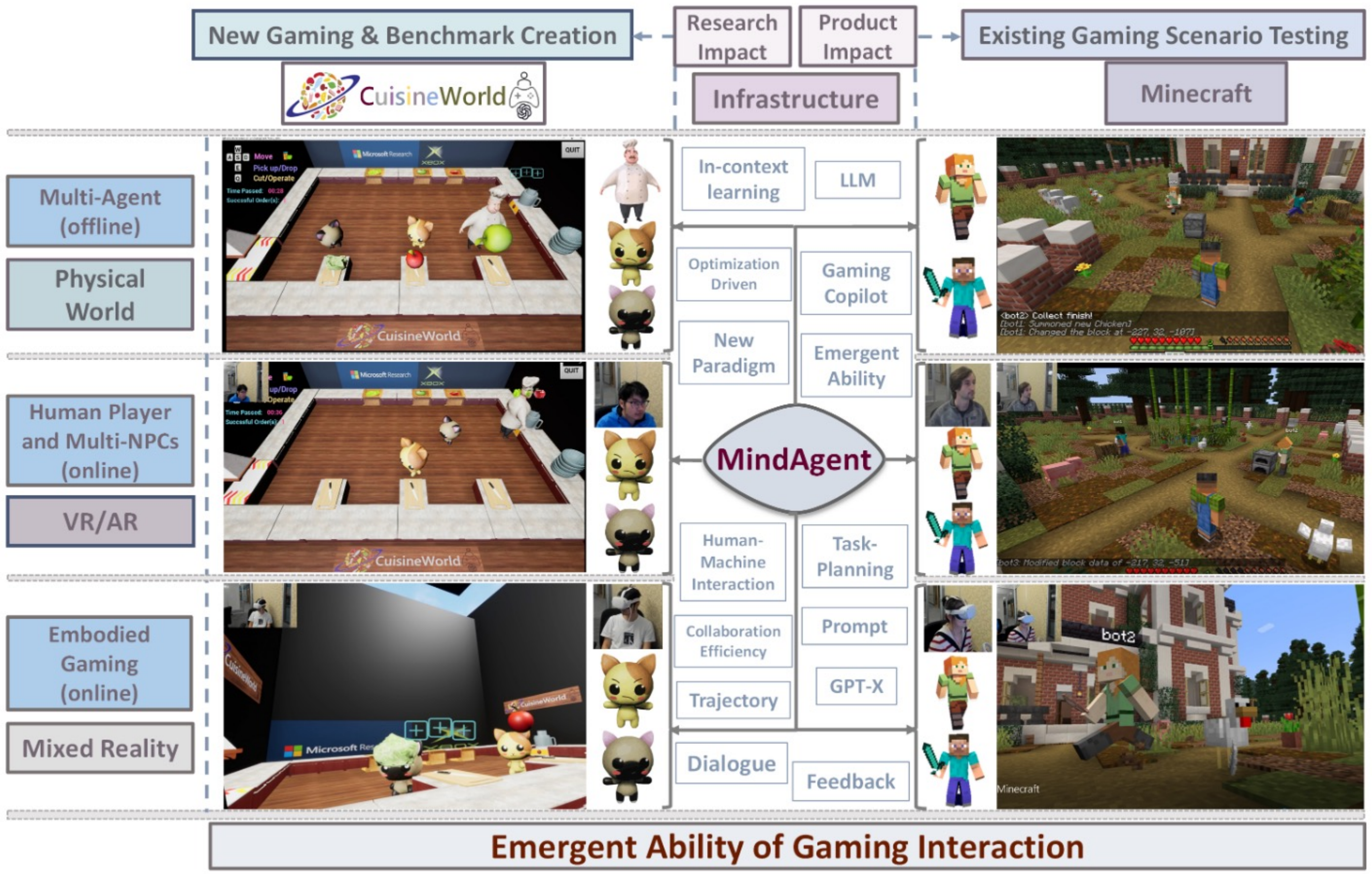
# MindAgent: Emergent Gaming Interaction

**Ran Gong[1†*], Qiuyuan Huang[2‡*], Xiaojian Ma[1*], Hoi Vo[3], Zane Durante[4†], Yusuke Noda[3], Zilong Zheng[5], Song-Chun Zhu[1567], Demetri Terzopoulos[1], Li Fei-Fei[4], Jianfeng Gao[2]**
[1]UCLA; [2]Microsoft Research, Redmond; [3]Xbox Team, Microsoft; [4]Stanford;[5]BIGAI; [6]PKU; [7]THU

# Overview

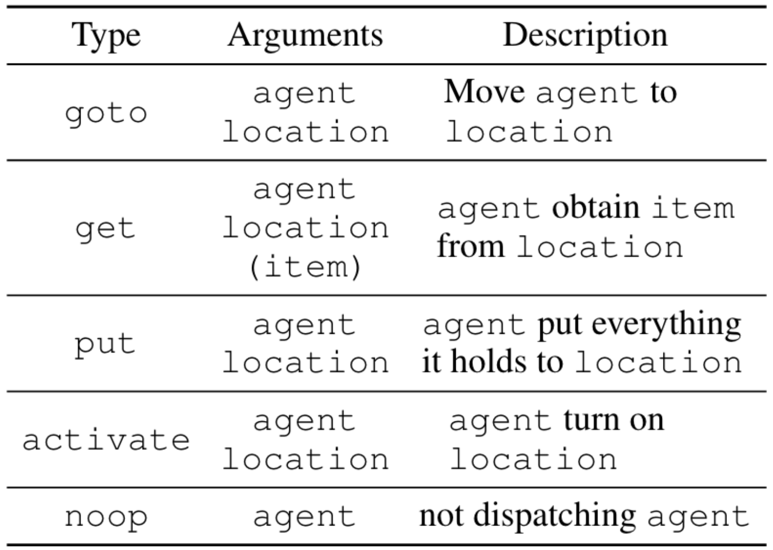

New Gaming & Benchmark Creation
Research Impact
Product Impact
Existing Gaming Scenario Testing
CuisineWorld
Infrastructure
Minecraft
Multi-Agent (offline)
Physical World
Human Player and Multi-NPCs (online)
VR/AR
Embodied Gaming (online)
Mixed Reality
In-context learning
LLM
Optimization Driven
Gaming Copilot
New Paradigm
Emergent Ability
MindAgent
Human-Machine Interaction
Task-Planning
Collaboration Efficiency
Prompt
GPT-X
Trajectory
Dialogue
Feedback
Emergent Ability of Gaming Interaction

## Main Contribution

- A new gaming scenario and related benchmark based on a multi-agent virtual kitchen environment, **CuisineWorld**.
- Introduce **MindAgent**, which demonstrates the in-context learning multi-agent planning capacity of LLMs and brings several prompting techniques that help facilitate their planning ability.
- Conduct extensive evaluations with multiple LLMs and prompting settings on the benchmark.
- Deploy the system into real-world gaming scenarios and demonstrate its capabilities in human-AI interactions. (using VR)

