TIDE-BENCH: TASK-AWARE AND DIAGNOSTIC EVALUATION OF TOOL-INTEGRATED REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-integrated reasoning has emerged as a promising paradigm for enhancing large language models with external computation, retrieval, and execution capabilities. However, the field still lacks a high-quality and unified evaluation benchmark, and existing TIR evaluations remain limited in dataset quality, task diversity, diagnostic comprehensiveness, and evaluation efficiency. In this work, we introduce TIDE-Bench, a holistic and efficient benchmark for evaluating TIR methods, featuring three key advantages. First, it provides diverse task settings, combining widely used mathematical reasoning and knowledge-intensive QA tasks with two newly designed tasks, tool-grounded experimental design task and dynamic interactive task, to probe models' abilities in complex tool invocation and multi-tool coordination. Second, TIDE-Bench adopts a comprehensive yet task-aware evaluation protocol, jointly measuring final answer quality, process reliability, tool-use efficiency, and inference cost across heterogeneous task settings. Third, TIDE-Bench constructs compact and discriminative evaluation sets through tool-aware sample filtering, retaining tool-supported instances and prioritizing those with lower no-tool performance. This substantially reduces evaluation cost while focusing evaluation on samples that are more informative for measuring the contribution of tool use. Extensive experiments on multiple foundation models and TIR methods reveal persistent bottlenecks in tool grounding, offering insights for future TIR research.

1 INTRODUCTION

Large Language Models (LLMs) have achieved significant progress in complex reasoning tasks across diverse domains. However, reasoning solely from internal model parameters remains insufficient for many real-world problems: model knowledge can be outdated or incomplete, symbolic operations require precise executable computation, and dynamic tasks often demand interaction with external environments or feedback signals. Tool-Integrated Reasoning (TIR) has therefore emerged as a promising paradigm that augments LLM reasoning with external retrieval, computation, code execution, and environment interaction (Yao et al., 2023; Schick et al., 2023; Qin et al., 2023; Zhao et al., 2025). As tool use becomes central to agentic systems, it is increasingly important to evaluate whether models can invoke appropriate tools, interpret their outputs, and ground final responses in tool-mediated evidence.

Despite recent progress, existing TIR evaluations remain limited in the following three aspects. First, **existing evaluation tasks lack sufficient diversity and quality**. In the absence of a unified benchmark for TIR, prior methods typically rely on ad hoc combinations of standard math reasoning and knowledge-intensive QA datasets, such as GSM8K and MATH for code-assisted mathematical reasoning (Cobbe et al., 2021; Hendrycks et al., 2021), and HotpotQA and MuSiQue for retrieval-augmented or multi-hop question answering (Yang et al., 2018; Trivedi et al., 2022). However, these datasets instantiate TIR mainly as tool use over static instances, such as code execution for math or retrieval for closed-ended QA. As a result, these tasks provide limited support for evaluating complex tool-use capabilities, including the ability to coordinate heterogeneous tools in open-ended problem solving, translate underspecified goals into executable tool-grounded procedures, and maintain reliable tool use across multi-turn and stateful interactions(Lu et al., 2025; Yao et al., 2024).

Second, existing evaluations often **lack diagnostic granularity** to measure how tools contribute to reasoning. Most benchmarks only report answer-level scores such as accuracy or pass@k, while

