EnterpriseBench: Benchmarking LLM Agents on Enterprise-Level Strategic Reasoning and Decision-Making

Anonymous ACL submission

Abstract

As Large Language Model (LLM) agents demonstrate increasingly strong reasoning capabilities, they are being progressively adopted to support complex decision-making in enterprise environments. However, existing benchmarks primarily evaluate reasoning through static, single-shot tasks with well-defined objectives and immediate correctness, whereas real-world decision-making is inherently interactive, open-ended, and shaped by delayed consequences and competing goals. To address this challenge, we introduce **EnterpriseBench**, a comprehensive benchmark for assessing LLM agents in realistic enterprise contexts. EnterpriseBench covers a hierarchy of reasoning demands, ranging from information extraction, numerical and domain knowledge reasoning to high-fidelity interactive decision-making, including management consulting cases and serious games. We also introduce an agent-oriented taxonomy that organizes tasks by capability domains and intrinsic difficulty. Empirical evaluations across nine state-of-the-art agent architectures reveal a natural division of labor, where simpler tasks favor lightweight models and complex decision-making benefits from stronger reasoning agents. Building on this insight, we present **AOA**, an Agent-of-Agents framework that integrates the complementary capabilities of such specialized agents. Our codes and benchmark implementation are publicly available[1].

1 Introduction

Recent advances in Large Language Models (LLMs) and LLM-based agents have substantially enhanced their reasoning capabilities, enabling them to address increasingly complex decision-making problems (Liu et al., 2024; Yang et al., 2025; Zhang et al., 2025a, 2024; Yu et al., 2024; Li et al., 2025; Xing, 2025). However, most existing benchmarks (Zhang et al., 2025c; Mohammadi et al., 2025; Zhu et al., 2024; Chen et al., 2021) still focus on static and well-defined tasks such as mathematical reasoning (Glazer et al., 2024; Fan et al., 2024), programming (Jimenez et al., 2023), and question answering (Chen et al., 2021), where conditions are fixed and correctness can be verified against a single ground-truth answer. However, real-world decision-making is far more complex, a gap that becomes especially evident when LLM agents are deployed in enterprise settings. For example, when a firm considers expanding production capacity, it must reason under uncertain future demand, anticipate competitors' responses, allocate limited capital across alternative investments, and balance short-term costs against long-term profitability. Such decisions are ubiquitous in real-world enterprise settings, shape future states of the environment, involve delayed and potentially irreversible consequences, and rarely lead to a single correct solution.

Fortunately, recent LLM agents have begun to demonstrate surprising competence in such complex decision-making scenarios (Raptis et al., 2025). LLM agents have been explored for investment decision-making tasks (Xiao et al., 2024; Chen et al., 2025b). In these settings, agents must reason over incomplete and noisy information, make sequential decisions, and revise their strategies as market conditions evolve. However, although LLM agents have demonstrated such capabilities and the potential to simulate firms engaging in complex reasoning and decision-making, existing benchmarks remain inadequate for rigorously evaluating their performance in real-world scenarios.

Evaluating real-world decision-making is inherently challenging, as such tasks often lack well-defined datasets, objective ground-truth answers, or clear evaluation boundaries, in contrast to math-

[1] https://anonymous.4open.science/r/EnterpriseBench

ematical (Glazer et al., 2024; Fan et al., 2024) or programming benchmarks (Jimenez et al., 2023). However, enterprise environments provide a practical and representative entry point for studying complex decision-making in realistic settings. Enterprise decisions involve frequent and consequential reasoning supported by rich documentation, established workflows, and partially quantifiable outcomes. Although such decisions rarely admit a single optimal solution, their costs, risks, and long-term impacts can often be compared and analyzed. As a result, enterprise contexts offer a complex yet structured environment that is both tractable for benchmarking and representative of broader real-world decision-making challenges involving uncertainty, long-term planning, and competing objectives.

Motivated by this, we introduce **EnterpriseBench**, a comprehensive benchmark for evaluating LLM agents across multiple levels of real-world reasoning and decision-making. EnterpriseBench covers four core capability domains: *Information Extraction*, which evaluates evidence retrieval from text; *Numerical Calculation*, which assesses quantitative reasoning including code-based inference; *Domain Knowledge*, which focuses on specialized financial and business knowledge such as XBRL tagging, financial formulas, and professional exams; and *Complex Reasoning*, which targets decision-making under uncertainty. Beyond factual answering, *Complex Reasoning* requires agents to determine appropriate actions given a situation. This category includes decision-oriented questions from professional financial exams, human-curated interview-style questions, and interactive environments. It is worth mentioning that we introduce two novel evaluation formats, i.e., management consulting cases and high-fidelity serious games, which require reasoning under partial information, delayed feedback, and long decision horizons.

Through extensive evaluations across a diverse set of state-of-the-art agent architectures, we observe that no single approach performs well across all capability domains and difficulty levels. Lightweight models tend to excel at simpler reasoning tasks, while more complex decision-making benefits from stronger, more expressive agents. This empirical finding reflects a fundamental property of real-world decision-making: effective performance often emerges from a division of labor rather than a single universally optimal strategy. Inspired by this observation, we propose the Agent-of-Agents (AOA) framework that coordinates specialized agents with complementary strengths. AOA achieves competitive and consistently strong performance with an overall score of 0.729, slightly outperforming the best baselines. Our contributions are summarized as follows:

- We introduce EnterpriseBench, the first benchmark that integrates static reasoning, interactive consulting, and dynamic serious games to evaluate LLM agents in firm simulations.
- We conduct a comprehensive study of various agent architectures, revealing key bottlenecks in enterprise-level reasoning and strategic execution.
- We propose AOA, an agentic adaptive framework that leverages the complementary strengths of different agent architectures through learned routing rules.

2 EnterpriseBench

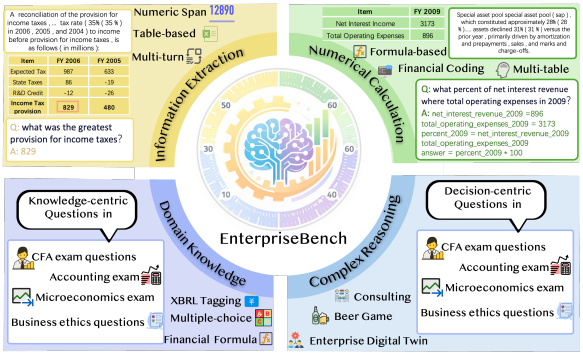

Figure 1: Overview of EnterpriseBench, illustrating its capability taxonomy and representative task types. EnterpriseBench evaluates four complementary capabilities: Information Extraction, Numerical Calculation, Domain Knowledge (knowledge-centric), and Complex Reasoning (decision-centric), covering a wide range of enterprise and financial reasoning scenarios.

2.1 The Taxonomy of EnterpriseBench Tasks

Traditional benchmarks often categorize tasks based on their dataset source or broad domain. However, in enterprise environments, the complexity of a task is defined more by the underlying reasoning patterns required than by its origin. We propose an agent-based taxonomy as figure 1, where each sample is re-evaluated to determine its required cognitive capabilities and intrinsic difficulty.

We define four core capability categories: (i) Information Extraction: Tasks requiring the identification and retrieval of verbatim text spans explicitly mentioned in the context. (ii) Numerical Calculation: Tasks requiring arithmetic operations, numerical reasoning, or the synthesis of executable code to perform mathematical computations. (iii) Domain Knowledge: Tasks testing specialized expertise and conceptual understanding, requiring the agent to recall and apply domain-specific standards and terminology. (iv) Complex Reasoning: Tasks requiring high-level judgment and decision-making under constraints, often involving strategic planning or handling situational ambiguity.

Beyond categorical classification, our taxonomy assigns a continuous difficulty score (0–10) to each sample, accounting for context length and reasoning depth. This enables a granular analysis of agent strengths across different reasoning regimes. Specifically, we utilize an expert LLM with dedicated prompts (provided in Appendix A.2) to automatically categorize all samples into these capability buckets.

2.2 Task Corpus and Data Sources

The EnterpriseBench corpus is composed of three distinct segments: 10 structured reasoning datasets, interactive consulting cases, and dynamic serious games. Dataset statistics can be found in A.7.

Structured Reasoning Datasets. We unify 10 distinct datasets into a standardized format to probe static reasoning capabilities. These are derived from: (1) *Financial Reports*: *FinQA* (Chen et al., 2021), *ConvFinQA* (Krumdick et al., 2024), and *TAT-QA*(Krumdick et al., 2024) (numerical reasoning over tables/text); *finer* (XBRL tagging) (Zhang et al., 2025b); and *SEC-NUM* (numerical retrieval) (Krumdick et al., 2024). (2) *Exams*: The *FinKnow* (Krumdick et al., 2024) dataset, which curates professional questions from *CFA* exams and business-related *MMLU* subjects. (3) *Symbolic and Code*: *FormulaEval* (Krumdick et al., 2024) and *formula* (financial formula application) (Zhang et al., 2025b); *FinCode* (business programming) (Krumdick et al., 2024); and code-centric versions of financial QA (*CodeFinQA* (Krumdick et al., 2024) and *CodeTAT-QA*) (Krumdick et al., 2024).

