Role Playing Leader Hallucination: A Lightweight Agent-Based Multi-Collaboration System

Kaiwen Bian*
Halıcıoğlu Data Science Institute
UC San Diego
La Jolla, CA 92093
kbian@ucsd.edu

Ziyu Huang
Halıcıoğlu Data Science Institute
UC San Diego
La Jolla, CA 92093
zih029@ucsd.edu

Weijie Zhang
Halıcıoğlu Data Science Institute
UC San Diego
La Jolla, CA 92093
wez042@ucsd.edu

1 Introduction

Multi-agent learning and collaboration between artificial agents have always been a challenging planning problem for the machine learning community, especially when this problem is placed in an adversarial environment. Due to the impressive planning and reasoning abilities of Large Language Models (LLMs) recently, many novel inference schematics have been developed to solve long-horizon planning problems via LLMs, including multi-agent planning during inference using cognitively modularized structures [13] and scalable multi-agent planning [1]. Similarly, others utilized this reasoning ability to move towards general level-1 artificial intelligence [2] by building agent models. In this work, we use the strong in-context learning ability of Language Models (LMs) to conduct agent-based reasoning and multi-agent collaboration in an adversarial environment. More importantly, our system is designed to be customizable and lightweight, meaning that it can be run with less parametrized LMs (GPT-4o-mini, Llama-14B, Qwen-14B) to avoid high API and computation costs [15]. We introduce our work of **R**ole **P**laying **L**eader **H**allucination [2] (RPLH) that addresses multi-agent collaboration problem in an adversarial environment through the L-policy [14] perspective of using LMs' "hallucination" ability [2] to infer other agents' intention from their behaviors, hence, coming to a desirable action plan by incorporating helpful agent's plans and avoid vicious plans from adversarial agents. We have created an system that treats the inference loop as somewhat a sequential learning process for the LM agent to reason about each agent.

2 Backgrounds and Related Works

Large Language Models (LLMs) have recently brought attention for dealing with complex collaborative tasks [3, 6, 5] in both single-agent and multi-agent planning. The cognitively modular approach, as demonstrated in [13], modularized LLM to handle complex tasks in decentralized environments, allowing agents to specialize, communicate, and make decisions according to their role in the system. LLMs have also been used for single-agent planning in virtual settings like VirtualHome as described in [10], highlighting how they enable real-world task simulations with procedural knowledge representation. While these models are exceptionally good at reasoning and decision-making skills, they come at huge computational costs, which makes their scalability in multi-agent settings limited

*Documentation at kbian.org/RPLH
[2]Code of the system is available at https://github.com/KevinBian107/RPLH

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

when cost efficiency is a constraint. To address the limitations of LLMs, researchers have explored **less parametrized** LMs [11, 9, 12] as a viable alternative for multi-agent systems. These models offer a promising solution for tasks requiring frequent interactions and dynamic planning. Although these LMs lack the power of LLMs, they can approximate LLM-like reasoning capabilities when combined with tailored frameworks such as ours, providing a scalable and computationally efficient option for multi-agent coordination. On the other frontier, **hallucination**, often viewed as an artifact in language generation, has recently been adopted as a strategic mechanism in multi-agent systems. It can be leveraged to simulate other agents' thought processes [2], which allows models to infer likely behaviors of other agents based on prior interactions and agent behavior. It provides a mechanism to handle uncertainty and ambiguity in agent communication, reducing reliance on constant recalibration and direct inter-agent communication. It is particularly valuable in resource-constrained environments where maintaining a constant exchange of information is impractical. Such approach can be framed as **Agent-based modeling** (ABM), which focuses on understanding the behaviors and interactions of individual agents within a system. Practically, it has been used in **adversarial environment** where LM agents would try to reason viscous intentions of other LM agents in unstructured environments like card games [7]. These types of formulation fully uses LMs' hallucination ability with the drawback that they are hard to scale to actual structured environment. RPLH draws inspiration from ABM principles by treating each agent as an independent decision-maker constrained by the collective dynamics of the group. It builds on this idea by using LMs to dynamically generate predictions about other agents' intentions, fostering collaboration through inferred reasoning rather than exhaustive communication. Through such novel use of hallucination, RPLH enables agents to act as autonomous planners, collaborators, and viscous spy. By combining the computational efficiency of LMs, the use of hallucination, and agent-based modeling, it provides a cost-effective framework.

3 Methods: Problem Formulation

We will define a few mathematical notations in this section to better introduce the information flow in our RPLH system and hopefully, this notation may be adapted as a formal definition for describing multi-agent collaboration systems. We will first go over the formulation of our problem (AC-MDP). Then we will go over the data formulation and data flow in our system. At last, we will introduce some specific feature related to each agent.

