2026 29th International Conference on Computer Supported Cooperative Work in Design (CSCWD) | 979-8-3315-8470-2/26/$31.00 ©2026 IEEE | DOI: 10.1109/CSCWD68734.2026.11582521

KBC-Coop: Knowledge-Grounded Role Cooperation for Single-Model LLMs in Collaborative Tasks

Haoyu Yang[1] Linquan Wu[2] Shaochao Lin[3] Zixuan Wang[2] Jacky Keung[2]
[1] University of Electronic Science and Technology of China
[2] City University of Hong Kong
[3] Harbin Engineering University

***Abstract*—Recent studies have shown that common multi-agent collaboration schemes for large language models (LLMs)—such as multi-round group discussion, debate, and unconstrained multi-role prompting—do not consistently outperform carefully designed single-agent baselines and can even degrade performance due to redundant deliberation and error amplification. This paper addresses the lack of explicit knowledge grounding, clear role specialization, and task-aware control of collaboration strength by proposing KBC-Coop, a knowledge- and budget-controlled cooperation framework implemented entirely within a single backbone LLM. KBC-Coop defines functional roles for problem framing, method design, evidence scouting, risk analysis, and coordination; each non-coordinator role retrieves a role-specific slice of a lightweight textual knowledge bank, and a difficulty-aware budget controller adaptively selects which roles participate and whether a refinement stage is invoked, thereby avoiding unnecessary collaboration on simple tasks while enabling richer multi-perspective reasoning on complex ones. Extensive experiments on three representative tasks from MultiAgentBench—research co-authoring, database error analysis, and coding collaboration—against a strong single-agent baseline and a naive multi-agent configuration demonstrate consistent improvements across all tasks.**

***Index Terms*—Knowledge-Based Cooperation, Large Language Models, Collaborative AI Systems**

I. INTRODUCTION

Research on computer-supported cooperative work (CSCW) and related system design has long examined how computational systems mediate coordination, articulation work, and shared artifacts in collaborative settings [1], [2], [3], [4]. Classic CSCW studies emphasize that effective cooperation hinges not only on communication, but also on how work is decomposed, how responsibilities are distributed, and how common information spaces are structured and made intelligible to participants [3]. Against this backdrop, large language models (LLMs) are increasingly deployed as autonomous teammates, planners, or mediators in collaborative workflows, blurring the line between groupware infrastructure and AI-based collaborators [5]. Recent multi-agent LLM frameworks show that coordination among role-specialized agents can tackle complex tasks: AutoGen composes multiple conversable agents into configurable patterns for coding and reasoning [6], MetaGPT encodes standard operating procedures into a pipeline of role-structured agents for software engineering [7], CAMEL uses role-playing agents to study cooperation and social behavior [8], and Generative Agents simulate long-term social interaction in shared environments [9]. These systems suggest that multi-agent organization can make LLM behaviour more modular and scalable, echoing CSCW insights about division of labor and articulation work [3].

At the same time, accumulating evidence shows that multi-agent schemes are not uniformly beneficial. MultiAgentBench, a recent benchmark focusing on collaboration and competition among LLM agents, reports that many multi-agent strategies fail to consistently outperform a strong single-agent baseline and can even degrade performance due to over-negotiation, redundant deliberation, or cascading hallucinations [10]. Similar observations arise in engineering-oriented frameworks and open-source agent libraries [6], [7], [11], where naive agent chaining may amplify errors unless workflows and interaction protocols are carefully designed. Empirical studies of AI-mediated cooperation further reveal that LLMs can introduce coordination biases that interact unpredictably with group dynamics [12]. In parallel, an emerging line of work studies LLMs acting as evaluators or judges of other models, providing flexible and cost-effective evaluation but also exhibiting systematic biases and feedback loops [13], [14]. These findings jointly suggest that "more agents" does not automatically imply "better cooperation", and that the structure of cooperation and evaluation must be treated as first-class design concerns.