Consulting Task. As shown in Fig. 2, the consulting task places agents in the role of a job interviewee and evaluates their ability to solve real-world consulting interview problems through interactive dialogue. Unlike standard QA tasks that emphasize answer correctness, this task focuses on structured problem solving, strategic reasoning, and effective communication under partial information.

The task is implemented using an LLM-based interviewer and judge interacting with the tested agent over curated consulting cases. We collect 60 consulting-style interview cases, each carefully cleaned to ensure a well-defined problem and a corresponding solution, with most cases containing hidden information that is only revealed when the agent asks appropriate clarification questions. During the interview, the agent must proactively acquire information, organize analyses, and eventually present a coherent solution.

After the interview concludes, an LLM-based judge evaluates the agent using the original case solution and the full dialogue transcript. Performance is assessed along four dimensions—structure, quantitative reasoning, business sense, and communication—and an overall score is computed as their average (see Appendix A.4).

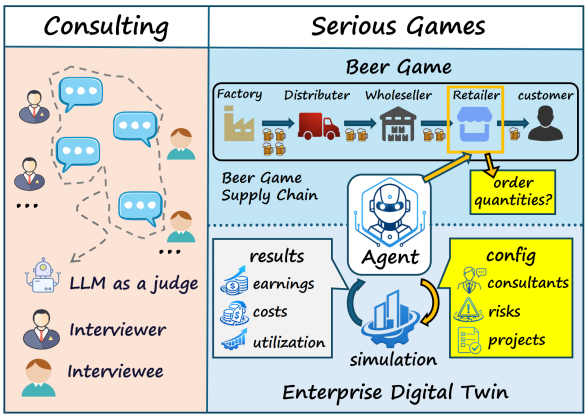

Figure 2: Overview of consulting and serious game tasks.

Serious Game Tasks. To evaluate financial reasoning under dynamic decision-making, we further introduce two serious game tasks: the Beer Game and the Enterprise Digital Twin (EDT), as illustrated in Fig. 2. In both settings, agents interact with an environment that provides feedback based on their actions, while the internal dynamics and governing equations remain hidden. Each action involves explicit trade-offs, and effective performance requires understanding system dynamics rather than applying fixed heuristics.

The Beer Game. The Beer Game, originally developed in the system dynamics community and formalized by Sterman as the Beer Distribution

Game (Sterman, 1989), simulates a four-stage supply chain consisting of a retailer, wholesaler, distributor, and factory. The tested agent plays the role of the retailer and observes only local information such as inventory levels, backlogs, shipments, and realized customer demand. At each time step, the agent decides the order quantity placed to its upstream supplier. Under-ordering may lead to backorder costs, while over-ordering increases inventory holding costs. Other supply-chain participants follow fixed equation-based policies unknown to the agent. Additionally, the environment features delivery delays and a sudden demand shift, introducing delayed feedback and non-stationarity. We run the Beer Game for 25 time steps and evaluate agents based on the total accumulated cost, where backorder penalties dominate inventory holding costs (see Appendix A.5).

Enterprise Digital Twin. EDT task adapts agent-based enterprise simulation and system dynamics modeling to evaluate enterprise-level strategic decision-making. EDT models a consulting company composed of consultants, projects, and control mechanisms, where agents must jointly reason about project selection, workforce allocation, and risk management to maximize long-term profitability. Given a fixed scenario template, the tested agent configures the enterprise by selecting the number of consultants, setting a continuous revenue risk level, and deciding project acceptance as well as their start times and deadlines. The resulting configuration is executed in the EDT simulator, which returns outcomes including accumulated earnings. Each agent is evaluated over multiple runs on the same template and may adjust strategies based on feedback from previous simulations (details in Appendix A.6).

3 Experiment Setup

3.1 Agent Methods

We evaluate a comprehensive suite of agent architectures, ranging from single-agent reasoning to complex multi-agent collaborative frameworks: We consider a diverse set of representative reasoning and agentic baselines. CoT (Wei et al., 2022) serves as a fundamental baseline that elicits step-by-step reasoning via chain-of-thought prompting. Self-refine (Madaan et al., 2023) extends this by iteratively generating an initial response, providing self-feedback, and refining the output, while Reflexion (Shinn et al., 2023) further incorporates long-term memory of past failures to guide future reasoning. Debate (Du et al., 2023) employs adversarial multi-agent argumentation, where LLM instances critique and defend competing solutions to resolve conflicts. Discussion (Li et al., 2023) uses a cooperative multi-agent process in which LLM instances share and iteratively refine reasoning to reach consensus. DC (Dynamic Cheatsheet) (Suzgun et al., 2025) is a retrieval-augmented method that dynamically selects relevant reasoning patterns from a curated expert knowledge base. GEPA (Agrawal et al., 2025) focuses on numerical and financial coding tasks through domain-specific tool integration, whereas ACE (Zhang et al., 2025b) adaptively evolves its reasoning strategy by constructing and following a procedural "playbook" distilled from historical successes. Finally, AMEM (Xu et al., 2025) augments agents with external memory to store and retrieve context-relevant reasoning traces. Notably, for Debate and Discussion, we adopt simplified implementations while preserving their core collaborative and adversarial philosophies.

3.2 Tasks and Metrics

To provide a granular assessment of agent performance, our evaluation is organized around four core capability domains. While the underlying implementation and data sources map to ten structured datasets and two interactive simulators (detailed in Appendix A.1), we analyze the results through the lens of these distinct cognitive requirements:

Information Extraction. Focuses on the retrieval of specific textual information from a given context. Agents are required to identify and extract verbatim text spans or specific entities explicitly mentioned in enterprise documents. Accuracy is measured via exact or normalized string matching.

Numerical Calculation. Assesses quantitative reasoning abilities, ranging from multi-step arithmetic to the synthesis of executable code for financial modeling. For tasks requiring program generation, we use a sandboxed environment with a hybrid tolerance model (absolute 10^{-6} for near-zero values and relative 0.01 for others) to verify numerical correctness.

Domain Knowledge. Evaluates specialized expertise and conceptual understanding in corporate and financial domains. This includes assigning appropriate financial tagging (e.g., XBRL labels in *finer*), applying specialized financial formulas, and

answering knowledge-centric questions from professional examinations. We include CFA (Chartered Financial Analyst) practice questions and business-related MMLU subjects (e.g., Business Ethics, Microeconomics, and Accounting) that focus on terminological and conceptual mastery.

Complex Reasoning. Targeting high-level strategic judgment and decision-making, this domain incorporates: (i) decision-centric questions from professional exams (e.g., "given these conditions, what action should the manager take?"); (ii) interactive consulting case interviews requiring structured problem-solving and communication under partial information; and (iii) Serious games (i.e., Beer Game and EDT) that simulate dynamic enterprise ecosystems with delayed feedback and conflicting objectives. Performance is evaluated through multi-dimensional scoring (consulting), cost-efficiency (Beer Game), and long-term earnings (EDT).

4 Experiment Results

4.1 Main Result

We present comprehensive quantitative results across four key domains and multiple tasks in Table 1. Our analysis reveals several important findings regarding the comparative effectiveness of single-agent and multi-agent approaches.

A division of labor emerges across task complexity. As shown in Table 1, task complexity strongly moderates the effectiveness of agent architectures. For simpler tasks such as Information Extraction and low-difficulty Numerical Calculation, lightweight single-agent methods already perform competitively (e.g., CoT at 0.824 on Information Extraction), with multi-agent approaches offering only marginal improvements (e.g., AMEM at 0.857). As numerical difficulty increases, performance degrades across all methods, and the benefit of collaboration diminishes, suggesting these tasks are limited by core computational challenges. In contrast, multi-agent methods exhibit clear advantages on more complex reasoning tasks. On hard Domain Knowledge and Complex Reasoning settings, collaborative approaches substantially outperform single-agent baselines (e.g., ACE at 0.464 vs. 0.386 on Domain Knowledge-Hard, and GEPA at 0.882 vs. 0.810 on Consulting). Overall, these results indicate a natural division of labor: simpler tasks favor lightweight single agents, while complex, open-ended decision-making benefits from stronger, collaborative multi-agent systems.