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

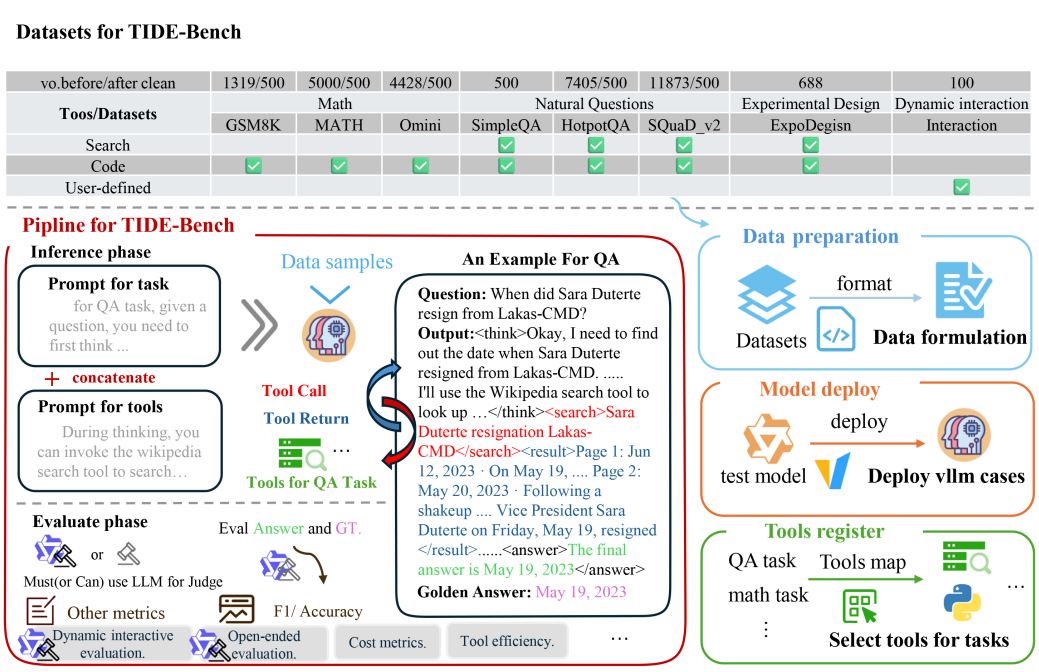

Figure 1: The overall framework of TIDE-Bench. The upper part summarizes the benchmark datasets and the tool types supported by each task. And the lower part illustrates the inference and evaluation pipeline of TIDE-Bench.More details about user-defined tools in Interaction is provided in Appendix B.

only some additionally report the number or success rate of tool calls. These metrics do not show whether a correct answer is grounded in tool outputs, whether the model misuses retrieved evidence or execution results, or whether success relies on many unnecessary calls and long contexts.

Third, existing evaluations also **lack sufficient efficiency and scalability** to support iterative model development. As LLMs continue to improve, early benchmarks become less discriminative, with many instances now solvable by foundation models even without invoking external tools. These low-discrimination instances provide limited value for evaluating tool-integrated reasoning, yet they still incur substantial inference time, tool-calling overhead, and long-context computation when included in large evaluation suites. At the same time, samples that are difficult without tools are not necessarily informative for TIR if their answers cannot be supported by the available tool outputs. This makes it important to identify samples that are both tool-supported and challenging without tool access, while keeping the resulting evaluation suite computationally efficient.

To address these limitations, we introduce the **T**ool-**I**ntegrated **D**iagnostic **E**valuation **Bench**mark (TIDE-Bench), a diverse, diagnostic, and efficient benchmark for TIR. As illustrated in Figure 1, TIDE-Bench combines community-standard mathematical and knowledge-intensive QA datasets with two new tracks: Tool-Grounded Experimental Design (ExpoDesign) and Dynamic Interaction (Interaction). This design preserves comparability with established closed-ended benchmarks while extending evaluation to discovery-oriented experimental planning and deployment-oriented interactive tool orchestration that requires complex tool invocation and multi-tool coordination. TIDE-Bench also adopts a unified task-aware protocol to evaluate answer quality, process reliability, tool-use efficiency, latency, and token cost, diagnosing whether tools improve reasoning reliably and economically. Finally, it adopts a tool-aware filtering strategy that first retains tool-supported samples and then prioritizes those with lower performance under the no-tool setting, reducing benchmark cost while focusing evaluation on samples that are both tool-relevant and challenging without tool access.

Our main contributions are:

1. We introduce TIDE-Bench, a unified benchmark for TIR that preserves comparability with standard math and knowledge-intensive QA tasks while extending evaluation to two underexplored settings: ExpoDesign and Interaction.

2. We construct a compact yet discriminative evaluation suite through systematic sample filtering. We first retain instances whose tool-enabled responses are supported by the corresponding tool outputs, and then prioritize samples with lower no-tool F1 or accuracy scores. This procedure produces 500-sample subsets that are tool-relevant and challenging without tool access, substantially reducing the cost of repeated evaluation.

3. We propose a task-aware evaluation protocol that measures final outcomes, reasoning processes, and tool-use cost. Beyond answer quality, TIDE-Bench diagnoses whether tool calls are grounded in the reasoning process, contribute to reliable answers, and justify their computational cost.

