ProAgent: Building Proactive Cooperative Agents with Large Language Models

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

[1]SSE, The Chinese University of Hong Kong, Shenzhen
[2]Institute for Artificial Intelligence, Peking University
[3]Institute of Automation, Chinese Academy of Sciences
[4]ReLER, AAII, University of Technology Sydney
[5]National Key Laboratory of General Artificial Intelligence, BIGAI
ceyaozhang2@link.cuhk.edu.cn, yaodong.yang@pku.edu.cn

Abstract

Building agents with adaptive behavior in cooperative tasks stands as a paramount goal in the realm of multi-agent systems. Current approaches to developing cooperative agents rely primarily on learning-based methods, whose policy generalization depends heavily on the diversity of teammates they interact with during the training phase. Such reliance, however, constrains the agents' capacity for strategic adaptation when cooperating with unfamiliar teammates, which becomes a significant challenge in zero-shot coordination scenarios. To address this challenge, we propose **ProAgent**, a novel framework that harnesses large language models (LLMs) to create *pro*active *agent*s capable of dynamically adapting their behavior to enhance cooperation with teammates. ProAgent can analyze the present state, and infer the intentions of teammates from observations. It then updates its beliefs in alignment with the teammates' subsequent actual behaviors. Moreover, ProAgent exhibits a high degree of modularity and interpretability, making it easily integrated into various of coordination scenarios. Experimental evaluations conducted within the *Overcooked-AI* environment unveil the remarkable performance superiority of ProAgent, outperforming five methods based on self-play and population-based training when cooperating with AI agents. Furthermore, in partnered with human proxy models, its performance exhibits an average improvement exceeding 10% compared to the current state-of-the-art method. For more information about our project, please visit https://pku-proagent.github.io.

Introduction

Large Language Models (LLMs) have rapidly emerged as powerful tools, achieving remarkable advancements across various domains, including long conversations (Ouyang et al. 2022), reasoning (Bubeck et al. 2023), and text generation (Brown et al. 2020). These models, by leveraging a vast amount of training data, can capture and embody a significant amount of common sense knowledge. Notable LLM-based agents like SayCan (Ahn et al. 2022), ReAct (Yao et al. 2023), DEPS (Wang et al. 2023b), RAP (Hao et al. 2023), Reflexion (Shinn et al. 2023), and JARVIS-1 (Wang et al. 2023c) have demonstrated the ability to make decisions interactively through appropriate prompts or feedback. However, these works have primarily focused on exploring the potential of LLMs as individual agents, whether in games or robotics. The untapped potential lies in investigating how LLM-based agents can effectively cooperate with other AI agents or humans.

This research delves into the capabilities of LLMs in tackling the intricate challenges of multi-agent coordination (Yang and Wang 2021; Zhang, Yang, and Başar 2021; Gronauer and Diepold 2022), particularly in the realm of policy generalization (Strouse et al. 2021; Zhao et al. 2023; Li et al. 2023b, 2024). Current approaches (Carroll et al. 2019; Jaderberg et al. 2017; Strouse et al. 2021; Zhao et al. 2023; Li et al. 2023b, 2024) to developing cooperative agents rely primarily on learning-based methods, whose policy generalization depends heavily on the diversity of teammates they interact with during the training phase. Such reliance, however, constrains the agents' capacity for strategic adaptation when cooperating with unfamiliar teammates, which becomes a significant challenge in zero-shot coordination scenarios. We present **ProAgent**, an innovative and adaptable framework specifically designed to excel in coordination scenarios alongside novel agents. ProAgent comprises four essential modules: `Planner`, `Verificator`, `Controller` and `Memory`, along with the mechanism of `Belief Revision`. These modules synergistically enable ProAgent to actively predict teammates' intentions and achieve adaptive cooperative reasoning and planning without the need for prior training or finetuning. To assess the adaptive cooperative capabilities of ProAgent, we conducted performance evaluations using the well-established multi-agent coordination testing suite, *Overcooked-AI* (Carroll et al. 2019). In this environment, two players must work together to maximize their score. The empirical findings

[*] These authors contributed equally.

[†] Work done when Ceyao Zhang visited Peking University.