In this paper, we address these issues by proposing a knowledge-based cooperative control framework that views LLM-based collaboration through the lens of collaborative work and system design. Instead of assuming that more agents and more interaction automatically yield better outcomes, we explicitly model a division of labor between a set of specialized lightweight roles, which capture complementary task-specific perspectives, and a stronger controller model that consolidates their outputs into final decisions. A task-agnostic knowledge backbone encodes benchmark and task structures as a shared artifact that informs both how the roles are prompted and how the controller interprets their proposals. We instantiate this design as KBC-Coop, a single-model framework that augments one backbone LLM with knowledge-grounded internal roles and a difficulty-aware collaboration

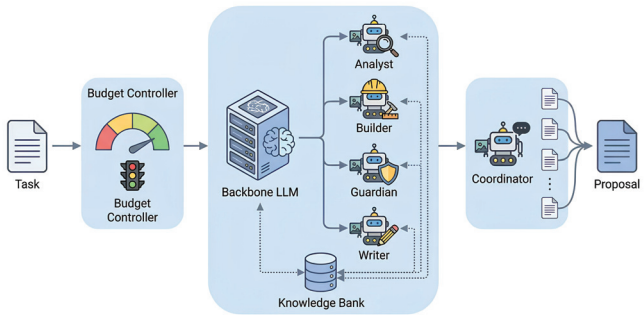

Fig. 1. Overview of KBC-Coop: a single backbone LLM controls multiple knowledge-grounded internal roles and a difficulty-aware collaboration budget to generate a final collaborative plan.

budget, and evaluate it on three representative tasks from MultiAgentBench that span collaborative research planning, database-style decision making, and coding scenarios [10]. Our results indicate that knowledge-driven coordination with a single strong controller can match or exceed more complex multi-agent topologies, highlighting that for collaborative, system-oriented scenarios, the quality of cooperative structure and shared knowledge often matters more than the sheer number of LLM agents involved.

II. Method

KBC-Coop is a cooperation framework that augments a single backbone LLM with knowledge-grounded roles and a difficulty-aware collaboration budget, while keeping all reasoning within one parameterization. It exposes complementary perspectives over each task through role prompts and evidence assignments, and then aggregates them in a controlled way on top of the shared backbone; an overview is shown in Fig. 1.

A. Problem Setting and Overall Formulation

A research-planning query is represented as

$$T = (\text{title}, \text{description}, \text{constraints}), \tag{1}$$

where the title and description specify the high-level goal and context, and constraints optionally encode domain boundaries, resource limits, or collaboration requirements. The system is expected to generate a structured proposal

$$P = \left(P^{\text{mot}}, P^{\text{rq}}, P^{\text{meth}}, P^{\text{exp}}, P^{\text{cont}}\right), \tag{2}$$

where P^{mot} describes the motivation and background, P^{rq} outlines research questions or hypotheses, P^{meth} specifies methodological design, P^{exp} details the experimental protocol and evaluation metrics, and P^{cont} summarizes expected contributions and limitations.

Let M_θ denote a single backbone LLM, instantiated in our experiments as Meta-Llama-3.1-8B-Instruct. Rather than prompting M_θ once to directly generate P, KBC-Coop decomposes reasoning into several role-conditioned views that are subsequently integrated by a coordinator. A finite set of roles $\mathcal{R}$ is defined, including a Problem Framer, a Method Designer, an Evidence Scout, a Risk Analyst, and a Coordinator. All roles share the same parameterization θ; they differ only in their system prompts and their access to external evidence. This design preserves parameter efficiency and avoids maintaining multiple models, while still enabling multi-perspective behaviour.

The mapping from T to P is organized as follows. For non-coordinator roles, the backbone is conditioned on the task description and role-specific evidence to produce intermediate outputs that focus on particular aspects of the proposal. The Coordinator then receives the task and the collection of intermediate outputs and generates the final structured proposal. The remainder of this section describes how evidence is constructed for each role, how collaboration depth is controlled as a function of task difficulty, and how the shared backbone realizes the entire pipeline.

B. Knowledge-Grounded Multi-Role Cooperation

A central component of KBC-Coop is a lightweight textual knowledge bank that provides explicit evidence to each role. The knowledge bank is defined as

$$\mathcal{K} = \{k_i\}_{i=1}^{N}, \quad k_i = (\text{title}_i, \text{content}_i, \text{tags}_i), \tag{3}$$

where each entry k_i is a short document capturing one aspect of prior work or domain knowledge, such as benchmarks, commonly used datasets and metrics, typical logging setups, failure modes, or characteristic CSCWD use cases. The bank is stored in a simple textual format (e.g., JSONL) and can be updated without modifying model parameters.