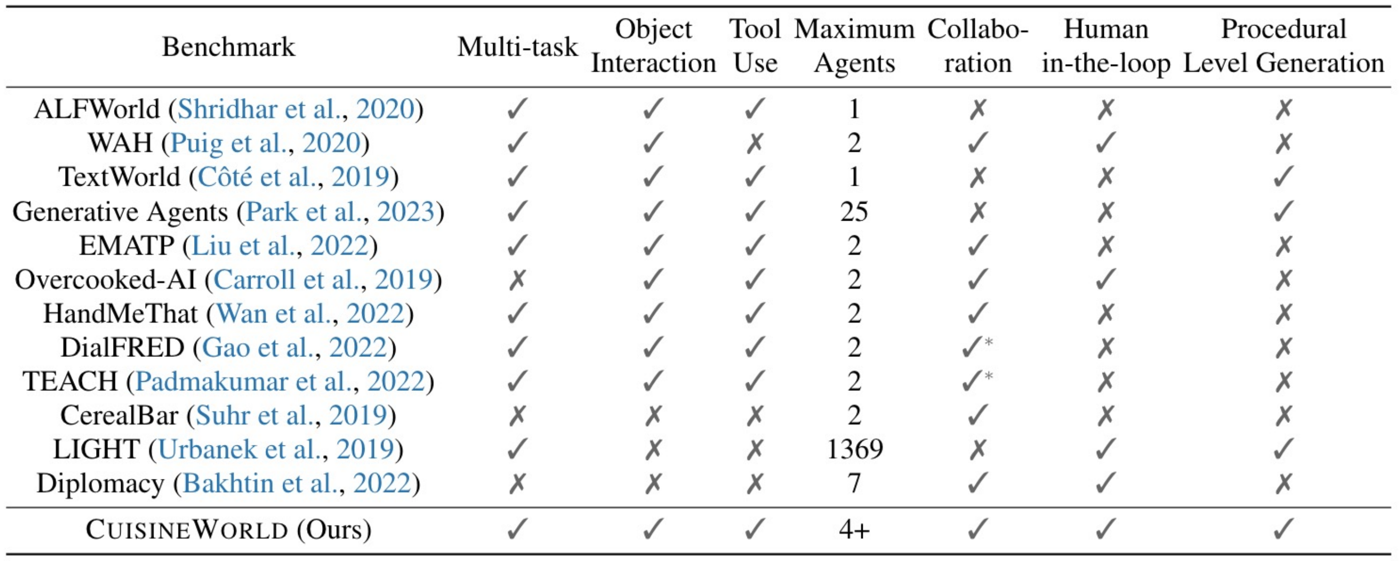
# CuisineWorld

- Multi-agent text game.



| Type | Arguments | Description |
|---|---|---|
| `goto` | `agent` `location` | Move `agent` to `location` |
| `get` | `agent` `location` `(item)` | `agent` obtain `item` from `location` |
| `put` | `agent` `location` | `agent` put everything it holds to `location` |
| `activate` | `agent` `location` | `agent` turn on `location` |
| `noop` | `agent` | not dispatching `agent` |

Table 2: Action space in CUISINEWORLD.



| Benchmark | Multi-task | Object Interaction | Tool Use | Maximum Agents | Collaboration | Human in-the-loop | Procedural Level Generation |
|---|---|---|---|---|---|---|---|
| ALFWorld (Shridhar et al., 2020) | ✓ | ✓ | ✓ | 1 | ✗ | ✗ | ✗ |
| WAH (Puig et al., 2020) | ✓ | ✓ | ✗ | 2 | ✓ | ✓ | ✗ |
| TextWorld (Côté et al., 2019) | ✓ | ✓ | ✓ | 1 | ✗ | ✗ | ✓ |
| Generative Agents (Park et al., 2023) | ✓ | ✓ | ✓ | 25 | ✗ | ✗ | ✓ |
| EMATP (Liu et al., 2022) | ✓ | ✓ | ✓ | 2 | ✓ | ✗ | ✗ |
| Overcooked-AI (Carroll et al., 2019) | ✗ | ✓ | ✓ | 2 | ✓ | ✓ | ✗ |
| HandMeThat (Wan et al., 2022) | ✓ | ✓ | ✓ | 2 | ✓ | ✗ | ✗ |
| DialFRED (Gao et al., 2022) | ✓ | ✓ | ✓ | 2 | ✓* | ✗ | ✗ |
| TEACH (Padmakumar et al., 2022) | ✓ | ✓ | ✓ | 2 | ✓* | ✗ | ✗ |
| CerealBar (Suhr et al., 2019) | ✗ | ✗ | ✗ | 2 | ✓ | ✗ | ✗ |
| LIGHT (Urbanek et al., 2019) | ✓ | ✗ | ✗ | 1369 | ✗ | ✓ | ✓ |
| Diplomacy (Bakhtin et al., 2022) | ✗ | ✗ | ✗ | 7 | ✓ | ✓ | ✗ |
| CUISINEWORLD (Ours) | ✓ | ✓ | ✓ | 4+ | ✓ | ✓ | ✓ |

Table 1: Comparsion between CUISINEWORLD and other related benchmarks. **Multi-task**: The benchmark contains multiple different tasks. **Object Interaction**: Agents have to manipulate or engage with different items or environmental elements to achieve certain goals with irreversible actions. **Tool Use**: Completing tasks necessitates the use of specific tools by the agents. **Maximum Agents**: This denotes the upper limit of agents that can be present in a single experiment. **Collaboration**: Many tasks mandate teamwork and collaboration between different agents. **Human in-the-loop**: The framework allows humans to join the game and collaborate actively with the agents. **Procedural Level Generation**: There's flexibility in adding new tasks, making the game dynamic and adaptable. *: Notably, even though multiple agents can be present, the second agent is limited to communicating with the first agent. The second agent cannot interact with the environment in an active gaming capacity.

# MindAgent

Within the prompt component, there are four distinct sub-components: recipes, general instructions, inference knowledge, and a one-shot demo.