[‡] Corresponding author

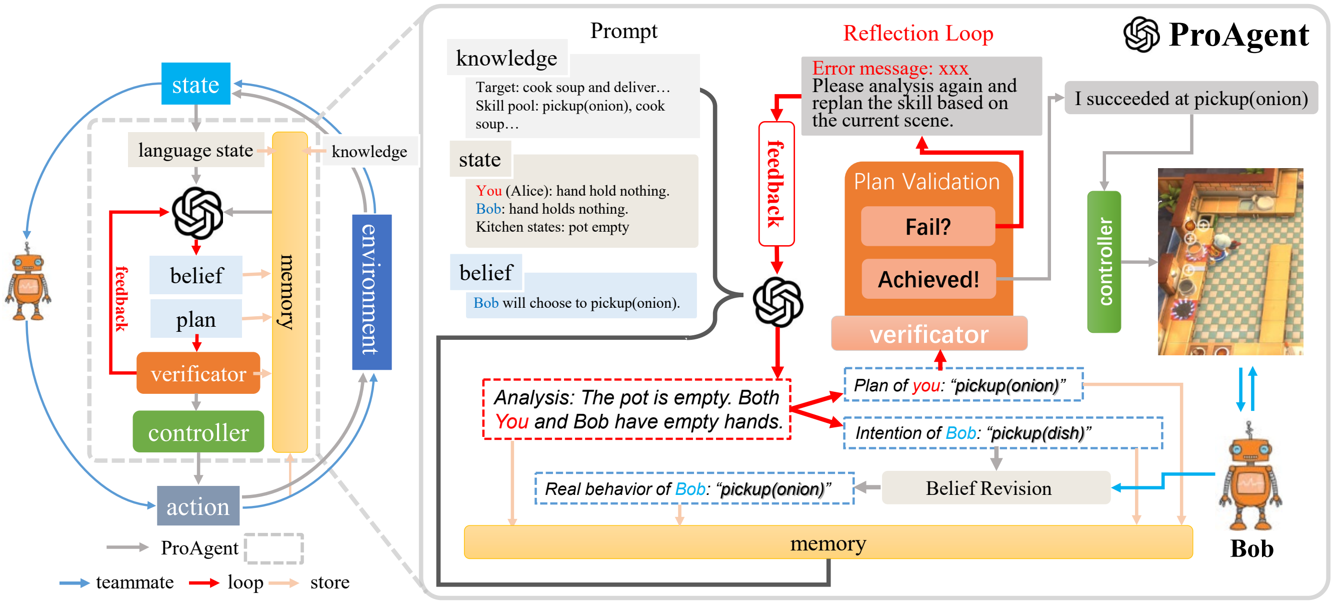

state
language state
knowledge
belief
plan
feedback
verificator
controller
memory
environment
action
ProAgent
teammate
loop
store
Prompt
knowledge
Target: cook soup and deliver…
Skill pool: pickup(onion), cook soup…
state
You (Alice): hand hold nothing.
Bob: hand holds nothing.
Kitchen states: pot empty
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
ProAgent
I succeeded at pickup(onion)
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 Revision
memory
Bob

Figure 1: Overview of our proposed ProAgent framework including the coordination task workflow (left) and inner details of ProAgent pipeline (right). The teammate agent's decision-making loop is formed by the blue solid arrows in the outer circle, while the decision process of the ProAgent is formed by the middle gray dotted box and the outer gray solid arrows. ProAgent commences its operation by translating the initial state into natural language. Then the `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 belief about predicted intention will be updated through the `Belief Revision` mechanism, which involves comparing it with the subsequent actual behavior of the teammate agent. As to the planned skill, the `Verificator` validates whether it can be performed under the current state. In case of skill failure, the `Verificator` will 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 low-level *actions*, to be executed within the environment.

from our evaluations reveal the following key insights: 1) ProAgent demonstrates remarkable proficiency in coordinating with various types of AI teammates across diverse scenarios. 2) ProAgent exhibits a notable preference for collaborating with rational teammates, such as the human proxy, which showcases human-like behavior and suggests its active effort to understand teammates' intentions to enhance cooperation. These results collectively highlight the effectiveness of ProAgent as a cooperative agent across a wide range of scenarios.

In summary, our work makes three key contributions: **Firstly**, we successfully integrate LLMs into the field of cooperative multi-agents and propose the ProAgent framework, which serves as a comprehensive guideline for leveraging the powerful reasoning and planning capabilities of LLMs in cooperative settings. **Secondly**, we demonstrate the remarkable capability of our ProAgent to interpretably analyze the current scene, explicitly infer teammates' intentions, and dynamically adapt its behavior accordingly. This proactive nature empowers ProAgent to actively collaborate with teammates, enabling more efficient cooperative scenarios. **Thirdly**, through a comprehensive series of experiments, we provide compelling evidence of ProAgent's superiority over other agents when engaging in cooperation with diverse types of teammates.

Related Works

Reasoning and Planning with Large Language Models. In the realm of LLMs (Huang and Chang 2023; Mialon et al. 2023; Bubeck et al. 2023), reasoning often entails decomposing intricate queries into sequential intermediate steps, referred to as Chain-of-Thought (CoT; Wei et al. 2022; Kojima et al. 2022), to attain a final solution. Some research focuses on minimizing errors as the number of steps increases (Wang et al. 2023a), while others explore decomposition techniques that break down complex problems into simpler subproblems (Zhou et al. 2023). Recent endeavors have translated LLMs' reasoning capability into planning by constructing a monologue with feedback (Welleck et al. 2023; Shinn et al. 2023; Paul et al. 2023) to facilitate the reasoning and planning process. Notably, the challenge of open-ended long-term planning in the MineDojo environment (Fan et al. 2022) has been addressed by utilizing LLMs as central planners (Wang et al. 2023b,c), thereby demonstrating the extensive capabilities of LLMs-based agents in overcoming complex decision-making tasks. As of the camera-ready version of our paper, the application of LLM-based agents in cooperative games remains little explored. Li et al. (2023a) employs a centralized LLM-based planner for both two players. In contrast, our framework adopts a decentralized planning paradigm and uses one planner for one player. While Zhang et al. (2023) also decentralized planning, their method facilitates cooperation through explicit communication. Our work, on the other hand, fosters cooperation by observing and inferring the intentions of teammates.

Multi-agent Coordination. The goal of multi-agent coordination is to enable multiple autonomous agents to collaborate effectively towards a shared goal (Rashid et al. 2018; Hu and Foerster 2020; Hu et al. 2021a; Yu et al. 2022; Zhong et al. 2023). However, traditional approaches have limitations in fixed task settings and struggle to handle multiple tasks or unseen scenarios. One approach to address this challenge is to enable an agent to learn multiple tasks concurrently (Hu et al. 2021b; Meng et al. 2022; Wen et al. 2022). However, these methods may still limit the agent's cooperation ability in familiar tasks and fail to handle unseen tasks or new agent interactions. Another line of research focuses on zero-shot coordination (ZSC), utilizing Population-Based Training (PBT; Strouse et al. 2021; Zhao et al. 2023; Lupu et al. 2021; Lucas and Allen 2022; Li et al. 2023b, 2024) and Theory of Mind (ToM; Hu et al. 2021a; Wu et al. 2021; Wang et al. 2021) to facilitate adaptive policy development for coordinating with various counterparts without prior coordination experience. However, these ZSC methods demand significant computational resources for data collection and model optimization, and the resulting policies often lack interpretability.