4. We conduct systematic experiments on multiple open-source and proprietary models, comparing tool-enabled and no-tool settings where applicable. The results show that tool benefits are strongly task-dependent: retrieval-heavy tasks benefit more consistently, while the benchmark exposes persistent challenges in multi-hop evidence composition, faithful mathematical formalization, and multi-turn tool orchestration.

2 Related Work

Tool-Use Evaluation. Existing evaluations of tool use have primarily focused on whether models can select appropriate tools, generate valid API or function calls, and execute them successfully. Representative benchmarks construct executable API environments, scale evaluation to large collections of real-world tools, or assess structured serial and parallel function calling under predefined schemas (Li et al., 2023; Patil et al., 2024; Qin et al., 2023; Patil et al., 2025). Subsequent studies further decompose, consolidate, or extend these evaluations by examining benchmark design, tool-learning protocols, and models' decisions about when and what types of tools to invoke (Chen et al., 2024; 2025a; Huang et al., 2024; Ning et al., 2024). While these studies provide an important foundation for evaluating tool-augmented language models, their focus remains largely on tool-use behavior itself. In contrast, TIR evaluates whether models can interpret tool outputs, integrate them into intermediate reasoning, and use them to produce reliable final answers. Thus, tool-selection accuracy, invocation success, or execution correctness alone cannot fully characterize TIR capabilities.

TIR Evaluation Practices. Recent TIR studies have developed diverse methods for improving tool-integrated reasoning, including reasoning-trajectory construction, automated tool invocation, reinforcement learning, search-augmented reasoning, and tool-preference optimization (Xue et al., 2025; Wei et al., 2025; Qian et al., 2025; Chen et al., 2025b; Jin et al., 2025a; Dong et al., 2025a; Zhang et al., 2025). Their evaluations commonly rely on mathematical reasoning, retrieval-augmented question answering, demonstrating the utility of tools such as search and code interpreters for complex reasoning. However, these evaluations are primarily method-oriented: they are designed to measure performance gains on specific task settings, with metrics such as accuracy, F1, pass@k, or tool-call counts. As a result, they offer limited diagnostic insight into when tools help, whether tool outputs are reliably grounded in reasoning, and how gains trade off against cost across task formats.

Generalist Agent Evaluation. A related line of work evaluates generalist agents in interactive or real-world environments. Representative benchmarks assess multi-step decision making, environment interaction, web navigation, computer use, and broader abilities involving reasoning, tool use, multimodal understanding, and information integration (Liu et al., 2023; Mialon et al., 2023; Zhou et al., 2023; Deng et al., 2023; Xie et al., 2024). While these benchmarks are closer to practical applications, they evaluate broad agent performance and often entangle planning, exploration, interaction, perception, tool invocation, and long-horizon task completion. This makes it difficult to isolate TIR-specific bottlenecks; we instead target whether models can invoke tools across task formats and turn tool outputs into accurate, cost-effective reasoning.

Table 1: Performance of Qwen3-32B before and after cleaning on Natural Questions and MATH benchmarks. Among them, for the SimpleQA dataset, we directly used the dataset provided in Tool-Light (Chen et al., 2025b) rather than the original source dataset, which contains 4,326 samples (Wei et al., 2024).

		Natural Questions			Math		
		SimpleQA	HotpotQA	SQuAD_v2	GSM8K	MATH	OmniMath
Before Cleaning	samples	500	7405	11873	1319	500	4428
	test time (s)	2004.7	29382.9	35177.0	11189.4	36878.9	557120.5
	f1 score/accuracy	0.09	0.30	0.12	0.92	0.79	0.27
After Cleaning	samples	500	500	500	500	500	500
	test time (s)	2004.7	2747.7	1419.7	3435.4	36878.9	60612.1
	f1 score/accuracy	0.09	0.06	0.05	0.86	0.79	0.00

Table 2: Coverage of reported evaluation suites in representative TIR lines. **NQ**/Search: retrieval- or web-search-based QA; **Math**/Code: math with code execution; **Interaction** and **ExpoDesign** denote the two newly introduced tracks.

