CoLLAB: A Framework for Designing Scalable Benchmarks for Agentic LLMs

Saaduddin Mahmud
University of Massachusetts Amherst
smahmud@umass.edu

Eugene Bagdasarian
University of Massachusetts Amherst
eugene@umass.edu

Shlomo Zilberstein
University of Massachusetts Amherst
shlomo@umass.edu

Abstract

Agents capable of making decisions from complex, unstructured instructions have seen a surge with the rise of large language models (LLMs). However, their ability to coordinate with other agents while following instructions is still an active area of research. To facilitate research in this area, we introduce a framework for designing scalable environments to evaluate coordination in agentic LLM networks, called *Coordinating LLM Agents Benchmark* (CoLLAB). CoLLAB adapts a widely used classical cooperative multi-agent problem-solving framework called Distributed Constraint Optimization Problems (DCOPs), and extends it with unstructured instructions and communication, making it directly relevant for studying coordination in agentic LLM networks. We provide a design blueprint for how CoLLAB environments can scale across multiple dimensions. Finally, we implement three case study environments within this framework and evaluate several LLM-based agent configurations. We then quantitatively analyze LLM-generated solutions against classical symbolic solvers to directly assess their quality. In addition, we demonstrate how CoLLAB supports seamless scaling of environment complexity, allowing us to design increasingly challenging coordination tasks and assess how different agents adapt.

Code: https://github.com/Saad-Mahmud/CoLLAB_SEA
Page: https://agents-collab.github.io/

1 Introduction

Large language models (LLMs) have rapidly advanced to the point where they are increasingly deployed as autonomous agents capable of planning, reasoning, and acting based on complex, unstructured instructions [1–3]. While most existing research has centered on single-agent capabilities, recent work has begun to explore the emergent behaviors of LLMs in multi-agent settings [4, 5]. An important direction in this area focuses on understanding how a network of LLM agents communicates and coordinates to jointly solve tasks. However, progress remains limited by the lack of standardized, scalable benchmarks for evaluating coordination in agentic LLM networks. Current testbeds are typically ad hoc—narrow in scope, difficult to scale systematically, and lacking well-defined baselines for meaningful quantitative comparison across studies.

To address this gap, we introduce the *Coordinating LLM Agents Benchmark* (CoLLAB), a framework that adapts the widely studied cooperative multi-agent problem-solving paradigm of Distributed Constraint Optimization Problems (DCOPs) [6, 7] to settings involving unstructured instructions and

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Scaling Environments for Agents (SEA).

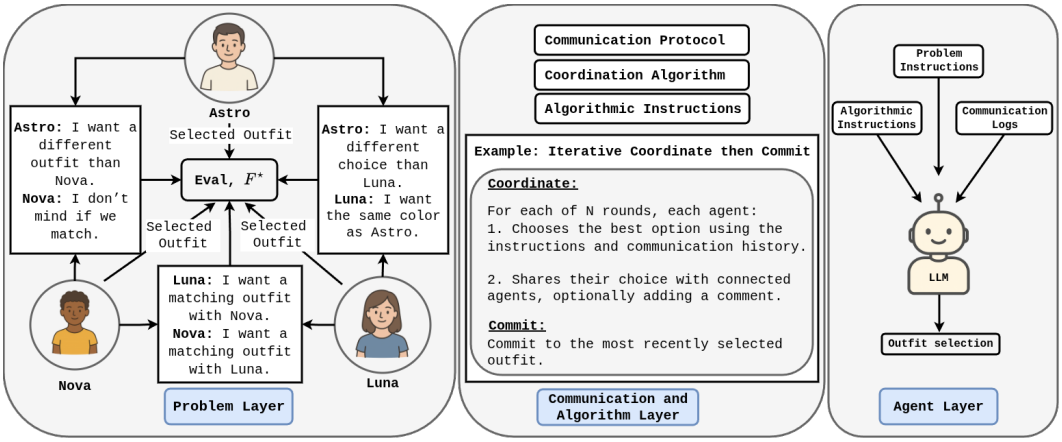

Figure 1: Overview of the CoLLAB framework architecture. **Left: Problem Layer.** Each agent receives personalized natural-language instructions and context. **Center: Communication and Algorithm Layer.** Agents coordinate through structured protocols and algorithms, sharing decisions and rationales over multiple rounds. **Right: Agent Layer.** Each LLM agent integrates problem instructions, algorithmic rules, and communication logs to make a decision.

natural-language communication. CoLLAB is organized into three layers: the *problem layer*, the *communication and algorithm layer*, and the *agent layer* (Figure 1). The problem layer combines a symbolic DCOP backbone with rich, multimodal instructions that specify goals and provide context for each agent. The communication and algorithm layer supports both structured and unstructured exchanges between agents and defines the protocols and algorithms for network interaction. Finally, the agent layer details how LLM agents interpret inputs and generate decisions.