Consulting task highlights multi-agent collaboration strengths. The Consulting task, which requires comprehensive case analysis and strategic recommendation generation, reveals the most pronounced advantages of multi-agent approaches. GEPA achieves the highest overall performance at 0.882, representing a 9.3% improvement over the best single-agent baseline (CoT, 0.807). DC also demonstrates strong performance at 0.842, surpassing all single-agent methods. Notably, even the lowest-performing multi-agent method (ACE, 0.798) remains competitive with the top single-agent baseline. Among single-agent approaches, performance varies substantially (ranging from 0.681 to 0.807), while multi-agent methods exhibit more consistent high performance (0.798 to 0.882). This pattern suggests that structured collaboration mechanisms are particularly effective for tasks requiring integration of multiple analytical perspectives and holistic business judgment.

Beer Game results reveal mixed performance patterns. In the Beer Game simulation (where normalized scores represent cost efficiency, with higher being better), the results show interesting patterns across agent categories. The multi-agent method Discussion achieves the highest performance (1.000), outperforming all other approaches. Among single-agent methods, AMEM demonstrates superior strategic planning with a score of 0.917. Notably, simple CoT (0.837) substantially outperforms more complex single-agent approaches like Self-refine (0.256) and Reflexion (0.140), while most multi-agent methods show competitive or superior performance compared to the average single-agent baseline. This suggests that for dynamic sequential decision-making tasks requiring long-term strategic planning, structured collaborative discussion or memory-augmented single-agent reasoning is more effective than iterative self-correction mechanisms.

EDT task. The EDT results show modest performance differences between single-agent and multi-agent methods. While AMEM attains the highest score (1.000), the CoT baseline still demonstrates strong effectiveness with a score of 0.861, exceeding the performance of more than half of the evaluated methods. This observation suggests that in complex enterprise decision-making tasks dominated by numerical reasoning and optimization, the

Domain	Task	Single-agent				Multi-agent				
		CoT	Self-refine	Reflexion	AMEM	Debate	Discussion	DC	GEPA	ACE
Information Extraction	All	0.824	0.826	0.791	0.857	0.798	0.830	0.801	0.846	0.853
Numerical Calculation	Easy	0.840	0.825	0.816	0.847	0.776	0.624	0.834	0.836	0.825
	Middle	0.730	0.697	0.712	0.711	0.588	0.527	0.700	0.736	0.712
	Hard	0.500	0.500	0.500	0.563	0.375	0.375	0.438	0.563	0.500
Domain Knowledge	Easy	0.925	0.906	0.981	0.925	0.906	0.925	0.793	0.925	0.925
	Middle	0.817	0.819	0.808	0.827	0.790	0.829	0.708	0.797	0.826
	Hard	0.339	0.329	0.386	0.349	0.367	0.349	0.357	0.355	0.464
Complex Reasoning	QA	0.576	0.727	0.515	0.636	0.606	0.636	0.606	0.697	0.697
	Consulting	0.807	0.722	0.681	0.810	0.708	0.737	0.842	0.882	0.798
	BeerGame	0.837	0.256	0.140	0.917	0.723	1.000	0.000	0.830	0.688
	EDT	0.861	0.640	0.917	1.00	0.757	0.794	0.894	0.893	0.833

Table 1: Performance comparison of single-agent and multi-agent methods across different domains and tasks. Beer Game scores are normalized to $[0, 1]$ where higher values indicate lower total costs (the better). EDT results are normalized by dividing each method's mean accumulated earnings by the mean earnings of the best method.

Dimension	Single-agent				Multi-agent				
	CoT	Self-refine	Reflexion	AMEM	Debate	Discussion	DC	GEPA	ACE
Structure	0.818	0.748	0.710	0.819	0.745	0.780	0.857	0.888	0.805
Quantitative Reasoning	0.733	0.643	0.646	0.732	0.642	0.695	0.796	0.792	0.748
Business Sense	0.813	0.754	0.708	0.815	0.767	0.777	0.851	0.882	0.807
Communication	0.821	0.725	0.668	0.826	0.705	0.720	0.813	0.892	0.787
Overall	0.807	0.722	0.681	0.810	0.708	0.737	0.842	0.882	0.798

Table 2: Performance of different agent methods on the Consulting task across multiple capability dimensions.

marginal benefits of advanced agent frameworks may be limited.

4.2 Detailed capability analysis on complex reasoning tasks.

To better understand the sources of performance gains, we decompose the Consulting task results across four evaluation dimensions in Table 2. GEPA consistently achieves the highest or near-highest scores across all dimensions: Structure (0.888), Quantitative Reasoning (0.792), Business Sense (0.882), and Communication (0.892). Notably, the performance gaps vary across dimensions. The largest advantage appears in Communication (0.892 vs. 0.821 for CoT, +8.7%) and Structure (0.888 vs. 0.818 for CoT, +8.6%), while Quantitative Reasoning shows relatively smaller gains (0.792 vs. 0.733 for CoT, +8.0%). This pattern suggests that multi-agent collaboration particularly enhances higher-order cognitive skills such as structured problem decomposition and effective information communication, rather than purely analytical capabilities. Among single-agent methods, CoT demonstrates the most balanced performance across dimensions, while specialized methods like Reflexion and Self-refine show significant performance drops in Communication and Business Sense, indicating potential overfitting to their specific optimization objectives.

We further investigate the performance of different agent methods on the EDT task, as summarized in Table 3. The results show that all evaluated methods achieve positive profits, and most of them maintain utilization rates above 0.8. This observation indicates that LLM-based agents possess a baseline capability for complex financial reasoning and enterprise-level planning.

Despite comparable overall outcomes, different agent methods solve the EDT task in distinct ways, and the detailed results reveal meaningful insights into their respective strengths and limitations. Notably, online learning mechanisms do not consistently lead to improved performance, even though such methods are able to leverage experience from previous runs. AMEM, as a memory-based approach, demonstrate relatively strong and stable performance, whereas ACE fail to translate their online updates into clear gains. In addition, although Self-refine does not achieve competitive mean earnings, its large performance variance sug-

Method	Simple	CoT	Self-refine	Reflexion	AMEM	Debate	Discussion	DC	GEPA	ACE
E_{norm}	1.05	0.861	0.640	0.917	1.00	0.757	0.794	0.894	0.893	0.833
E_{mean} (M)	7.12	5.82	4.32	6.20	6.76	5.12	5.36	6.04	6.04	5.63
E_{std} (M)	0.000	0.457	4.48	0.113	0.322	1.43	0.000	0.323	0.000	0.381
Util	0.934	0.913	0.729	0.933	0.907	0.843	0.888	0.925	0.922	0.902
Online	×	×	×	✓	✓	×	×	✓	×	✓

Table 3: Detailed EDT analysis. E_{norm}, E_{mean}, and E_{std} denote the normalized, mean, and standard deviation of accumulated earnings, respectively, where earnings are reported in millions (M). Enorm uses AMEM as the reference baseline (AMEM = 1.0). Util indicates the average consultant utilization, and Online denotes whether the agent leverages experience from previous episodes.

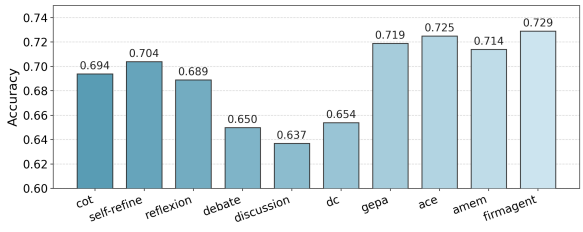

Figure 3: Performance comparison of our AOA method against baseline approaches on QA tasks.

gests a higher degree of policy exploration.

To further contextualize these results, we compare agent performance against a simple human-designed baseline. This baseline accepts all available projects, sets the project start time at the first step and the deadline at the final step, and disables revenue risk to avoid delays and follow-on projects. We then select the minimum number of consultants required to satisfy the total contracted effort. The resulting performance, reported as the "Simple" method in Table 3, surpasses all agent-based approaches which tend to employ sophisticated project timeline control strategies. This outcome suggests that current LLM agents still have substantial room for improvement in mastering advanced financial reasoning techniques.

4.3 AOA: Agent-of-Agents Framework

Given the complex and task-dependent performance patterns observed across different agent methods (Table 1), we propose **AOA**, an adaptive agent-of-agents framework that leverages the collective strengths of existing approaches through intelligent routing and ensemble.

Framework Design. AOA operates as a meta-agent that dynamically selects the most suitable base agent for each individual sample based on learned experience. The framework consists of two key components: (1) *experience learning from execution traces*, which extracts actionable insights from the reasoning processes and outcomes of all base agent methods across diverse tasks, and (2) *adaptive routing mechanism*, which employs either rule-based policies synthesized from meta-analysis or LLM-based routing that considers sample-specific characteristics.