Benchmark	**Natural Questions**	**Math**	**Interaction**	**ExpoDesign**
	Search	Code	User-defined	Code+Search
TIDE-Bench (ours)	✓	✓	✓	✓
Understand TIR(Lin & Xu, 2025)	✗	✓	✗	✗
Tool-Light(Chen et al., 2025b)	✓	✓	✗	✗
ToolRL(Qian et al., 2025)	✓	✓	✗	✗
AutoTIR(Wei et al., 2025)	✓	✓	✗	✗
SimpleTIR(Xue et al., 2025)	✗	✓	✗	✗
Agentic Reasoning(Wu et al., 2025)	✓	✓	✗	✗
Tool-Star(Dong et al., 2025a)	✓	✓	✗	✗
Search-R1(Jin et al., 2025a)	✓	✗	✗	✗
OTC-PO(Wang et al., 2025a)	✓	✓	✗	✗
ReasonZoo(Zhao et al., 2025)	✓	✓	✗	✗

3 METHOD

3.1 HIGH-QUALITY AND DIVERSE DATASET RECONSTRUCTION

In the absence of a unified benchmark specifically designed for tool-integrated reasoning, existing TIR methods typically construct their evaluations from ad hoc combinations of standard QA and mathematical reasoning datasets. We analyze these commonly used evaluation sources and identify two limitations that hinder a reliable and comprehensive assessment of TIR abilities.

First, existing QA and mathematical reasoning datasets contain many low-discrimination samples that can be solved effectively without tool use. On popular datasets such as HotpotQA (Yang et al., 2018), SQuADv2 (Rajpurkar et al., 2018), and GSM8K (Cobbe et al., 2021), we find that Qwen3-32B can solve a substantial portion of samples without invoking external tools. However, samples that are difficult without tools are not necessarily suitable for TIR evaluation if the available tools cannot provide useful support. We therefore evaluate Qwen3-32B under both tool-enabled and no-tool settings, first retaining samples with $\mathrm{Supported} = 1$, and then selecting the 500 samples with the lowest no-tool F1 scores for knowledge-intensive QA tasks or accuracy scores for mathematical reasoning tasks. As shown in Table 1, this filtering strategy substantially reduces evaluation time while retaining samples that are tool-supported and more challenging without tool access. Appendix A details the corresponding data-cleaning procedure, screening configuration, and evaluation hyperparameters.

Second, as shown in Table 2, the evaluation of existing methods is largely restricted to QA and mathematical reasoning tasks, where both task complexity and the necessity of tool use remain limited.

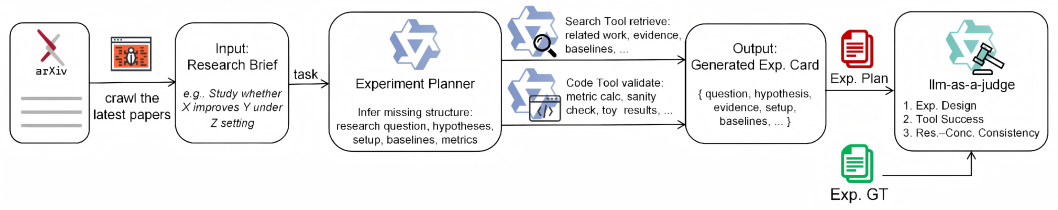

Figure 2: Pipeline of Pipeline of ExpoDesign. Given an underspecified research prompt, the model retrieves external evidence, uses code for lightweight execution or validation, and produces a structured experiment card that is evaluated by an LLM judge for design quality, tool-grounded plausibility, and result–conclusion consistency.

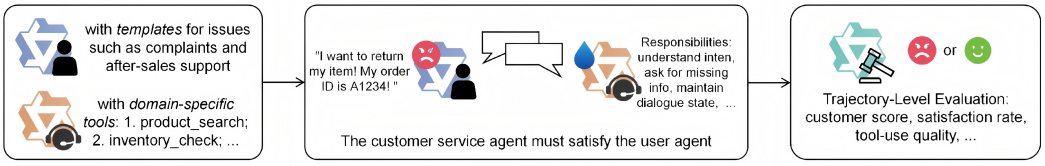

Figure 3: Pipeline of Interaction. In a simulated e-commerce service scenario, the agent interacts with a customer across multiple turns, invokes domain-specific tools to verify products, orders, policies, inventory, and prices, and is evaluated over the full trajectory for dialogue success and tool-use quality.