This layered design enables CoLLAB to flexibly scale problem complexity along multiple axes within the same underlying coordination problem: the problem layer can vary instruction modality, decision-making complexity, and agent count; the communication layer controls message structure, volume, and the coordination algorithm used to reach consensus; and the agent layer allows for scaling of reasoning capabilities, context window, or deliberation style. Researchers can thus tune problem difficulty and systematically observe the effects of environment complexity and protocol variation on coordination performance. Importantly, the symbolic backbone enables direct, quantitative comparison of LLM-generated solutions against established DCOP solvers, providing a strong and transparent baseline for measuring progress.

We instantiate three case study environments within this framework: a *Personal Assistant* domain for organizing social events, a *Meeting Scheduling* domain with overlapping and conflicting availabilities, and a *Smart-Grid/Home* domain for collaborative energy optimization. Across these environments, we systematically scale key dimensions of complexity and evaluate multiple LLM-based agent configurations, including both CoT-augmented and standard models. By directly comparing their performance to symbolic DCOP solvers, we obtain quantitative measures of coordination quality and a clear understanding of how increasing environment complexity affects agent behavior. Our results demonstrate that CoLLAB offers a principled and extensible foundation for benchmarking agentic LLMs in controlled, scalable, and comparable settings. Beyond comprehensive evaluation, our discussion also reveals the framework's future potential for enabling systematic study of multi-agent safety, security, and privacy.

2 Related Work

LLM agent environments. A growing body of work has developed multi-agent evaluation suites for LLMs that go beyond single-agent testbeds [8–11]. Notable examples include AgentsNet [4], which explores whether LLM agents can self-organize over graph topologies and solve canonical distributed problems; MultiAgentBench [5], which orchestrates diverse collaborative and competitive tasks under configurable communication structures; and Decrypto [12], which focuses on interactive language games to probe theory of mind and coordination under information hiding. In contrast, CoLLAB

is grounded in a shared formal backbone (DCOPs), layering on natural language and image-based instructions along with a broad range of communication protocols. This design affords two key advantages: (i) environment complexity can be scaled orthogonally within the same optimization paradigm, and (ii) agent performance can be quantitatively measured and compared against symbolic DCOP solvers, yielding reproducible, solver-anchored baselines rather than bespoke or heuristic scoring.

LLMs on combinatorial problems. Several recent works evaluate LLMs on standalone combinatorial or constraint reasoning tasks (e.g., Sudoku, logical puzzles, NP-hard optimization) to assess their capacity for internal inference and search from static descriptions [13–15]. These approaches treat the model as a single solver, assuming full observability and no interactive coordination. In contrast, COLLAB preserves the combinatorial optimization core but embeds it within a cooperative multi-agent setting. This design reveals coordination phenomena such as ambiguity resolution, negotiation, exception handling, and asymmetric information—factors absent from single-agent puzzle settings.

Distributed Constraint Optimization Problems (DCOPs). DCOPs formalize coordination in multi-agent systems by assigning each agent control over one or more variables; agents choose values to maximize a global utility, typically defined as the sum of local utility functions [16]. Classical DCOP solvers—complete search, asynchronous bounding (e.g. ADOPT), inference/message-passing (e.g. DPOP), and local search methods (e.g. DSA)—assume fully specified symbolic models: well-formed variables, deterministic numeric utility functions, and engineered message schemas [6, 17–19]. These methods perform best when the problem structure, constraint scopes, and communication protocols are known in advance. In contrast, our work treats LLMs as agents operating under more realistic, language-mediated interfaces: COLLAB retains a DCOP backbone but replaces symbolic descriptions with natural language and image-based specifications, and makes communication protocols customizable—from structured message schemas to free-form dialogue. This framing preserves the evaluative rigor of DCOPs while surfacing LLM-specific coordination failure modes (e.g., misinterpretation, hallucination, and ambiguity).

3 Background: Distributed Constraint Optimization Problems

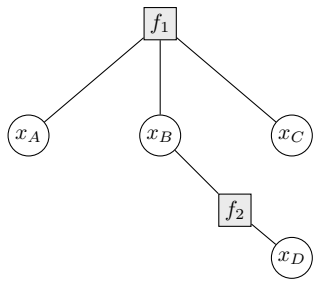

Figure 2: Factor graph: variables/agents (circles), factors/utility (squares).