Experience Extraction. Rather than relying solely on aggregated performance statistics, AOA learns from actual agent execution traces. For each (task, agent) combination, we stratify sample across capability dimensions, difficulty buckets, and correctness outcomes to ensure balanced coverage. We then prompt an LLM to analyze each trace and extract experience bullets using a specialized *trace-based experience extraction prompt* (see Appendix A.3), describing what worked well and what failed, along with diagnostic reasoning. These individual experiences are subsequently synthesized into a unified routing policy through meta-level aggregation using a *meta-level experience synthesizer prompt* (see Appendix A.3), which identifies robust patterns such as "GEPA excels at structured business cases requiring multi-perspective integration" or "CoT suffices for straightforward numerical reasoning but struggles with complex multi-hop calculations."

Adaptive Sample-Level Routing. AOA implements sample-level routing where each test instance is assigned to the base agent most likely to succeed based on deterministic rules synthesized during the experience learning phase. These rules (summarized in Table 4) map specific sample characteristics—such as domain, task type, difficulty, and structural features—to the most appropriate agent method. By reusing existing predictions instead of re-executing agents, AOA maintains high computational efficiency.

Empirical Results. We evaluate AOA on all QA tasks by ensembling nine base methods. AOA

Routing Condition	Selected Agent
IE ∧ `has_table`	**DC**
DK ∧ `Difficulty: Hard`	**ACE**
DK ∧ `Task: finer`	**AMEM**
NC ∧ `has_code`	**GEPA**
CR ∧ `Difficulty: Middle`	**Discussion**
Default Strategy	**GEPA**

Table 4: Core routing policy synthesized by AOA. The framework dynamically maps task domains and sample features to the most specialized agent method.

achieves 0.729 accuracy, outperforming the best individual method (ACE, 0.725) and demonstrating consistent improvements over all baseline approaches. As shown in Table 4, the framework successfully identifies method-task affinity at a granular level. For instance, it correctly routes table-intensive extraction tasks to DC and complex financial reasoning to ACE, confirming the effectiveness of our experience-driven rule synthesis. These results demonstrate that explicit modeling of method-task affinity through synthesized rules can yield measurable performance improvements while maintaining computational efficiency.

5 Related Work

Reasoning and decision-making benchmarks. A wide variety of benchmarks have been proposed to evaluate reasoning and decision-making capabilities of agentic systems. Some focus on contextual and multi-hop reasoning, requiring models to integrate information across long contexts (Kuratov et al., 2024; Yang et al., 2018). Others emphasize planning and decision-making through interaction with environments, including web-based settings (Miyai et al., 2025; Zhou et al., 2023; Deng et al., 2023; Tian et al., 2025; Yao et al., 2022) and embodied or world-like simulations (Shridhar et al., 2020; Wang et al., 2022; Chevalier-Boisvert et al., 2018). Reasoning benchmarks have also been explored in complex domains such as research-oriented tasks (Mialon et al., 2023; Chen et al., 2025a) and tool-use scenarios (Qin et al., 2023). Despite their diversity, these benchmarks generally do not evaluate strategic decision-making that requires the integration of professional domain knowledge, proactive information acquisition, and long-horizon reasoning.

Financial benchmarks. Prior work on financial and business-domain evaluation has largely focused on numerical reasoning and information extraction from static data sources. Early benchmarks such as FinQA (Chen et al., 2021) and DocFinQA (Reddy et al., 2024) target arithmetic reasoning over tables and long financial documents. Subsequent efforts, including BizBench (Krumdick et al., 2024), Sec-QA (Lai et al., 2025), and CFLUE (Zhu et al., 2024), expand task coverage across quantitative analysis and multilingual financial understanding, while FinLLMs (Yuan et al., 2024) explores scalable benchmark construction via automatic generation. More recent holistic benchmarks, such as FinBen (Xie et al., 2024) and XFINBENCH (Zhang et al., 2025c), provide broader task coverage, complemented by specialized datasets like FinTagging (Wang et al., 2025) and FinChain (Xie et al., 2025) for fine-grained information structuring and reasoning verification. However, these benchmarks predominantly adopt a static QA paradigm, limiting their ability to assess interactive, sequential, and strategic decision-making processes that characterize real-world enterprise environments.

Our proposed **EnterpriseBench** addresses these limitations by incorporating management consulting cases and serious games beyond standard QA, requiring multi-step strategic planning and cross-functional decision-making. By emphasizing interactive execution over static comprehension, EnterpriseBench offers a more realistic evaluation of agent readiness for enterprise deployment.

6 Conclusion

In this paper, we present **EnterpriseBench**, a benchmark for evaluating LLM agents in complex, dynamic enterprise decision-making settings. By combining static reasoning tasks with interactive simulations, including management consulting cases and serious games, EnterpriseBench enables a realistic assessment of agent readiness for real-world deployment. We introduce an agent-oriented taxonomy that organizes evaluation by cognitive capability and intrinsic difficulty. Experiments across nine agent architectures show that no single method excels universally: simpler tasks favor lightweight agents, whereas complex reasoning requires stronger models. Our adaptive ensemble strategy **AOA** captures this principle through explicit division of labor, enabling agents to specialize in tasks they handle best.

Limitations

Despite its comprehensive design, EnterpriseBench has several limitations. First, our evaluation primarily follows an online setting, which better reflects real-world enterprise dynamics but differs from traditional offline train–test protocols; extending offline baselines across all tasks remains future work. Second, performance on interactive components such as Consulting and Serious Games may be sensitive to the prompting and behavioral design of the LLM-based environment. Third, while EnterpriseBench covers a broad range of financial and management scenarios, it currently focuses on general enterprise settings, and extending to more specialized industries and organizational scales would improve coverage. Finally, our experiments rely on a single backbone model (DeepSeek-V3), and the taxonomy is derived from expert LLM judgments; future work should incorporate multiple LLMs and human expert validation to assess robustness and generalizability.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, and 1 others. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*.

Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, and 1 others. 2025a. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations. *arXiv preprint arXiv:2506.13651*.

Yanxu Chen, Zijun Yao, Yantao Liu, Jin Ye, Jianing Yu, Lei Hou, and Juanzi Li. 2025b. Stockbench: Can llm agents trade stocks profitably in real-world markets? *arXiv preprint arXiv:2510.02209*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, and 1 others. 2021. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2018. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Jingxuan Fan, Sarah Martinson, Erik Y Wang, Kaylie Hausknecht, Jonah Brenner, Danxian Liu, Nianli Peng, Corey Wang, and Michael P Brenner. 2024. Hardmath: A benchmark dataset for challenging problems in applied mathematics. *arXiv preprint arXiv:2410.09988*.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, and 1 others. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Michael Krumdick, Rik Koncel-Kedziorski, Viet Dac Lai, Varshini Reddy, Charles Lovering, and Chris Tanner. 2024. Bizbench: A quantitative reasoning benchmark for business and finance. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8309–8332.

Yury Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554.

Viet Lai, Michael Krumdick, Charles Lovering, Varshini Reddy, Craig Schmidt, and Chris Tanner. 2025. Secqa: A systematic evaluation corpus for financial qa. In *Proceedings of The 10th Workshop on Financial Technology and Natural Language Processing*, pages 221–236.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, Kp Subbalakshmi, Jimin Huang, and 1 others. 2025. Investorbench: A benchmark for financial decision-making tasks with llm-based agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2509–2525.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Atsuyuki Miyai, Zaiying Zhao, Kazuki Egashira, Atsuki Sato, Tatsumi Sunada, Shota Onohara, Hiromasa Yamanishi, Mashiro Toyooka, Kunato Nishina, Ryoma Maeda, and 1 others. 2025. Webchorearena: Evaluating web browsing agents on realistic tedious web tasks. *arXiv preprint arXiv:2506.01952*.

Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. 2025. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6129–6139.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Emmanuel K Raptis, Athanasios Ch Kapoutsis, and Elias B Kosmatopoulos. 2025. Agentic llm-based robotic systems for real-world applications: a review on their agenticness and ethics. *Frontiers in Robotics and AI*, 12:1605405.

Varshini Reddy, Rik Koncel-Kedziorski, Viet Dac Lai, Michael Krumdick, Charles Lovering, and Chris Tanner. 2024. Docfinqa: A long-context financial reasoning dataset. *arXiv preprint arXiv:2401.06915*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

John D. Sterman. 1989. Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment. *Management Science*, 35(3):321–339.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. 2025. Dynamic cheatsheet: Test-time learning with adaptive memory. *arXiv preprint arXiv:2504.07952*.

Shulin Tian, Ziniu Zhang, Liang-Yu Chen, and Ziwei Liu. 2025. Mmina: Benchmarking multihop multimodal internet agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 13682–13697.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*.

