Benchmark Real-time Adaptation and Communication Capabilities of Embodied Agent in Collaborative Scenarios

Shipeng Liu†§‡, Boshen Zhang†∗, Zhehui Huang∗
† Equal Contribution ‡ Corresponding Author
§ Ming Hsieh Department of Electrical and Computer Engineering
∗ Thomas Lord Department of Computer Science
University of Southern California
Los Angeles, California, USA
{shipengl,boshenzh,zhehuihu}@usc.edu

Abstract

Advancements in Large Language Models (LLMs) have opened transformative possibilities for human-robot interaction, especially in collaborative environments. However, Real-time human-AI collaboration requires agents to adapt to unseen human behaviors while maintaining effective communication dynamically. Existing benchmarks fall short in evaluating such adaptability for embodied agents, focusing mostly on the task performance of the agent itself. To address this gap, we propose a novel benchmark that assesses agents' reactive adaptability and instantaneous communication capabilities at every step. Based on this benchmark, we propose a Monitor-then-Adapt framework (MonTA), combining strong adaptability and communication with real-time execution. MonTA contains three key LLM modules, a lightweight *Monitor* for monitoring the need for adaptation in high frequency, and two proficient *Adapters* for subtask and path adaptation reasoning in low frequency. Our results demonstrate that MonTA outperforms other baseline agents on our proposed benchmark. Further user studies confirm the high reasonability adaptation plan and consistent language instruction provided by our framework.

Introduction

Embodied agents powered by Large Language Models (LLMs) show great potential for interpreting human instructions (Bubeck et al. 2023; Ouyang et al. 2022; Liu et al. 2023a) and performing tasks through sequential actions in diverse environments (Zhang et al. 2023a; 2024; Agashe, Fan, and Wang 2023; Liu et al. 2023a). To evaluate embodied agents effectively, their capabilities can be assessed from multiple perspectives, including language instruction interpretation, subtask decomposition, action sequence generation, etc (Li et al. 2024). In highly cooperative scenarios (Liu et al. 2023b; 2024; Carroll et al. 2019) where agents must interact with humans at every step, real-time adaptation on both subtask and action sequence becomes especially crucial (Weber, Mateas, and Jhala 2011).

Current LLM-powered agents (Zhang et al. 2023a; 2024; Agashe, Fan, and Wang 2023; Liu et al. 2023a) still rely on scripted-based policy or RL-based controller to achieve atomic action, which makes them often struggle to achieve reactive, real-time adaptation when collaborating with true humans, highlighting the need to evaluate and improve this capability for better performance in dynamic, collaborative scenarios.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

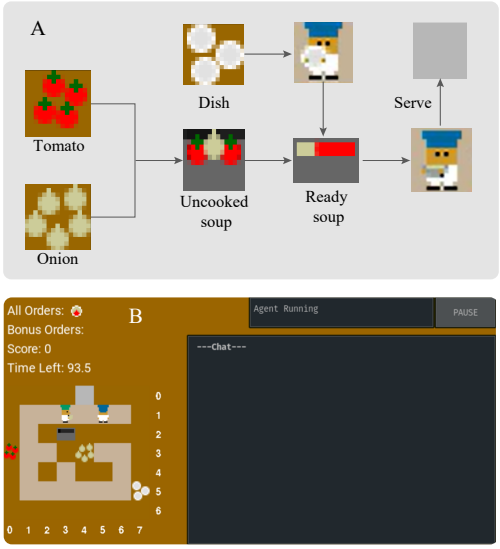


Figure 1: (A) The cooking procedure to finish one order. (B) The game interface that we use to test agents and conduct user studies.

The existing benchmarks (Agashe et al. 2024; Cui et al. 2021), such as Overcooked-AI (Carroll et al. 2019), designed for highly cooperative multi-agent scenarios, lack modularity and fail to provide metrics for evaluating both real-time adaptability and dynamic communication efficiency of LLM-powered agents. Specifically, the following limitations hinder the effectiveness.

- **Lack of layouts required frequent adaptation**: Current benchmarks do not require agents to frequently adapt strategies and coordinate sequential tasks due to conflicting spaces, limiting their ability to assess dynamic teamwork scenarios.
- **No explicitly real-time adaptation evaluation**: The way of accessing the performance using the overall game score

is straightforward yet not clear. The game score can be influenced by agents' capabilities from different perspectives, such as subtask reasoning, adaptation, and human goal integration.

- **No Communication efficiency evaluation**: For LLM-based agents, communication is a critical capability and advantage to achieve seamless human-agent collaboration.

To better evaluate LLM agents in cooperative scenarios, we introduce an enhanced Overcooked-AI (Carroll et al. 2019) benchmark, which is specifically designed to assess the capability of both reactive adaptation and the effective communication of LLM-based coordination agents. Furthermore, we also introduce **MonTA**, a framework designed to enable embodied agents to execute real-time adaptation by combining fast monitoring and slow adaptation.

To summarize, our key contributions include:

- We proposed a fine-grained benchmark to evaluate LLM-powered agents' adaptability and communication capabilities, focusing on dynamic real-time interactions and cooperative scenarios.
- We developed **MonTA**, a modular and adaptive framework that integrates fast monitoring and deliberate adaptation to enable LLM agents to perform reactive, real-time adaptations in highly cooperative environments.
- We conducted thorough experiments and user studies on our benchmarks, using both MonTA and other frameworks with various LLMs, which demonstrate the benchmark's effectiveness in evaluating real-time agent adaptability.
- Further comparison and analysis confirm MonTA's superior adaptability and seamless collaboration in dynamic human-agent cooperation.

Related Work

Embodied Agent Benchmark

Numerous studies have proposed benchmarks for embodied multi-agent systems. (Zhang et al. 2023b; Agashe, Fan, and Wang 2023; Chang et al. 2024; Puig et al. 2023; Chang et al. 2024). Several language-based benchmarks, such as (Das et al. 2018; Majumdar et al. 2024), have focused on question answering, which emphasizes information gathering but does not consider the physical interaction made by embodied agents. Notably, Li et al. (2024) introduced the Embodied Agent Interface, a modular framework for evaluating embodied decision-making processes by considering factors beyond overall task performance. These benchmarks usually focus on assessing overall performance, limiting their ability to evaluate agents' adaptability and communication ability during collaboration. This work focuses on evaluating agents' reactive and proactive ability in language instruction during interaction. This work uniquely benchmarks real-time adaptability within collaborative scenarios.

Real-time Human-AI Collaboration

Human-AI collaboration has been a long-standing challenge. Prior works study human-AI cooperation in games such as Hanabi (Agashe et al. 2024; Cui et al. 2021), diplomacy ((FAIR)† et al. 2022), and overcooked (Carroll et al. 2019; Fontaine et al. 2021; Strouse et al. 2021). Several studies also leverage LLMs for decision-making tasks in the game of overcooked. For instance, Zhang et al. (2023a) use LLMs to infer other agents' intentions and plan subtasks, while Zhang et al. (2024) explore long-horizon inference for improved multi-agent cooperation. Recent work, such as (Liu et al. 2023a), proposed an LLM-based hierarchical framework to follow human high-level instruction, but still lack adaptability during execution of assigned subtask. Prior work primarily focuses on agents with high-level task planning and often lacks low-level adaptability. Our work differs by enabling agents' real-time proactive adaptation at the atomic action level.

An Enhanced Overcooked-AI Benchmark for Embodied Agents

To thoroughly evaluate the real-time adaptability and performance of different AI agents, we extended the original Overcooked benchmark and designed different modular tests. Specifically, we first constructed 22 layouts with varying complexity and coordination demands, following the teaming fluency metrics discussed in Hoffman (2019) and Nikolaidis (2024). Secondly, we designed a communication panel(Figure 1B) to test the effectiveness of real-time language instruction during human-agent collaboration. With the designed layouts, we develop three different evaluation modes to access the agent's real-time adaptation on subtasks and pathes. Details are presented in the following section.

Environments and Collaboration Process

The Overcook-AI environment (Carroll et al. 2019) is designed to test the coordination skills of multiple agents or human agents. Agents work together in a layout (Figure 1B left) to achieve higher scores by preparing and serving soups within a set time frame, following recipe-specific cooking procedures (Figure 1A) to complete the available recipes. Unlike end-to-end AI agents (e.g., behavior cloning or reinforcement learning), the collaboration process between human and embodied agents can generally be divided into two key steps: determining the current subtask and executing atomic actions to finish each subtask.