A distributed constraint optimization problem (DCOP) is a tuple

$$\mathcal{P} = \langle \mathcal{A}, \mathcal{X}, \{\mathcal{D}_x\}_{x\in\mathcal{X}}, \sigma, \mathcal{F} \rangle,$$

where $\mathcal{A}$ is a set of agents; $\mathcal{X}$ is a set of variables; each $x \in \mathcal{X}$ has a finite domain $\mathcal{D}_x$; $\sigma : \mathcal{X} \to \mathcal{A}$ maps each variable to its controlling agent; and $\mathcal{F} = \{f_\alpha\}$ is a set of local utility factors, each with scope $\mathcal{X}^\alpha \subseteq \mathcal{X}$ and function $f_\alpha : \prod_{x\in\mathcal{X}^\alpha} \mathcal{D}_x \to \mathbb{R}$. The objective is

$$\max_{x\in\prod_{x\in\mathcal{X}} \mathcal{D}_x} \sum_\alpha f_\alpha\big(x^\alpha\big).$$

A DCOP can be visualized as a bipartite factor graph with variable/agent nodes $\{x\}$, factor nodes $\{f_\alpha\}$, and edges whenever $x \in \mathcal{X}^\alpha$; This structure (Figure 2) supports search, message-passing, and local improvement–based solvers. Common extensions retain the same objective while enriching modeling: *Dynamic DCOPs* (e.g., PD - DCOPs) explicitly model evolving factors, domains, or agent sets and can exploit predictions about future change [20]; *Contextual DCOPs* (often formalized as *Probabilistic DCOPs*, P - DCOPs) augment factors with exogenous variables c, yielding context-conditioned costs $f_\alpha(\cdot; c)$ [16]; *Asymmetric DCOPs (ADCOPs)* assign agent-specific utilities on shared scopes to capture heterogeneous preferences and privacy [21]. Standard DCOP algorithms formalize the protocols by which agents exchange information during the coordination phase and delineate the mechanisms for achieving consensus and committing to a joint assignment.

4 The CoLLAB Framework

CoLLAB is organized into three layers: (i) the *problem layer*, which defines the DCOP instance and encodes local constraints through both structured representations and unstructured modalities such as text or images; (ii) the *communication and algorithm layer*, which governs how agents exchange information and coordinate their decisions; and (iii) the *agent layer*, which specifies the internal reasoning process of each agent, including its inference style, memory usage, and decoding strategy.

4.1 Problem Layer

Instance. Each instance is

$$\Theta \;=\; \big\langle \mathcal{A},\, \mathcal{X},\, \{\mathcal{D}_x\}_{x\in\mathcal{X}},\, \mathcal{C},\, \mathcal{F}^\star,\, \rho \big\rangle. \tag{1}$$

Agents and roles. Agents split into instructors and actors:

$$\mathcal{A} \;=\; \mathcal{A}^{\mathrm{inst}} \,\dot\cup\, \mathcal{A}^{\mathrm{act}}, \qquad \mu : \mathcal{A}^{\mathrm{act}} \to \mathcal{A}^{\mathrm{inst}},$$

where $\mu(i)$ pairs actor i to its instructor.

Decision variables and ownership. Let $\mathcal{X} = \{x_1, \dots, x_m\}$ with finite domains $\mathcal{D}_x$. Actors own a disjoint partition of variables:

$$\mathcal{X} \;=\; \biguplus_{i\in\mathcal{A}^{\mathrm{act}}} \mathcal{X}_i, \tag{2}$$

and actor i's assignment is $\mathbf{x}_i \in \prod_{x\in\mathcal{X}_i} \mathcal{D}_x$. The joint assignment is

$$\mathbf{x} \;=\; \big(\mathbf{x}_i\big)_{i\in\mathcal{A}^{\mathrm{act}}} \;\in\; \prod_{x\in\mathcal{X}} \mathcal{D}_x. \tag{3}$$

Context. Here, $\mathcal{C}$ denotes the context space. For each instructor $u \in \mathcal{A}^{\mathrm{inst}}$, let $\mathcal{C}_u$ denote the local context space and $c_u \in \mathcal{C}_u$ its realization. Collectively,

$$\mathbf{c} \;=\; (c_u)_{u\in\mathcal{A}^{\mathrm{inst}}} \;\in\; \prod_{u\in\mathcal{A}^{\mathrm{inst}}} \mathcal{C}_u. \tag{4}$$