Method

The overview of our ProAgent framework, as is depicted in Fig. 1, involves constant interaction between agents and the environment. The inference pipeline of ProAgent is a hierarchical process that involves multiple interactions between the LLMs and the task at hand. We break down the pipeline into five key stages:

Knowledge Library and State Grouding. The pipeline starts with acquiring *Knowledge Library* specific to the current task and transforming the raw tensor state information into *Language-based State* description that the LLM can effectively comprehend.

High-level Skill Planning. Receiving the aligned language-based state, the LLM-based `Planner` then analyzes the current scene, infers the *intentino* about the teammate agent's intentions, and plans a skill for the current agent.

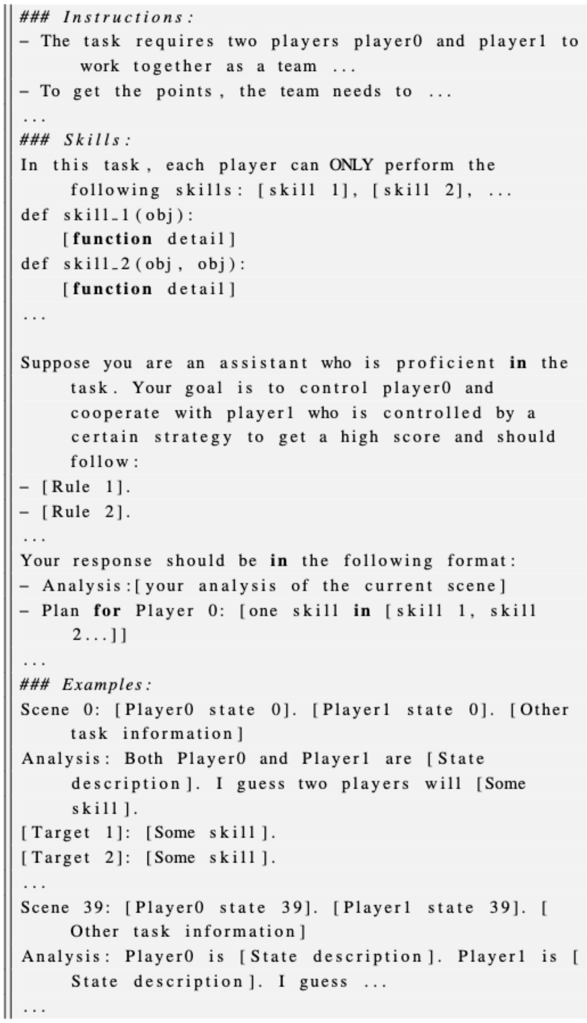
Belief Revision. The belief in the teammate agent's intention is further corrected by the `Belief Revision` mechanism.

Skill Validation and Action Execution. The selected skill will be validated by the `Verificator` and a replan is needed if the current skill fails. If a valid skill is selected, and the `Controller` module decomposes it into low-level actions, allowing ProAgent to effectively interact with the task or environment. The controller can be rule-based, or RL-based methods.

Memory Storage. Throughout the pipeline, all relevant information involved in the prompt, planning process, validation process, and belief revision process is stored in the `Memory` module. This accumulated knowledge helps in making informed decisions and adjusting behavior over time.

Prompt Construction

Knowledge library The planning ability of LLMs is closely related to the prompt at the beginning, which is also the standard practice in automated planning. ProAgent is no exception, and the knowledge library should be fed into LLMs at the initial stage before the cooperation task begins. The main difficulty lies in how to build structured knowledge. In practice, we find that the best combination of knowledge library needs to be described from three perspectives, including `Instructions`, `Skills`, and `Examples`.



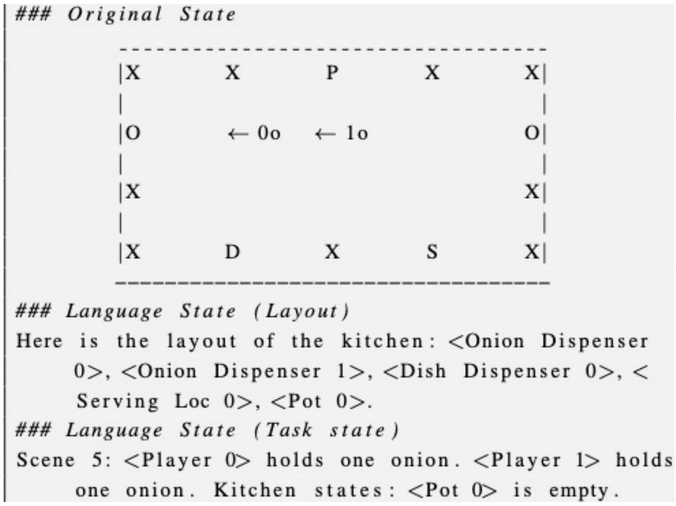
```
### Instructions:
- The task requires two players player0 and player1 to
    work together as a team ...
- To get the points, the team needs to ...
...
### Skills:
In this task, each player can ONLY perform the
    following skills: [skill 1], [skill 2], ...
def skill_1(obj):
    [function detail]
def skill_2(obj, obj):
    [function detail]
...

Suppose you are an assistant who is proficient in the
    task. Your goal is to control player0 and
    cooperate with player1 who is controlled by a
    certain strategy to get a high score and should
    follow:
- [Rule 1].
- [Rule 2].
...
Your response should be in the following format:
- Analysis:[your analysis of the current scene]
- Plan for Player 0: [one skill in [skill 1, skill
    2...]]
...
### Examples:
Scene 0: [Player0 state 0]. [Player1 state 0]. [Other
    task information]
Analysis: Both Player0 and Player1 are [State
    description]. I guess two players will [Some
    skill].
[Target 1]: [Some skill].
[Target 2]: [Some skill].
...
Scene 39: [Player0 state 39]. [Player1 state 39]. [
    Other task information]
Analysis: Player0 is [State description]. Player1 is [
    State description]. I guess ...
...
```

Figure 2: A template to construct the knowledge library.