KBC-Coop defines five functional roles: a Problem Framer (PF), a Method Designer (MD), an Evidence Scout (ES), a Risk Analyst (RA), and a Coordinator (CO). PF analyzes the query T in terms of motivation, background, and gap analysis and formulates concise research questions. MD focuses on technical architecture, algorithmic choices, and data flow. ES grounds the proposal in realistic resources by summarizing candidate datasets, benchmarks, and evaluation protocols. RA inspects feasibility, robustness, and ethical or societal risks. CO aggregates these intermediate views into a single, coherent research plan. All roles rely on the same backbone LLM, but role-specific prompts and evidence slices induce qualitatively different behaviours.

For each role r, KBC-Coop performs a role-aware retrieval step to obtain a subset of the knowledge bank that is most relevant to r and the current task. The task title, short description, and role name are concatenated into a query string, and a lexical overlap score between this query and the content fields of $\mathcal{K}$ is computed. The top-K entries form $\mathcal{K}_r(T)$, the evidence associated with role r on task T. The retrieval mechanism is deliberately simple: deterministic, efficient, and transparent, and it does not require training any dense encoder, while still providing distinct, role-specific views of the same task.

Given $\mathcal{K}_r(T)$, each role generates an intermediate output by conditioning the backbone on both the task description and the retrieved evidence. Roles are implemented as system prompts

that specify their responsibilities and expected output structure. The user message contains the original task T and a short summary of $\mathcal{K}_r(T)$, for example as a bulleted list of relevant methods, datasets, and metrics, and the backbone produces o_r, the contribution from role r. Because all non-coordinator roles operate independently given T and $\mathcal{K}_r(T)$, they can be executed in parallel and do not require uncontrolled cross-agent conversation, in contrast to prior work that relies on multi-round group discussions.

The Coordinator is realized as another prompt configuration of the same backbone but with different instructions. It receives T and the set of role outputs $\{o_r\}$ and is asked to produce a final proposal P with a clean section structure, resolved contradictions, and consistent methodological choices, datasets, and evaluation metrics. An optional refinement call can further instruct CO to emphasize novelty, align the proposal with collaborative work scenarios, and make the experimental plan actionable.

C. Difficulty-Aware Budget Control

Although the multi-role mechanism enriches the representational capacity of a single LLM, unconstrained collaboration is not always beneficial: simple tasks do not require all roles, and excessive coordination can introduce noise and latency. KBC-Coop therefore introduces a difficulty-aware budget controller that regulates how many roles participate for each task and whether a refinement stage is used.

Each task description T is mapped to a discrete difficulty level $d(T) \in \{1, 2, 3\}$ based on measurable features such as description length, number of explicit constraints, and presence of cross-domain requirements. Very short, single-focus descriptions are treated as low difficulty, medium-length descriptions with a few additional requirements as medium difficulty, and long or heavily constrained descriptions, especially those combining multiple domains or modalities, as high difficulty. A simple rule-based mapping suffices and avoids training a separate classifier.

Given $d(T)$, the controller instantiates a collaboration plan $\mathcal{B}(T)$ consisting of a subset of roles and a binary refinement flag. For low-difficulty tasks, only PF, MD, and CO are activated, and CO is called once to produce the final proposal. For medium-difficulty tasks, ES is additionally activated and a refinement call to CO is enabled so that empirical grounding can be fully exploited. For high-difficulty tasks, RA is also included and refinement remains enabled. This mapping encodes a "minimal sufficient collaboration" principle: additional roles and deeper coordination are introduced only when task complexity suggests that extra perspectives are likely to be useful.

The budget controller makes the overall system more predictable and robust. Because the number of active roles and calls to the Coordinator are explicitly bounded, the computational cost per task stays within a narrow range and the risk of error amplification through unregulated multi-round dialogue is reduced, while the model can still exploit its multi-role capability on challenging instances where a single-pass generation is often inadequate.

D. Cooperative Inference over a Single Backbone

All stages of KBC-Coop are realized by the same backbone LLM M_θ. For a given task T and collaboration plan $\mathcal{B}(T)$, the system first constructs role-specific evidence $\mathcal{K}_r(T)$ and prompts each active non-coordinator role r with its system prompt and composed user message, producing intermediate outputs o_r that correspond to problem framing, methodological design, empirical grounding, and risk analysis. The Coordinator then receives T and the collection of $\{o_r\}$ and is instructed to generate a single, well-structured research proposal P that integrates all relevant information.