Utility factors. The idealized utility decomposes by instructor:

$$\mathcal{F}^\star \;=\; \big\{\mathcal{F}^\star_u\big\}_{u\in\mathcal{A}^{\mathrm{inst}}}, \quad \mathcal{F}^\star_u \;=\; \big\{f^\star_{u,\beta}\big\}_{\beta=1}^{k_u}, \quad f^\star_{u,\beta} : \Big(\prod_{x\in S_{u,\beta}} \mathcal{D}_x\Big) \times \mathcal{C}_u \to \mathbb{R}, \tag{5}$$

and the ground-truth objective is

$$F^\star(\mathbf{x};\mathbf{c}) \;=\; \sum_{u\in\mathcal{A}^{\mathrm{inst}}} \sum_{\beta=1}^{k_u} f^\star_{u,\beta}\big(\mathbf{x}_{S_{u,\beta}};\, c_u\big). \tag{6}$$

Instructions. Let $\mathcal{I}$ denote the instruction space (e.g., multimodal). For each instructor $u \in \mathcal{A}^{\mathrm{inst}}$,

$$\rho_u :\; \mathcal{F}^\star_u \times \mathcal{C}_u \;\to\; \mathcal{I}. \tag{7}$$

The instruction delivered to actor i is generated from its paired instructor's local context:

$$I_i(\mathbf{c}) \;=\; \rho_{\mu(i)}\big(\,\mathcal{F}^\star_{\mu(i)},\, c_{\mu(i)}\,\big) \;\in\; \mathcal{I}. \tag{8}$$

Policies. Each actor i maps the received instruction to an assignment:

$$\pi_i :\; \mathcal{I} \;\longrightarrow\; \prod_{x\in\mathcal{X}_i} \mathcal{D}_x, \qquad \mathbf{x}_i \;=\; \pi_i\big(I_i(\mathbf{c})\big). \tag{9}$$

Execution and learning objective. Given $\pi = (\pi_i)_{i\in\mathcal{A}^{\mathrm{act}}}$ and local contexts $\mathbf{c}$, the induced joint assignment is

$$\mathbf{x}(\pi,\mathbf{c}) \;=\; \big(\,\pi_i\big(\,I_i(\mathbf{c})\big)\,\big)_{i\in\mathcal{A}^{\mathrm{act}}}. \tag{10}$$

Let $P_{\mathcal{C}}$ be a distribution over contexts. The goal is to find policies maximizing expected utility:

$$\max_{\pi} \;\; \mathbb{E}_{\mathbf{c}\sim P_{\mathcal{C}}}\big[\,F^\star\big(\mathbf{x}(\pi,\mathbf{c});\,\mathbf{c}\big)\,\big]. \tag{11}$$

4.2 Communication and Algorithm Layer

This layer defines how LLM agents coordinate to solve a shared problem instance, specifying the interaction protocol, communication topology, and prompting structure used to elicit decisions. Coordination can follow different paradigms, such as *centralized coordination*, where agents communicate summaries of their local preferences to a coordinating entity that proposes a joint solution, or iterative decentralized schemes like the *Iterative Coordinate–then–Commit (ICC)* protocol, in which agents exchange partial decisions over multiple rounds before finalizing their commitments. In this work, we focus on a simple greedy variant of ICC, where agents iteratively select their best local action given the current context. However, the framework is compatible with a wide range of distributed optimization and coordination algorithms from the DCOP literature, including DSA, MGM, Max-Sum, and ADOPT [16]. Communication within each coordination round may range from minimal *decision-only* exchanges to richer formats that incorporate natural-language rationales. Moreover, a key strength of this layer is its ability to flexibly impose normative behavioral framings through prompt and instruction design, allowing agents to adopt self-interested, team-oriented, or neutral perspectives that shape the balance between individual and collective objectives.

4.3 Agent layer.

The *agent layer* defines how each agent interprets its instructions, contextual information, and received messages—potentially leveraging multi-step reasoning, function calling, or constrained JSON-mode generation. Regardless of the internal reasoning process, each agent ultimately produces a structured output (e.g., JSON) to ensure that decisions can be consistently parsed and evaluated by the problem layer.