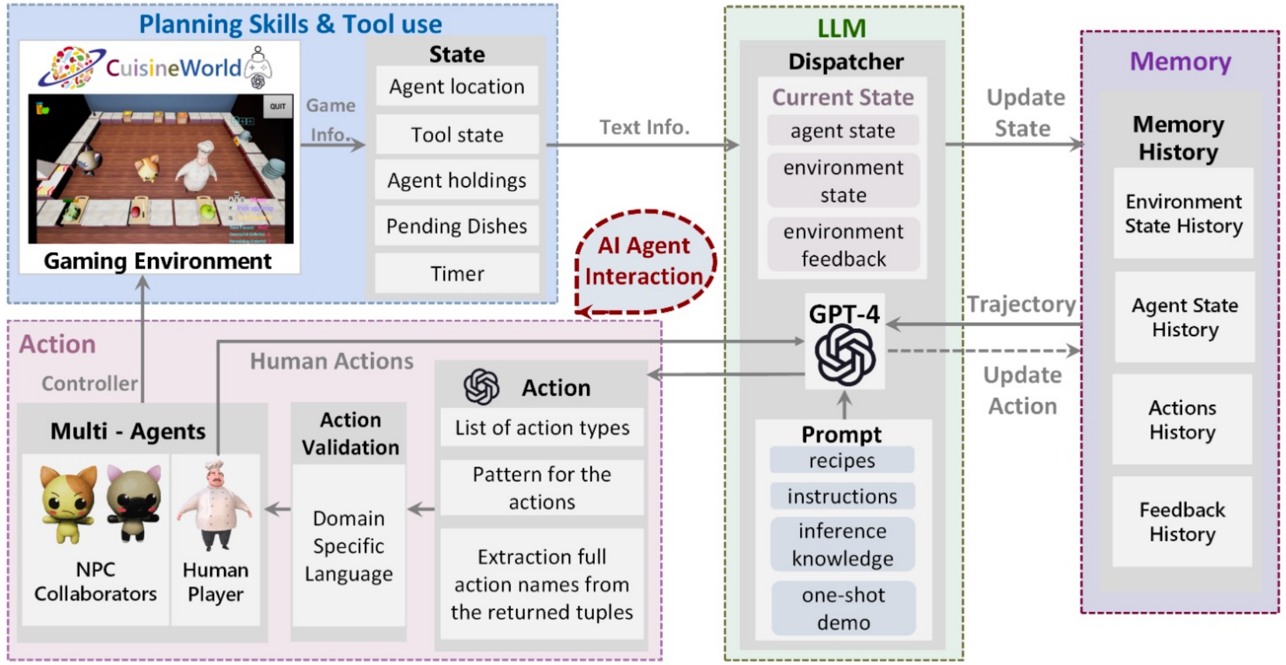
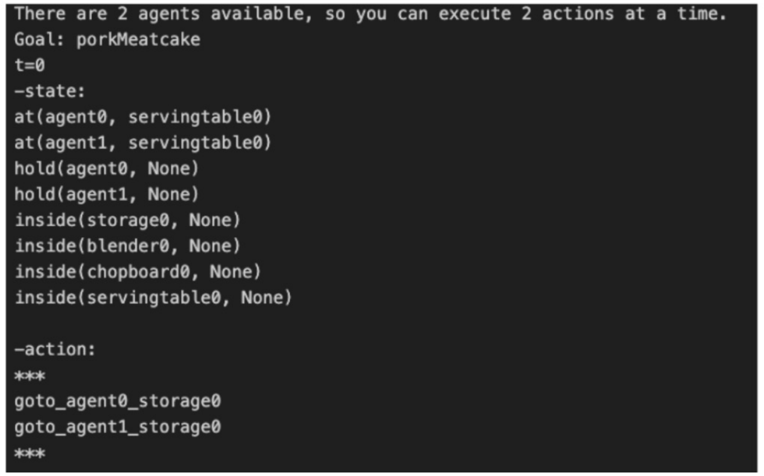


Figure 3: Our overview of our MINDAGENT architecture. **Planning Skill & Tool Use**: The game environment requires diverse planning skills and tool use to complete tasks. It emits related game information. This module also converts relevant game data into a structured text format so the LLMs can process it. **LLM**: The main workhorse of our infrastructure makes decisions, which is a dispatcher for the multi-agent system. **Memory History**: A storage utility that stores relevant information. **Action Module**, extract actions from text inputs and convert them into domain-specific language. Validate DSLs so they don't cause errors when executing.



```
There are 2 agents available, so you can execute 2 actions at a time.
Goal: porkMeatcake
t=0
-state:
at(agent0, servingtable0)
at(agent1, servingtable0)
hold(agent0, None)
hold(agent1, None)
inside(storage0, None)
inside(blender0, None)
inside(chopboard0, None)
inside(servingtable0, None)

-action:
***
goto_agent0_storage0
goto_agent1_storage0
***
```

One-shot demo

**Recipes.** outline the hierarchical procedure for preparing various dishes at the given level. They specify the necessary ingredients for each intermediate or final product, the appropriate tools required, and the expected outcome post-cooking.

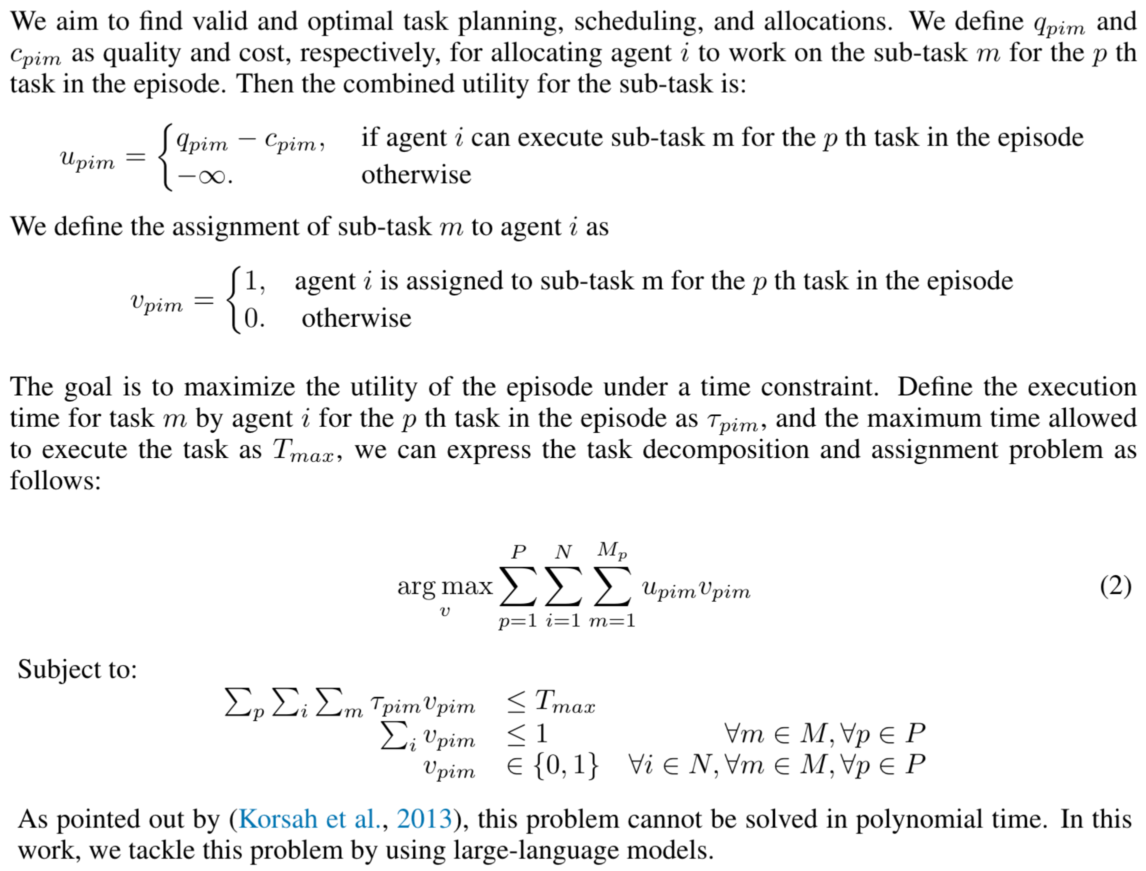
**Instructions.** detail the foundational rules of CUISINEWORLD. These instructions delineate the array of actions agents can undertake within the game and enumerate the characteristics of every tool available in the current kitchen scenario. Moreover, they inform agents about the base ingredients retrievable from storage, as well as all potential intermediate products they can procure. Agents are also explicitly advised to remain cautious about feedback from the environment.