3.1 Adversarial-Collaborative-MDP

We define the problem that we want to solve as an Adversarial Collaborative MDP (AC-MDP) problem, which can be defined as:

$$\langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}, T, H, \mathcal{I}, \mathcal{J}, \mathcal{R} \rangle$$

Where $\mathcal{N}$ is the set of n agents: $A_1, A_2, \ldots, A_n$ and one of them would be an adversarial agent with sabotage intention. $\mathcal{S}$ is the global state space, which is shared across all agents. $\{\mathcal{A}_i\}$ is the set of actions available to each agent $\mathcal{A}_i$, including both environment actions where $\mathcal{A}_i^E$ and communication actions where $\mathcal{A}_i^C$ (including dialogues, reasoning, and hallucination of future states). Additionally, $T : s_t, a_t \to s_{t+1},\ T(s_t, a_t) = s_{t+1}$ is the transition function, defining the probability of transitioning from state s_t to s_{t+1}. For our base simple environment of Ad-BoxNet in section 5, such transition probability T is deterministic. H is the planning horizon, specifying how far into the future the agents plan. $\mathcal{I}$ is the information constant, representing the agents' episodic memory, which includes previous memory information fo state-action history pairs $\bigcup_{i=1}^{n} \{s_{0\to t}\}_{A_i} \cup \{a_{0\to t}\}_{A_i}$ (depending on the setting, this may be Markovian, no-memory, or all-interactions). For AC-MDP, the reward $\mathcal{R}$ is implicit in the completeness of matching boxes.

3.2 Sensory Information and Information Constant

Sensory information and information constant are essentially what everyone can observe in the environment, which is crucially to keep apart from private information flows such as attitude (subsection 3.4) and agent models (subsection 3.3). In our system, the information constant I_c is a subset of the larger super set of sensory information I_s, which is an analogy to semantic and episodic memory

in cognitive science and human communication. We define I_c as:

$$I_c = \{s_{0\to t}\}_{A_1} \cup \{a_{0\to t}\}_{A_1} \cup \{s_{0\to t}\}_{A_2} \cup \{a_{0\to t}\}_{A_2} \cup \cdots = \bigcup_{i=1}^{n} \{s_{0\to t}\}_{A_i} \cup \{a_{0\to t}\}_{A_i}$$

Where A_i is the i^{th} agent, the first term $\{s_{0\to t}\}_{A_i}$ is the set of states for all iterations, and the second term $\{a_{0\to t}\}_{A_i}$ is the set of actions for all iterations. We can then extend from information constant I_c and define sensory information $I_s^{(t)}$ at each environmental step as the set of tokens including: (1) the goals for each agents G_i, (2) information constants $I_c^{(t)}$ at each environmental step, and, at last, (3) the dialogues from each agent $D_i^{(t)}$, which includes this local agent's action plan: $\{a_{(t+1)}\}_{A_i}\ \forall\ i \in \{1, n-1\}$ if the dialogue comes from a local agent and includes the hallucination $h_i^{(t)}$ and reasoning $r_i^{(t)}$ if it coems from an HCA. This sensory information I_s is the super set of what we will be using to pass around in practice. Mathematically, we define all dependency of the sensory information $I_s^{(t)}$ at iteration t as:

$$I_s^{(t)} = \{G_i, I_c^{(t)}, D_i^{(t)}\}_{i=1}^{n}$$

In this paper, we will refer to any notation that can be shared among multiple agents with the subscript i, and to any notation that can exist between multiple time stamps with the superscript t.

3.3 HCA Input / Output: Action Belief and Agent Modeling

In our system, we explicitly create three different models for the HCA to conduct agent-based reasoning and conduct spy detections from each local agent's behaviors. We define LMs as $\mathbb{P}_{LM}(prompt) = tokens$ and we define the prior belief of action $a^{(0)} \in \{\mathcal{A}_i\}$ for agent i (set of actions that the HCA i believes what other agents should do). Further, we define the information that each HCA planner or HCA judge can receive in iteration t as $I^t_{\text{HCA}} = \{M_a^{(t)}, M_s^{(t)}, M_j^{(t)}, I_s^{(t)}\}$ where $M_a^{(0)} = M_s^{(0)} = M_j^{(0)} = \emptyset$ at $t = 0$. For each model, we define $M_a^{(t)}$ as reasoning of the attitude or characteristic of each agent at t, $M_s^{(t)}$ as the speculation of the agent that has characteristic with spy at t, and $M_j^{(t)}$ as the justification its action plan at t. Thus, we have the HCA as:

$$\mathbb{P}_{\text{HCALM}}(I^t_{\text{HCA}}, s_t) = \{a_t^{(0)}, M_a^{(t+1)}, M_s^{(t+1)}, M_j^{(t+1)}\}$$

All $M_a^{(t)}$, $M_s^{(t)}$, and $M_j^{(t)}$ is a property of HCA agent just like C_i is a property of local agent, which is why we are separating both of them out of the sensory information I_s that everyone can access.

3.4 Local Input / Output: Attitudes and Action Plans

We define the "characteristics" or "attitude" of the i^{th} agent as C_i and it is defined as the abstract "personality" that each agent has picked up from the dialogue. The characteristics will be assigned to each local agent before we start the iteration. Example of it is given in A.3.

$$C = \{\text{"spy", "nice", "critical", "agreeing"}\}$$

Notice that attitude is a property of each local agent, so at the end, we define the information that each local agent can receive in iteration as t as $I^t_{\text{Local}\{i\}} = \{C_i, I_s^{(t)}\}$. Thus we have local agents as:

$$\mathbb{P}_{\text{LocalLM}\{i\}}(I^t_{\text{Local}\{i\}}, s_t) = \{a_t^{(i)}\}$$

3.5 Prior and Posterior Action Belief

In each iteration, there exists a prior belief of the optimal action and an agent playing a judge role that "convert" prior belief to posterior belief. In section 3.4, we defined prior belief as $a_t^{(0)}$

$$a_t^{(0)} = [a_{A_{i1}}, a_{A_{i2}}, \dots] = \{a_{A_{ij}}\}_{j=1}^{n}$$

where $a_{A_{ij}}$ is the i^{th} agent's recommended instant action for the j^{th} agent, reflecting what we think about others. We are defining a general notion here. However, in our practical implementation, each i^{th} agent being HCA is conducted independently.