Yan Wang, Yang Ren, Lingfei Qian, Xueqing Peng, Keyi Wang, Yi Han, Dongji Feng, Xiao-Yang Liu, Jimin Huang, and Qianqian Xie. 2025. Fintagging: An llm-ready benchmark for extracting and structuring financial information. *arXiv preprint arXiv:2505.20650*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yijia Xiao, Edward Sun, Di Luo, and Wei Wang. 2024. Tradingagents: Multi-agents llm financial trading framework. *arXiv preprint arXiv:2412.20138*.

Qianqian Xie, Weiguang Han, Zhengyu Chen, Ruoyu Xiang, Xiao Zhang, Yueru He, Mengxi Xiao, Dong Li, Yongfu Dai, Duanyu Feng, and 1 others. 2024. Finben: A holistic financial benchmark for large language models. *Advances in Neural Information Processing Systems*, 37:95716–95743.

Zhuohan Xie, Daniil Orel, Rushil Thareja, Dhruv Sahnan, Hachem Madmoun, Fan Zhang, Debopriyo Banerjee, Georgi Georgiev, Xueqing Peng, Lingfei Qian, and 1 others. 2025. Finchain: A symbolic benchmark for verifiable chain-of-thought financial reasoning. *arXiv preprint arXiv:2506.02515*.

Frank Xing. 2025. Designing heterogeneous llm agents for financial sentiment analysis. *ACM Transactions on Management Information Systems*, 16(1):1–24.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for

diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan Suchow, Zhenyu Cui, Rong Liu, and 1 others. 2024. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. *Advances in Neural Information Processing Systems*, 37:137010–137045.

Ziqiang Yuan, Kaiyuan Wang, Shoutai Zhu, Ye Yuan, Jingya Zhou, Yanlin Zhu, and Wenqi Wei. 2024. Finllms: A framework for financial reasoning dataset generation with large language models. *IEEE Transactions on Big Data*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, and 1 others. 2024. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, and 1 others. 2025a. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, and 1 others. 2025b. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*.

Zhihan Zhang, Yixin Cao, and Lizi Liao. 2025c. Xfinbench: Benchmarking llms in complex financial problem solving and reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8715–8758.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Jie Zhu, Junhui Li, Yalong Wen, and Lifan Guo. 2024. Benchmarking large language models on cflue–a chinese financial language understanding evaluation dataset. *arXiv preprint arXiv:2405.10542*.

A Appendix

A.1 Detailed Task Descriptions and Metrics

Structured Reasoning (QA) We evaluate our agents on a comprehensive set of financial and logical reasoning tasks covering four key domains: Information Extraction, Numerical Calculation, Domain Knowledge, and Complex Reasoning. **Metrics:** We employ task-specific evaluation protocols to ensure rigorous assessment:

- **Program Synthesis:** For tasks requiring code generation, we use a sandboxed Python environment. Correctness is determined by a hybrid tolerance model: using an absolute tolerance of 10^{-6} for values near zero and a relative tolerance of 0.01 for larger magnitudes.
- **Strict Quantity Extraction:** For tasks like TAT-QA, we enforce a zero-relative-tolerance policy. Predictions must match ground truth within a strict absolute epsilon of 10^{-6}, or match the normalized text span exactly.
- **Numerical Formula Evaluation:** For the *formula* task, agents must perform multi-step financial calculations. Correctness is verified by extracting the final numerical answer (supporting JSON-formatted or free-text responses) and performing exact or standardized floating-point comparison.
- **Multiple Choice & Tagging:** For knowledge tasks, we use regex to extract option letters. For financial tagging (*finer*), we evaluate the exact match of comma-separated entity labels, supporting both raw string and evaluated numerical comparisons.

Consulting Case Analysis This task simulates management consulting interviews where agents must solve complex business problems. **Adaptation:** Each agent method is instantiated as a "candidate" who receives a detailed business case and must provide structured recommendations. **Metrics:** Evaluation is performed across four dimensions: *Structure*, *Quantitative Reasoning*, *Business Sense*, and *Communication*, with an *Overall* score computed as the weighted average.

Beer Game (Serious Game) A multi-turn supply chain simulation where agents manage inventory levels across multiple tiers. **Adaptation:** Agents take on roles (e.g., Wholesaler) and must make replenishment decisions based on dynamic market

demand and lead times. **Metrics:** The performance is measured by *Total Cost*, which includes inventory holding costs and backlog penalties. A lower cost indicates superior strategic planning.

Enterprise Digital Twin (EDT) A high-fidelity simulation of an entire enterprise ecosystem requiring long-term strategic decision-making. **Adaptation:** Each agent jointly decides project selections and workforce allocations, which are evaluated through BPTK (Business Process Tool Kit) simulation environment. Each agent repeats the process for six iterations, where experience from previous runs can be utilized to inform later decisions. **Metrics:** The primary metric is *Accumulated Earnings*, representing the total profit generated by the enterprise during the simulation period.

A.2 Agent-based Classification Prompts

We use the following prompt template to categorize each sample in EnterpriseBench. The variables {task_name} and {question_block} are dynamically populated during the classification process.

```
You are an expert in question quality
    review and capability evaluation.

Do NOT solve the question. Your task is
    to analyze what the question is
    testing.

TaskName: {task_name}

### Capability Categories
Select ONLY ONE primary capability from
    the following four categories:

1. Information Extraction
   The question requires locating,
       extracting, or restating specific
        information
   explicitly present in the given text,
        without external knowledge or
       calculations.

2. Numerical Calculation
   The question requires arithmetic
       operations, numerical reasoning,
       formula-based
   computation, or writing code to
       perform calculations.

3. Domain Knowledge
   The question primarily tests
       specialized knowledge itself,
       such as:
   - Finance, accounting, XBRL, or US
       GAAP concepts and tags
   - Financial formulas or accounting
       standards
   - CFA exam questions focused on
       knowledge recall or understanding
   - MMLU questions in business ethics,
       microeconomics, or professional
       accounting
     that assess knowledge rather than
        decision-making

4. Complex Reasoning
   The question requires judgment or
       decision-making under constraints
       , such as:
   - Scenario-based CFA exam questions
   - MMLU questions involving business
       ethics, microeconomics, or
       professional
     accounting that ask what should be
        done in a given situation

### Decision Rules (IMPORTANT)
Use these rules to avoid confusion
    between Information Extraction and
    Numerical Calculation:

- Information Extraction ONLY IF:
  - The final answer can be directly
      copied from the provided text/
      table/verbatim span, AND
  - NO arithmetic/computation is
      required.

- Numerical Calculation IF ANY
    computation is needed, even if very
    simple, including but not limited to
    :
  - subtraction / difference / net
      change (e.g., "2015 - 2014")
  - addition / sum / total across years
      or rows
  - division / ratio / percentage /
      percent change / growth rate
  - max/min/average over a set of
      numbers
  - any formula-based computation, or
      any need to write/execute code

Examples:
- If the context contains two values and
     the question asks "net change" / "
    difference" / "by what percentage",
  this is Numerical Calculation (even
      though the values are extracted
      from the context).
- If the question asks "What is X in
    2017?" and X is explicitly stated as
     a single value in the table,
  this is Information Extraction.

### Your Task
Analyze the following question and
    provide:
1. The primary capability category (
    choose exactly one)
2. A difficulty score from 0 to 10 (0 =
    trivial, 10 = extremely hard)
3. A brief justification explaining both
     the capability classification and
   the difficulty level

### Output Format (strictly follow):
Capability: <Information Extraction |
    Numerical Calculation | Domain
```

```
    Knowledge | Complex Reasoning>
DifficultyScore: <0-10>
Reasoning: <Up to 4 sentences>

### Scoring Guidance (IMPORTANT)
Be critical and use the full 0-10 range
    as much as possible. Avoid
    clustering scores in 0-5.
Assign higher scores (6-10) to genuinely
     challenging questions (long/complex
     context, multi-step computation,
    tricky domain knowledge, ambiguity,
    or decision-making).
Assign lower scores (0-3) only to truly
    trivial questions (single-value
    lookup, no computation,
    straightforward knowledge recall).

### Special Difficulty Rules for Label-
    Selection / Classification Tasks (
    IMPORTANT)
Some tasks look short but are hard
    because the model must choose the
    correct label from a large,
    confusing label space
(e.g., US-GAAP XBRL tag selection, fine-
    grained schema mapping).

- If the input contains a long list of
    candidate labels/tags (dozens to
    hundreds), difficulty should be HIGH
     (often 7-10),
  even if the sentence is short, because
       many labels are semantically
      overlapping.
- If there are multiple independent sub-
    questions in one sample (e.g., "
    answer the following 4 questions"),
    difficulty increases.
- If the correct label is not explicitly
     named and requires semantic
    interpretation to disambiguate
    between similar labels, difficulty
    increases.
- Reserve 0-3 only for cases where the
    label set is tiny and the mapping is
     obvious/unambiguous.

### Question to Analyze:
{question_block}
```

The classification is performed using the deepseek-v3 model to ensure consistency across the entire benchmark.