**Inference Knowledge.** houses insights and helpful hints for the agent. When utilized appropriately, these hints can guide agents to sidestep potential errors and enhance their collaborative efficiency.

**One-shot Demo.** presents a step-by-step demonstration of the preparation of a distinct dish, different from other dishes at the current level. This demonstration spans several time steps, each of which is incorporated as part of the prompt. The demonstration illustrates the major procedures for cooking one dish in CUISINEWORLD, including obtaining ingredients, putting ingredients into different tools, transporting intermediate ingredients, and delivering the final dish to the serving table.

# MindAgent

- Multi-agent task allocation



We aim to find valid and optimal task planning, scheduling, and allocations. We define $q_{pim}$ and $c_{pim}$ as quality and cost, respectively, for allocating agent $i$ to work on the sub-task $m$ for the $p$ th task in the episode. Then the combined utility for the sub-task is:

$$u_{pim} = \begin{cases} q_{pim} - c_{pim}, & \text{if agent } i \text{ can execute sub-task m for the } p \text{ th task in the episode} \\ -\infty. & \text{otherwise} \end{cases}$$

We define the assignment of sub-task $m$ to agent $i$ as

$$v_{pim} = \begin{cases} 1, & \text{agent } i \text{ is assigned to sub-task m for the } p \text{ th task in the episode} \\ 0. & \text{otherwise} \end{cases}$$

The goal is to maximize the utility of the episode under a time constraint. Define the execution time for task $m$ by agent $i$ for the $p$ th task in the episode as $\tau_{pim}$, and the maximum time allowed to execute the task as $T_{max}$, we can express the task decomposition and assignment problem as follows:

$$\arg\max_{v} \sum_{p=1}^{P} \sum_{i=1}^{N} \sum_{m=1}^{M_p} u_{pim} v_{pim} \tag{2}$$

Subject to:

$$\begin{aligned} \textstyle\sum_p \sum_i \sum_m \tau_{pim} v_{pim} &\leq T_{max} \\ \textstyle\sum_i v_{pim} &\leq 1 & \forall m \in M, \forall p \in P \\ v_{pim} &\in \{0,1\} & \forall i \in N, \forall m \in M, \forall p \in P \end{aligned}$$

As pointed out by (Korsah et al., 2013), this problem cannot be solved in polynomial time. In this work, we tackle this problem by using large-language models.

Reformulate **q** or **r** in natural language.
e.g. "collect finish"

Prevent bad assignment.
e.g. "agent ids cannot be the same"

Using state-action memory history as rewards are not immediately observable

## Experiments

**Overview.** We conduct extensive experiments in CuisineWorld. We first introduce the experiment settings and present an analysis of empirical results in CuisineWorld. Our experiments focus on addressing the following research questions:

**Q1:** How efficiently can the model dispatch multiple agents?

**Q2:** Can the model dispatch agents for dynamic, on-the-fly goals across different tasks?

**Q3:** How do various components of the input prompt influence the model's performance?

**Q4:** How do other LLMs perform compared to GPT-4?

**Q5:** To what extent can the existing methods collaborate with human users?

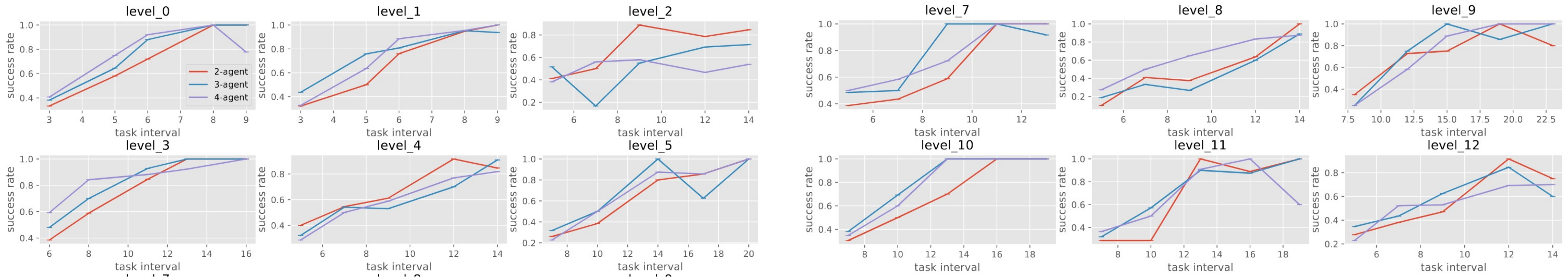
**Q6:** What's the human perception of collaborating with numerous intelligent agents?

We perform experiments on CuisineWorld through OpenAI APIs and anthropic APIs. All GPT-4 experiments are using gpt-4-0613 model, and all chat-GPT experiments are using gpt-3.5-turbo-0613. For Llama 2 experiments, we use hugging face inference endpoints Llama-2-70b-chat-hf. We set the temperature for all experiments to 0.1 following (Wang et al., 2023a). We report the average results over three episodes.

## Experiment 1: LLMS Dispatch Multi-Agents (NPC)

- Collaboration Efficiency

$$\mathbf{CoS} = \frac{1}{M}\sum_{i=1}^{M} \frac{\#\textit{completed task}\left[\tau_{\text{int},(i)}\right]}{\#\textit{completed task}\left[\tau_{\text{int},(i)}\right] + \#\textit{failed task}\left[\tau_{\text{int},(i)}\right]},$$

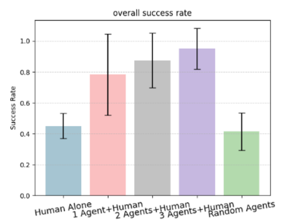


- LLM dispatcher can coordinate more agents to execute tasks more efficiently.
- LLM dispatcher struggles when there are fewer tasks

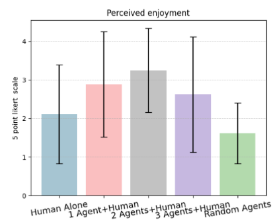
# Experiment 2: Human and Multi-NPCs with LLMs

**Hypotheses**. The user study tests the following hypotheses:

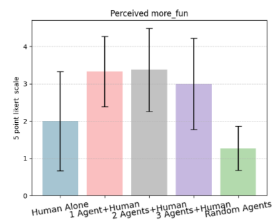
- **H1: Task productivity**. Participants have higher productivity if collaborating with AI agents.
- **H2: Task productivity with more agents**. Participants have higher productivity if collaborating with more AI agents.
- **H3: Perception of the robot.** Participants would have higher perceived task efficiency and have more fun playing the game due to collaboration.



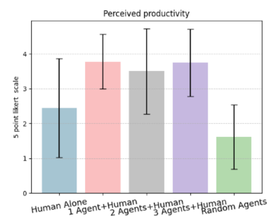
(a) **Collaboration score** We can tell that the collaboration score is higher if more agents are collaborating with human players, even though the difference is not significant.