The possible subtasks in overcooked environments include: 1. Collect ingredients (e.g., onions, tomatoes) and add to the pot. 2. Cook the ingredients and wait for the timer to indicate the soup is ready. 3. Serve the soup in clean dishes and deliver it to the serving location. 4. Agents can also determine to drop the items in their hands on any empty counter. To finish one order, agents have to infer the current state and determine the next subtask based on the recipe. Once the agent determined a subtask and its corresponding position, the agent can be directed through a sequence of atomic actions: *up*, *down*, *left*, *right*, *stay*, and *interact* to finish the subtask. During the collaboration, adaptations of

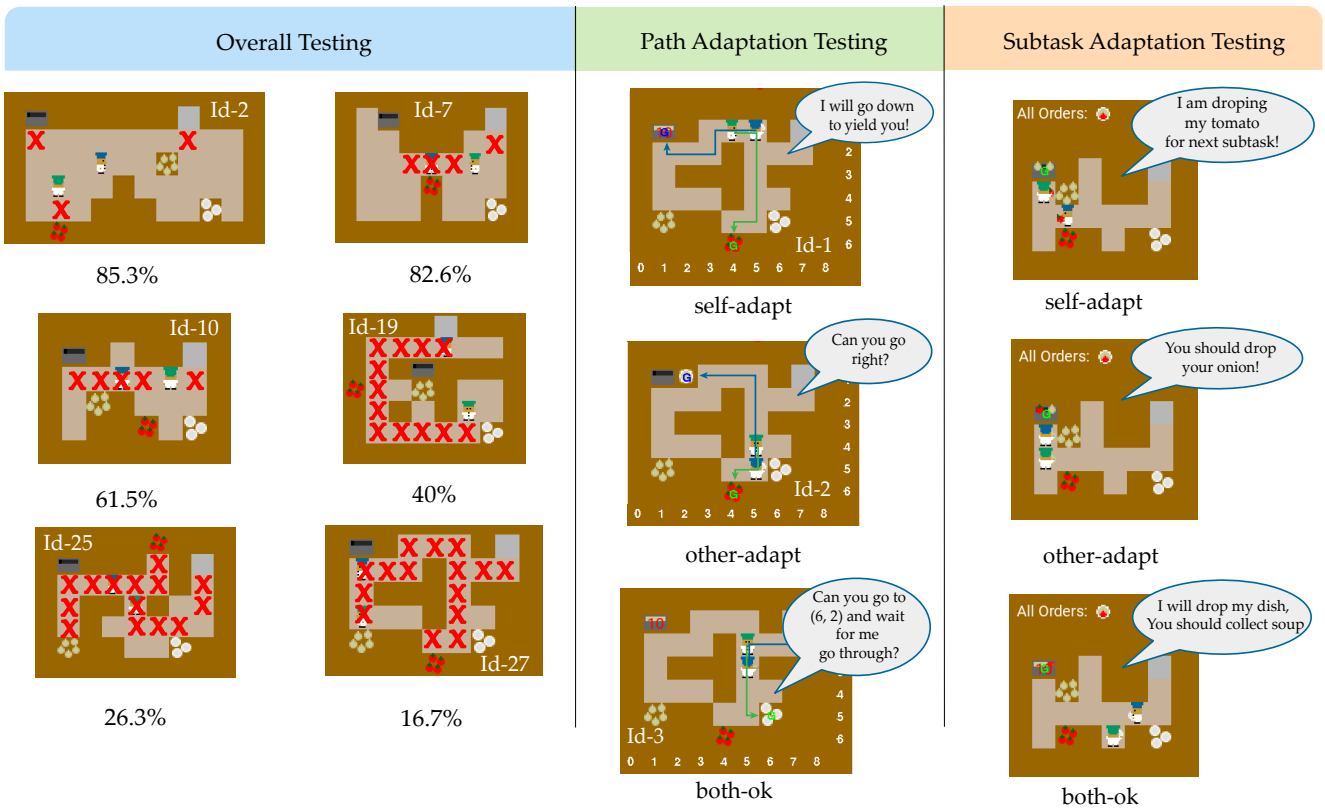


Figure 2: Benchmark for evaluating agent's real-time adaptation capabilities. (A) Six representative layouts with different teaming fluency from 85.3% to 16.7%. The red cross represents a critical point that would interfere with another agent's workflow. (B) Three representative path adaptation testing frames designed by human experts: self-adapt, other-adapt, and both-ok types, viewed from the perspective of the blue agent. The subtask goal locations for the blue and green agents are marked as blue "G" and green "G", respectively, with their greedy paths shown as blue and green lines. The blue agent is giving language instruction. (C) Three representative subtask adaptation testing frames where the blue agent is giving language instructions.

both subtasks and action sequences need to happen in real-time due to the uncertainty of humans or paired agents.

Mode 1: Overall testing

In order to better evaluate the agent's real-time adaptation capabilities, we adopted the teaming fluency metrics discussed in Hoffman (2019) and Nikolaidis (2024) to design layouts for our real-time adaptation benchmark. The teaming fluency of a layout is defined as the percentage of non-obstructed areas within the total free area of a layout. If one agent stays at one position and does not adapt, the other agent cannot finish the task independently, we then mark this position as an obstructed area (red crosses as shown in Figure 2A). With this definition, a higher teaming fluency score suggests an open layout where agents can operate independently without much need to account for each other's presence. Conversely, a lower teaming fluency indicates a more confined and narrow layout, necessitating agents to adapt to one another.

To generate layouts with different teaming fluency, we adopt the following two steps: first, we use GPT-4o to generate layouts with symbolic text representation by prompting it to vary the positions of interaction points (e.g., onion, tomato, pot) and adjust the number and positions of empty counters to change the free space. After generating enough layouts, we run a script to filter layouts based on whether they are solvable and teaming fluency. Finally, we manually review the layouts to ensure they are suitable for investigating different adaptation skills and revise them if needed.

We have selected 22 layouts with teaming fluency scores ranging from 88.37% to 7.14%. More details are shown in the Appendix A.2. These layouts impose constraints on concurrent motions with gradually increasing complexity, requiring agents to adapt to dynamic situations in real-time.

Evaluation criteria For different layouts, we measure the overall score achieved in a certain time threshold as a metric. To achieve good performance in the overall evaluation, the agent needs to show the ability to adapt subtasks as well as low-level paths, especially when paired with non-adaptive agents.

Mode 2: Path adaptation testing

The overall score reflects the agent's general adaptation capability. However, it cannot independently assess path adaptation and subtask adaptation, providing less insight. To explicitly evaluate the path adaptation capabilities of embodied agents, we carefully design short-horizon scenarios and frames where the subtask of each agent is provided. For each short scenario, we first select layouts where teaming fluency is below 50%. Then, we vary the agent's starting position and target position for each agent to form a test. The criteria are that the greedy path of the agent has to collide, and possible adaptation plans exist. Then, we define and label three types of adaptation situations during the execution of subtasks: self-adapt, other-adapt, and both-ok. In the self-adapt scenarios, as shown in the Figure 2B, the agent must yield human trajectory to achieve better fluency or finish both players' subtasks. In other adapt scenarios, task success can only be achieved by asking human teammates to yield. In both-acceptable situations, either can yield or communicate and achieve both subtasks assigned. Finally, human experts will determine the suitable adaptation plan and calculate timesteps for each scenario, and we will set the time limit for each scenario accordingly. In total, we designed 16 self-adaptation scenarios, 14 other-adaptation scenarios, and 13 scenarios where both agents may yield.

Evaluation criteria We provided both quantitative and qualitative evaluations of the agent's path adaptation capabilities based on the short scenarios. For the quantitative evaluation, we have the agent start on the designed start frame, and the scenario is counted as successfully finished if both agents can complete their assigned subtasks within the limited timesteps. The success rate and stuck time on different scenarios provide us with a direct assessment of the agent's ability in path adaption and spatial reasoning. For the qualitative evaluation, we provide frame testing by asking the agent to reason about the start frame and output a language-based adaptation plan. The reasonableness and consistency of the proposed adaptation plan are evaluated by comparing it to a human-labeled adaptation plan.

Mode 3: Subtask adaptation testing

For subtask adaptation testing, we follow a similar process by first labeling the frame where the human experts think that the subtask adaptation is needed. Among those frames where subtask adaptation is needed, it will be further labeled as self-adapt, other-adapt, or both-ok. Finally, a ground subtask adaptation is provided for each frame. Figure 2C shows three frames that require adaptation with their labeled ground truth message.