As shown in Fig 2, `Instructions` are for LLMs to understand the objective of the task and information about other cooperative agents. `Skills` is designed to regulate the planning pattern of the LLM, defining which skills are legal and which are not. We also enforce the format of LLMs' responses to follow the CoT: output analysis and then plan according to the analysis instead of directly outputting a plan. `Examples` is an optional component of the three. Its main functionality is to provide real cases for LLMs to strengthen their memories and behave following the regulations set by `Skills`. Normally, `Examples` should contain a scene description followed by an analysis and the desired behavior, such as the selected skill. With these three parts, LLMs can understand the task and what is expected of them in the subsequent planning and reasoning stages.

Grounding tensor state to language-based state To facilitate interaction between LLMs and the environment, it is essential to establish a bridge between the original symbolic state provided by the environment and the language-based state for LLMs. In most scenarios, the raw state is not directly applicable to LLMs' usage. Hence, finding an effective alignment between the original symbolic state and the language-based state is crucial to enhancing LLMs' accurate understanding of the current situation. To illustrate this, we present a simplified example based on the *Overcooked-AI* environment, demonstrating how the state can be transformed into language within our ProAgent framework. With the knowledge library and initial state information prepared, ProAgent is equipped to tackle the cooperative task alongside its teammates. This marks the transition to the subsequent stage, where ProAgent engages in reasoning and planning, progressing step by step to achieve its objectives. An illustrative instance can be found in Fig 3.



```
### Original State
        ---------------------------------
        |X      X       P       X      X|
        |                               |
        |O      ← 0o    ← 1o           O|
        |                               |
        |X                             X|
        |                               |
        |X      D       X       S      X|
        ---------------------------------
### Language State (Layout)
Here is the layout of the kitchen: <Onion Dispenser
    0>, <Onion Dispenser 1>, <Dish Dispenser 0>, <
    Serving Loc 0>, <Pot 0>.
### Language State (Task state)
Scene 5: <Player 0> holds one onion. <Player 1> holds
    one onion. Kitchen states: <Pot 0> is empty.
```

Figure 3: Grounding the tensor state to language-based state.

Cooperative Reasoning and Planning

ProAgent is a specialized system tailored for cooperative tasks, where information from teammate agents plays a pivotal role in the coordination process. Existing works mainly utilize information in two ways: firstly, through explicit incorporation, involving communication and exchange of information before decision-making; secondly, through implicit modeling of teammate agents to facilitate cooperative learning. Each approach comes with its own set of advantages and disadvantages concerning cooperative reasoning and planning: The integration of teammate information can be achieved efficiently by sending teammate agent information to LLMs. However, this approach may jeopardize the overall generalization of ProAgent's reasoning capabilities. On the other hand, modeling the teammate agent offers a more flexible approach, while the modeling process is inherently unstable as the teammate agent's strategy may continuously evolve, demanding additional resources for maintenance.

In order to strike a balance between the generalization ability of built agents and the efficiency of incorporating teammate information, particularly for LLMs that possess excellent reasoning capabilities but face challenges in fine-tuning or learning extra belief modules, ProAgent introduces three core components along with a cooperative reasoning and planning mechanism. The three modules encompass: 1) The `Memory` module, which stores information about task trajectory and general knowledge in the task domain. 2) The `Verificator` module, consisting of one component for skill failure analysis and another for transforming skills into atomic actions. 3) The `Controller` module, dedicated to the transformation of skills into atomic actions. To further align the LLMs' belief regarding the teammate agent's intentions with actual behavior, and thereby continually enhance prediction accuracy, ProAgent implements the `Belief Revision` mechanism. This process effectively strengthens the LLMs' beliefs, leading to improved cooperative reasoning and planning.

Memory Module: Leveraging History for Cooperative Behavior In ProAgent, the `Memory` module plays a crucial role in supporting information storage and retrieval processes. It consists of two components: `Knowledge Library` and `Trajectory`. The `Knowledge Library` acts as a persistent repository, retaining a comprehensive record of the task, including its layout, rules, and demonstrations throughout game play sessions. On the other hand, the `Trajectory` component serves as a python list. It stores essential information, such as the latest `Language-based State`, `Analysis`, `Belief` of teammates' intentions, and the `Skill` used. When needed, only specific parts of the `Memory` are retrieved, depending on the chosen strategy, such as the `recent-K` strategy[1] or `relevent-K` strategy[2]. Those strategies focus on the immediate context, facilitating efficient decision-making and planning during ongoing interactions. Overall, the `Memory` module significantly enhances ProAgent's capacity to access pertinent information and cooperate efficiently with teammate agents. By leveraging past experiences and learning from historical data, the `Memory`

[1] only retrieve the K most recent trajectories.

[2] retrieve the most relevent K trajectories based on their embedding similarity.

module empowers ProAgent to make informed decisions during cooperation tasks.

Planner Module: Reasoning with Chain of Thought With the history information and current state description ready, ProAgent utilizes the strong reasoning ability of LLMs to make decisions in the current situation. The `Planner` module, which follows the Chain of Thought (CoT) approach commonly used in LLMs' reasoning and planning work (Yao et al. 2023; Hao et al. 2023; Shinn et al. 2023). Instead of directly outputting a plan, the `Planner` module makes the final decision step by step. The provided information is first thoroughly analyzed, and the intention of the teammate agent's plan for the current step is predicted. Based on this `Analysis` and the `Belief` about the teammate agent, LLMs formulate a plan that ensures it is the most reasonable and effective strategy for the given situation. In the experiment part, we design three level prompts (L1: directly planning without analysis and intention; L2: with analysis but no intention; L3: with both analysis and intention) and conduct an ablation study to assess how this design enhances ProAgent's performance in a cooperative scenario.