4.4 Scaling dimensions.

CoLLAB enables systematic scaling of difficulty and complexity along several independent axes, at each layer of the framework:

1. **Scaling agent size and connectivity.** The environment can be scaled by increasing the number of agents or by expanding each agent's interaction neighborhood, thereby raising coordination complexity.
2. **Decision complexity.** This dimension captures the combinatorial difficulty faced by each agent, and can be scaled by increasing the number of variables per agent, enlarging domain sizes, or raising the arity of local factors—expanding the space of possible assignments and the depth of required reasoning.
3. **Instruction modality and abstraction.** Through the rendering map ρ, instructions I_i can be delivered as text, images, or multimodal inputs.
4. **Interaction protocol and communication budget.** The communication layer accommodates both structured and open-ended coordination protocols. Agents can exchange concise decision messages or extended natural-language rationales, with the latter demanding additional LLM reasoning. The number of coordination rounds and message modalities can be varied to examine the trade-off between communication cost and cooperative effectiveness.
5. **Normative instruction.** Beyond pure utility maximization, agents can be guided by different normative frames—such as *self-oriented*, *team-oriented*, or *neutral* instructions. These prompts shape agents' social objectives and can induce complex cooperative or competitive behaviors, often requiring deeper contextual and ethical reasoning.
6. **Agent scaling and heterogeneity.** The framework supports scaling not only in the number of agents but also in their capabilities, e.g., varying agent reasoning strategies, access to history, or different policy classes.

5 Case Studies: Three CoLLAB Environments

We instantiate CoLLAB across three cooperative domains, each highlighting distinct coordination challenges.

Meeting Scheduling.

- **Setup:** Agents act as meeting participants who must coordinate attendance over a shared discrete timeline. Each instance defines a set of meetings with varying participation requirements and temporal overlaps, creating interdependent scheduling choices across agents.
- **Instructions:** Agents receive descriptions of their assigned meetings and simple scheduling rules. In the multimodal variant, this information is presented visually through timeline-based depictions rather than text.
- **Objective:** Encourage coordinated attendance among participants while avoiding schedule conflicts, balancing collective meeting overlap with individual feasibility constraints.

Personal Assistant.

- **Setup:** Agents select outfits from their wardrobes. Each garment has categorical attributes and optionally an image. Pairwise factors encourage (mis)matching color palettes across socially linked agents.
- **Instructions:** Text describes each agent's likes, dislikes, and requested matches with neighbors. For the vision setting, we automatically generate per-agent collages summarizing available wardrobe options.
- **Objective:** Maximize the sum of individual preference scores, plus rewards or penalties for satisfying requested color relationships with neighbors.

Smart Grid.

- **Setup:** Agents represent sites that must assign each of their devices to an available renewable energy source from which it draws power. When a selected source lacks sufficient capacity, devices default to non-renewable energy, introducing a coordination dependency across agents sharing the same renewable sources.
- **Instructions:** Agents receive information about their connected energy sources and devices, along with guidance for balancing loads to minimize overuse of shared capacity. In the multimodal variant, this information is presented visually through energy-flow and capacity diagrams rather than text.
- **Objective:** Coordinate source assignments to reduce reliance on non-renewable energy by efficiently balancing renewable usage across all agents.

6 Experimental Analysis

In this section, we examine the scalability of different layers within the CoLLAB framework across the three problem domains introduced earlier. We evaluate how varying these layers impacts overall solution quality relative to symbolic baselines. Finally, we discuss prospective research directions informed by these observations.

6.1 Experiment Configurations

Problem Configurations. For each domain, we generate problem instances under two regimes: Dense and Sparse. Dense instances contain a greater number of decision variables per agent and more shared factors between agents compared to the sparse setting. We generate 20 instances for each regime, and Table 1 summarizes the structural statistics averaged across these instances. The number of agents is fixed at 10, as the decomposed nature of DCOPs makes problem complexity more sensitive to density than to the raw agent count. Nevertheless, our implementation readily supports the generation of problems with an arbitrarily large number of agents. For each instance, we also generate images that visually represent a subset of the textual description. In the multimodal setting, the corresponding text is hidden, and agents are instructed to rely solely on the provided images for information.

Meeting Scheduling: Each instance includes roughly 15 meetings distributed over a compact 7-slot timeline. Participation arity scales with density: *sparse* instances sample 2–6 participants per meeting, while *dense* instances sample 3–8.

Domain	Regime	Agents	Decision Variables / Agent	Factors / Agent
Meeting Scheduling	Dense	10	8.4	49.7
	Sparse	10	6.2	29.3
Personal Assistant	Dense	10	1.0	4.2
	Sparse	10	1.0	2.9
Smart Grid	Dense	10	5.0	5.5
	Sparse	10	3.6	4.1

Table 1: Average structural properties of the problem domains.

Personal Assistant: Each agent is assigned a wardrobe containing 4–7 possible outfit combinations. Sparsity is controlled through a graph edge density parameter in the instance generator, with *sparse* instances using a density of approximately 0.15 and *dense* instances around 0.50.