Evaluation criteria For subtask adaptation testing, we can directly compare the generated subtask goal location with human-provided subtask adaptation plans.

MonTA Framework

For seamless teaming between AI agents and unpredictable humans, the AI agent must have effective communication, high-level reasoning, and real-time adaptive action capabilities. Previous approaches using LLMs for coordination and subtask reasoning (Liu et al. 2023a; Zhang et al. 2023a) still rely on pre-defined scripts for atomic actions, limiting low-level adaptability. Inspired by cognitive studies showing that humans interchange between fast and intuitive thinking versus slow and deliberate thinking (Kahneman 2011), We introduce **MonTA** agent (Figure 3), which leverages two LLMs for explainable, real-time adaptation. MonTA features three key modules:

- **Monitor**: Continuously checks atomic actions to determine the need for adaptation or adjustments.
- **Path Adapter**: Adjusts paths in real-time to accommodate changing environments.
- **Subtask Adapter**: Oversees subtask execution and dynamically adjusts tasks as needed.

This section focuses on the design of the framework and how the adapters and monitor enable adaptive collaboration and communication. Detailed prompts for each module can be found in Appendix A1.

MonTA uses a DFS planner to compute path as a sequence of atomic actions, given a target location. These atomic actions include *up*, *down*, *left*, *right*, *interact*, and *stay*. It's worth noting that this planner can be replaced by any existing algorithm, highlighting the generalizability of our proposed framework.

Monitor

To enable real-time reactive adaptation, latency must be imperceptible to ensure seamless collaboration. Inspired by human decision-making, which involves both fast, intuitive responses (System 1) and deliberate, analytical reasoning (System 2), we employed an active *Monitor* capable of high-frequency inference. Like System 1, the *Monitor* quickly handles straightforward planning and execution, reserving more deliberate reasoning akin to System 2 for moments when adaptations are required. Specifically, it decides to launch subtask-level or path-level adaptations as needed, dynamically switching between a greedy plan and an adaptive plan. The chosen plan is executed by an underlying planner. For instance, subtask-level adaptation is required when agents face conflicting task allocations (e.g., preparing duplicate items), while path-level adaptation is needed for resolving conflicts in planning paths (e.g., navigating a narrow corridor in opposite directions). The agent reverts to its original path once the *Monitor* detects further adaptation unnecessary. By deploying an LLM to monitor actions at atomic level during decision-making, we achieve low inference latency, enabling real-time adaptation.

Monitor model selection To meet the inference speed requirements of the *Monitor*, we tested various models of different sizes to evaluate their inference speed and reasoning capabilities. Based on the most updated LLM benchmark (Beeching et al. 2023), we selected the representative models including GPT-4o, Llama 3.1-8B-Instruct, Llama 3.2-3B-Instruct, and Llama 3.2-1B-Instruct. To further op-

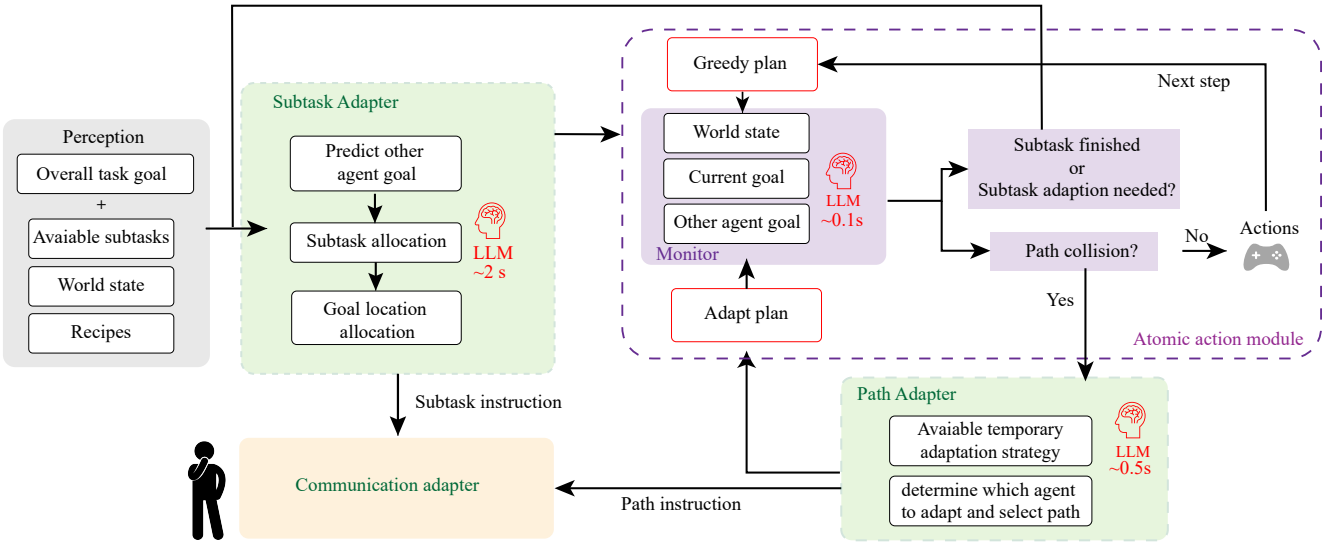


Figure 3: MonTA Framework. The framework comprises a real-time atomic monitor and two primary adapter modules: the subtask adapter and the path adapter. The monitor operates at a high frequency to continuously assess the collaboration status and determine whether adaptation is necessary. The adapters are invoked only upon the monitor's request, and they decide how language instructions should be sent to the communication adapter to guide the human collaborator.

timize inference speed, we utilized SGLang (Zheng et al. 2023). Table 1 summarizes each model's inference speed.

Subtask Adapter

The *Subtask Adapter* is designed to determine the next or ongoing subtask for all agents, such as *picking up onions*, when prompted by the *Monitor*. It considers the overall task goal, world state, and recipes to infer the current goals of other agents and the agent's own goal. It then identifies potential target locations (which may include multiple options) and passes a path to the selected target location to the *Monitor* for active oversight. We leverage a proficient LLM, GPT-4o (OpenAI et al. 2024), to implement Chain-of-Thought reasoning (Wei et al. 2023). Based on the chosen adaptation strategy, the *Subtask Adapter* generates subtask-level messages to coordinate with other agents.

Path Adapter

The *Path Adapter* is designed to determine the best alternative plan beyond the original greedy plan when prompted by the *Monitor*. Using the provided alternative paths and associated costs, the *Path Adapter* reasons about the optimal temporary adaptation path to avoid collisions. The agent responsible for adapting is selected based on the available alternatives. Similarly, we leverage GPT-4o to perform Chain-of-Thought reasoning, enabling the *Path Adapter* to determine which agent should adapt and to select the best adaptation path from the options provided by the planner. The *Path Adapter* then generates a contextually relevant message based on the chosen adaptation strategy. An example prompt is provided in Appendix A1.

Experiments and quantitative results

To demonstrate the advantages of our proposed MonTA framework, we compare our MonTA framework with two baselines. The baseline models consist of a rule-based greedy agent (Carroll et al. 2019) (GA) and a subtask adapter agent (SAA). GA uses a rule-based subtask planner combined with Depth First Search (Tarjan 1972) methods to compute atomic actions (Carroll et al. 2019). It also includes an auto-unstuck function that selects randomly from the available actions when stuck. SAA uses use LLM to generate the next subtask when the current subtask is finished and employs a greedy planner to execute atomic actions without real-time adaptation. We then evaluate and compare our MonTA with baselines on the proposed benchmark.

Overall performance

In multi-agent collaboration, an adaptive agent must recognize when to adjust its actions based on the actions of other agents. To evaluate this adaptability, we selected three layouts — 7, 19, and 27 (Figure 2) — with teaming fluency 82.6%, 40%, and 16.7%, respectively. For each layout, we paired our agent and baseline models with GA. We then conducted experiments 3 times to get the average game score. During the game, the target agent does not communicate with paired GA, requiring adaptive behavior from our agent to successfully complete tasks and get scores as the layout complexity increases.