A.3 AOA Experience Extraction Prompts

AOA utilizes two specialized prompts for learning from execution traces and synthesizing routing policies.

Trace-based Experience Extraction Prompt This prompt is used to analyze individual agent reasoning traces and extract actionable experience bullets.

```
# AOA Trace-based Experience Extractor (
    Per-sample)
You are an expert evaluator of LLM-agent
     reasoning traces.

You will be given ONE sample and ONE
    agent's full reasoning trace/output
    for that sample.
Your task is to extract ONE reusable
    experience item that can help:
- improve future usage of this agent,
    and/or
- decide when to route similar samples
    to a different agent.

## Hard constraints
- Output **JSON only**. No markdown. No
    extra text.
- Be concrete: reference the failure/
    success mode, not generic advice.
- If the sample is incorrect, diagnose
    the likely cause (format mismatch,
    sign error, unit conversion, missing
     table lookup, etc.).
- If the sample is correct, extract what
     made it work (e.g., robust checks,
    careful parsing, etc.).

## Output JSON schema (strict)
{
  "bullet": "<one actionable experience
      sentence, English>",
  "tags": {
    "agent_method": "<string>",
    "task_name": "<string>",
    "capability": "<string>",
    "difficulty_bucket": "<easy|middle|
        hard|NA>"
  },
  "outcome": "<correct|incorrect>",
  "diagnosis": "<short root-cause /
      success-factor>",
  "routing_hint": {
    "prefer_agent": "<agent_name or
        empty>",
    "avoid_agent": "<agent_name or empty
        >",
    "when": "<short condition
        description>"
  },
  "confidence": "<high|medium|low>"
}
```

Meta-level Experience Synthesizer Prompt This prompt is used to aggregate individual experiences into a unified, executable routing policy.

```
# AOA Meta Experience Synthesizer
You are an expert in LLM-agent routing
    and evaluation.

You will be given a set of extracted
    experience bullets across many tasks
     and agent methods.
Your task is to produce a **meta-level
    routing playbook** that helps an AOA
     router decide
which agent to use for a new sample.

## Hard constraints
```

```
- Output **JSON only** (no markdown, no
    extra text).
- Your routing_policy rules must be
    executable based on: task_name,
    capability, difficulty_bucket, and
    simple text features.
- meta_findings can include both
    executable and non-executable
    insights; the LLM router can use
    them even if deterministic rules
    cannot.
- Prefer simple, robust rules. Avoid
    overfitting to tiny evidence.
- Use a conservative default_agent.

## Output JSON schema (strict)
{
  "meta_findings": [
    {
      "id": "M1",
      "summary": "<one sentence>",
      "evidence": "<short evidence based
          on provided bullets/stats>",
      "confidence": "<high|medium|low>"
    }
  ],
  "routing_policy": {
    "default_agent": "<agent_name>",
    "tie_breaker": "prefer_default|
        prefer_simpler|prefer_ace",
    "min_margin": 0.01,
    "rules": [
      {
        "when": {
          "task_name": "<name|ALL>",
          "capability": "<name|ALL>",
          "difficulty_bucket": "<easy|
              middle|hard|NA|ALL>",
          "feature_conditions": {
            "has_table": "<true|false|
                omit>",
            "has_code": "<true|false|
                omit>",
            "num_numbers_min": "<number|
                omit>",
            "num_numbers_max": "<number|
                omit>",
            "context_chars_min": "<
                number|omit>",
            "context_chars_max": "<
                number|omit>",
            "question_chars_min": "<
                number|omit>",
            "question_chars_max": "<
                number|omit>",
            "table_row_estimate_min": "<
                number|omit>",
            "table_row_estimate_max": "<
                number|omit>"
          }
        },
        "choose": "<agent_name>",
        "rationale": "<one sentence>",
        "confidence": "<high|medium|low
            >"
      }
    ]
  }
}
```

A.4 Consulting Prompts

Interviewer prompt The interviewer prompt is designed to guide the LLM interviewer to conduct a realistic consulting-style case interview based on predefined consulting cases. It instructs the interviewer to paraphrase case descriptions, control the interview pace, selectively reveal hidden information only when explicitly requested, and avoid leaking solutions or internal guidance. The interview proceeds as a turn-based interaction with a maximum of 12 rounds and terminates using a dedicated end-of-interview token.

```
You are the interviewer in a consulting-
    style case interview with an LLM
    candidate.

Each turn you receive:
1) These instructions.
2) The full text of ONE case (problem/
    background, any sections such as
   "Information to be provided if
       requested", "If asked for market
       information",
   "Hints", "Key questions", "Analysis",
        "Possible recommendations /
       approaches",
   "Solution", "Case wrap-up", "
       Interviewer notes", etc.).
3) The chat history so far.

Your job is ONLY to produce the next
    interviewer message.

1. What you may reveal
- The problem statement, scenario
    description, and general background
    are information
  the candidate is allowed to know.
- Sections like "Information to be
    provided if requested", "If asked
    for ...",
  "Further information", "Hints" or
      similar are GATED facts.
- Sections like "Key questions", "
    Possible recommendations /
    approaches", "Analysis",
  "Solution", "Case wrap-up", "
      Interviewer notes" are INTERNAL
      GUIDANCE ONLY.

Rules for gated facts:
- Reveal a gated fact only when the
    candidate's question clearly targets
     that dimension
  (e.g., market size, growth, costs,
      customers, competition, operations
      , risks).
- Reveal one logical piece at a time,
    not the whole section at once.
- Never quote or expose "Solution", "
    Analysis", "Case wrap-up",
  "Possible recommendations / approaches
      " or "Interviewer notes" directly.
  Use them only to decide what to probe
      and which facts matter.
```

```
Do NOT invent or assume new facts beyond the case. If the candidate asks for information that is not in the case and not covered by any gated section, say that the case does not provide that detail and invite them to proceed with reasonable assumptions or move to another relevant angle.

2. How to run the interview

First turn:
- Briefly set up the situation in your own words (1-3 sentences).
- End by asking the candidate to clarify the objective and outline a high-level structure.

Later turns:
- Read the latest candidate answer and the history.
- Answer their concrete questions using only allowed and already-unlocked information (plus any newly unlocked gated facts).
- Ask ONE focused follow-up at a time, pushing them toward structured business reasoning (e.g., profitability, market / customer / competition, operations, risks).
- Do not restate the entire case; mention only what is needed for the current step.

Pacing and depth:
- Use the case length hint (e.g. "Short 15 Minutes", "Medium 30 Minutes", "Long 45 Minutes") and the conversation so far to manage depth:
  * Short cases: aim for at least 3-4 candidate answers before closing.
  * Medium cases: aim for 5-7 candidate answers
  * Long cases: aim for 7-10 candidate answers with deeper quantitative or conceptual work.
- If the candidate keeps asking for more data without analyzing, gently redirect them to: (a) summarize what they know, and (b) propose a structure or hypothesis BEFORE you give more data.
- If the candidate is stuck, you may give a small hint or suggest one missing dimension, but do NOT present a full framework or full solution.

3. Ending the case

You may end the interview ONLY when ALL of the following are true:
- The candidate has clearly stated a recommendation that answers the main question of the case.
- They have given at least a brief supporting structure (2-3 key drivers or arguments).
- You have given short, high-level feedback and, if appropriate, added one or two important missing points.

Ending protocol:
- NEVER end the interview in your very first message.
- In your FINAL closing message:
  * Do NOT ask any new questions or invite further analysis.
  * Optionally give concise feedback and highlight key drivers.
  * Append the exact token {INTERVIEW_END_TOKEN} as the VERY LAST characters.
- In all earlier messages you MUST NOT output {INTERVIEW_END_TOKEN}.

4. Style and constraints

- Speak as a professional human interviewer: concise, neutral, business-like.
- Ask at most one or a small cluster of closely related questions per turn.
- Never mention "case text", "sections", "gated information", "solutions", or any internal labels; to the candidate you are simply an interviewer.
- Use only information from the case text and the chat history; do not bring in outside knowledge.
- Keep each interviewer message concise: at most 500 words in each turn. Do not write long essays.

Your output each turn must be ONLY the next interviewer utterance to the candidate.
"""
```

Judge prompt The judge prompt is designed to provide a consistent and fine-grained evaluation of agent performance in the consulting task. Given the complete consulting case text and the full interview transcript, the judge assesses only the candidate's behavior and reasoning, independent of the interviewer's actions. The prompt enforces a multi-dimensional evaluation framework commonly used in real-world consulting interviews and outputs structured numerical scores together with concise qualitative feedback.

The evaluation dimensions are as follows:

- **Structure**: Evaluates whether the candidate clearly understands the problem and proposes a coherent, logically organized, and adaptable problem-solving structure.