During action decision-making, we start with $a_t^{(0)}$, which is generated by the HCA A_i. Let $a_t^* \in \{\mathcal{A}_i\}$ as the true optimal action at time t. Let $a_t^{(k)} \in \{\mathcal{A}_i\}$ be the action that is modified by local agent and HCA at k^{th} agent. During the action decision-making process, we are trying to minimize the difference between a_t^* and $a_t^{(k)}$. With n different agents, we have $k \in \{1, \dots, n\} - \{i\}$, since A_i is the HCA. We define $f : \{a^{(k-1)}, C_{k-1}, s_t\} \to a_t^{(k)}$.

$$f_k(a^{(k-1)}, C_{k-1}, s_t) = proj_{\{\mathcal{A}_i\}}(\mathbb{P}_{LM}(a^{(k-1)}, C_{k-1}, s_t, I_s^t))$$

At k = n, we have

$$a_t^{(n)} = (f_{n-1} \circ f_{n-2} \circ \cdots \circ f_1)(a_t^{(0)}, C_i, s_t)$$

We are using $a_t^{(n)}$ to estimate a_t^* and come to the optimal action to be executed and steps by the environmental engine. Our method of searching for the optimal action is analogous to a neural network. Each modification to the action can be thought of as adding another layer to this network.

4 Methods: System and Framework

The RPLH system is designed to be lightweight, customizable, and scalable for conducting agent-based reasoning through the lens of adversarial collaboration, hopefully moving a step closer to level-one agent [2]. We believe that the key of collaboration in a structured environment, lays in the communication process and the data we can extract from such communication for an agent to gradually build up its agent model. Hence, a major component of the RPLH system is about dealing with information flow. The name of hallucination in our title comes from the fact that the same LMs need to play many different roles in the reasoning process to better collaborate.

4.1 Vanilla and Efficient Instances

We have created two different instances of our system, which we named them as the RPLH-vanilla system and RPLH-efficient system. The first is the vanilla version of our construct where no limits have been posted on the structure of the LMs' output (unstructured string token output), which benefits the agent modeling process since more chain-of-thought reasoning can happen. However, a significant drawback exist in practice where it's extremely hard to converge in a structured system as less parametrized models constantly make syntactic errors in their responses, causing retaking actions as it can not be stopped by our environment in subsection 5.1. To address to this, we created the RPLH-efficient system where we use a finite state machine package [4] to tune to output probability of the LMs to be structured into a dictionary format, which could consider as a projection of action that is generated by LM to doable actions. The efficient system significantly reduces the number of syntactic checks and increases the rate of convergence with sacrifices on the length of the output that the LMs can give. By establishing these models, we are creating an analogy for "weights" in traditional machine learning using "tokens".

4.2 Agent and Standard Reasoning

To encourage the language model to reason about other agents' actions, we introduced a novel characteristic of spy where number of spy must be greater or equal than 2 as spy agents would sometimes be the HCA and they would have no memory of themselves being the spy when being HCA. Agents with the spy characteristic attempt to disrupt HCA's decision-making process, ultimately leading to a failure in convergence. Hence, this setup would force HCA to think about other agents as it needs to deduct out the suspicious agent. To demonstrate the effects of conducting agent-based reasoning, we have set up two different inference system. The first (RPLH-Agent) explicitly

constructs a model of each other agent in the environment through agent, spy, and justification model in subsection 3.3, and incrementally updates them through each environmental step's interaction with the local agent. In this system HCA would play two different roles: the first is to act as central planner and does hallucination of future steps with agent reactions to better plan for the actions and the second is to act as a judge to critically evaluate each local agent's behavior (action) and reason about such agent's intention for updating the three models. On the other hand, the RPLH-Standard system only hallucination of future steps, reasons other agent's reactions, and acts as a central planning agent to give action plan. A separate judge would be involved when there exist a conflict between the local agent and the central agent to decide on the optimal action plan. Importantly, the judge does not propose any new action plans but only picks the ones that's being proposed. If a synthetic error occurs and retake action is required, it will fall back onto the central planner's action plan directly, which has been checked earlier in the iteration for executability.

4.3 Information Flow

This section will provide a higher-level overview of how data flows in our system. This is an abstract framework that can be adapted to any structured environment (i.e. VirtualHome environment). The illustration of our framework is in figure 1.