Effectiveness of the Benchmark The Figure 4 shows that all three agents, including MonTA and the two baselines, achieved their best performance on Layout 7. This is expected, as Layout 7 has high teaming fluency, minimizing

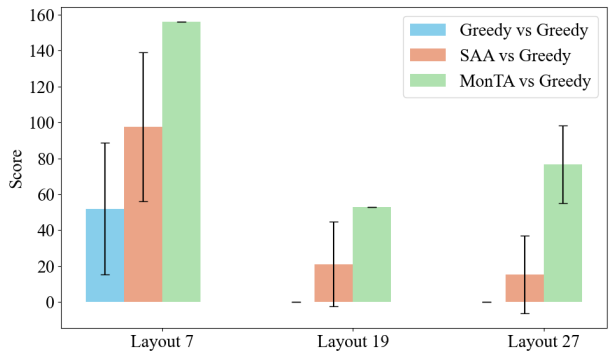


Figure 4: Overall evaluation results. The average score comparison between different agent pairs includes MonTA (ours) v.s. greedy, SAA v.s. greedy, and greedy v.s. greedy.

the likelihood of agents getting stuck and reducing the need for subtasks or path adaptations. As the layout becomes complex and has a lower teaming fluency, we note that the GA consistently scores 0 in layouts 10 and 27 over 5 trials. This potential results from the high ratios of adaptation requirements to finish an order in these two layouts, and the random action used to unstuck is less ineffective or even impossible to succeed. Similarly, SAA also experienced a performance drop as teaming fluency decreased but got slightly better performance than GA by leveraging a proficient LLM to infer subtasks, providing subtask flexibility compared to the rule-based subtask transitions used by GA. These results highlight that as team fluency metrics decrease, agents must exhibit higher adaptability, showcasing the benchmark's ability to evaluate varying levels of adaptability effectively.

MonTA Agent Excels in Layouts with Lower Teaming Fluency Our results demonstrate that the MonTA agent outperforms the two baseline agents across all three tested layouts, achieving scores of 156 $\pm$ 0, 53 $\pm$ 0, and 76.6 $\pm$ 26.5, as shown in Figure 4. This highlights the superiority of MonTA in collaborative scenarios. Additionally, the SAA struggles to match MonTA's performance in Layouts 19 and 27, emphasizing the importance of path adaptation and real-time execution of adaptive strategies. This is further supported by the higher adaptation ratios of the MonTA agent in Layouts 19 and 27 compared to the ratio in Layout 7.

Another interesting observation is that the MonTA agent demonstrates significantly lower variance compared to GA and SAA. This indicates that MonTA consistently identifies potential adaptation needs and executes adaptive plans to prevent failure rather than relying on the paired agent to adapt. This capability is particularly crucial when pairing with agents with unknown dynamics like human.

Path adapter evaluation

While overall performance across layouts is a useful metric for evaluating an agent's adaptability, there is significant randomness in scores, particularly for GA and SAA. For instance, SAA achieved a very high score in one trial on layout 10 due to a rare occurrence where the agents' subtasks did not collide, resulting in an unusually high score. Furthermore, when paired with an auto-unstuck agent, a greedy decision can force the other agent with the auto-unstuck function to yield, progressing step by step at a very slow speed. To better evaluate adaptation capabilities, we conducted experiments on the path adaptation benchmark.

The agents are asked to complete two conflicting subtasks within a limited timeframe. Further details about the designed scenarios are provided in Benchmark Section and Table 2. To balance real-time responsiveness with effective adaptation detection, we varied the LLMs used for atomic monitoring. For the adapter, which requires high reasoning capabilities but does not need frequent queries, GPT-4o was selected in this experiment. In total, we evaluated four agents using GPT-4o, Llama-3.1-8B, Llama-3.2-3B, and Llama-3.2-1B as monitors, alongside two greedy planners—one with and one without an auto-unstuck function. Each of the six agents was paired with the greedy planner featuring the auto-unstuck function to assess their ability to handle expert-designed scenarios. We run 5 trials for each configuration on 21 scenario evaluations. Additional details about the scenarios can be found in Table 3.

In scenarios requiring the target agent to adapt, our MonTA framework achieves an almost 100% (Figure 5A green, red, blue bars) success rate even when utilizing a lightweight large language model like Llama3.2-3B. In contrast, the two greedy planners almost always fail to complete these scenarios, experiencing significant stuck time (Figure 5A, brown bar), whereas the MonTA agents successfully avoids being stuck. This clearly demonstrates the superior adaptability of our framework.

In scenarios labeled as Both-OK and Other-Adapt, the MonTA agent using GPT-4o, Llama3.2-8B, or Llama3.2-3B achieves performance similar to the greedy planner. In these cases, the MonTA agent may request the paired agent to adapt while following its original plan. This approach can still achieve the subtask goal because the paired agent executes an auto-unstuck function. The standard deviation is high here because we are averaging all scenarios which belong to the same type. Notably, in such cases, MonTA retains an advantage when paired with a human or an agent capable of interpreting language input, as it can effectively communicate and inform adaptation plans to allow the paired agent to adapt more effectively.

Lauguage instruction study True reactive collaborators not only adapt their behavior but also provide instructions to others when necessary. In our framework, the adapter leverages a proficient LLM, GPT-4o, to reason about adaptation plans and generate language instructions. The adapter autonomously adjusts its behavior when self-adaptation is required and sends language instructions only when another agent needs to adapt.

To evaluate the language instructions generated by our adapter, we queried GPT-4o to produce instructions for both self-adaptation and other-adaptation scenarios at each spe-

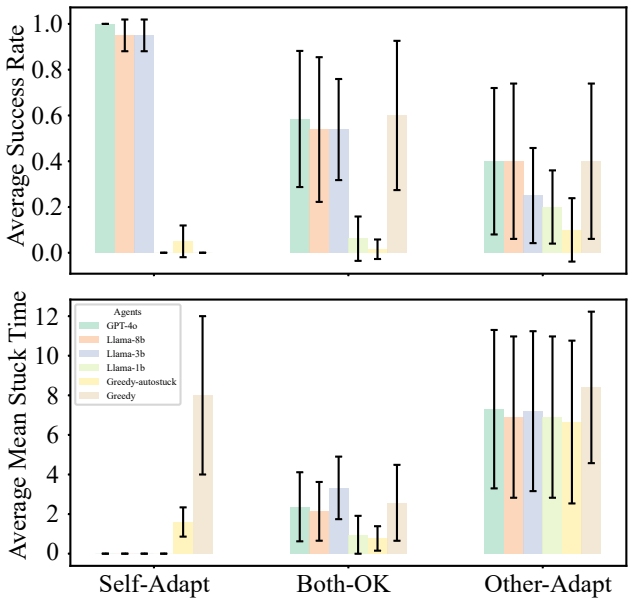


Figure 5: Path adaptation evaluation results. (A) Success rates of different agents on scenarios of Self-Adapt, Other Adapt, and Both-OK types. (B) Stuck time experienced by two agents across three scenario types.

Metrics	GPT-4o	Llama-8B	Llama-3B	Llama-1B
L_m (s)	0.42 (0.07)	0.14 (0.002)	0.08 (0.0006)	0.04 (0.001)
L_{sa} (s)	2.09 (0.29)	2.11 (1.0)	2.62 (1.34)	2.84 (1.31)
L_{pa} (s)	0.79 (0.08)	0.48 (0.002)	0.26 (0.03)	0.15 (0.001)
N_a (%)	9.7 (5.5)	52 (4)	84 (5)	4 (0)

Table 1: The monitor latency, L_m, subtask adapter latency, L_{sa}, and path adapter latency, L_{pa}, as well as the ratio of adapter queries, N_a. The monitor decides on every step if we use different LLM. The format is Mean (Standard deviation).

cific step. This process was repeated five times across 20 benchmark scenarios. Seven human experts assessed the reasonableness and consistency of these instructions. Detailed evaluation criteria are provided in Appendix A2.

The evaluation results, presented in Figure 6, reveal that human experts found the suggestions generated by MonTA to be reasonable and consistent in nearly 75% of scenarios. This indicates that the adapter effectively identifies who should adapt and determines the correct adaptation plan. Such adaptability can reduce the cognitive burden on human collaborators and facilitate seamless human-agent collaboration.

An interesting observation is that in some of the frames, our agent received evaluations with a larger variance. A closer examination of the scenarios revealed divergent human preferences regarding costs and the choice between self-adaptation and other-adaptation solutions. These differences indicate that the optimal solution can vary depending on individual preferences, highlighting the importance of considering human preferences when designing agent instructions. Further details of the evaluation are provided in Table 6.