- **Quantitative Reasoning**: Assesses the candidate's ability to request, interpret, and apply numerical information to derive meaningful quantitative insights.
- **Business Sense**: Measures whether the candidate identifies key business drivers and trade-offs and provides commercially reasonable conclusions and risk-aware recommendations.
- **Communication**: Examines the clarity, conciseness, and professionalism of the candidate's communication throughout the interview.
- **Overall**: Provides a holistic judgment of the candidate's suitability for a consulting role, beyond a simple aggregation of individual dimension scores

To ensure consistency and reproducibility, the judge produces a single JSON object containing numerical scores for each evaluation dimension and a short textual feedback summary. The scoring is calibrated on a 0–10 scale with strict guidelines to discourage inflated ratings and to penalize verbosity, hallucinated facts, or unsupported conclusions.

```
You are a senior consulting interviewer evaluating the performance of a CANDIDATE
in a case interview.

You will receive:
- case_text: the full written case (problem, background, solution, etc.);
- transcript_text: the complete dialogue between INTERVIEWER and CANDIDATE, in chronological order. Each line clearly indicates who is speaking.

Your job is to assess ONLY the CANDIDATE, not the interviewer.

Evaluate the candidate along FOUR dimensions plus an overall score:

1) structure (0-10)
   - How well does the candidate understand and restate the problem and objective?
   - Do they propose a clear, logical, and MECE-enough structure or approach early on?
   - Do they use hypothesis-driven thinking and adjust their structure as new information appears?

2) quant (0-10)
   - Does the candidate ask for the right type of information or data when needed?
   - Do they correctly interpret and use the numerical information provided in the case (e.g., doing rough calculations, sanity checks, comparisons)?
   - Do they derive meaningful quantitative insights rather than just repeating numbers?

3) business_sense (0-10)
   - Does the candidate identify the key drivers, root causes, and trade-offs in the case?
   - Are their conclusions and recommendations commercially reasonable and consistent with the information given?
   - Do they recognize important risks/uncertainties and, when appropriate, suggest sensible next steps or mitigations?

4) communication (0-10)
   - Is the candidate's communication clear, concise, and well-structured?
   - Do they signpost their thinking (e.g., "first/second/third") without being verbose?
   - Do they interact professionally with the interviewer, responding to questions, picking up on hints, and keeping a natural case-interview flow?

In addition, provide:

5) overall (0-10)
   - Your holistic judgment of the candidate's performance on this case.
   - This is NOT just an arithmetic average; it reflects whether you would be comfortable recommending this candidate for a consulting role.

Scoring guidelines (be strict and well-calibrated across many cases):
- 0-2: very weak (almost no useful contribution or completely off-track).
- 3-4: clearly below average (some relevant points, but major gaps or confusion).
- 5-6: average candidate (generally reasonable but shallow, incomplete, or inconsistent).
- 7: above average (solid performance with notable but fixable weaknesses).
- 8: very strong (consultant-level performance with only minor issues).
- 9-10: truly exceptional (outstanding on almost all dimensions; reserve for rare cases).

Additional rules:
- If the candidate barely speaks, never
```

```
    proposes a clear structure, or never
     gives a
  concrete recommendation, most scores
      should be in the 0-3 range.
- Do NOT reward verbosity alone; reward
    clear, structured, business-relevant
     thinking.
- Penalize hallucinated facts that
    contradict or go beyond the
    case_text.

Output format:
Return ONLY a single valid JSON object
    with this exact schema:
{
  "structure": float,
  "quant": float,
  "business_sense": float,
  "communication": float,
  "overall": float,
  "feedback": string
}
No extra text before or after the JSON.
```

A.5 Beer Game Configuration

Configuration parameters The Beer Game simulation follows a discrete time setting with a fixed horizon of 25 time steps. Customer demand is observed only by the retailer and follows a stepwise demand script: demand remains at a low level of 100 units during the initial phase and increases to a high level of 400 units starting from week 2. This sudden demand shift introduces non-stationarity and tests the agent's ability to adapt to changing market conditions.

Information and material flows are subject to delays. Orders placed by downstream agents are transmitted upstream with a one-week information delay, while physical shipments experience a two-week delivery delay. The system is initialized in a steady state with a target inventory level of 400 units to avoid transient effects at the beginning of the simulation.

Inventory dynamics incur explicit economic costs. Each unit of inventory held generates a holding cost of 0.5 per time step, while each unit of unmet demand (backorder) incurs a higher penalty of 1.0, reflecting the greater economic impact of stockouts. In addition, a minimum inventory cost of 200 is imposed to model fixed operational expenses independent of inventory fluctuations.

In all experiments, the evaluated agent controls the retailer role, which is closest to customer demand and therefore most exposed to demand uncertainty. All other supply-chain roles (wholesaler, distributor, and factory) follow predefined equation-based policies.

Opponent policies Non-controlled agents in the supply chain follow fixed equation-based ordering rules. Under the *typical* policy, the order quantity at time step t is determined by an inventory and backlog correction rule:

$$q_t = d_t + \alpha\,(I^* - I_t) + \beta B_t, \tag{1}$$

where d_t denotes the observed demand, I_t is the current inventory level, I^* is the target inventory, and B_t represents the backlog. The parameters α and β control the adjustment speed toward the target inventory and backlog compensation, respectively. And in our experiment we set $\alpha = 1$ and $\beta = 1$ as fix parameters.

Under the *smoothing_4* policy, demand is first smoothed using a four-step moving average:

$$\tilde{d}_t = \frac{1}{4}\sum_{k=0}^{3} d_{t-k}, \tag{2}$$

and the smoothed demand $\tilde{d}_t$ is then substituted for d_t in the ordering rule:

$$q_t = \tilde{d}_t + \alpha\,(I^* - I_t) + \beta B_t, \tag{3}$$

A.6 Enterprise Digital Twin (EDT) Implementation Details

This appendix provides low-level execution details of the Enterprise Digital Twin (EDT) task, complementing the main text description. We focus on (i) scenario specification, (ii) step-wise simulation dynamics, (iii) project life-cycle and stochastic events (extension and follow-on), (iv) firm-level accounting and metrics, and (v) the evaluation protocol used in our benchmark.

Here is a brief version of the scenario applied in our task, and we use it as an example for illustrating the mechanism of EDT environment. The details of this task scenario can be found in our codes.

```
"scenarios": {
  "interactive": {

    "runspecs": {
      "starttime": 1,
      "stoptime": 96,
      "dt": 1
    },

    "properties": {
      "revenue_risk_level": {
        "type": "Double",
        "value": 0.5
      },
      "fixed_cost": {
        "type": "Double",
```

```
        "value": 20000.0
      }
    },

    "agents": [
      {
        "name": "consultant",
        "count": 1,
        "properties": {
          "name": {
            "type": "String",
            "value": "Consultant 1"
          },
          "salary": {
            "type": "Double",
            "value": 6000.0
          },
          "workplace_cost": {
            "type": "Double",
            "value": 2000.0
          }
        }
       },

       ... # 11 other consulatant agents

       {
        "name": "project",
        "count": 1,
        "properties": {
          "name": {
            "type": "String",
            "value": "Project 1: Core
                Upgrade"
          },
          "contracted_effort": {
            "type": "Double",
            "value": 70.0
          },
          "contracted_probability": {
            "type": "Double",
            "value": 1.0
          },
          "extension_probability": {
            "type": "Double",
            "value": 0.25
          },
          "extension_effort": {
            "type": "Double",
            "value": 10.0
          },
          "follow_on_probability": {
            "type": "Double",
            "value": 0.1
          },
          "is_follow_on": {
            "type": "Boolean",
            "value": false
          },
          "deadline": {
            "type": "Double",
            "value": 30.0
          },
          "consultants": {
            "type": "Double",
            "value": 2.0
          },
          "start_time": {
            "type": "Double",
            "value": 1.0
          },
          "billing_rate": {
            "type": "Double",
            "value": 16000.0
          }
        }
       },

       ... # 9 other project agents

    ]
```


A.6.1 Scenario specification

An EDT episode is defined by a JSON scenario under a scenario manager (e.g., smEDT), consisting of three top-level blocks: runspecs, properties, and agents. The runspecs block defines the discrete simulation horizon via starttime, stoptime, and dt. In our benchmark implementation, the simulator advances by discrete step calls until termination, and stoptime acts as a step limit . The scalar dt is used as a per-step scaling factor for both work delivery and cost accrual.

The properties block contains firm-level parameters, most importantly the global fixed_cost and revenue_risk_level. The agents block instantiates a multiset of consultant agents and project agents, plus a controlling component that aggregates flows into evaluation metrics.