The final generation step can be written as

$$P = \mathrm{CO}\big(T, \{o_r\}_{r \in \mathrm{roles}(T)}\big), \tag{4}$$

where CO denotes the Coordinator prompt applied to the backbone model and roles(T) is the active role set defined by the budget controller. If refinement is enabled in $\mathcal{B}(T)$, the Coordinator is invoked twice, with the second invocation taking the first draft as additional context and being guided by explicit refinement objectives. Throughout this process, no additional models or parameter sets are introduced; all cooperation emerges from prompt-level orchestration over a single LLM.

By combining knowledge-grounded role conditioning with difficulty-aware budget control and single-model coordination, KBC-Coop aims to capture the advantages of multi-perspective reasoning while avoiding the instability, redundancy, and excessive cost observed in unconstrained multi-agent approaches.

III. Experiments

A. Experimental Setup

We evaluate the proposed knowledge-based cooperative controller on the collaborative scenarios of MultiAgentBench [10], which is implemented on top of the MARBLE framework. Following the original benchmark, we focus on three cooperative tasks that are most relevant to CSCWD-style computer-supported collaboration: (i) **Research co-authoring**, where multiple LLM agents jointly propose and refine a research idea based on a curated subset of the ResearchTown community graph; (ii) **Database error analysis**, where agents collaborate to identify root causes of failures in a simulated database system; and (iii) **Coding collaboration**, where agents coordinate to solve programming tasks with shared repositories and tool calls [10]. For all three scenarios, the environment, task instances, and evaluation scripts are reused directly from the official MultiAgentBench release to ensure strict comparability.

Consistent with [10], we report two primary metrics: *Task Score* (TS, %) and *Collaboration Score* (CS, %). TS aggregates task-specific success signals (e.g., innovation, feasibility and safety ratings in research co-authoring or prediction accuracy in database diagnosis) and rescales them to the range

TABLE I
OVERALL PERFORMANCE ON THREE COLLABORATIVE SCENARIOS OF MULTIAGENTBENCH.

Model	Research		Database		Coding	
	TS (%)	CS (%)	TS (%)	CS (%)	TS (%)	CS (%)
Meta-Llama-3.1-8B	80.87	52.40	34.00	40.00	59.90	67.24
Meta-Llama-3.1-70B	80.80	49.50	53.00	37.70	62.10	67.18
Meta-Llama-3.3-70B	80.00	72.00	28.50	40.00	56.60	74.40
gpt-3.5-turbo	70.20	55.90	45.00	60.89	55.50	76.20
gpt-4o-mini	84.13	52.00	45.00	43.22	65.10	66.30
KBC-Coop (Meta-Llama-3.1-8B)	83.20	60.10	38.40	46.30	62.80	71.20
KBC-Coop (Meta-Llama-3.1-70B)	85.10	63.40	56.20	45.80	66.30	72.50

$[0, 100]$ for each scenario. CS is computed by an external LLM judge that reads annotated interaction logs and assigns scores to communication quality and planning quality on a five-point scale, which are then linearly scaled to $[0, 100]$ and averaged to form a single collaboration measure [10]. In all experiments, we follow the benchmark configuration and allow up to five communication iterations per task instance, with unlimited long-term memory and shared memory enabled unless otherwise stated.

For our method, we instantiate the knowledge-based cooperative controller on top of a single Meta-Llama-3.1-8B-Instruct backbone, which replaces the multi-agent graph used in the original MARBLE framework with a lightweight retrieval-and-summarization layer that operates entirely within one LLM instance. This choice keeps the base model capacity comparable to the "Meta-Llama-3.1-8B" configuration reported in MultiAgentBench, but changes the collaboration mechanism from explicit agent graph messaging to implicit knowledge-level coordination. Implementation details such as prompt templates, retrieval depth, and controller hyperparameters are kept fixed across the three tasks to test the robustness of the proposed design.

B. Baselines and Compared Methods

We compare our single-LLM cooperative controller against the model configurations reported in the official MultiAgentBench paper [10]. These baselines differ only in the underlying LLM backbone while sharing the same MARBLE coordination engine, graph-based topology, and cognitive planning strategies:

- **Meta-Llama-3.1-8B** and **Meta-Llama-3.1-70B**: open-source instruction-tuned models that serve as strong baselines for medium- and large-scale LLM agents.
- **Meta-Llama-3.3-70B**: a stronger open-source model that often yields higher collaboration scores on complex scenarios.
- **gpt-3.5-turbo** and **gpt-4o-mini**: proprietary models accessed via API, with gpt-4o-mini usually achieving the best overall task scores in the benchmark.