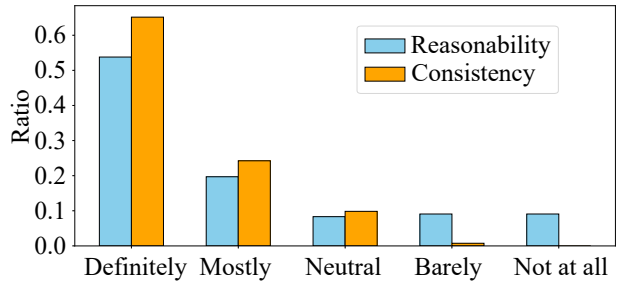


Figure 6: Language instruction evaluation results. Blue and yellow bars show the ratio of LLM instruction reasonability levels and the consistency of LLM suggestions reported by human experts.

Trade-off between performance and latency To use LLMs to aid with real-time decision-making, there is a latency requirement, where high latency may influence the human experience of collaboration. Therefore, we experimented to compare the latency estimation of each query. The reactive adaptation process includes two latency levels: monitor latency and adapter latency. The monitor is queried every atomic action step, which largely determines the overall frequency of collaboration, however, if the adapter has to be called multiple times due to the incapability of the monitor, it might still result in a latency issue. We selected four language models to test, including GPT-4o, Llama3.1-8b, Llama3.2-3b, and Llama3.1-1b. The GPT-4o is queried using API, where the Llama is run locally through SGLang in an RTX3090 GPU computer.

We report the average latency estimation results from our path adapter experiments in Table 1. As expected, the adapter latency is consistently higher than the monitor latency across all LLMs, aligning with the design goal for the monitor to prioritize speed over complex reasoning. Notably, as the model size decreases to 8B and 3B, the monitor latency is reduced to approximately 0.1 seconds, enabling a monitoring frequency of 8Hz for real-time decision-making, and these smaller models can run locally.

Furthermore, the GPT-4o monitor only needs to query the adapter in 9.7% of cases, demonstrating excellent performance in identifying when adaptation is necessary. In contrast, the Llama models struggle to achieve efficient state transitions for monitoring. To improve generalization to real-world systems, fine-tuning or distilling the Llama models would be an ideal approach to enhance their performance within this framework.

Conclusion

In this paper, we focus on leveraging LLMs to enable agents with real-time adaptation capabilities. To achieve this, we

first create a benchmark consisting of diverse layouts, carefully designed scenarios that require agents to demonstrate reactive and adaptive behaviors. We then introduce the MonTA framework, the core idea of which is only to utilize the LLM to monitor whether and what type of adaptation is needed. This allows the monitor to assess the agents' status at a higher rate in real time. When necessary, the LLM transitions to more deliberate "slow thinking" to adapt plans or provide user instructions. Our experiments demonstrate that the real-time adaptation capabilities significantly enhance performance and robustness when two agents collaborate in low-teaming-fluency layouts and execute our designed subtasks.

Acknowledgment

The authors would like to thank Siddharth Srikanth for his assistance with prompt engineering and code refactoring.

References

Agashe, S.; Fan, Y.; Reyna, A.; and Wang, X. E. 2024. Llm-coordination: Evaluating and analyzing multi-agent coordination abilities in large language models.

Agashe, S.; Fan, Y.; and Wang, X. E. 2023. Evaluating multi-agent coordination abilities in large language models. *arXiv preprint arXiv:2310.03903*.

Beeching, E.; Fourrier, C.; Habib, N.; Han, S.; Lambert, N.; Rajani, N.; Sanseviero, O.; Tunstall, L.; and Wolf, T. 2023. Open llm leaderboard (2023-2024). `https://huggingface.co/spaces/open-llm-leaderboard-old/open_llm_leaderboard`.

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint 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. *Advances in neural information processing systems* 32.

Chang, M.; Chhablani, G.; Clegg, A.; Cote, M. D.; Desai, R.; Hlavac, M.; Karashchuk, V.; Krantz, J.; Mottaghi, R.; Parashar, P.; et al. 2024. Partnr: A benchmark for planning and reasoning in embodied multi-agent tasks. *arXiv preprint arXiv:2411.00081*.

Cui, B.; Hu, H.; Pineda, L.; and Foerster, J. 2021. K-level reasoning for zero-shot coordination in hanabi. *Advances in Neural Information Processing Systems* 34:8215–8228.

Das, A.; Datta, S.; Gkioxari, G.; Lee, S.; Parikh, D.; and Batra, D. 2018. Embodied question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–10.

(FAIR)†, M. F. A. R. D. T.; Bakhtin, A.; Brown, N.; Dinan, E.; Farina, G.; Flaherty, C.; Fried, D.; Goff, A.; Gray, J.; Hu, H.; Jacob, A. P.; Komeili, M.; Konath, K.; Kwon, M.; Lerer, A.; Lewis, M.; Miller, A. H.; Mitts, S.; Renduchintala, A.; Roller, S.; Rowe, D.; Shi, W.; Spisak, J.; Wei, A.; Wu, D.; Zhang, H.; and Zijlstra, M. 2022. Human-level play in the game of ¡i¿diplomacy¡/i¿ by combining language models with strategic reasoning. *Science* 378(6624):1067–1074.

Fontaine, M. C.; Hsu, Y.-C.; Zhang, Y.; Tjanaka, B.; and Nikolaidis, S. 2021. On the importance of environments in human-robot coordination. *arXiv preprint arXiv:2106.10853*.

Hoffman, G. 2019. Evaluating fluency in human–robot collaboration. *IEEE Transactions on Human-Machine Systems* 49(3):209–218.

Kahneman, D. 2011. Thinking, fast and slow. *Farrar, Straus and Giroux*.

Li, M.; Zhao, S.; Wang, Q.; Wang, K.; Zhou, Y.; Srivastava, S.; Gokmen, C.; Lee, T.; Li, L. E.; Zhang, R.; et al. 2024. Embodied agent interface: Benchmarking llms for embodied decision making. *arXiv preprint arXiv:2410.07166*.

Liu, J.; Yu, C.; Gao, J.; Xie, Y.; Liao, Q.; Wu, Y.; and Wang, Y. 2023a. Llm-powered hierarchical language agent for real-time human-ai coordination. *arXiv preprint arXiv:2312.15224*.

Liu, S.; Wilson, C. G.; Krishnamachari, B.; and Qian, F. 2023b. Understanding human dynamic sampling objectives to enable robot-assisted scientific decision making. *ACM Transactions on Human-Robot Interaction*.

Liu, S.; Wilson, C. G.; Lee, Z. I.; and Qian, F. 2024. Modelling experts' sampling strategy to balance multiple objectives during scientific explorations. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, 452–461.

Majumdar, A.; Ajay, A.; Zhang, X.; Putta, P.; Yenamandra, S.; Henaff, M.; Silwal, S.; Mcvay, P.; Maksymets, O.; Arnaud, S.; et al. 2024. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16488–16498.

Nikolaidis, S. 2024. Algorithmic scenario generation as quality diversity optimization. *arXiv preprint arXiv:2409.04711*.

OpenAI; Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; Avila, R.; Babuschkin, I.; Balaji, S.; Balcom, V.; Baltescu, P.; Bao, H.; Bavarian, M.; Belgum, J.; Bello, I.; Berdine, J.; Bernadett-Shapiro, G.; Berner, C.; Bogdonoff, L.; Boiko, O.; Boyd, M.; Brakman, A.-L.; Brockman, G.; Brooks, T.; Brundage, M.; Button, K.; Cai, T.; Campbell, R.; Cann, A.; Carey, B.; Carlson, C.; Carmichael, R.; Chan, B.; Chang, C.; Chantzis, F.; Chen, D.; Chen, S.; Chen, R.; Chen, J.; Chen, M.; Chess, B.; Cho, C.; Chu, C.; Chung, H. W.; Cummings, D.; Currier, J.; Dai, Y.; Decareaux, C.; Degry, T.; Deutsch, N.; Deville, D.; Dhar, A.; Dohan, D.; Dowling, S.; Dunning, S.; Ecoffet, A.; Eleti, A.; Eloundou, T.; Farhi, D.; Fedus, L.; Felix, N.; Fishman, S. P.; Forte, J.; Fulford, I.; Gao, L.; Georges, E.; Gibson, C.; Goel, V.; Gogineni, T.; Goh, G.; Gontijo-Lopes, R.; Gordon, J.; Grafstein, M.; Gray, S.; Greene, R.; Gross, J.; Gu, S. S.; Guo, Y.; Hallacy, C.; Han, J.; Harris, J.; He, Y.; Heaton, M.; Heidecke, J.; Hesse, C.; Hickey, A.;