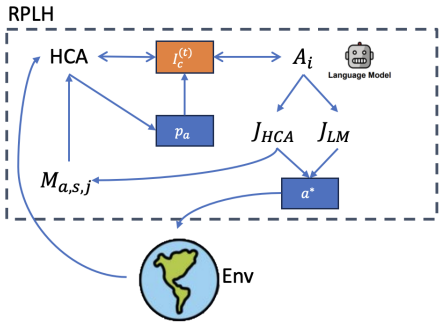

Figure 1: RPLH framework

The main idea is that each agent in our environment will take turns to play as a central agent, in which we refer to as HCA (hallucination central agent) in our system and conduct central planning reasoning and hallucination to give an action plan based on the state-action memory setting given (i.e. Markovian, no-memory, all-previous-interactions). Then such information will flow to each local agent who receives a specific attitude that is assigned by the user and they would react to the HCA message with their characteristics baked into it (i.e. spy agent would try to sabotage and nice agent would try to play alone). Then each local agent's response would be given to the judge agent (either a separate judge agent or HCA agent playing as a judge) to conduct corresponding reasoning and update (i.e. agent model and spy model) depending on the system used. At last, after all local agents have responded, the environment is stepped with the current optimal action plan. For more details, we have a concise algorithm representation of our system summarized in A.1 and additional details for each step summarized in A.2.

5 Experimentation

5.1 Scalable, Structured, and Adversarial Ad-BoxNet Environment

We utilized an adversarial warehouse-like environment in which we call Ad-BoxNet, inspired by [1]. Ad-BoxNet involves multiple LMs agents operating in a grid-like structure environment. We tested the following three methods in the Ad-BoxNet, the standard method with spy (RPLH-Standard-Spy), the agent method with spy (RPLH-Agent-Spy), and the standard method with no spy (RPLH-Standard-Nospy). We also compared these with a traditional decentralized communication framework. Centralized framework were excluded in our experiment as adversarial spy agents would have no effect in such setups. Experiments were conducted for 20 trials with constant randomized initial configurations to evaluate coordination and scalability. In our setup, a task is considered a

failure if (1) does not reach a consensus before a pre-specified number of in the dialogue round, (2) exceeds a pre-specified planning iterations limits before reaching the goal, and (3) surpassing a pre-specified tolerance limit for syntactic error round. Data recorded for each trial included state-action history, agent dialogues, and model-specific output.

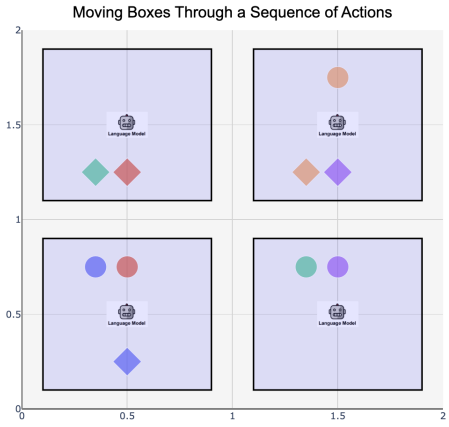

Figure 2: Structured 2x2 Ad-BoxNet

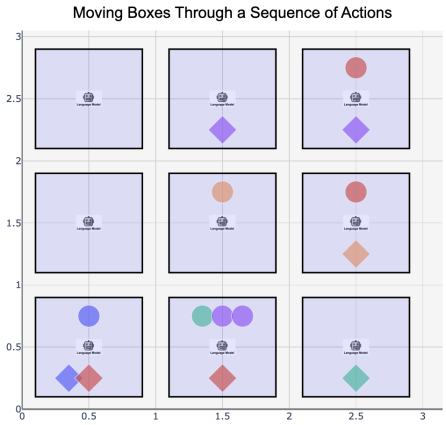

Figure 3: Structured 3x3 Ad-BoxNet

The Ad-BoxNet in figure 2 and figure 3 is a grid-like structure environment with robot arms, colored boxes, and the corresponding goal location in distinct cell regions. Each robotic arm, controlled by LMs system, operates independently and can perform one of three actions: moving a box to a neighboring cell, placing it at a goal location within its cell, or remaining idle. The objective is to collaboratively move boxes to their matching targets in the fewest steps. The environment includes adversarial "spy" agents aiming to distract others, framing it as both collaborative and adversarial. There are 5 different colored boxes and multiple boxes and targets of the same color can exist. Actions are collision-free if valid. A trial is successful if all boxes are matched to their targets within a specified number of steps.

Our environment combines both collaboration and adversarial elements, requiring agents to work together to complete tasks optimally while handling "spy" agents designed to distract the plan. Our environment is structured and scalable. The LMs agents operate in our environment under strict, structured action inputs while communicating and reasoning in natural language, thus, requiring LMs models to strictly follow instruction prompt. This setup supports scalability to similar structured environments with unstructured communications, serving as a base case for more complex virtual scenarios. The environment is fully observable in states but partially actuatable in actions. Agents have complete knowledge of box and target locations but rely on collaboration or adversarial interactions to achieve goals due to their limited control over the environment.

5.2 Evaluation Metrics

To evaluate the effectiveness of our multi-agent communication system for task planning and social reasoning. We conduct 20 trials for each method. Experiment configurations were consistent across comparisons and randomized between trials. All experiments used the GPT-4o-mini model, initialized with no state-action history, with max 2 or 3 boxes of the same color.

5.2.1 Standard Metrics

We report key metrics: task success rate, number of steps for success, and number of boxes matched to target per step. Success was defined as all boxes being matched to targets, with success rate and steps averaged across trials. Steps resulting from syntactic errors were excluded.

5.2.2 Energy Metrics

We introduced energy metrics, a distance-based evaluation metric measuring the proximity of boxes to targets at each step. Distances are measured in two ways:

- Norm-1: The smallest number of grid steps required to move a box to a target.
- Norm-2: The shortest straight-line distance from each box to the target.