Verificator Module: Analyzing Skill Failures With Multi-rounds Prompts In the cooperative setting, the `Verificator` module plays a crucial role in scrutinizing and identifying any unreasonable or flawed planning generated by the LLMs. Its primary function involves analyzing the underlying reasons for these inadequacies and providing valuable insights and suggestions for improvement. In the ProAgent framework, this process entails conducting a thorough investigation through multiple rounds of prompt and response between the agent and the LLMs.

To illustrate this process, we first employ a three-round prompt and response approach, including `Preconditions Check`, `Double-check` and `Error Conclusion`. It's important to note that the number of rounds or the specific interaction style is not restricted, and we found in experiments that usually one round prompt is enough.

The `Preconditions Check` involves signaling the LLMs if the current plan is illegal due to internal checks before its actual execution. A robust internal checking mechanism can prevent failures when the LLMs haven't fully understood the consequences of their chosen skill under the current state. In the *Overcooked-AI* example, we design the condition check prompt by leveraging both the current scene and the failed skill as inputs. We employ a trigger prompt to enable the LLMs to individually verify each precondition of the skill and pinpoint the specific one that led to the failure. To aid in solving multi-step reasoning problems, prompting techniques like CoT are also adopted. An instance of the trigger prompt in Overcooked-AI could be: "*Analysis of why I cannot execute this skill in the current scene step by step.*" or just "*Why did Player 0 fail ?*" The preconditions of each skill can be expressed either in natural language or in pseudo-code form, which can be more effective as proposed in previous works (Liang et al. 2023; Singh et al. 2023).

Belief revision: Rectifying Belief on Teammate Agents The `Belief Revision` mechanism plays a pivotal role in rectifying any incorrect beliefs during cooperation. ProAgent makes predictions about their teammates' future behavior and stores relevant analyses in their memory. At the beginning, the observed behavior of the teammate agent may deviates from the assumed intentions recorded in `Memory`. In subsequent steps, ProAgent verifies the accuracy of their predictions and corrects any erroneous beliefs. Specifically, the `Belief Revision` mechanism works to remember all past infered intentions and really behaviors. The really behaviors enforces ProAgent to learn from ground truth, which help it to revise the wrong belief, thereby avoiding similar mistakes in the future. When we use L3 level prompts with revision, those information will be added into the prompts for next query. This iterative process allows ProAgent to refine their beliefs over time and enhance their ability to make accurate predictions about their teammate's intentions. In summary, the `Belief Revision` mechanism ensures that ProAgent maintains accurate and up-to-date information about their teammate agent's real behavior. By referencing the `Belief` part of `Memory` before making decisions, ProAgent continually improves the accuracy of their beliefs regarding their teammate's future behavior.

Controller Module: Grounding High-Level Skills to Low-Level Actions Based on the modules and mechanisms discussed above, ProAgent effectively engages in cooperative reasoning and plans a high-level skill. However, it is worth noting that there is a gap between the skill space and the environment's action space. Therefore, we also need a `Controller` module which is imperative, aiming to convert language-based skills into low-level actions that can be executed in the environment. Although this transformation process is closely tied to the specific task at hand, making the `Controller` module highly flexible, it necessitates the establishment of fixed rules capable of decomposing the skill into multiple steps of low-level actions and providing a feedback signal to the reasoning component once the action is fully executed. The controller can be a rule-based path search algorithm or a policy trained by language-grounded reinforcement learning (Hanjie, Zhong, and Narasimhan 2021; Ding et al. 2023; Hu and Sadigh 2023; Du et al. 2023) methods. Considering that the controller is not our main concern, we choose the built-in controller in the Overcooked-AI environment , which is implemented based on the search strategy. On this basis, we made small improvements so that when a road blockage is found, a new path will be searched again. A better controller can definitely reach better performance.

Experiments

Experimental Settings

Following previous works on cooperative AI and human-AI cooperation, we choose Overcooked-AI as our test environment, in which two agents swiftly prepare and serve soups by placing up to three ingredients in a pot, cooking the soup, filling the soup with the dish, and delivering

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

the soup. Agents must dynamically allocate tasks and cooperate effectively. Five classical layouts are used: *Cramped Room*, *Asymmetric Advantages*, *Forced Coordination*, *Coordination Ring*, and *Counter Circuit*. A detailed description of each layout can be found in the appendix.

Our primary concern behind this work is how well the agents developed so far based on ZSC methods can cooperate with diverse teammates, ranging from different AI agents to humans. In previous works on Overcooked-AI, the cooperative performance of an agent is often evaluated with two held-out populations: self-play (SP) agent and human proxy model. We conduct a comparative analysis between our proposed ProAgent and five alternatives prevalent in the field including SP (Tesauro 1994; Carroll et al. 2019), PBT (Jaderberg et al. 2017), FCP (Strouse et al. 2021), MEP (Zhao et al. 2023), and COLE (Li et al. 2023b, 2024). We combined the above six algorithms in pairs to construct 36 pairs. For example, we choose the SP algorithm as player 0 and the PBT algorithm as player 1, and these two algorithms can form an agent pair (SP, PBT). Since the two players are not all homogeneous, we will also form a (PBT, SP) algorithm pair. For each algorithm pair, we ran five episodes and collected the mean and standard variation of the episode returns. Besides, we also select the human proxy model proposed by (Carroll et al. 2019) to test the agent's ability to cooperate with humans. In the main experiments, we use L2 level prompts and recent-1 strategy.