Hickey, W.; Hoeschele, P.; Houghton, B.; Hsu, K.; Hu, S.; Hu, X.; Huizinga, J.; Jain, S.; Jain, S.; Jang, J.; Jiang, A.; Jiang, R.; Jin, H.; Jin, D.; Jomoto, S.; Jonn, B.; Jun, H.; Kaftan, T.; Łukasz Kaiser; Kamali, A.; Kanitscheider, I.; Keskar, N. S.; Khan, T.; Kilpatrick, L.; Kim, J. W.; Kim, C.; Kim, Y.; Kirchner, J. H.; Kiros, J.; Knight, M.; Kokotajlo, D.; Łukasz Kondraciuk; Kondrich, A.; Konstantinidis, A.; Kosic, K.; Krueger, G.; Kuo, V.; Lampe, M.; Lan, I.; Lee, T.; Leike, J.; Leung, J.; Levy, D.; Li, C. M.; Lim, R.; Lin, M.; Lin, S.; Litwin, M.; Lopez, T.; Lowe, R.; Lue, P.; Makanju, A.; Malfacini, K.; Manning, S.; Markov, T.; Markovski, Y.; Martin, B.; Mayer, K.; Mayne, A.; McGrew, B.; McKinney, S. M.; McLeavey, C.; McMillan, P.; McNeil, J.; Medina, D.; Mehta, A.; Menick, J.; Metz, L.; Mishchenko, A.; Mishkin, P.; Monaco, V.; Morikawa, E.; Mossing, D.; Mu, T.; Murati, M.; Murk, O.; Mély, D.; Nair, A.; Nakano, R.; Nayak, R.; Neelakantan, A.; Ngo, R.; Noh, H.; Ouyang, L.; O'Keefe, C.; Pachocki, J.; Paino, A.; Palermo, J.; Pantuliano, A.; Parascandolo, G.; Parish, J.; Parparita, E.; Passos, A.; Pavlov, M.; Peng, A.; Perelman, A.; de Avila Belbute Peres, F.; Petrov, M.; de Oliveira Pinto, H. P.; Michael; Pokorny; Pokrass, M.; Pong, V. H.; Powell, T.; Power, A.; Power, B.; Proehl, E.; Puri, R.; Radford, A.; Rae, J.; Ramesh, A.; Raymond, C.; Real, F.; Rimbach, K.; Ross, C.; Rotsted, B.; Roussez, H.; Ryder, N.; Saltarelli, M.; Sanders, T.; Santurkar, S.; Sastry, G.; Schmidt, H.; Schnurr, D.; Schulman, J.; Selsam, D.; Sheppard, K.; Sherbakov, T.; Shieh, J.; Shoker, S.; Shyam, P.; Sidor, S.; Sigler, E.; Simens, M.; Sitkin, J.; Slama, K.; Sohl, I.; Sokolowsky, B.; Song, Y.; Staudacher, N.; Such, F. P.; Summers, N.; Sutskever, I.; Tang, J.; Tezak, N.; Thompson, M. B.; Tillet, P.; Tootoonchian, A.; Tseng, E.; Tuggle, P.; Turley, N.; Tworek, J.; Uribe, J. F. C.; Vallone, A.; Vijayvergiya, A.; Voss, C.; Wainwright, C.; Wang, J. J.; Wang, A.; Wang, B.; Ward, J.; Wei, J.; Weinmann, C.; Welihinda, A.; Welinder, P.; Weng, J.; Weng, L.; Wiethoff, M.; Willner, D.; Winter, C.; Wolrich, S.; Wong, H.; Workman, L.; Wu, S.; Wu, J.; Wu, M.; Xiao, K.; Xu, T.; Yoo, S.; Yu, K.; Yuan, Q.; Zaremba, W.; Zellers, R.; Zhang, C.; Zhang, M.; Zhao, S.; Zheng, T.; Zhuang, J.; Zhuk, W.; and Zoph, B. 2024. Gpt-4 technical report.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35:27730–27744.

Puig, X.; Shu, T.; Tenenbaum, J. B.; and Torralba, A. 2023. Nopa: Neurally-guided online probabilistic assistance for building socially intelligent home assistants. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 7628–7634. IEEE.

Strouse, D.; McKee, K.; Botvinick, M.; Hughes, E.; and Everett, R. 2021. Collaborating with humans without human data. *Advances in Neural Information Processing Systems* 34:14502–14515.

Tarjan, R. 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing* 1(2):146–160.

Weber, B. G.; Mateas, M.; and Jhala, A. 2011. Building human-level ai for real-time strategy games. In *2011 AAAI Fall symposium series*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Zhang, C.; Yang, K.; Hu, S.; Wang, Z.; Li, G.; Sun, Y.; Zhang, C.; Zhang, Z.; Liu, A.; Zhu, S.-C.; et al. 2023a. Proagent: Building proactive cooperative ai with large language models. *arXiv preprint arXiv:2308.11339*.

Zhang, H.; Du, W.; Shan, J.; Zhou, Q.; Du, Y.; Tenenbaum, J. B.; Shu, T.; and Gan, C. 2023b. Building cooperative embodied agents modularly with large language models. *arXiv preprint arXiv:2307.02485*.

Zhang, H.; Wang, Z.; Lyu, Q.; Zhang, Z.; Chen, S.; Shu, T.; Du, Y.; and Gan, C. 2024. Combo: Compositional world models for embodied multi-agent cooperation. *arXiv preprint arXiv:2404.10775*.

Zheng, L.; Yin, L.; Xie, Z.; Huang, J.; Sun, C.; Hao Yu, C.; Cao, S.; Kozyrakis, C.; Stoica, I.; Gonzalez, J. E.; et al. 2023. Efficiently programming large language models using sglang. *arXiv e-prints* arXiv–2312.

Appendix

A1. Prompt Construction

We have distinct prompts for Subtask Adapter, Path Adapter, and Monitor. There are two prompts for Monitor as it serves different purposes depending on whether an agent is adapting or following the original greedy path.

Subtask Adapter prompt Subtask Adapter prompt contains environment context, Current game state, filtered actions, and goals.

```
Context:
You are a chef that works with another human chef in a
      kitchen ...
You should follow these rules: ...
The procedure to finish one dish is ...
Recipe book:
Recipe 0: Requires ingredients: [ingredient 1],[
     ingredient 2], [ingredient 3]
====================
Kitchen state:
    [Kitchen Items in text]
====================
Your current state:
    1. You are at the coordinates (x,y)
    2. You are facing [item name]
    3. You are holding [item name]

The state of the other human chef:
    1. The other chef is at the coordinates (x,y)
    2. They are facing [item name]
    3. They are holding [item name]
====================
Your available actions:
    \Option 1: [available subtask]
    \Option 2: [available subtask]
    ...
Human available actions:
    \Option 1: [available subtask]
    \Option 2: [available subtask]
====================
Goal:
Your first step will be: analyze the state of the
     kitchen and items, as well as the recipe to get
     next best action.select an action from your
     avaiable actions. and select the target position
     to interact. choose your target position from
     kitchen items. do not select target position not
     listed in kitchen state list.
Your second step will be: analyze human state,world
     state and human message/human preference, reason
     about human intention. choose a human intended
     position from Delivery location, pot, dispenser
     listed in kitchen items. do not select target
     position not listed in kitchen item list.

Return the final data with human intented target
     position, human intented action id, your
     final_action_id, target position,
...
```

Path Adapter prompt Path Adapter takes information about agents' greedy paths, potential adaptation plans with associated costs, and goals.

```
Here is your planned greedy path:
    [agent greedy path]

Human is likely to take following path:
    [human greedy path]
These two paths overlap path points, which causes
     collisions.
Your potential adapt plans:
    \Plan 1 : [ available plan, plan length]
    \Plan 2 : [available plan, plan length]
...
human potentail adapts plans:
    \Plan 1 : [available plan, plan length]
    \Plan 2 : [available plan, plan length]
...

First check the adaptation plan works, by checking if
     the adaptation plan of one agent will still
     overlap with the other agent's original path.
After identifying a valid adaptation plan, choose one
     with the lowest cost and decide which agent to
     adapt., please check carefully if the adaptation
     plan has conflict with other agnet's original
     path
Return the probability of humans adapting with 1 to
     p_human_adapt if human adaptation has low cost,
     and the probability of agent adapting with 1 to
     p_agent_adapt if agent adapt has lowerst cost and
      valid and adapt_index, give me detailed analysis
```