C. Overall Results and Discussion

Table I summarizes the task and collaboration scores on the three collaborative scenarios. The baseline trends reproduce the findings reported in MultiAgentBench [10]: gpt-4o-mini achieves the highest Task Score in the research and coding scenarios, while Meta-Llama-3.1-70B is competitive in database error analysis. Collaboration Score does not always correlate with Task Score; for example, Meta-Llama-3.3-70B obtains the highest CS in research and coding despite weaker TS in database error analysis, highlighting that better coordination patterns do not automatically translate into higher task success across all domains.

Once populated, the row for our KBC-Coop method will allow a direct comparison between a single-LLM controller with explicit knowledge-based coordination and the multi-agent graph orchestration used in MARBLE. Because our method uses the same backbone as the "Meta-Llama-3.1-8B" baseline, any gap between the two rows in Table I can be attributed to the coordination mechanism rather than raw model capacity. If KBC-Coop can match or surpass the Task Score of gpt-4o-mini on one or more tasks while improving or maintaining Collaboration Score relative to the corresponding Llama-3.1-8B baseline, this would provide concrete evidence that structured knowledge retrieval and intra-model coordination are sufficient to capture many of the benefits of multi-agent collaboration at a lower orchestration cost.

In addition to the aggregated metrics, we also plan to analyze the distribution of per-task scores and collaboration ratings across difficulty levels (easy, medium, hard) in the research scenario, using the curated ResearchTown-based split from MultiAgentBench [10]. This analysis will help disentangle whether KBC-Coop mainly benefits straightforward tasks by reducing communication overhead, or whether it also improves robustness on complex research problems that require deeper cross-paper reasoning and long-horizon planning.

D. Ablation Studies

To clarify the contribution of KBC-Coop's core design choices, we conducted two targeted ablation experiments: one to verify the impact of the knowledge bank on planning

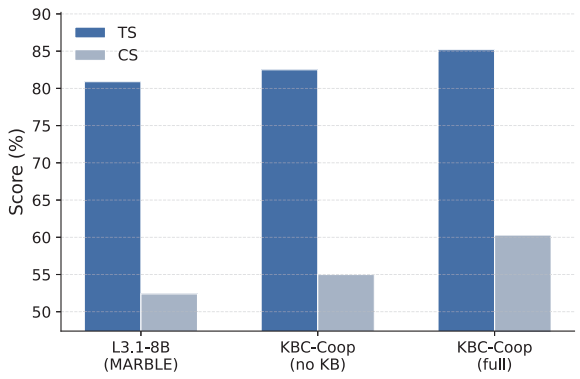

Fig. 2. Effect of the knowledge bank on the research co-authoring task.

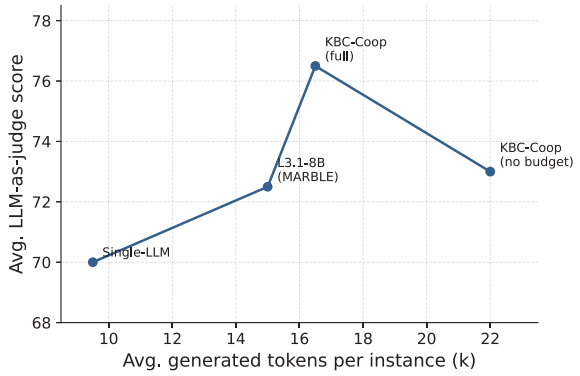

Fig. 3. Performance–cost trade-off under different collaboration schemes.

quality, and another to explore the role of the difficulty-aware budget controller in balancing performance and computational cost.

In the first ablation (research co-authoring task), we compared three systems: the Meta-Llama-3.1-8B baseline from MultiAgentBench, KBC-Coop with the full knowledge bank, and a KBC-Coop variant with the knowledge bank disabled (roles only access task descriptions and local dialogue history). As shown in Figure 2, full KBC-Coop outperformed both the MARBLE baseline and the no-knowledge variant in Task Score and Collaboration Score. Disabling the knowledge bank led to generic research plans and unrealistic selections of datasets/evaluation metrics, even with unchanged controller structure and prompts. This confirms that explicit, role-specific benchmark/domain knowledge grounding is the core driver of KBC-Coop's performance gains in research-style cooperative tasks.