Collaborating with AI Agents

Quantitative Results Table 1 illustrates the average performance of SP, PBT, FCP, MEP, COLE, and ProAgent when paired with all the 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 results indicate that ProAgent outperforms the baselines in all layouts when acting as Playe 0. Taking the role of Player 1, ProAgent only slightly underperforms FCP in cramped room layout and loses to PBT in forced coordination layout. We will examine this failure further in the appendix. In previous studies, it is rare to compare different AI agent combinations with each other, and our experimental results also reveal that none of the other ZSC methods is consistently better than other methods. Considering that ProAgent requires no specific training with distinct teammates and in distinct layouts, it presents a stronger adaptive ability than the other AI agents. These results show our LLM-based agent is a better cooperator.

Qualitative Results To gain deeper insights into the fundamental components of effective cooperation, we perform a qualitative examination of our ProAgent's behaviors exhibited during our experiments, leading us to identify several cooperative behaviors.

ProAgent excels in making strategic plans. For example, when pot one is cooking and pot two lacks an onion, we observed that ProAgent would prioritize putting one onion into pot two. After this, the agent will fetch the plate. At the same time, cooking can be completed in the first pot, and this agent with a plate can directly fill the plate with soup. This process is very effective. Besides, after making a failure plan, ProAgent can promptly recognize this failure, and make a new and often better plan.

ProAgent demonstrates a remarkable capacity to dynamically adjust low-level actions while executing high-level plans. For instance, when ProAgent intends to deposit an onion into a pot, it's underlying `Controller` identifies a blocked path caused by its teammate. Swiftly, the `Controller` will identify an alternative interconnected route, skillfully bypassing any potential obstructions. This adaptive strategy enables ProAgent to discover unhindered pathways. Moreover, when `Planner` has no clear goal, the `Controller` will move randomly. This dynamic operation helps ProAgent to break the deadlock caused by other AI agents due to conventions formed during the training phase.

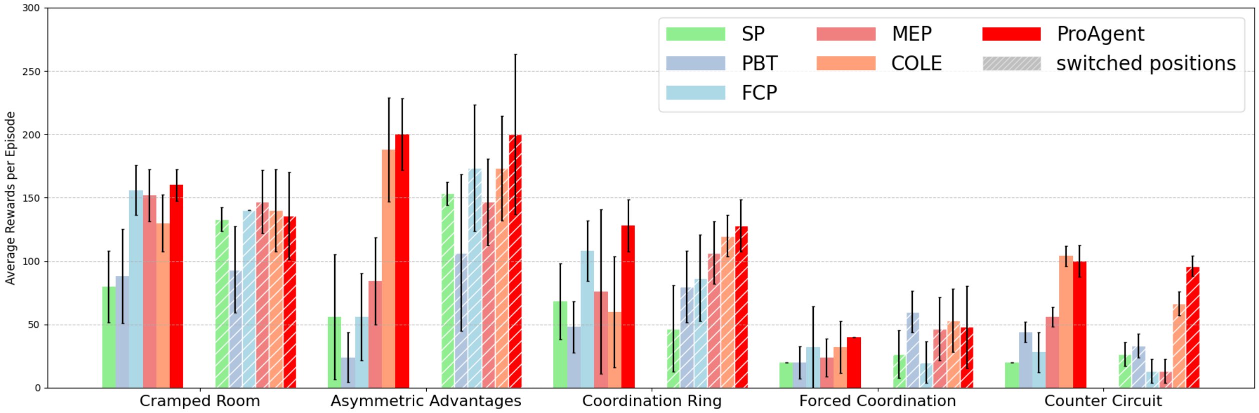


Figure 4: Performance with human proxy partners. In each layout, the reward bar represents the average performance of one algorithm collaborating with the unseen human proxy partners over 400 timesteps on five BC models, and the error lines represent the standard error. The hashed bars indicate the rewards obtained where the starting positions are switched.

Collaborating with Humans

Apart from cooperation with AI agents, our concern also involves the generalization to human partners. Due to the limitation of collecting human interaction data, we follow the previous work (Carroll et al. 2019) that uses a behavior cloning (BC) model trained on human data as a proxy of humans. Fig. 4 presents the average cumulative rewards achieved for 400 timesteps by ProAgent when engaged in collaboration with BC. The reported outcomes encompass both the mean value and standard error across five distinct BC models. Analysis of the experimental findings reveals that across the five environments, ProAgent outperforms the baseline in four environments, exhibiting particularly noteworthy superiority when functioning as Player 0 in the context of *Forced Coordination*. Notably, the positioning discrepancy between the left and right starting positions had a negligible impact on ProAgent's performance. However, this difference led to substantial performance disparities among the baselines, particularly in asymmetric layouts, where the cumulative rewards achieved by all baselines were superior in the left position compared to the right position, consistent with the findings in COLE (Li et al. 2023b, 2024).

Discussion

Does analysis and belief help in better planning? To gauge the influence of *analysis* and *intention* on the accuracy and efficiency of decisions made by the `Planner` Module, we conducted an ablation study within the context of the *Cramped Room* layout. The experiment considered three distinct conditions and their respective scores were: 1) 204 for L3 level prompts (with both *analysis* and *intention*), 2) 184 for L2 level prompts (with *analysis* but no *intention*), and 3) 100 for L1 level prompts (making a skill plan directly, neither *analysis* nor *intention*). We believe that the significance of analysis in the `Planner` Module lies in its provision of in-context for final planning just as CoT will improve the effect of reasoning. Additionally, inferring teammate intentions provides further improvements.

Is Verificator effective in feedback-based reasoning? Upon removing the `Verificator` Module and allowing ProAgent to engage in planning without feedback, we computed success rates over 100 steps. Notably, the success rate dropped significantly to 20%, underscoring the critical role of our `Verificator` Module in furnishing feedback when the `Planner` Module generates inaccurate plans.