Monitor prompt Monitor has two prompt. One prompt monitoring if agents need to shift to the adaptation path, and one prompt monitoring if agents need to switch back to the original greedy path. We show one prompt here as they only vary in prompt goals. Prompt contains grid layout, agent positions, target positions, agents' greedy path to target, and goals.

```
Context:
Grid layout:
This is a 9x7 grid world. The top left corner is (0,
     0) and the bottom right corner is (8, 6). Moving
     down will result second pos coordinates +1, e.g.
     (0,0) -> (0,1), moving right will results the
     first pos coordinate +1, e.g (0,0)->(1,0) The
     Grid contains the following items:
X is obstacle, a is your position, and A is your
     target position, b is position of human partner
     and B is human partner's target position(inferred
     ).
[grid layout]

You are at the coordinates: [agent position]
Your target positions:[agent target position]

The other chef is at the coordinates: [others position
     ]
Human Target Position: [others target position]

your planned greedy path:
[agent greedy path]

Human is likely to take following path:
```

```
[human greedy path]
You are current doing a clear temperary adaptation
    path for collision avoidance:
[adaptation path]

***Your goal: Only using all the information above ***
analyze Do you need to adapt to human behavior?
for example, you should adapt to human when you want
    to avoid collision (human current position is on
    your way).
otherwise, do not adapt. For example, if you see that
    both agent are stucked, then it could be good to
    adapt.

respond your analysis and if you follow greedy or not.
    respond true if follow greedy, false if not.
```

A2. Additional details of benchmark

Layouts The layouts is selected based on the teaming fluency metrics from high to low. Table 2 provides a visualization of the selected 22 layouts and the corresponding ID as well as the teaming fluency.

Frames We generated 41 frames with three types, including self-adapt, other-adapt, and both-ok. We use 20 frames for quantitative testing, which is shown in Table 3 and 21 frames for qualitative testing(Table 6).

A3. Additional results

Figure 7 presents the success rate of path adaptation testing on 20 frames shown in Table 2. Figure 8 presents the detailed evaluation of llm-generated language instructions for each frame shown in Table 6. Table 4 and Table 5 shows detailed success rate and stuck time for path adaptation testing.

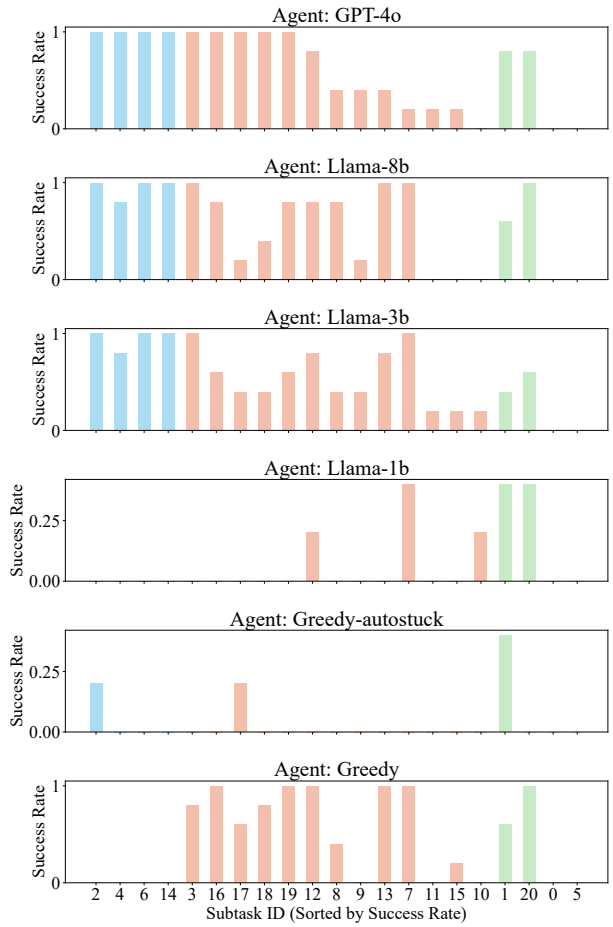


Figure 7: Success rate of path adaptation testing on different frames. The blue, red, and green bars represent the success rate of self-adapt, both-ok, and other-adapt frames.

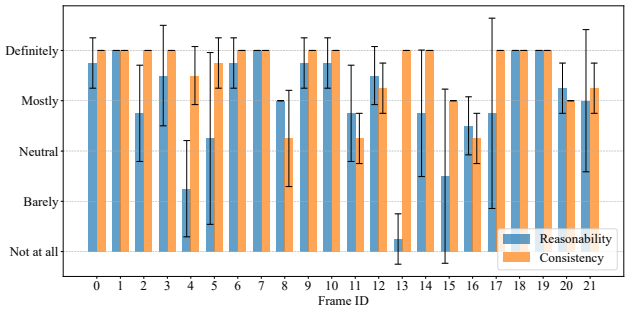


Figure 8: Human reported reasonability and consistency on the llm generated suggestions on each frame.

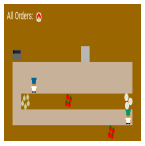
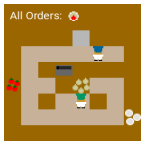
ID	Layout	Collision Points	Fluency	ID	Layout	Collision Points	Fluency
1		5	88.37%	2		5	85.29%
5		6	80.00%	6		5	85.71%
7		4	82.61%	8		5	77.27%
10		5	61.5%	11		5	68.75%
14		7	68.18%	15		7	22.22%
16		5	73.68%	17		9	40.00%
18		7	41.67%	19		12	40.00%
20		9	18.18%	21		8	33.33%
22		13	7.14%	23		12	40.00%
24		12	42.86%	25		14	26.32%
26		15	16.67%	27		15	16.67%

Table 2: All 22 layouts with corresponding number of collision points and team fluency scores.

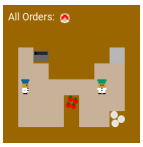
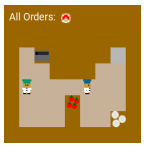
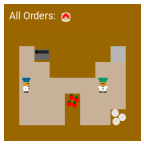
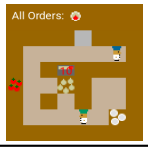
ID	Frame	Description	Adaptation type	ID	Frame	Description	Adaptation type
0		**Blue:** pickup onion **Green:** pot ingredient	Other-adapt	1		**Blue:** pickup onion **Green:** pickup tomato	Other-adapt
2		**Blue:** pickup tomato **Green:** pickup onion	Self-adapt	3		**Blue:** pickup dish **Green:** pickup onion	Both-ok
4		**Blue:** pickup onion **Green:** pickup dish	Self-adapt	5		**Blue:** pickup dish **Green:** pickup onion	Other-adapt
6		**Blue:** pickup onion **Green:** pickup tomato	Self-adapt	7		**Blue:** pickup tomato **Green:** pickup tomato	Both-ok
8		**Blue:** pickup onion **Green:** pickup tomato	Both-ok	9		**Blue:** pickup tomato **Green:** pickup onion	Both-ok
10		**Blue:** pickup dish **Green:** pickup tomato	Both-ok	11		**Blue:** pickup onion **Green:** pot ingredient	Both-ok
12		**Blue:** pickup tomato **Green:** pot ingredient	Both-ok	13		**Blue:** pickup tomato **Green:** pickup onion	Both-ok
14		**Blue:** pickup onion **Green:** pickup tomato	Self-adapt	15		**Blue:** pickup dish **Green:** pickup onion	Both-ok
16		**Blue:** pickup dish **Green:** pot ingredient	Both-ok	17		**Blue:** pot ingredient **Green:** pickup dish	Both-ok
18		**Blue:** pot ingredient **Green:** serve soup	Both-ok	19		**Blue:** serve soup **Green:** pot ingredient	Both-ok
20		**Blue:** pickup dish **Green:** pickup tomato	Other-adapt				

Table 3: All 21 path adaptation testing evaluation with description and adaptation type.