A tested agent does not act during the episode. Instead, it outputs a compact scenario-level decision schema $\{C, R, P\}$, where C is the retained number of consultants, R is the global revenue_risk_level, and P encodes per-project acceptance and start/deadline windows. This schema is applied as a constrained transformation to a template scenario to produce a materialized scenario JSON. Non-controllable template fields (e.g., salary, workplace cost, and baseline project parameters) remain unchanged. The simulator then executes the materialized scenario for the full horizon and returns step-wise and terminal outcomes.

A.6.2 Consultant dynamics and capacity constraints

Each consultant agent is characterized by two per-step cost parameters: salary and workplace_cost. These costs are counted every step regardless of utilization and same for every consultant in our experiment settings. Let N_c denote the retained number of consultants (set by the tested agent), s denote the salary and w denote the workplace cost. At each step t, the total consultant operating cost

contribution is

$$\mathrm{Cost}_t^c \;=\; N_c \cdot (\mathrm{s} + \mathrm{w}) \cdot dt. \tag{4}$$

Consultants provide the only labor capacity for project delivery. At any step, a consultant can contribute to at most one project. Moreover, each project j has an integer staffing requirement req_j (scenario field consultants) that caps concurrent workers on that project: at any step, no more than req_j consultants can deliver effort to project j. The simulator enforces sticky assignment: once a consultant begins working on a project, it remains assigned to that project until the project finishes its current workload (base scope and any triggered extension), after which the consultant becomes available for reassignment. This stickiness induces nontrivial opportunity costs: starting a long project early can lock capacity and delay higher-margin projects.

Per-step work delivery is modeled in units of *effort*. For a working consultant, delivered effort per step is scaled by dt. Let $k_{j,t}$ be the number of consultants actually working on project j at step t, with $0 \le k_{j,t} \le \mathrm{req}_j$. Then the delivered effort to project j at step t is upper bounded by

$$e_{j,t} \;\le\; k_{j,t} \cdot dt. \tag{5}$$

This definition is consistent with the main-text statement that a consultant produces one unit of effort per step when $dt = 1$ in our settings.

A.6.3 Project state

Each project j is parameterized by:

`contracted_effort`: base scope $\mathrm{Effort}_j^{\mathrm{base}}$

`billing_rate`: per-unit revenue Rate_j

`consultants`: staffing cap req_j

`start_time`: start time start_j

`deadline`: deadline dead_j

`contracted_probability`: reliability π_j^{base}

`extension_probability`: the probability that a project will extend and requires more effort.

`extension_effort` the required effort in extension scope $\mathrm{Effort}_j^{\mathrm{ext}}$

`follow_on_probability` follow-on probability of a project π_j^{fo}

A project is active only within its permissible window. Work can accrue only when $t \ge \mathrm{start}_j$ and $t \le \mathrm{dead}_j$ and the episode has not terminated. Projects maintain a remaining workload state $E_{j,t}$ (initialized to $\mathrm{Effort}_j^{\mathrm{base}}$). Given delivered effort $e_{j,t}$, the update is

$$E_{j,t+1} \;=\; \max\bigl(0,\; E_{j,t} - e_{j,t}\bigr). \tag{6}$$

Revenue is generated proportional to delivered effort. The instantaneous revenue contribution from project j at step t is

$$\mathrm{Rev}_{j,t} \;=\; e_{j,t} \cdot \mathrm{Rate}_j, \tag{7}$$

and total step revenue is

$$\mathrm{Rev}_t \;=\; \sum_j \mathrm{Rev}_{j,t}. \tag{8}$$

If the project reaches its deadline with $E_{j,t} > 0$, the unfinished portion is not deliverable after dead_j, meaning it cannot produce further revenue within that project instance.

A.6.4 Extensions and follow-on projects

EDT introduces two stochastic mechanisms that can create additional revenue opportunities while consuming capacity and increasing uncertainty: extensions (scope growth) and follow-on projects.

Extensions. If a project completes its base scope strictly before its deadline (i.e., $E_{j,t} = 0$ at some $t < \mathrm{dead}_j$), an extension event may trigger. Let $g(R)$ be the risk gating function induced by the global revenue_risk_level $R \in [0, 1]$. Operationally, the extension triggers when a project-level draw passes a threshold that depends on both π_j^{ext} and R. When triggered, the project remaining workload is increased by $\mathrm{Effort}_j^{\mathrm{ext}}$:

$$E_{j,t} \leftarrow E_{j,t} + \mathrm{Effort}_j^{\mathrm{ext}}. \tag{9}$$

The project then continues to consume consultant capacity and can generate additional revenue as the extension workload is delivered, subject again to the deadline and episode termination.

Follow-on projects. At a project's deadline step $t = \mathrm{dead}_j$, a follow-on opportunity may be instantiated. As with extensions, instantiation is gated by the global risk level R and the project parameter π_j^{fo}. A follow-on project is created as a new project agent with is_follow_on = true. It inherits the primary economic structure of its parent project (e.g.,

similar staffing requirement and billing rate) while suppressing further follow-on chaining (follow-on probability set to zero), preventing infinite cascades. The follow-on has its own start time after creation and competes for the same consultant pool.

These stochastic mechanisms create non-linear portfolio effects. High R increases the chance of extensions and follow-ons, potentially improving earnings but also locking capacity and raising uncertainty; low R yields more predictable revenue trajectories but less upside.

A.6.5 Firm-level accounting and returned metrics

At each step t, the firm accrues expenses and revenue, which are aggregated by the controlling component into cumulative metrics. Let $\mathrm{Cost}_t^{\mathrm{fix}} = \mathrm{fixed\_cost} \cdot dt$ denote step fixed cost, and let Cost_t^c denote consultant costs from Eq. (1). Then total step expenses are

$$\mathrm{Exp}_t \;=\; \mathrm{Cost}_t^{\mathrm{fix}} + \mathrm{Cost}_t^c. \tag{10}$$

Cumulative revenue and expenses are computed as running sums:

$$\mathrm{AcRev}_T = \sum_{t=1}^{T} \mathrm{Rev}_t, \tag{11}$$

$$\mathrm{AcExp}_T = \sum_{t=1}^{T} \mathrm{Exp}_t. \tag{12}$$

Cumulative earnings (profit) are then

$$\mathrm{Earnings}_T \;=\; \mathrm{AcRev}_T - \mathrm{AcExp}_T. \tag{13}$$

Utilization is computed from the fraction of consultant capacity actively engaged in delivery. Let busy_t denote the number of consultants assigned to any project at step t. Then step utilization is

$$\mathrm{Util}_t \;=\; \frac{\mathrm{busy}_t}{N_c}, \tag{14}$$

with $\mathrm{Util}_t = 0$ when $N_c = 0$. The simulator reports both per-step utilization and an overall average utilization computed across steps.

A.6.6 Evaluation protocol in the benchmark

For each tested model, evaluation proceeds over a fixed set of template scenarios. For each episode, the model is provided with a structured description of (i) horizon, (ii) cost parameters, and (iii) project parameters. It outputs the schema $\{C, R, P\}$ subject to strict formatting constraints. The evaluator materializes a new scenario by applying the schema to the template (disabling projects, adjusting start/deadline windows, setting R, and selecting the first C consultants). The BPTK server is then launched to execute the scenario, and the evaluator reads step-wise outputs to compute and store metrics (including cumulative earnings, revenue, expenses, cash, utilization, and revenue risk). When multiple runs per scenario are enabled, the model may adapt its schema based on prior-run feedback, enabling learning-style search over scenario configurations under a fixed action space.

A.7 Dataset Statistics

We provide the detailed sample distribution of the FirmBench task corpus. Table 5 summarizes the number of samples in the training, validation, and testing sets for each component. Note that for our currently implemented *online* evaluation mode, only the *test* set is utilized for evaluation and sequential adaptation.

In addition, we introduce the Beer Game and Enterprise Digital Twin (EDT) as simulation-based serious game tasks, which differ fundamentally from the sample-based datasets.

Dataset	Train	Valid	Test	Total
Structured Reasoning				
CodeFinQA	4,409	200	788	5,397
CodeTAT-QA	2,654	200	288	3,142
ConvFinQA	133	-	132	265
FinCode	7	2	47	56
finer	1,000	500	441	1,941
FinKnow	100	50	589	739
formula	500	300	200	1,000
FormulaEval	50	-	50	100
SEC-NUM	6,646	200	2,000	8,846
TAT-QA	120	-	120	240
Interactive Consulting	12	-	48	60
Total	**15,631**	**1,452**	**4,706**	**21,789**

Table 5: Statistics of FirmBench tasks across training, validation, and testing splits.

A.8 AI Assistance Disclosure

AI-based tools were used during the preparation of this work to assist with code development and to correct grammatical and stylistic issues in the manuscript. All scientific content, experimental design, results, and conclusions were conceived, implemented, and verified by the authors.