Conclusion

In this work, we propose **ProAgent**, a proactive LLM-based agent framework, with the primary objective of addressing the multi-agent coordination predicament. By leveraging the inherent faculties of LLMs encompassing common sense comprehension and language-centric task understanding, coupled with explicit mechanisms for reasoning and planning, ProAgent demonstrates remarkable performance within various coordination scenarios. Experiments on cooperating with both AI agents and human proxies in the Overcooked-AI demonstrate the effectiveness of ProAgent over state-of-the-art methods. Moreover, ProAgent's reasoning and planning are based on natural language, which is interpretable and friendly to humans. These encouraging results pave the way for further advancements in both cooperative multi-agent and human-compatible AI systems built upon LLMs.

Acknowledgements

This work is sponsored by the National Natural Science Foundation of China (62376013), by the Basic Research Project No. HZQB-KCZYZ-2021067 of Hetao Shenzhen-HK S&T Cooperation Zone, Beijing Municipal Science & Technology Commission (Z231100007423015), by the Shenzhen Outstanding Talents Training Fund 202002, by the Guangdong Research Projects No. 2017ZT07X152 and No. 2019CX01X104, by the Guangdong Provincial

Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001), by the Shenzhen Key Laboratory of Big Data and Artificial Intelligence (Grant No. ZDSYS201707251409055), by the NSFC under Grant No. 62271433, and by Shenzhen Science and Technology Program under Grant No. JCYJ20220530143806016 and No. RCJC20210609104448114.

References

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Ho, D.; Hsu, J.; Ibarz, J.; Ichter, B.; Irpan, A.; Jang, E.; Ruano, R. J.; Jeffrey, K.; Jesmonth, S.; Joshi, N. J.; Julian, R.; Kalashnikov, D.; Kuang, Y.; Lee, K.-H.; Levine, S.; Lu, Y.; Luu, L.; Parada, C.; Pastor, P.; Quiambao, J.; Rao, K.; Rettinghouse, J.; Reyes, D.; Sermanet, P.; Sievers, N.; Tan, C.; Toshev, A.; Vanhoucke, V.; Xia, F.; Xiao, T.; Xu, P.; Xu, S.; Yan, M.; and Zeng, A. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. arXiv:2204.01691.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language Models are Few-Shot Learners. In *Advances in neural information processing systems*, volume 33, 1877–1901.

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; Nori, H.; Palangi, H.; Ribeiro, M. T.; and Zhang, Y. 2023. Sparks of Artificial General Intelligence: Early Experiments with GPT-4. arXiv:2303.12712.

Carroll, M.; Shah, R.; Ho, M. K.; Griffiths, T.; Seshia, S.; Abbeel, P.; and Dragan, A. 2019. On the Utility of Learning about Humans for Human-AI Coordination. In *Advances in neural information processing systems*, volume 32.

Ding, Z.; Zhang, W.; Yue, J.; Wang, X.; Huang, T.; and Lu, Z. 2023. Entity Divider with Language Grounding in Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*, 8103–8119. PMLR.

Du, Y.; Watkins, O.; Wang, Z.; Colas, C.; Darrell, T.; Abbeel, P.; Gupta, A.; and Andreas, J. 2023. Guiding Pretraining in Reinforcement Learning with Large Language Models. arXiv:2302.06692.

Fan, L.; Wang, G.; Jiang, Y.; Mandlekar, A.; Yang, Y.; Zhu, H.; Tang, A.; Huang, D.-A.; Zhu, Y.; and Anandkumar, A. 2022. MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge. In *NIPS Processing Systems Datasets and Benchmarks Track*.

Gronauer, S.; and Diepold, K. 2022. Multi-Agent Deep Reinforcement Learning: A survey. *Artificial Intelligence Review*, 1–49.

Hanjie, A. W.; Zhong, V. Y.; and Narasimhan, K. 2021. Grounding Language to Entities and Dynamics for Generalization in Reinforcement Learning. In *International Conference on Machine Learning*, 4051–4062. PMLR.

Hao, S.; Gu, Y.; Ma, H.; Hong, J. J.; Wang, Z.; Wang, D. Z.; and Hu, Z. 2023. Reasoning with Language Model is Planning with World Model. arXiv:2305.14992.

Hu, H.; and Foerster, J. N. 2020. Simplified Action Decoder for Deep Multi-Agent Reinforcement Learning. In *International Conference on Learning Representations*.

Hu, H.; Lerer, A.; Cui, B.; Pineda, L.; Brown, N.; and Foerster, J. 2021a. Off-Belief Learning. In *International Conference on Machine Learning*, 4369–4379. PMLR.

Hu, H.; and Sadigh, D. 2023. Language Instructed Reinforcement Learning for Human-AI Coordination. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR.

Hu, S.; Zhu, F.; Chang, X.; and Liang, X. 2021b. UPDeT: Universal Multi-agent Reinforcement Learning via Policy Decoupling with Transformers. arXiv:2101.08001.

Huang, J.; and Chang, K. C.-C. 2023. Towards Reasoning in Large Language Models: A Survey. arXiv:2212.10403.

Jaderberg, M.; Dalibard, V.; Osindero, S.; Czarnecki, W. M.; Donahue, J.; Razavi, A.; Vinyals, O.; Green, T.; Dunning, I.; Simonyan, K.; Fernando, C.; and Kavukcuoglu, K. 2017. Population Based Training of Neural Networks. arXiv:1711.09846.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large Language Models are Zero-Shot Reasoners. In *Advances in neural information processing systems*, volume 35, 22199–22213.

Li, W.; Qiao, D.; Wang, B.; Wang, X.; Jin, B.; and Zha, H. 2023a. Semantically Aligned Task Decomposition in Multi-Agent Reinforcement Learning. arXiv:2305.10865.

