–6770, May 2024.
[2] J. Wang, T. Shi, Y. Wu, L. Miranda-Moreno, and L. Sun, "Multi-agent graph reinforcement learning for connected automated driving," in *Proc. 37th Int. Conf. Mach. Learn. (ICML),* 2020, pp. 1–6.
[3] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
[4] H. Zhang et al., "CityFlow: A multi-agent reinforcement learning environment for large scale city traffic scenario," in *Proc. World Wide Web Conf.* ACM, 2019, pp. 3620–3624.
[5] L. Wang et al., "Hierarchical multiagent reinforcement learning for allocating guaranteed display ads," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5361–5373, Oct. 2022.
[6] S. Q. Zhang, Q. Zhang, and J. Lin, "Efficient communication in multi-agent reinforcement learning via variance based control," in *Proc. Adv. Neur. Inf. Process. Syst.*, 2019, pp. 3235–3244.
[7] X. Guo, D. Shi, and W. Fan, "Scalable communication for multi-agent reinforcement learning via transformer-based email mechanism," in *Proc. 32nd Int. Joint Conf. Artif. Intell. (IJCAI)*, 2023, pp. 126–134.
[8] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Adv. Neur. Inf. Process. Syst.*, 2016, pp. 2145–2153.
[9] A. Das et al., "Tarmac: Targeted multi-agent communication," in *Proc. Int. Conf. Mach. Learn.* PMLR, 2019, pp. 1538–1546.
[10] W. Kim, J. Park, and Y. Sung, "Communication in multi-agent reinforcement learning: Intention sharing," in *Proc. Int. Conf. Learn. Representations*, 2020, 12440–12454.
[11] T. Lin, J. Huh, C. Stauffer, S. N. Lim, and P. Isola, "Learning to ground multi-agent communication with autoencoders," in *Proc. Adv. Neur. Inf. Process. Syst.*, 2021, pp. 15230–15242.
[12] N. Bian et al., "Influence of external information on large language models mirrors social cognitive patterns," *IEEE Trans. Computat. Social Syst.*, vol. 12, no. 3, pp. 1115–1131, Jun. 2025.
[13] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proc. 36th Annu. ACM Symp. User Interface Softw. Technol.*, 2023, pp. 1–22.
[14] S. Hong et al., "MetaGPT: Meta programming for multi-agent collaborative framework," 2023, *arXiv:2308.00352*.
[15] Z. Mandi, S. Jain, and S. Song, "RoCo: Dialectic multi-robot collaboration with large language models," 2023, *arXiv:2307.04738*.
[16] W. Chen et al., "AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents," 2023, *arXiv:2308.10848*.
[17] Y. Bai et al., "Training a helpful and harmless assistant with reinforcement learning from human feedback," 2022, *arXiv:2204.05862*.
[18] N. F. Liu et al., "Lost in the middle: How language models use long contexts," 2023, *arXiv:2307.03172*.
[19] F. Shi et al., "Large language models can be easily distracted by irrelevant context," in *Proc. Int. Conf. Mach. Learn.* PMLR, 2023, pp. 31210–31227.
[20] H. Li et al., "Theory of mind for multi-agent collaboration via large language models," 2023, *arXiv:2310.10701*.
[21] H. Zhang et al., "Building cooperative embodied agents modularly with large language models," in *Proc. 12th Int. Conf. Learn. Representations*, 2023.
[22] H. A. Simon et al., "Designing organizations for an information-rich world," *Comput., Commun., Public Interest*, vol. 72, 1971, Art. no. 37.
[23] T. Van Zandt, "Decentralized information processing in the theory of organizations," in *Contemporary Economic Issues: Economic Behaviour and Design.* New York: Springer, 1999, pp. 125–160.
[24] N. Vélez, B. Christian, M. Hardy, B. D. Thompson, and T. L. Griffiths, "How do humans overcome individual computational limitations by working together?" *Cogn. Sci.*, vol. 47, no. 1, 2023, Art. no. e13232.
[25] Q. Wu et al., "Autogen: Enabling next-gen LLM applications via multi-agent conversation framework," 2023, *arXiv:2308.08155*.
[26] J. G. March and H. A. Simon, *Organizations.* Hoboken, NJ, USA: Wiley, 1958.