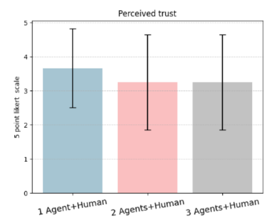
(b) **Perceived Enjoyment** Humans enjoy the game more if they collaborate with the right number of agents



(c) **Perceived more fun** due to collaboration. Players enjoy the game more because of collaborating with competent agents.

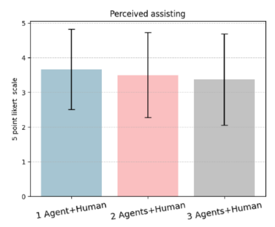


(g) **Perceived productivity**. Players think collaborating with AI agents will improve productivity.

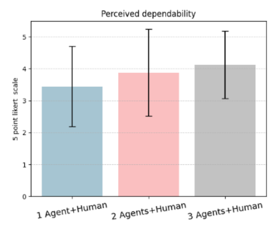


(h) **Perceived Trust**. There is no difference in terms of trust when collaborating with more agents.

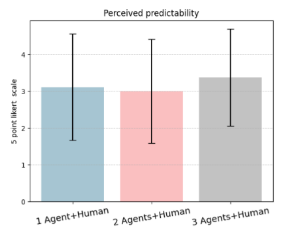
Figure 5: Human Evaluations



(d) **Perceived Assisting**. There is no significant difference in terms of human perceptions of helpfulness when collaborating with more agents, even though the task success rate is higher.



(e) **Perceived dependability**. When collaborating with more agents, players depend on the agents more.

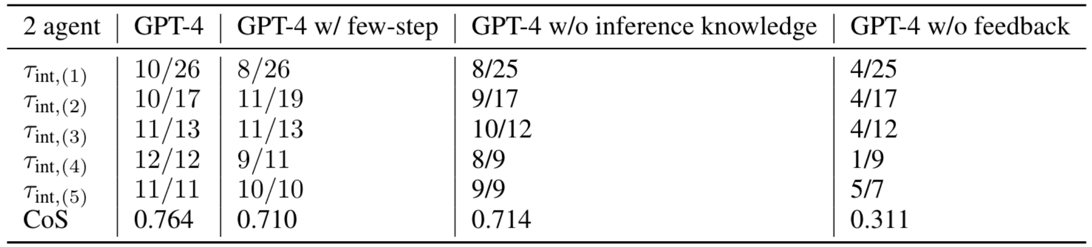


(f) **Perceived Predictability**. There is no difference in terms of the predictability of agents' behaviors when collaborating with more agents.

**Findings.** We find significant effects on team collaboration success rate $F(4,55) = 28.11, p < 0.001$. Post-hoc comparisons using the Tukey HSD tests revealed that the team of the player with LLM agents achieves a higher success rate than a human working alone, $p < 0.001$ across different numbers of agents, **confirming H1**. Even though the success rate is generally higher when collaborating with more agents, there is no significant effect compared with collaborating with one agent, collaborating with two agents $p = 0.774$, or collaborating with three agents $p = 0.231$. We observe that human players have more fun playing the game when collaborating with LLM-powered intelligent agents than playing alone, $p = 0.0126$. Players feel that collaboration with intelligent agents leads to higher productivity, $p = 0.0104$, thus **confirming H3**.

# Analysis and Emergent Gaming Abilities

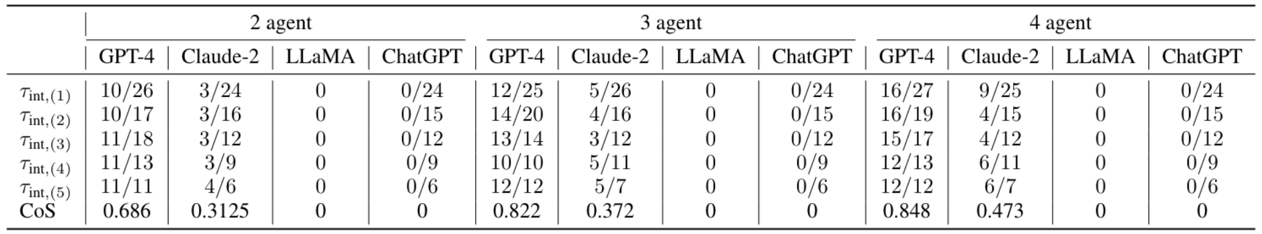
- How do various components of the input prompt influence the model's performance?



| 2 agent | GPT-4 | GPT-4 w/ few-step | GPT-4 w/o inference knowledge | GPT-4 w/o feedback |
|---|---|---|---|---|
| $\tau_{\text{int},(1)}$ | 10/26 | 8/26 | 8/25 | 4/25 |
| $\tau_{\text{int},(2)}$ | 10/17 | 11/19 | 9/17 | 4/17 |
| $\tau_{\text{int},(3)}$ | 11/13 | 11/13 | 10/12 | 4/12 |
| $\tau_{\text{int},(4)}$ | 12/12 | 9/11 | 8/9 | 1/9 |
| $\tau_{\text{int},(5)}$ | 11/11 | 10/10 | 9/9 | 5/7 |
| CoS | 0.764 | 0.710 | 0.714 | 0.311 |

Table 7: Additional Ablation

- Other LLM's performance



| | 2 agent | | | | 3 agent | | | | 4 agent | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GPT-4 | Claude-2 | LLaMA | ChatGPT | GPT-4 | Claude-2 | LLaMA | ChatGPT | GPT-4 | Claude-2 | LLaMA | ChatGPT |
| $\tau_{\text{int},(1)}$ | 10/26 | 3/24 | 0 | 0/24 | 12/25 | 5/26 | 0 | 0/24 | 16/27 | 9/25 | 0 | 0/24 |
| $\tau_{\text{int},(2)}$ | 10/17 | 3/16 | 0 | 0/15 | 14/20 | 4/16 | 0 | 0/15 | 16/19 | 4/15 | 0 | 0/15 |
| $\tau_{\text{int},(3)}$ | 11/18 | 3/12 | 0 | 0/12 | 13/14 | 3/12 | 0 | 0/12 | 15/17 | 4/12 | 0 | 0/12 |
| $\tau_{\text{int},(4)}$ | 11/13 | 3/9 | 0 | 0/9 | 10/10 | 5/11 | 0 | 0/9 | 12/13 | 6/11 | 0 | 0/9 |
| $\tau_{\text{int},(5)}$ | 11/11 | 4/6 | 0 | 0/6 | 12/12 | 5/7 | 0 | 0/6 | 12/12 | 6/7 | 0 | 0/6 |
| CoS | 0.686 | 0.3125 | 0 | 0 | 0.822 | 0.372 | 0 | 0 | 0.848 | 0.473 | 0 | 0 |