These distances were tracked per trial and step, showing a decreasing trend as unmatched boxes decreased. The distance curves provide information about task performance and convergence rate. The slope of the distance curve represents the convergence speed of tasks. The area under the curve (AUC) measures task performance efficiency. We defined AUC as *Energy*. Our evaluation focuses on an interesting point where low-energy trials converge faster and prioritize early box matching. Energy metric penalizes no or fewer boxes matched in the beginning. Example are provided in A.7

5.2.3 Embedding Similarity and Feature Importance

For the RPLH-Agent system, we evaluated social reasoning by comparing the agent's reasoning with the one-line spy/justification example that we have provided. Using sentence embedding [8], we calculated the cosine similarity between LM-generated outputs and human-annotated heuristic sentences. This is a vanilla way of assessing how well the agent can reason about the spy's attitude and propose an appropriate justification. Examples are provided in A.4. Furthermore, we transformed the standard, energy, and embedding metrics into features for a Random Forest Regressor to identify the most important predictors of trial performance using feature importance score (steps to converge and targets matched per step).

6 Experiment Results

6.1 Main Findings

Metric	rplh-agent-spy	rplh-standard-spy	rplh-standard-nospy
Box-To-Target*	**0.406**	0.271	0.277
Steps*	**28.410**	35.773	36.958
Success*	**69.9%**	55.0%	64.9%
AUC-Norm-1*	**187.369**	228.863	241.998
AUC-Norm-2*	**108.398**	133.285	139.057
Slope-Norm-1	**-0.388**	-0.333	-0.279
Slope-Norm-2	**-0.270**	-0.240	-0.191
Similarity-Spy-1	0.563	—	—
Similarity-Spy-2	0.647	—	—
Similarity-Spy-3	0.622	—	—
Similarity-Justification	**0.627**	0.553	0.554

Table 1: Evaluation results for different methods. We report standard metrics, energy metrics, and embedding similarity metrics over 20 runs with a max of 2 boxes of the same color in a 3x3 grid environment. Asterisk* denotes that the value is bootstrapped 10,000 times for a robust comparison.

We have illustrated our RPLH-Agent-Spy outperforms any of the other two systems (RPLH-Standard) no matter when box upper bound is 2 or 3 (in table 6.1, A.9). In all cases, fewer total boxes in the environment make the environment easier to be completed (in terms of convergence rate, convergence speed, and number of responses needed). On average, RPLH-Agent would: (1) take the least energy to complete the task, (2) have the fastest speed of convergence, have the highest rate of convergence, (3) have the highest average box-to-targets per response ratio, and (4) have the least number of responses needed for convergence. Furthermore, traditional methods like decentralized communication takes extremely long time to converege under this setting since it is really hard for agreement to be reached.

Interestingly, the box to targets per response metric did not change for both RPLH-Standard systems neglecting the presence of spy, pointing to the potential conclusion that their underlying mechanism may be the same, independent of number of boxes. But with RPLH-Agent, more boxes would result in a more experienced understanding of the spy agent, improving the justification embedding similarity, hence, resulting in higher target match rate.

Furthermore, it takes longer for our system to solve the task when no-spy is around, which aligns with our hypothesis that the presence of spy agent "forces" the HCA to conduct better agent modeling, hence, better convergence in solving the task. Aligning with our idea again, though convergence is more efficient, the actual convergence rate is actually lower, signifying that spy does have effect on disrupting the central agent as well (when HCA doesn't explicitly model spy in RPLH-standard).

At last, we looked at the feature similarity A.5 and sentence embedding similarity A.8 in which we find that our RPLH-Agent system detects some spy agents and these features seem to be the most correlative to predicting number of responses or average targets matched per response when putting into a Random Forest Regressor.

6.2 Hypothesis Testing

To statistically demonstrate the effects of the "SPY" characteristic and LM reasoning, multiple hypothesis tests were conducted to assess their statistical significance. We want to investigate whether the presence of the SPY characteristic influences system performance. The hypothesis test examines matrices representing the number of responses (or iterations—fewer indicating a faster convergence rate) and the presence or absence of the SPY characteristic. H_0 : The spys has no effect on the performance. The mean number of iterations of system without spy - The mean number of iterations of system with spy $= 0$. H_1 :The mean number of iterations of system without spy - The mean number of iterations of system with spy $\neq 0$. A significance level of $\alpha = 0.1$ was chosen, yielding $\alpha/2 = 0.05$. Since the observed p-value (0.049) is less than 0.05, the alternative hypothesis is rejected in favor of the null. We have included our result in A.6.

7 Conclusion

To conclude our contributions again, in this work, we proposed a novel adversarial learning environment under the context of multi-agent collaboration task. We proposed the mathematical formulation of discussing about multi-agent collaboration and introduced a unique energy metrics for evaluating such system. To our best knowledge, we are also the ones to propose solving multi-agent planning in an adversarial environment by leveraging the "hallucination" capabilities and agent-based reasoning ability of LMs. We extended the traditional strategical/logical reasoning and communication approach ("pair-review") used in traditional multi-agent setups and introduced agent-based reasoning that explores the possibility of moving towards a more general agent-based model. Our contribution includes a unique lightweight (workable for Llama-14B, Qwen-14B, and GPT-4o-mini) and customizable system that enhances inference through agent-based reasoning (RPLH-Agent) and showed that our method outperforms traditional multi-agent methods and our own baseline method (RPLH-Standard). Most importantly, while our experiments focus on warehouse-related tasks (MoveBox), our system is designed for a structured environment and can scale to more complex warehouse-related tasks and generalize to many other multi-agent task environments to address common challenges in complex long-horizon multi-agent planning tasks, such as redundant repetition in agent dialogues and context information dilution. We believe that our communication system will be a competitive alternative to a centralized and decentralized system, even using a small language model as the backbone.