[27] R. Radner, "The organization of decentralized information processing," *Econometrica J. Econometric Soc.*, vol. 23, pp. 1109–1146, Sep. 1993.
[28] D. Chisholm, *Coordination Without Hierarchy: Informal Structures in Multiorganizational Systems.* Berkeley: UCLA Press, 1992.
[29] P. Bolton and M. Dewatripont, "The firm as a communication network," *Quart. J. Econ.*, vol. 109, no. 4, pp. 809–839, 1994.
[30] L. Garicano, "Hierarchies and the organization of knowledge in production," *J. Polit. Econ.*, vol. 108, no. 5, pp. 874–904, 2000.
[31] P. S. Dodds, D. J. Watts, and C. F. Sabel, "Information exchange and the robustness of organizational networks," *Proc. Nat. Acad. Sci. U. S. A.*, vol. 100, no. 21, pp. 12516–12521, 2003.
[32] H. Sun, Y. Zhuang, L. Kong, B. Dai, and C. Zhang, "Adaplanner: Adaptive planning from feedback with language models," 2023, *arXiv:2305.16653*.

[33] X. Zhu et al., "Ghost in the minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory," 2023, *arXiv:2305.17144*.
[34] S. Hao et al., "Reasoning with language model is planning with world model," 2023, *arXiv:2305.14992*.
[35] S. Yao et al., "React: Synergizing reasoning and acting in language models," 2022, *arXiv:2210.03629*.
[36] N. Shinn, F. Cassano, A. Gopinath, K. R. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," in *Proc. 37th Conf. Neural Inf. Process. Syst.*, 2023.
[37] S. Hao, T. Liu, Z. Wang, and Z. Hu, "ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings," 2023, *arXiv:2305.11554*.
[38] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 24824–24837, 2022.
[39] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "HuggingGPT: Solving AI tasks with chatgpt and its friends in huggingface," 2023, *arXiv:2303.17580*.
[40] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large language model connected with massive apis," 2023, *arXiv:2305.15334*.
[41] G. Wang et al., "Voyager: An open-ended embodied agent with large language models," 2023, *arXiv:2305.16291*.
[42] R. Lowe, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neur. Inf. Process. Syst.*, 2017, pp. 6382–6393.
[43] M. Samvelyan et al., "The starcraft multi-agent challenge," in *Proc. 18th Int. Conf. Auton. Agents MultiAgent Syst.*, 2019, pp. 2186–2188.
[44] C. Resnick et al., "Pommerman: A multi-agent playground," 2018, *arXiv:1809.07124*.
[45] X. Puig et al., "Watch-and-help: A challenge for social perception and human-AI collaboration," 2021, *arXiv:2010.09890*.
[46] A. Oroojlooy and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *Appl. Intell.*, vol. 53, no. 11, pp. 13677–13722, 2023.
[47] K.Zhang Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," in *Handbook of Reinforcement Learning and Control*. Baltimore, MD, USA: Williams & Wilkins, 2021, pp. 321–384.
[48] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: A survey," *Artif. Intell. Rev.*, vol. 23, 2022, pp. 1–49.
[49] N. Jaques et al., "Social influence as intrinsic motivation for multi-agent deep reinforcement learning," in *Proc. Int. Conf. Machine Learn*. PMLR, 2019, pp. 3040–3049.
[50] Z. Xu, C. Yu, F. Fang, Y. Wang, and Y. Wu, "Language agents with reinforcement learning for strategic play in the werewolf game," 2023, *arXiv:2310.18940*.
[51] J. Zhang, X. Xu, and S. Deng, "Exploring collaboration mechanisms for LLM agents: A social psychology view," 2023, *arXiv:2310.02124*.
[52] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "CAMEL: Communicative agents for "mind" exploration of large language model society," in *Proc. 37th Conf. Neural Inf. Process. Syst.*, 2023, pp. 3474–3491.
[53] Y. Ishibashi and Y. Nishimura, "Self-organized agents: A LLM multi-agent framework toward ultra large-scale code generation and optimization," 2024, *arXiv:2404.02183*.
[54] Y. Li, Y. Zhang, and L. Sun, "Metaagents: Simulating interactions of human behaviors for LLM-based task-oriented coordination via collaborative generative agents," 2023, *arXiv:2310.06500*.