Table 6: Performance of Other LLMs on Level 3

- Emergent Capabilities

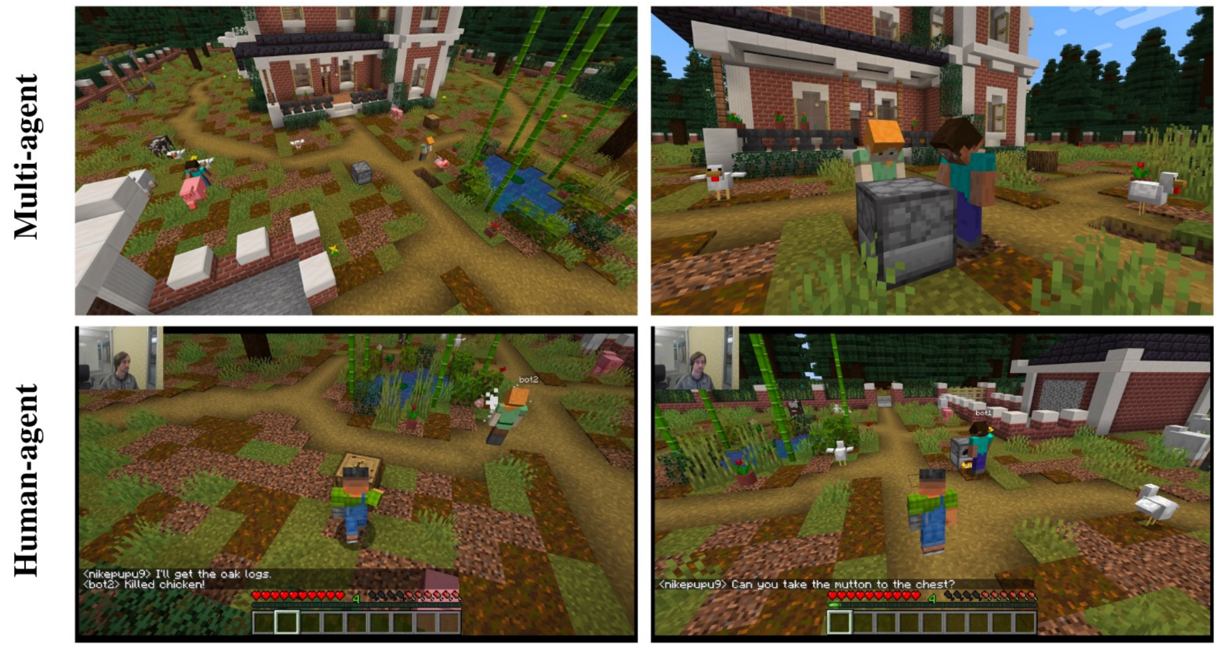
Yet, despite this limited input, GPT-4's performance is remarkable. This underscores GPT-4's impressive **emergent zero-shot multi-agent planning** capabilities. Beyond simply completing unseen tasks, GPT-4 also demonstrates adaptability by dynamically prioritizing multiple different tasks as they arise, emphasizing its **emergent multi-task, on-the-fly planning** skills.

**Emergent Multi-agent Reasoning Capabilities.** Referencing Table 8, GPT-4 has the capability to deploy more agents based on demonstrations of fewer agents. For instance, GPT-4 can effectively dispatch four agents having only seen demonstrations involving two agents. Moreover, the efficiency of collaboration is higher as the number of agents increases, spotlighting its **emergent collaboration** prowess.

# Novel Game Adaptation (Minecraft)

We define the following actions for the multi-agent system in our Minecraft game: 1) `goto(agent, location)`; 2) `killMob(agent, mobType)`; 3) `mineBlock(agent, blockType)`; 4) `putFuelFurnace(agent, fuelType)`, to put the item from agent's inventory to the furnace's bottom slot. 5) `putItemFurnace(agent, itemType)`, to put the item from agent's inventory to the furnace's top slot; 6) `takeOutFurnace(agent)`, take out the cooked item from the furnace 7) `putInChest(agent, itemType)` ;

The state space in Minecraft contains the following: 1) nearby blocks for each agent 2) nearby entities for each agent. 3) each agent's inventory 4) items inside the furnace 5) items inside the chest. 6) human player's inventory if a human player is involved.



<nikepupu9> I'll get the oak logs.
<bot2> Killed chicken!

<nikepupu9> Can you take the mutton to the chest?

# ProAgent: Building Proactive Cooperative AI with Large Language Models

**Ceyao Zhang**[1,2,*†] **Kaijie Yang**[3,*] **Siyi Hu**[4,*]
**Zihao Wang**[2] **Guanghe Li**[2] **Yihang Sun**[2] **Cheng Zhang**[2] **Zhaowei Zhang**[2,5] **Anji Liu**[6]
**Song-Chun Zhu**[2,5] **Xiaojun Chang**[4] **Junge Zhang**[7]
**Feng Yin**[1] **Yitao Liang**[2] **Yaodong Yang**[2,‡]

[1] SSE&FNii, The Chinese University of Hong Kong, Shenzhen,
[2] Institute for Artificial Intelligence, Peking University,
[3] Monash University,
[4] ReLER, AAII, University of Technology Sydney,
[5] Beijing Institute for General Artificial Intelligence (BIGAI),
[6] University of California, Los Angeles,
[7] Institute of Automation, Chinese Academy of Sciences