8 Limitations and Future Works

Since our LMs are less parametrized, we observed that it tends to believe in the most recent event much more than relying on past experiences, leading to the bias of easier detecting a spy agent just right in case of after seeing a spy agent doing an action but not when environmental step walks more into the history, the detection ability drops. A natural extension of this work is to deploy the RPLH system onto more complicated environments with more complicated collaboration tasks (i.e. virtual home, optimal transport, multi-agent coding) where we can assign more complex objectives, more complex constraint, and with increased uncertainty given by environment engines. Due to the structured nature of our base environment and that our agent can interact with the environment well, we believe that our system can scale in such a way.

References

[1] Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. Scalable multirobot collaboration with large language models: Centralized or decentralized systems? 2023.

[2] Zhiting Hu and Tianmin Shu. Language models, agent models, and world models: The law for machine reasoning and planning. 2023.

[3] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of Advances in Neural Information Processing Systems*, 2022.

[4] Instructor-AI. Instructor. `https://github.com/instructor-ai/instructor`. Accessed: 2024-11.

[5] Arsalan Mousavian Ankit Goyal Danfei Xu Jonathan Tremblay Dieter Fox Jesse Thomason Ishika Singh, Valts Blukis and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *In Workshop on Language and Robotics at CoRL 2022*, 2022.

[6] Noah Brown Yevgen Chebotar Omar Cortes Byron David Michael Ahn, Anthony Brohan. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[7] Aidan O'Gara. Hoodwinked: Deception and cooperation in a text-based game for language models. University of Southern California, `aogara@usc.edu`, 2024. Manuscript in preparation.

[8] Ollama. nomic-embed-text: A high-performing open embedding model. `https://ollama.ai/models/nomic-embed-text`, 2024. Accessed: 2024-12-13.

[9] Tianduo Wang Peiyuan Zhang, Guangtao Zeng and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

[10] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of Advances in Neural Information Processing Systems*, 2018.

[11] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, and Faisal Azhar. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[12] Julien Chaumond Thomas Wolf Victor Sanh, Lysandre Debut. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv:1910.01108*, 2019.

[13] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. 2023.

[14] Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. *Proceedings of Advances in Neural Information Processing Systems*, 2023.

[15] Mu Li Alex Smola Zhuosheng Zhang, Aston Zhang. Automatic chain of thought prompting in large language models. *arXiv:2210.03493*, 2022.

A Appendix

A.1 Algorithm of RPLH

Algorithm 1 Agent/Standard RPLH

Initialize environment with i agents, m-step look-ahead, goals G_i, initial information constant $I_c^{(0)}$, and assigned attitudes C_i.
for each t environmental step **do**
  if environment step is 0 **then**
    Set $M_a^{(t)}$, $M_s^{(t)}$, $M_j^{(t)}$ as $\emptyset$
  end if
  Select one agent as Hallucinating Center Agent (HCA), others as local agents.
  while being HCA planning **do**:
    if RPLH-Agent **then**
      Use $M_a^{(t-1)}$, $M_s^{(t-1)}$, $M_j^{(t-1)}$ to conduct preventative measures.
    end if
    Plan r_t:
    Reason how agents react to commands and how agents behave as HCA.
    Hallucinate $h^{(k)}$ $\forall k \in \{t, t+m\}$ future steps to resolve goals.
    Define next action $a^{(t+1)}$ and pass $(r_t, h^{(t,\ldots,t+m)}, a^{(t+1)})$ to local agents.
    if RPLH-Agent **then**
      Pass M_a and M_s only to the HCA judge agent.
    end if
    if Synthetic error **then**
      If
    end if
  end while
  for each local agent i **do**
    if local agent i is not assigned action by HCA or has no boxes/targets in grid **then**
      Pass to next agent.
    else
      Process $I_c^{(t)}$ and attitude C_i.
      Compute suggested action $a_i^{(t+1)}$ or agree with HCA action.
      Output dialogue $D_i^{(t)}$ into sensory information $I_s^{(t)}$.
      Relay "relapsed" messages in $D_i^{(t)}$ and give to judge.
    end if
    while judging **do**
      Understand G_i, $D_i^{(t)}$, and $I_c^{(t)}$ depending on memory setting.
      if RPLH-Agent judging **then**
        HCA acts as judge and reason spy detection and prevention and give new a^*.
        Update to $M_a^{(t)}$, $M_s^{(t)}$, and $M_j^{(t)}$.
      else
        Judge J_{LM} resolves disagreements and give new a^*.
      end if
    end while
  end for
  Execute global action a^* and switch HCA to a new agent.
  if All grids are empty **then**
    Break, convergence
  end if
end for

A.2 Additional RPLH Algorithm Descriptions

The following go over the the inference loop again with some additional important details on the implementation side.