Smart Grid: Each instance spans a 24-slot scheduling horizon. Agents represent homes equipped with multiple renewable sources and household machines: *sparse* instances include 2–3 sources and 3–4 machines per home, while *dense* instances include 3–5 sources and 4–6 machines.

LLM agent and compute. We evaluate three large language models—gemma-3-27B-IT, gpt-4.1-nano, and qwen3-30B-A3B-Instruct—across two agent types: a JSON-only agent, which directly outputs structured responses, and a Chain-of-Thought (CoT) agent, which first performs step-by-step reasoning before producing its final structured output. Inference was executed on a server with $8\times$ A100 GPUs, totaling approximately 768 GPU-hours across all domains and instance seeds for the open-source models, while running the same experiments on GPT-4.1-Nano incurred an overall inference cost of roughly $\$10$.

Symbolic baselines. We evaluate three lightweight baselines using the symbolic backbone. The first, **Oracle-SA**, is a simulated annealing-based solver. The second, **Oracle-RS**, is a random sampling method that repeatedly generates joint assignments (100 assignments used)and reports the best observed utility. For comparison, we also include a purely **Random** baseline that reports the average utility of uniformly sampled complete assignments.

Communication and Normative Instruction We employ the ICC protocol with 5 coordination rounds (described in Section 4). Two communication modes are considered: one in which agents exchange only *decision messages (DM)*, and another in which they also exchange *natural language messages (NM)*. Furthermore, we examine three types of normative instruction: (1) *Neutral*, where no specific behavioral guidance is provided; (2) *Self-oriented*, where agents are instructed to optimize for their individual objectives; and (3) *Team-oriented*, where agents are encouraged to act in the collective interest. Unless otherwise specified, the default configuration is *DM + Neutral*.

Evaluation. Performance is measured using the average normalized utility (ANU). For each instance, the achieved utility is scaled relative to the range between the minimum and maximum attainable utilities for that instance. Results are then averaged over 20 randomly generated instances per setting, with both the mean and standard deviation reported.

6.2 Result Analysis

In Figure 3, we examine the impact of problem density on different agent types. Across all domains, LLM-based agents consistently underperform compared to the Oracle-SA solver. In most settings, their performance falls between Oracle-RS and the Random baseline, except in the *Personal Assistant* domain, where the LLM-based agent slightly surpasses Oracle-RS in the sparse regime. Overall, LLM agents appear more competitive with symbolic baselines in the sparse regime. We also observe no significant difference between the CoT and JSON-only modes.

Figure 4 illustrates the effect of instruction modality, comparing image-based versus text-based inputs. Interestingly, in the *Meeting Scheduling* domain, agents achieve higher performance when provided with image-based instructions, whereas in other domains, text-based instructions yield better results.

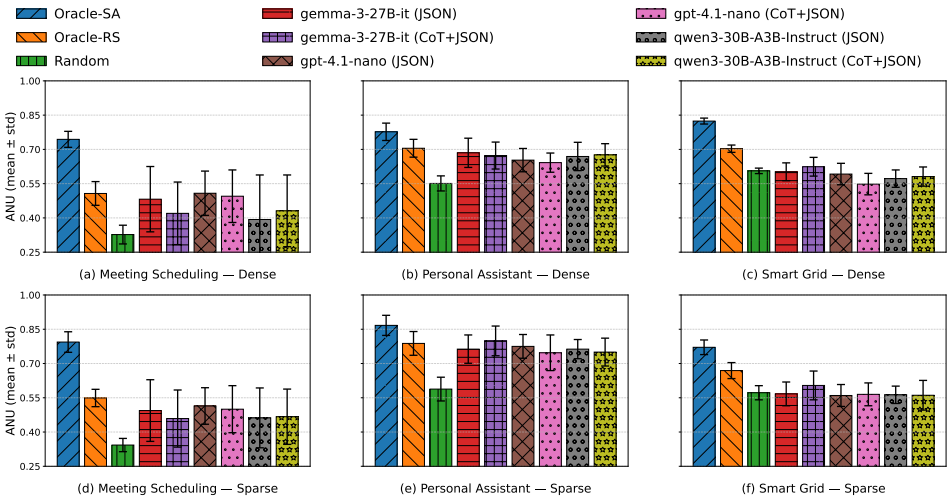

Figure 3: Performance of LLM agents and symbolic baselines across CoLLAB domains as problem size and interaction complexity are scaled.