# Overview

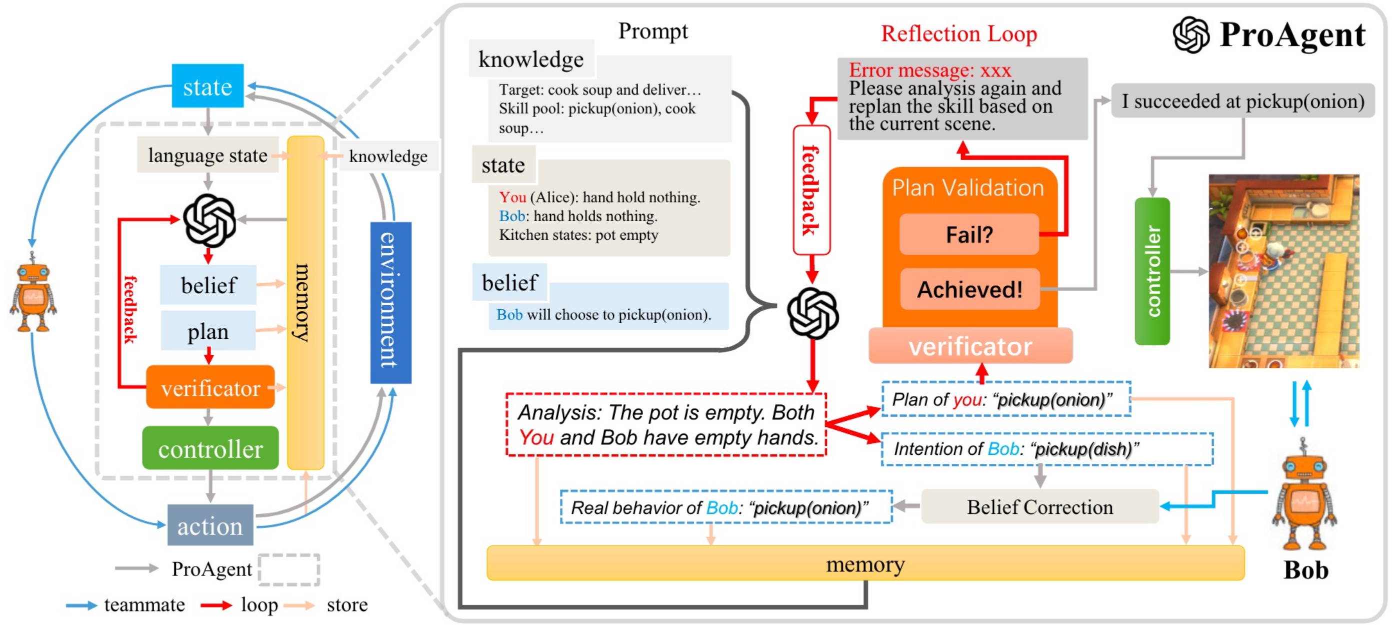

state
language state
knowledge
belief
plan
memory
environment
verificator
controller
action
feedback
ProAgent
teammate
loop
store
Prompt
knowledge
Target: cook soup and deliver…
Skill pool: pickup(onion), cook soup…
state
You (Alice): hand hold nothing.
Bob: hand holds nothing.
Kitchen states: pot empty
belief
Bob will choose to pickup(onion).
Reflection Loop
Error message: xxx
Please analysis again and replan the skill based on the current scene.
ProAgent
I succeeded at pickup(onion)
Plan Validation
Fail?
Achieved!
verificator
controller
Analysis: The pot is empty. Both You and Bob have empty hands.
Plan of you: "pickup(onion)"
Intention of Bob: "pickup(dish)"
Real behavior of Bob: "pickup(onion)"
Belief Correction
memory
Bob

Figure 1: Overview of our proposed ProAgent framework including the coordination task workflow (**left**) and inner details of ProAgent pipeline (**right**). ProAgent commences its operation by translating the initial state into natural language. A large language model (Planner) adeptly analyzes the provided language state in conjunction with historical information stored in the Memory. This analytical process allows the model to discern the *intentions* of the teammate and devise a high-level *skill* for the agent accordingly. The predicted intention is validated through the Belief Correction mechanism, which involves comparing it with the ground truth behavior of the teammate agent. In case of skill failure, the Verificator is summoned to assess the skill's preconditions and provide a detailed explanation for the encountered issue. Should the need arise, ProAgent enters into a re-plan loop, initiating a recalibration process. On the other hand, if the skill is deemed viable, the Controller further dissects it into several executive actions, to be executed within the environment.

LLM agent for cooperative scenarios, zero-shot coordination problem.

Environment: Overcooked-AI

Demonstrates the remarkable capability of ProAgent to interpretably analyze the current scene, explicitly infer teammates' intentions and dynamically adapt its behavior accordingly.

# Methods

- Knowledge Library
- State Alignment
- Cooperative Reasoning and Planning.

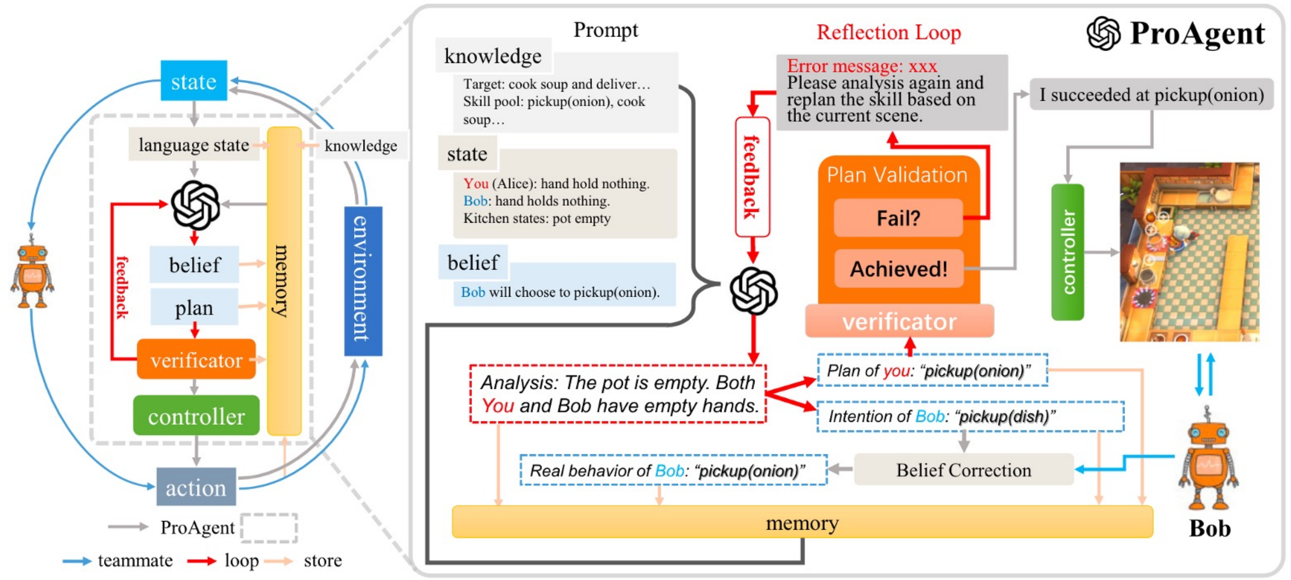
belief, plan (output by LLM)

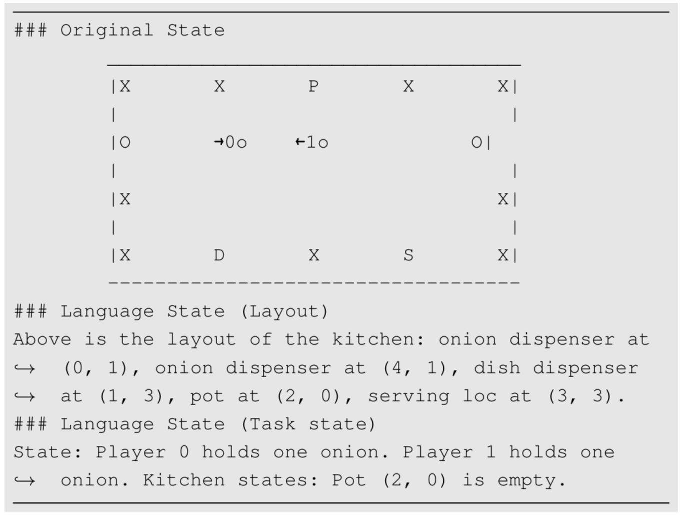
- Belief Correction

1) Replace the predicted intention with the actual behavior of the teammate.
2) Provide an annotation alongside the original prediction to flag it as incorrect.