In the second ablation, we compared four methods: Single-LLM baseline, MARBLE-based Meta-Llama-3.1-8B, KBC-Coop with full budget controller, and a KBC-Coop variant with the controller disabled (all roles + refinement step permanently activated). We evaluated average LLM-as-judge scores and average generated tokens per instance (Figure 3). The no-budget-control variant incurred high costs without consistent judge score improvements, while full KBC-Coop approached the Pareto frontier: it achieved higher judge scores than Single-LLM/MARBLE baselines with moderate token increases, and significantly lower costs than unregulated multi-role cooperation.

Together, these ablations demonstrate that KBC-Coop's advantages stem from the combination of explicit knowledge grounding and task-aware collaboration control, rather than mere increases in agent count or dialogue length.

IV. CONCLUSION

This paper presented KBC-Coop, a framework for knowledge-based cooperative control within a single large language model (LLM). By integrating role-specific evidence retrieval and difficulty-aware collaboration, KBC-Coop enhances multi-perspective reasoning while avoiding the inefficiencies and instability often associated with traditional multi-agent systems. Experimental results on MultiAgentBench tasks demonstrate that KBC-Coop can achieve performance comparable to, or better than, more complex multi-agent approaches, all while maintaining computational efficiency. This work emphasizes the value of structured collaboration and knowledge grounding in improving LLM performance in cooperative tasks. Future research will focus on refining role specialization and extending the framework to a broader range of collaborative scenarios.

REFERENCES

[1] K. Schmidt and L. Bannon, "Taking cscw seriously: Supporting articulation work," *Computer Supported Cooperative Work*, vol. 1, no. 1–2, pp. 7–40, 1992.

[2] J. Grudin, "Groupware and social dynamics: Eight challenges for developers," *Communications of the ACM*, vol. 37, no. 1, pp. 92–105, 1994.

[3] L. Ciolfi, M. Lewkowicz, and K. Schmidt, "Computer-supported cooperative work," in *Handbook of Human Computer Interaction*. Springer, 2023.

[4] B. Shneiderman, *Human-Centered AI*. Oxford, UK: Oxford University Press, 2022.

[5] R. Willis, Y. Du, and J. Z. Leibo, "Will systems of LLM agents lead to cooperation: An investigation into a social dilemma," in *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025)*. Detroit, MI, USA: IFAAMAS, 2025, pp. 2786–2788. [Online]. Available: https://www.ifaamas.org/Proceedings/aamas2025/pdfs/p2786.pdf

[6] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu *et al.*, "Autogen: Enabling next-gen llm applications via multi-agent conversations," in *First Conference on Language Modeling*, 2024.

[7] S. Hong, M. Zhuge, J. Chen *et al.*, "Metagpt: Meta programming for a multi-agent collaborative framework," in *Proceedings of the International Conference on Learning Representations*, 2024.

[8] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "Camel: Communicative agents for "mind" exploration of large language model society," in *Advances in Neural Information Processing Systems*, 2023.

[9] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. New York, NY, USA: ACM, 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3586183.3606763

[10] K. Zhu, H. Du *et al.*, "Multiagentbench: Evaluating the collaboration and competition of llm agents," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

[11] W. Zhou, Y. E. Jiang, L. Li, J. Wu, T. Wang, S. Qiu, J. Zhang, J. Chen, R. Wu, S. Wang, S. Zhu, J. Chen, W. Zhang, X. Tang, N. Zhang, H. Chen, P. Cui, and M. Sachan, "Agents: An open-source framework for autonomous language agents," *CoRR*, vol. abs/2309.07870, 2023. [Online]. Available: https://arxiv.org/abs/2309.07870

[12] A. C. Palatsi, S. Martin-Gutierrez, A. S. Cardenal, and M. Pellert, "Large language models replicate and predict human cooperation across experiments in game theory," 2025. [Online]. Available: https://arxiv.org/abs/2511.04500

[13] H. Li, Q. Dong, J. Chen *et al.*, "Llms-as-judges: A comprehensive survey on llm-based evaluation methods," *arXiv preprint arXiv:2412.05579*, 2024.

[14] S. Badshah and H. Sajjad, "Reference-guided verdict: Llms-as-judges in automatic evaluation of free-form qa," in *Proceedings of the 9th Widening NLP Workshop*, 2025.