Subtask id	MonTA (GPT-4o)	MonTA (Llama-8b)	MonTA (Llama-3b)	MonTA (Llama-1b)	Greedy (auto_unstuck)	Greedy
0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.8	0.6	0.4	0.4	0.4	0.6
2	1.0	1.0	1.0	0.0	0.2	0.0
3	1.0	1.0	1.0	0.0	0.0	0.8
4	1.0	0.8	0.8	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0
6	1.0	1.0	1.0	0.0	0.0	0.0
7	0.2	1.0	1.0	0.4	0.0	1.0
8	0.4	0.8	0.4	0.0	0.0	0.4
9	0.4	0.2	0.4	0.0	0.0	0.0
10	0.0	0.0	0.2	0.2	0.0	0.0
11	0.2	0.0	0.2	0.0	0.0	0.0
12	0.8	0.8	0.8	0.2	0.0	1.0
13	0.4	1.0	0.8	0.0	0.0	1.0
14	1.0	1.0	1.0	0.0	0.0	0.0
15	0.2	0.0	0.2	0.0	0.0	0.2
16	1.0	0.8	0.6	0.0	0.0	1.0
17	1.0	0.2	0.4	0.0	0.2	0.6
18	1.0	0.4	0.4	0.0	0.0	0.8
19	1.0	0.8	0.6	0.0	0.0	1.0
20	0.8	1.0	0.6	0.4	0.0	1.0

Table 4: Success rate for greedy agent, greedy agent with built-in auto-unstuck function, and MonTA agent with varying monitors for all 21 path adaptation evaluations.

Subtask id	MonTA (GPT-4o)	MonTA (Llama-8b)	MonTA (Llama-3b)	MonTA (Llama-1b)	Greedy (auto_unstuck)	Greedy
0	24.0 (0)	24.0 (0)	24.0 (0)	24.0 (0)	24.0 (0)	24.0 (0)
1	4.6 (2.1)	3.6 (5.1)	4.8 (4.6)	3.6 (0.89)	2.6 (1.5)	7.2 (8.2)
2	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	4.4 (4.2)	24.0 (0)
3	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)
4	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)
5	0.6 (1.3)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	2.4 (2.2)
6	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	2.0 (3.1)	8.0 (4.1)
7	6.8 (9.3)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	1.8 (1.6)
8	0.0 (0)	0.6 (1.3)	7.4 (3.2)	0.0 (0)	0.0 (0)	10.2 (9.3)
9	1.8 (1.6)	8.8 (6.9)	8.8 (8.1)	0.0 (0)	0.0 (0)	0.0 (0)
10	8.4 (4.7)	1.8 (1.6)	7.2 (1.3)	4.4 (3.4)	5.0 (1.9)	2.6 (3.6)
11	0.0 (0)	0.0 (0)	2.6 (5.8)	0.0 (0)	0.0 (0)	0.0 (0)
12	0.0 (0)	0.0 (0)	0.8 (1.8)	0.0 (0)	0.0 (0)	0.0 (0)
13	0.0 (0)	0.0 (0)	1.8 (4)	0.0 (0)	0.0 (0)	0.0 (0)
14	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)
15	13.8 (8.1)	10.8 (10)	11.6 (8.6)	8.0 (11)	2.6 (2.6)	15.8 (8.5)
16	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)
17	0.0 (0)	5.8 (5.7)	3.0 (4.5)	0.0 (0)	2.4 (2.5)	3.0 (5.2)
18	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)
19	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)
20	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)

Table 5: Stuck time for greedy agent, greedy agent with built-in auto-unstuck function, and MonTA agent with varying monitors for all 21 path adaptation evaluations. The format is Mean (Standard deviation).

ID	Frame	Language Instruction	Reasonability	Consistency	ID	Frame	Language Instruction	Reasonability	Consistency
0		I will adapt to location [4, 3]. I will adapt to location [4, 3]. I will adapt to location [4, 3]. I will adapt to location [4, 3]. I will adapt to location [4, 3].	3.83 (0.41)	4.00 (0.00)	1		Could you adapt to location [4, 3]? Could you adapt to location [4, 3]? Could you adapt to location [4, 3]? Could you adapt to location [4, 3]? Could you adapt to location [4, 3]? Thank you!	4.00 (0.00)	3.83 (0.41)
2		I will adapt to location [4, 3]. I will adapt to location [4, 3]. I will adapt to location [4, 3]. I will adapt to location [4, 3]. I will adapt to location [4, 3].	2.67 (1.21)	4.00 (0.00)	3		Could you adapt to location [5, 2]? Could you adapt to location [5, 2]? Could you adapt to location [5, 2]? Could you adapt to location [5, 2]? Could you adapt to location (6, 2)?	3.17 (1.33)	4.00 (0.00)
4		I will adapt to location [2, 2]. I will adapt to location [2, 2]. I will adapt to location [2, 2]. I will adapt to location [2, 2]. No adaptation	0.83 (0.98)	3.50 (0.55)	5		Could you adapt to location [2, 2]? Thank you! Could you adapt to location [2, 2]? Could you adapt to location [2, 2]? Could you adapt to location [2, 2]? Could you adapt to location [2, 2]?	1.67 (1.63)	3.67 (0.52)
6		I will adapt to location [1, 3]. I will adapt to location [1, 3]. I will adapt to location [1, 3]. I will adapt to location [1, 3]. I will adapt to location [1, 3]. Thank you!	3.83 (0.41)	4.00 (0.00)	7		Could you adapt to location [1, 3]? Could you adapt to location [1, 3]? Could you adapt to location [1, 3]? Could you adapt to location [1, 3]? Could you adapt to location [1, 3]? Thank you!	4.00 (0.00)	3.83 (0.41)
8		Could you adapt to location [2, 4]? Could you adapt to location [2, 4]? Thank you! Could you adapt to location [1, 3]? Thank you! Could you adapt to location [2, 4]? Thank you! I will adapt to location [2, 2].	3.17 (0.41)	2.17 (0.75)	9		I will adapt to location [2, 4]. I will adapt to location [2, 4]. I will adapt to location [2, 4]. I will adapt to location [2, 4]. I will adapt to location [2, 4].	3.83 (0.41)	4.00 (0.00)
10		I will adapt to location [2, 4]. I will adapt to location [2, 4]. I will adapt to location [2, 4]. I will adapt to location [2, 4]. I will adapt to location [2, 4].	3.83 (0.41)	4.00 (0.00)	11		Could you adapt to location [3, 2]? Could you adapt to location [6, 4]? Could you adapt to location [6, 4]? Could you adapt to location [6, 4]? Could you adapt to location [3, 2]?	3.00 (0.89)	2.33 (0.52)
12		Could you adapt to location (1, 3)? I will adapt to location [6, 2]. I will adapt to location [6, 2]. I will adapt to location [6, 2]. I will adapt to location [6, 2].	3.50 (0.55)	3.33 (0.52)	13		Could you adapt to [5, 5]? Could you adapt to location [5, 5]? Could you adapt to location [5, 5]? Could you adapt to location [5, 5]? Could you adapt to location [5, 5]?	0.33 (0.52)	4.00 (0.00)
14		I will adapt to location [6, 2]. I will adapt to location [6, 2]. I will adapt to location [6, 2]. I will adapt to location [6, 2]. I will adapt to location [6, 2].	3.17 (1.17)	4.00 (0.00)	15		I will adapt to location (1, 2). No adaptation No adaptation No adaptation No adaptation.	1.50 (1.64)	2.67 (0.52)
16		Could you adapt to location [1, 3]? I will adapt to location [2, 2]. I will adapt to location [2, 2]. Could you adapt to location [1, 3]? I will adapt to location [2, 2].	2.50 (1.05)	2.33 (0.52)	17		I will adapt to location [1, 3]. I will adapt to location [1, 3]. I will adapt to location [1, 3]. I will adapt to location [1, 3]. I will adapt to location [1, 3].	2.83 (1.60)	4.00 (0.00)
18		I will adapt to location [1, 4]. I will adapt to location [1, 4]. I will adapt to location [1, 4]. I will adapt to location [1, 4]. I will adapt to location [1, 4].	4.00 (0.00)	4.00 (0.00)	19		Could you adapt to location [1, 4]? Could you adapt to location [1, 4]? Thank you! Could you adapt to location [1, 4]? Thank you! Could you adapt to location [1, 4]? Could you adapt to location [1, 4]? Thank you!	4.00 (0.00)	3.83 (0.41)
20		I will adapt to location [4, 2]. I will adapt to location [4, 2]. I will adapt to location [4, 2]. No adaptation I will adapt to location [4, 2].	3.50 (0.55)	3.00 (0.00)	21		No adaptation I will adapt to location [4, 2]. I will adapt to location [4, 2]. I will adapt to location [4, 2]. I will adapt to location [4, 2].	2.83 (1.47)	3.33 (0.52)

Table 6: 22 frames user study evaluation and corresponding five agent generated responses, human rated reasonability and consistency.