- Skill Validation
- Controller

The controller can be a rule-based path search algorithm or a policy trained by language-grounded RL.


state
language state
knowledge
belief
plan
feedback
verificator
controller
memory
environment
action
ProAgent
teammate
loop
store
Prompt
knowledge
Target: cook soup and deliver…
Skill pool: pickup(onion), cook soup…
state
You (Alice): hand hold nothing.
Bob: hand holds nothing.
Kitchen states: pot empty
belief
Bob will choose to pickup(onion).
Reflection Loop
Error message: xxx
Please analysis again and replan the skill based on the current scene.
feedback
Plan Validation
Fail?
Achieved!
verificator
I succeeded at pickup(onion)
controller
ProAgent
Analysis: The pot is empty. Both You and Bob have empty hands.
Plan of you: "pickup(onion)"
Intention of Bob: "pickup(dish)"
Real behavior of Bob: "pickup(onion)"
Belief Correction
memory
Bob


```
### Original State

        ___________________________________
        |X       X       P       X       X|
        |                                 |
        |O      →0o     ←1o              O|
        |                                 |
        |X                               X|
        |                                 |
        |X       D       X       S       X|
        -----------------------------------
### Language State (Layout)
Above is the layout of the kitchen: onion dispenser at
↪  (0, 1), onion dispenser at (4, 1), dish dispenser
↪  at (1, 3), pot at (2, 0), serving loc at (3, 3).
### Language State (Task state)
State: Player 0 holds one onion. Player 1 holds one
↪  onion. Kitchen states: Pot (2, 0) is empty.
```

# Experiments

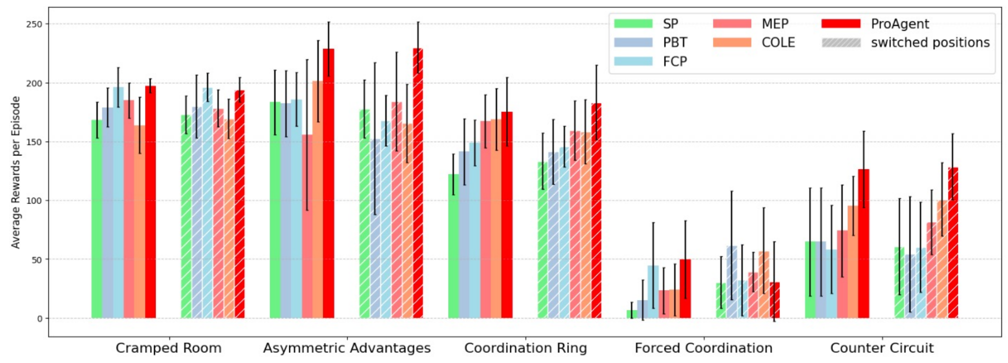


Figure 6: Cross-agent collaborative evaluation: the ZSC performance of ProAgent, COLE, FCP, MEP, PBT, and SP when paired with all the held-out populations. In each layout, the reward bar represents the average performance of one algorithm collaborating with all other algorithms, and the error lines represent the standard deviation. The gray and hashed bars indicate the rewards obtained where the starting positions are switched.

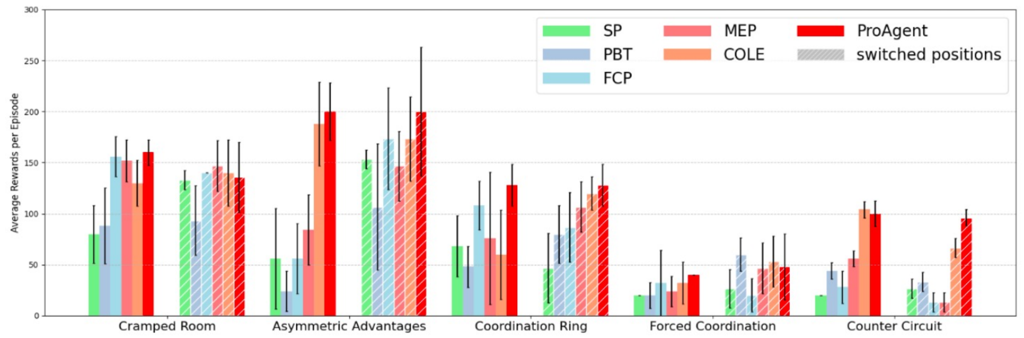


Figure 2: Performance with human proxy partners. In each layout, the reward bar represents the average performance of one algorithm collaborating with the unseen human proxy partners over 400 timesteps on five different random seeds, and the error lines represent the standard error. The hashed bars indicate the rewards obtained where the starting positions are switched. Zoom in for better visualization.

| **Layout** | **Baseline AI Agents** | | | | | **ProAgent** (ours) |
|---|---|---|---|---|---|---|
| | SP | PBT | FCP | MEP | COLE | |
| **Cramped Room** | $168.5 \pm 15.2$ | $178.8 \pm 16.5$ | $196.3 \pm 16.8$ | $185 \pm 15$ | $163.8 \pm 24.1$ | $\mathbf{197.3} \pm 6.1$ |
| | $172.8 \pm 16.1$ | $179.8 \pm 26.8$ | $\mathbf{196} \pm 11.9$ | $178.2 \pm 15.6$ | $169.2 \pm 16.8$ | $194.2 \pm 10.5$ |
| **Asymmetric Advantages** | $183.3 \pm 27.5$ | $182.2 \pm 27.9$ | $185.7 \pm 22.7$ | $155.7 \pm 63.9$ | $201.3 \pm 34.5$ | $\mathbf{228.7} \pm 23$ |
| | $177.8 \pm 24.6$ | $152.3 \pm 64.5$ | $167.8 \pm 21.3$ | $184 \pm 41.8$ | $165.5 \pm 33.3$ | $\mathbf{229.8} \pm 21.9$ |
| **Coordination Ring** | $122 \pm 17.2$ | $141.3 \pm 28$ | $148.8 \pm 19.4$ | $167.2 \pm 22.4$ | $168.8 \pm 26.1$ | $\mathbf{175.3} \pm 29$ |
| | $133.3 \pm 23.7$ | $141.3 \pm 27.5$ | $145.7 \pm 17.1$ | $159.3 \pm 25.3$ | $158.3 \pm 27.1$ | $\mathbf{183} \pm 31.7$ |
| **Forced Coordination** | $6.7 \pm 6.7$ | $15.3 \pm 17.1$ | $44.7 \pm 36.4$ | $23.3 \pm 19.8$ | $24 \pm 21.8$ | $\mathbf{49.7} \pm 33.1$ |
| | $30.2 \pm 21.9$ | $\mathbf{61.7} \pm 46$ | $32.2 \pm 30.2$ | $39.3 \pm 16.9$ | $57.3 \pm 36.4$ | $31 \pm 33.9$ |
| **Counter Circuit** | $64.7 \pm 45.8$ | $64.7 \pm 45.9$ | $58.3 \pm 37.5$ | $74.3 \pm 39.1$ | $95.5 \pm 25.2$ | $\mathbf{126.3} \pm 32.3$ |
| | $60.7 \pm 40.8$ | $54.3 \pm 49.1$ | $60 \pm 38.3$ | $81.5 \pm 27.5$ | $100.8 \pm 31.1$ | $\mathbf{128.5} \pm 28.1$ |

Table 1: Performance for all AI agent pairs. Each column represents the average reward and standard error of one algorithm playing with all others. For each layout, the first row represents the scenario where the agent takes the role of Player 0, and the AI partner takes the role of Player 1. The second row depicts the vice-versa scenario. The best results for each layout are highlighted in bold.