1. One agent hallucinates themselves being the central agent (HCA) and the rest agents being local agents. They take in the input of information constant $I_c^{(t)}$ depending on memory settings.
   (a) When the local spy agent takes turn to be the HCA agent, it does not know its identity being the spy anymore.
   (b) **RPLH-Agent**: Based on information in the 3rd degree Markovian (can be defined by the user) agent model $M_a^{(t-1)}$, spy model $M_s^{(t-1)}$, and justification model $M_j^{(t-1)}$, try to reason and conduct preventative measures based on the rules of the environment (i.e. not assigning action would pass the agent or not listening to the local agent).
   (c) Hallucinate two things and store them as r_t (HCA only plans when being the central planner):
      i. Reasons about how other agents would react when receiving HCA's command.
      ii. Reasons about how all other agents would give commands when they are the HCA agents.
   (d) After reasoning with other agents, based on these reasonings, hallucinate future n steps that would result in resolving the goal states, we refer to this as $h^{(k)} \quad \forall\, k \in \{t, t+m\}$.
   (e) Write the immediate next step action plan $a^{(t+1)}$ as executable format.
   (f) Pass all $r^{(t)}$, $h^{(t,\ldots t+m)}$, and $a^{(t+1)}$ to all the other i agents currently being the local agents.
   (g) **RPLH-Agent**: Pass the agent model M_a and spy model M_s only to the HCA judge agent.
2. Depending on the HCA command and environmental situation, local agent may be passed.
   (a) If local agent does not have assigned action by HCA, pass.
   (b) If local agent does not have any boxes or targets in its grid, pass.
3. Local agent receives messages from the HCA.
   (a) Understand the rules of the environment G_i and the current situation in information constant $I_c^{(t)}$ based on history methods.
   (b) From I_s each agent understands their assigned attitude C_i (spy agent takes role if environment is defined to be adversarial).
   (c) Conduct reasoning based on the attitude assigned and give suggested actions for $a_i^{(t+1)}$ or agree with HCA's $a^{(t+1)}$.
   (d) All local agent output (usually spy doesn't speak much) is given to the judge agent (depending on the system, this may be a new judge agent of HCA agent playing the judge agent) and pushes dialogue $D_i^{(t)}$ into sensory information $I_s^{(t)}$.
   (e) From the second local agents onward, each local i agent receives "relapsed" messages that come from $i-1$ local agent round.
4. Depending on RPLH-Standard or RPLH-Agent:
   (a) **RPLH-Standard**: Judge J_{LM} gets involved if there are disagreements in action plans.
   (b) **RPLH-Agent**: HCA play the role of the judge and gets the local feedback again and tries to conduct agent-based reasoning, namely:
      i. Whether this local agent has spy tendency and what plans should the HCA take to avoid being affected by the spy agent.
         A. Update spy model $M_a^{(t)}$.
         B. Update agent model $M_s^{(t)}$.
         C. Update justification model $M_j^{(t)}$.
5. After all available local HCA give their opinion, the global action a^* after discussion is executed, and the HCA agent is switched to a new agent to repeat from step 1.

A.3 Attitude Configurations

An example of the definition of each agent's unique additional prompt may look like the following. This field may be customizable by the user of our system in the config sections [3] of our system

- **nice agent:** "Be very easy going. Try to find agreement with the central agent if you can, the goal is to resolve conversation."
- **critical agent:** "Be very critical and try to propose many changes."
- **agreeing agent:** "Be super cooperative, agree to whatever the central agent says directly."
- **spy agent:** "Trying to mess up decision making."

A.4 Embedding Comparison Sentences

This is the standard embedding comparison sentence that we use, we only used the spy sentence for comparison in this study. However, teh same procedure can be down to reason with other attitudes as well.

- **nice agent**: "Seems to be very easy-going, trying to find agreement with the central agent and the goal is to resolve conversation."
- **critical agent**: "Seems to be very critical and proposing many changes."
- **agreeing agent**: "Seems to be super cooperative, agreeing to central command."
- **spy agent**: "Seems to be the spy agent, its goal is to prevent match targets."
- **justification**: "I suspect that this agent is the spy agent, thus, I would not listen to this agent."

A.5 Feature Importance Metrics

We made agent's performance metrics into features and used a Random Forest Regressor to see which of the features matter more to the prediction of number of response or average targets matched per response.

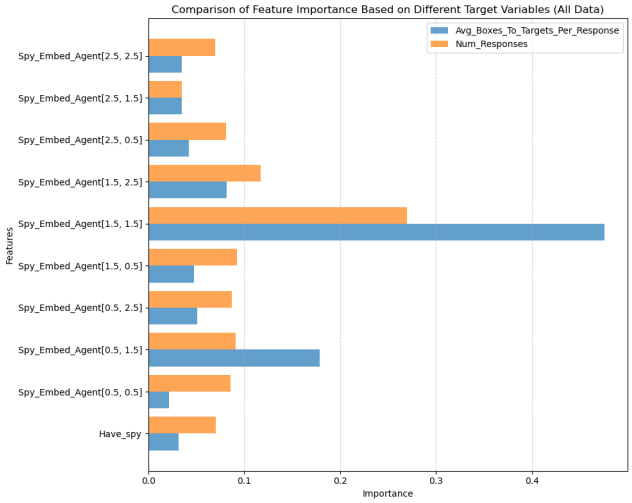

Figure 4: Feature importance using RandomForest Regressor

A.6 Hypothesis Testing

We performed follow-up bootstrapped hypothesis testing with $\alpha = 0.1$ and test-statistics as differences in mean to evaluate whether there the presence of spy is statistically significant for the number of responses needed.