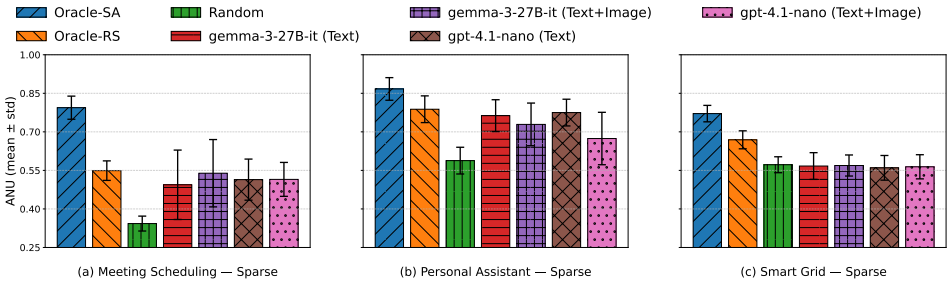

Figure 4: Influence of instruction modality on the performance of LLM agents.

This suggests that the design and clarity of the instruction modality can meaningfully influence agent reasoning—image representations may be more complex to parse, yet occasionally convey contextual information more effectively.

Figure 5 presents the impact of communication type, contrasting natural language messages (NM) with decision messages (DM). Performance remains similar across both modes, indicating that agents did not successfully leverage the additional communicative capacity provided by NM.

Finally, Figure 6 explores the role of normative strategy. Results show that behavioral framing can substantially affect coordination outcomes, though its effect varies by domain: in *Meeting Scheduling*, the team-oriented strategy performs best, while in *Personal Assistant*, the selfish strategy yields higher utility.

6.3 Discussion

Based on the above analysis, we identify several promising directions for improving coordination in LLM-based agent networks. (1) We observed that local solutions occasionally oscillate, with the best outcome emerging in intermediate rounds rather than at convergence. Symbolic solvers address such cases using *anytime mechanisms*, which track and retain the best solutions discovered throughout the optimization process. Because LLM agents currently lack access to the true global utility, designing an effective anytime mechanism is nontrivial; however, our results suggest that such methods could improve performance by approximately 1–3% in the domains we designed. (2) Iterative coordination protocols are computationally expensive when implemented with LLMs, motivating the study of *non-iterative* or *single-shot* coordination methods that reduce computational cost and mitigate convergence issues. (3) More sophisticated agent architectures are needed: simple prompt-based CoT reasoning and natural-language message generation do not yet exploit the additional reasoning capacity and communication bandwidth available to the agents. (4) Finally, future protocols could allow *actor*

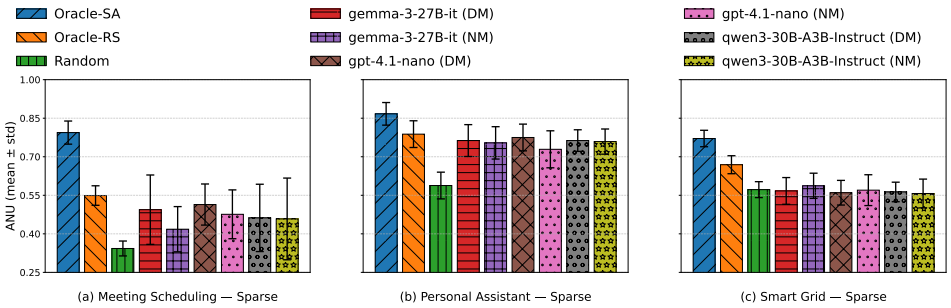

Figure 5: Influence of communication protocol on the performance of LLM agents.

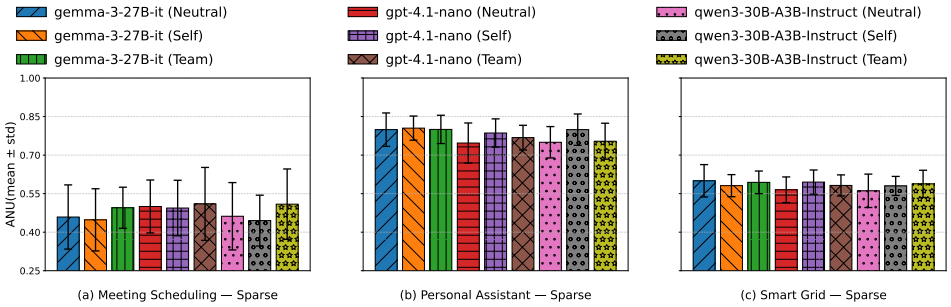

Figure 6: Influence of normative instruction on the performance of LLM agents.

agents ($\mathcal{A}^{\text{act}}$) to query *instructor agents ($\mathcal{A}^{\text{inst}}$)* for clarification during execution, enabling dynamic disambiguation of instructions.