Li, Y.; Zhang, S.; Sun, J.; Du, Y.; Wen, Y.; Wang, X.; and Pan, W. 2023b. Cooperative Open-ended Learning Framework for Zero-shot Coordination. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR.

Li, Y.; Zhang, S.; Sun, J.; Zhang, W.; Du, Y.; Wen, Y.; Wang, X.; and Pan, W. 2024. Tackling Cooperative Incompatibility for Zero-Shot Human-AI Coordination. arXiv:2306.03034.

Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; and Zeng, A. 2023. Code as Policies: Language Model Programs for Embodied Control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 9493–9500. IEEE.

Lucas, K.; and Allen, R. E. 2022. Any-Play: An Intrinsic Augmentation for Zero-Shot Coordination. In *International Foundation for Autonomous Agents and Multiagent Systems*, 853–861.

Lupu, A.; Cui, B.; Hu, H.; and Foerster, J. 2021. Trajectory Diversity for Zero-Shot Coordination. In *International conference on machine learning*, 7204–7213. PMLR.

Meng, L.; Wen, M.; Yang, Y.; Le, C.; Li, X.; Zhang, W.; Wen, Y.; Zhang, H.; Wang, J.; and Xu, B. 2022. Offline Pre-trained Multi-Agent Decision Transformer: One Big Sequence Model Tackles All SMAC Tasks. arXiv:2112.02845.

Mialon, G.; Dessì, R.; Lomeli, M.; Nalmpantis, C.; Pasunuru, R.; Raileanu, R.; Rozière, B.; Schick, T.; Dwivedi-Yu, J.; Celikyilmaz, A.; Grave, E.; LeCun, Y.; and Scialom, T. 2023. Augmented Language Models: a Survey. arXiv:2302.07842.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P. F.; Leike, J.; and Lowe, R. 2022. Training Language Models to Follow Instructions with Human Feedback. In *Advances in Neural Information Processing Systems*, volume 35, 27730–27744.

Paul, D.; Ismayilzada, M.; Peyrard, M.; Borges, B.; Bosselut, A.; West, R.; and Faltings, B. 2023. REFINER: Reasoning Feedback on Intermediate Representations. arXiv:2304.01904.

Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*, 4295–4304. PMLR.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K. R.; and Yao, S. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; and Garg, A. 2023. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 11523–11530. IEEE.

Strouse, D.; McKee, K.; Botvinick, M.; Hughes, E.; and Everett, R. 2021. Collaborating with Humans without Human Data. In *Advances in Neural Information Processing Systems*, volume 34, 14502–14515.

Tesauro, G. 1994. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2): 215–219.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q. V.; Chi, E. H.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023a. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.

Wang, Y.; Zhong, F.; Xu, J.; and Wang, Y. 2021. ToM2C: Target-oriented Multi-agent Communication and Cooperation with Theory of Mind. In *International Conference on Learning Representations*.

Wang, Z.; Cai, S.; Chen, G.; Liu, A.; Ma, X.; and Liang, Y. 2023b. Describe, Explain, Plan and Select: Interactive Planning with LLMs Enables Open-World Multi-Task Agents. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Wang, Z.; Cai, S.; Liu, A.; Jin, Y.; Hou, J.; Zhang, B.; Lin, H.; He, Z.; Zheng, Z.; Yang, Y.; Ma, X.; and Liang, Y. 2023c. JARVIS-1: Open-World Multi-Task Agents with Memory-Augmented Multimodal Language Models. arXiv:2311.05997.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; brian ichter; Xia, F.; Chi, E. H.; Le, Q. V.; and Zhou, D. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, 24824–24837.

Welleck, S.; Lu, X.; West, P.; Brahman, F.; Shen, T.; Khashabi, D.; and Choi, Y. 2023. Generating Sequences by Learning to Self-Correct. In *The Eleventh International Conference on Learning Representations*.

Wen, M.; Kuba, J.; Lin, R.; Zhang, W.; Wen, Y.; Wang, J.; and Yang, Y. 2022. Multi-Agent Reinforcement Learning is a Sequence Modeling Problem. In *Advances in Neural Information Processing Systems*, volume 35, 16509–16521.

Wu, S. A.; Wang, R. E.; Evans, J. A.; Tenenbaum, J. B.; Parkes, D. C.; and Kleiman-Weiner, M. 2021. Too Many Cooks: Bayesian Inference for Coordinating Multi-Agent Collaboration. *Topics in Cognitive Science*, 13(2): 414–432.

Yang, Y.; and Wang, J. 2021. An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective. arXiv:2011.00583.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K. R.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*.

Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; and Wu, Y. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems*, volume 35, 24611–24624.

Zhang, H.; Du, W.; Shan, J.; Zhou, Q.; Du, Y.; Tenenbaum, J. B.; Shu, T.; and Gan, C. 2023. Building Cooperative Embodied Agents Modularly with Large Language Models. arXiv:2307.02485.

Zhang, K.; Yang, Z.; and Başar, T. 2021. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. *Handbook of reinforcement learning and control*, 321–384.

Zhao, R.; Song, J.; Yuan, Y.; Hu, H.; Gao, Y.; Wu, Y.; Sun, Z.; and Yang, W. 2023. Maximum Entropy Population Based Training for Zero-Shot Human-AI Coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 5, 6145–6153.

Zhong, Y.; Kuba, J. G.; Feng, X.; Hu, S.; Ji, J.; and Yang, Y. 2023. Heterogeneous-Agent Reinforcement Learning. arXiv:2304.09870.

Zhou, D.; Schärli, N.; Hou, L.; Wei, J.; Scales, N.; Wang, X.; Schuurmans, D.; Cui, C.; Bousquet, O.; Le, Q. V.; and Chi, E. H. 2023. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *The Eleventh International Conference on Learning Representations*.