[3] Config system in https://github.com/KevinBian107/RPLH/tree/master/rplh/configs

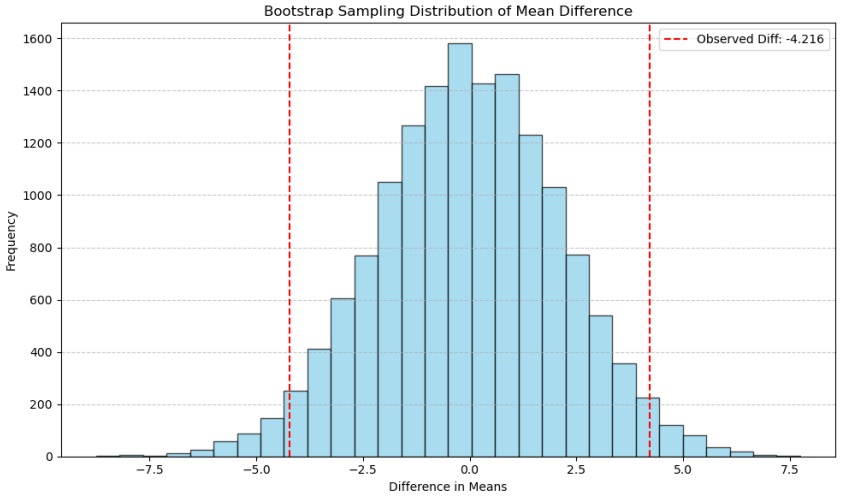

Figure 5: Bootstrapped Hypothesis Testing

A.7 Distance Energy Metric

Example of the energy metric in terms of norm 1 and norm 2 distance for each system at trial 3 with the same seed of 3.

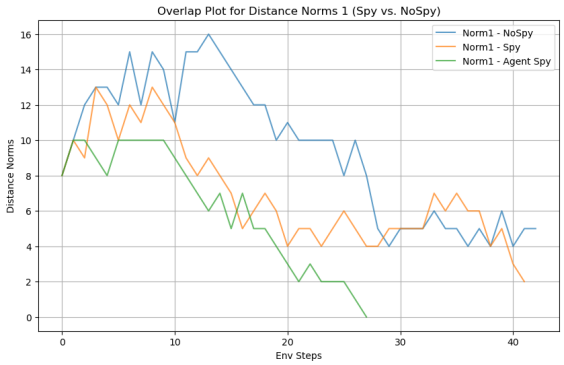

Figure 6: Norm 1 Energy Metric

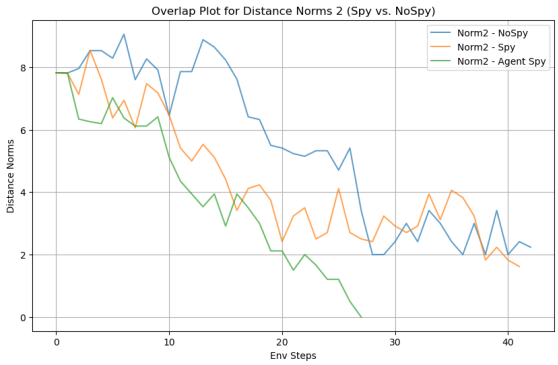

Figure 7: Norm 2 Energy Metric

A.8 Sentence Embedding Similarity

	Spy_Agent[0.5, 0.5]	Spy_Agent[1.5, 1.5]	Spy_Agent[2.5, 2.5]	Justification_Embed
count	14	14	14	14
mean	0.571	0.619	0.643	0.615
std	0.108	0.0748	0.110	0.050
min	0.354	0.530	0.354	0.500
25%	0.533	0.567	0.600	0.590
50%	0.588	0.600	0.683	0.619
75%	0.644	0.673	0.726	0.651
max	0.712	0.753	0.734	0.699

Table 2: Statistical comparison of embedding similarities for different agents and justification embeddings over 20 trials with max 2 boxes of the same color in a 3x3 grid environment.

A.9 Additional Result (Box Num Upperbound = 2)

Metric	rplh-agent-spy	rplh-standard-spy	rplh-standard-nospy
Box-To-Target*	**0.365**	0.333	0.290
Steps*	**24.096**	26.279	28.87
Success*	**73.6%**	70.2%	60.1%
AUC-Norm-1*	**156.3594**	163.61	160.321
AUC-Norm-2*	**85.789**	90.377	87.479
Slope-Norm-1	**-0.661**	-0.622	-0.567
Slope-Norm-2	**-0.417**	-0.371	-0.334
Similarity-Spy-1	0.592	—	—
Similarity-Spy-2	0.641	—	—
Similarity-Spy-3	0.613	—	—
Similarity-Justification	**0.611**	—	—

Table 3: Evaluation results for different methods. We report standard metrics, energy metrics, and embedding similarity metrics over 20 trials with a max of 2 boxes of the same color in a 3x3 grid environment. Asterisk* denotes the value is bootstrapped 10,000 times for a robust comparison.