Additionally, the proposed method is well-suited for investigating vulnerabilities and adversarial behaviors in distributed agent networks. For example, one could study the impact of compromised agents that deliberately issue misleading or malicious responses, or probe privacy risks related to information leakage through inter-agent communication. While prior work has focused on single-agent security challenges [22], our multi-agent framework opens the door to a broader understanding of both attack surfaces and potential mitigation strategies in collaborative agentic systems.

7 Conclusion

We introduced CoLLAB, a flexible framework for constructing scalable benchmarks to evaluate large language models as coordinating agents. By instantiating CoLLAB across three representative domains—Personal Assistant, Meeting Scheduling, and Smart Grid—we systematically varied interaction topology, decision complexity, instruction modality, communication protocol, and normative strategy to examine the capabilities and limitations of current LLM-based agents. Looking ahead, we plan to extend CoLLAB with additional problem dimensions, richer communication protocols, and improved agent architectures.

Acknowledgments

This research was supported in part by the National Science Foundation grants 2205153, 2321786, and 2416460, and by Schmidt Sciences under the AI Safety Science program.

References

[1] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, and et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. URL `https://arxiv.org/abs/2108.07258`.

[2] Shunyu Yao, Dian Zhao, Jeffrey Yu, Izhak Shafran, Tom Griffiths, Kuang-Huei Cao, and Karthik Narasimhan. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. URL https://arxiv.org/abs/2210.03629.

[3] Noah Shinn, Mark Labash, and Ashwin Gopinath. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

[4] Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems. *arXiv preprint arXiv:2504.00587*, 2025.

[5] Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. Multiagentbench: Evaluating the collaboration and competition of LLM agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8580–8622, 2025. URL https://aclanthology.org/2025.acl-long.421.pdf.

[6] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2): 149–180, 2005. doi: 10.1016/j.artint.2004.09.003.

[7] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, 1998. doi: 10.1109/69.729715.

[8] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 3135–3144, 2017. URL https://proceedings.mlr.press/v70/shi17a/shi17a.pdf.

[9] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://openreview.net/forum?id=ryTp3f-0-.

[10] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *arXiv preprint arXiv:2207.01206*, 2022. URL https://arxiv.org/abs/2207.01206.

[11] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL https://arxiv.org/abs/2307.13854.

[12] Andrei Lupu, Timon Willi, and Jakob Nicolaus Foerster. The decrypto benchmark for multi-agent reasoning and theory of mind. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=kFoJXqiGKz. Submitted; OpenReview ID: kFoJXqiGKz, version updated 2025-02-05.

[13] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. 2023. URL https://arxiv.org/abs/2305.10601. NeurIPS 2023 camera-ready version.

[14] Adam Ishay, Zhun Yang, and Joohyung Lee. Leveraging large language models to generate answer set programs. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning (KR 2023)*, 2023. URL https://proceedings.kr.org/2023/37/kr2023-0037-ishay-et-al.pdf.

[15] Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. https://machinelearning.apple.com/

research/illusion-of-thinking, 2025. Apple Machine Learning Research white paper. PDF: https://ml-site.cdn-apple.com/papers/the-illusion-of-thinking.pdf.

[16] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research*, 61:623–698, 2018. doi: 10.1613/jair.5565.

[17] Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005. URL https://www.ijcai.org/Proceedings/05/Papers/0445.pdf.

[18] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 639–646, 2008. URL https://www.robots.ox.ac.uk/~argus/papers/Decentralised%20Coordination%20of%20Low-Power%20Embedded%20Devices%20Using%20the%20Max-Sum%20Algorithm.pdf.

[19] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. Distributed stochastic search and distributed breakout: Properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2):55–87, 2005. doi: 10.1016/S0004-3702(04)00148-1.

[20] Khoi D. Hoang, Ferdinando Fioretto, Ping Hou, William Yeoh, Makoto Yokoo, and Roie Zivan. Proactive dynamic distributed constraint optimization problems. *Journal of Artificial Intelligence Research*, 74:179–225, 2022. doi: 10.1613/jair.1.13499.

[21] Tal Grinshpoun, Alon Grubshtein, Roie Zivan, Arnon Netzer, and Amnon Meisels. Asymmetric distributed constraint optimization problems. *Journal of Artificial Intelligence Research*, 47: 613–647, 2013. doi: 10.1613/jair.3945.

[22] Eugene Bagdasarian, Ren Yi, Sahra Ghalebikesabi, Peter Kairouz, Marco Gruteser, Sewoong Oh, Borja Balle, and Daniel Ramage. AirGapAgent: Protecting privacy-conscious conversational agents. In *CCS*, 2024.