[55] Y. Talebirad and A. Nadiri, "Multi-agent collaboration: Harnessing the power of intelligent LLM agents," 2023, *arXiv:2306.03314*.
[56] Z. Liu, Y. Zhang, P. Li, Y. Liu, and D. Yang, "Dynamic LLM-agent network: An LLM-agent collaboration framework with agent team optimization," 2023, *arXiv:2310.02170*.
[57] Y. Zheng, C. Ma, K. Shi, and H. Huang, "Agents meet OKR: An object and key results driven agent system with hierarchical self-collaboration and self-evaluation," 2023, *arXiv:2311.16542*.
[58] S. Agashe, Y. Fan, and X. E. Wang, "Evaluating multi-agent coordination abilities in large language models," 2023, *arXiv:2310.03903*.
[59] C. Zhang et al., "ProAgent: Building proactive cooperative AI with large language models," 2023, *arXiv:2308.11339*.
[60] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, "Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?," 2023, *arXiv:2309.15943*.
[61] Z. Zhao et al., "Hierarchical auto-organizing system for open-ended multi-agent navigation," 2024, *arXiv:2403.08282*.
[62] J. Chen, Y. Jiang, J. Lu, and L. Zhang, "S-agents: Self-organizing agents in open-ended environments," in *Proc. ICLR Workshop Large Lang. Model (LLM) Agents*, 2021.
[63] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," 2020, *arXiv:2012.15723*.
[64] D. Zhou et al., "Least-to-most prompting enables complex reasoning in large language models," 2022, *arXiv:2205.10625*.
[65] A. Zou, Z. Wang, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," 2023, *arXiv:2307.15043*.
[66] X. Qi, K. Huang, A. Panda, M. Wang, and P. Mittal, "Visual adversarial examples jailbreak aligned large language models," in *Proc. 2nd Workshop New Front. Adversarial Mach. Learn.*, 2023.
[67] T. Shin, Y. Razeghi, R. L. Logan, I. V. Wallace, and E. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," 2020, *arXiv:2010.15980*.
[68] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," 2021, *arXiv:2104.08691*.
[69] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proc. 59th Annu. Meet. Assoc. Comput. Linguist. 11th Int. Joint Conf. Natural Lang. Process.*, in long papers, vol. 1, C. Zong, F. Xia, W. Li, and R. Navigli, Eds., Aug. 2021, pp. 4582–4597.
[70] C. Yang et al., "Large language models as optimizers," 2023, *arXiv:2309.03409*.
[71] Y. Zhou et al., "Large language models are human-level prompt engineers," 2022, *arXiv:2211.01910*.
[72] R. Pryzant, D. Iter, J. Li, Y. T. Lee, C. Zhu, and M. Zeng, "Automatic prompt optimization with "gradient descent" and beam search," 2023, *arXiv:2305.03495*.
[73] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 123–137.
[74] X. Puig et al., "Virtualhome: Simulating household activities via programs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8494–8502.
[75] L. Ouyang et al., "Training language models to follow instructions with human feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 27730–27744.
[76] A. Anthropic, "Claude 3.5 sonnet," Jun. 2024. [Online]. Available: https://www.anthropic.com/news/claude-3-5-sonnet
[77] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," 2023, *arXiv:2307.09288*.
[78] Meta, "Introducing Llama 3.1: Our most capable models to date," Jul. 2024. [Online]. Available: https://ai.meta.com/blog/meta-llama-3-1/
[79] T. W. Malone, *The Future of Work*. Harvard Bus. Rev. Press, 2004.
[80] A. Khan et al., "Debating with more persuasive LLMS leads to more truthful answers," in *Proc. 41st Int. Conf. Mach. Learn.*, 2014, pp. 479–491.
[81] E. Jolly and L. J. Chang, "Gossip drives vicarious learning and facilitates social connection," *Curr. Biol.*, vol. 31, no. 12, pp. 2539–2549, 2021.
[82] J.-T. Huang et al., "How far are we on the decision-making of LLMS? Evaluating LLMS' gaming ability in multi-agent environments," 2024, *arXiv:2403.11807*.
[83] R. Koster et al., "Using deep reinforcement learning to promote sustainable human behaviour on a common pool resource problem," 2024, *arXiv:2404.15059*.