These benchmarks typically provide only a small set of tools, often search and code execution, and the tasks themselves usually do not require sophisticated tool coordination. For example, as shown in Table 7 in Appendix C, even when QA tasks allow access to both search and code tools, models tend to rely almost exclusively on the search tool to solve the problems. This suggests that existing TIR tasks are often limited in both task design and tool configuration, making them insufficient for effectively evaluating TIR methods—especially their ability to coordinate multiple tools to support complex reasoning. Moreover, compared with real-world decision-making scenarios, traditional math and QA tasks are relatively simple and less representative of practical tool-integrated applications. To address these limitations, we introduce two more complex tasks:

Tool-Grounded Experimental Design (ExpoDesign). As illustrated in Figure 2, this task starts from an underspecified research prompt and asks the system to construct a tool-grounded mini-experiment. For each evaluation case, TIDE-Bench provides a paper-agnostic minimal-experiment brief and an evaluation protocol specifying the interaction format and the expected final deliverable. Given the brief, the system needs to infer a complex structured experiment specification, including a testable research question, falsifiable hypotheses, relevant evidence, data and evaluation metrics, weak and strong baselines, experimental procedure, preliminary results, brief analysis, and limitations. The central challenge is that the system must not only complete the missing experimental structure from an open-ended research prompt, but also coordinate heterogeneous tools within a single reasoning process. Search is used to ground the research context, collect external evidence, and support hypothesis formation, while code operationalizes assumptions through preliminary checks, metric computation, lightweight simulation, or result validation. During inference, the system may invoke tools over multiple rounds and finally returns a structured experiment card in JSON format. To prevent retrieval shortcuts, the input omits metadata that could uniquely identify the source paper, such as the title, full abstract, author list, URL, or arXiv ID. We construct the dataset by sampling STEM and computer science papers according to the 2025 arXiv distribution and pairing each record with a human-written or model-generated task brief. Because the task does not admit a unique gold answer, we evaluate each experiment card with an LLM judge along three dimensions: design quality, tool-grounded plausibility, and result–conclusion consistency.

Dynamic Interaction (Interaction). As illustrated in Figure 3, Interaction is defined as a multi-scenario, multi-turn e-commerce service-resolution task. Given a scripted customer scenario, the

system acts as a merchant service agent, resolves the user's request through natural-language dialogue, and invokes domain-specific tools when needed to query product, inventory, policy, order, and pricing information. The task covers five service scenarios, including complaints, transactions, after-sales support, clarification, and misunderstanding or adversarial turns, requiring the system to progress toward service resolution over the full dialogue rather than produce a single-turn answer. In this task, TIR is instantiated as a stateful tool-orchestration problem. The agent must understand evolving user intent, ask for missing information, maintain task state across turns, follow business constraints, and ground user-facing responses in tool outputs. Therefore, the task evaluates whether TIR systems can coordinate multiple tools, track dialogue state, respect domain constraints, and produce tool-supported responses in deployment-oriented interactive settings. To instantiate this setting, we construct a dialogue-based e-commerce environment in which the evaluated model acts as the merchant service agent, while a fixed API model simulates customers with diverse goals, tones, and information conditions. The benchmark contains 100 scripted scenarios balanced across the five service categories. Each scenario specifies a customer persona, goal, tone, optional missing information, product and order context, and an opening user message. During each episode, the agent interacts with the customer for up to ten turns and may invoke five user-defined tools: product search, inventory checking, policy retrieval, order lookup, and price calculation. Tool outputs are visible only to the agent and are merged into its context, while the simulated customer observes only the plain dialogue. This design requires the agent to verify facts and constraints through tools before responding, rather than relying on unsupported assumptions. We evaluate each episode over the full trajectory, including both the dialogue outcome and the tool-use trace. The customer-side score is used as the primary measure of task success, while satisfaction rate and average tool calls serve as auxiliary diagnostics for interaction quality and tool-use behavior.

3.2 Evaluation Metrics

In this work, we propose a comprehensive and task-aware evaluation protocol that assesses TIR methods from four complementary perspectives: answer quality, process reliability, tool-use efficiency, and computational cost.

Answer quality. For mathematical and knowledge-intensive tasks, an **LLM-as-a-judge** protocol produces binary correctness decisions whose mean is the primary **accuracy**. Auxiliary diagnostics include EM, token-level F1, and substring-hit accuracy for QA, and math-level accuracy for mathematical tasks. ExpoDesign scores **experimental design**, **tool success**, and **result–conclusion consistency** on the original 1–5 rubric and averages the three dimensions. Interaction reports the **customer-facing score** on its original 0–5 scale, with satisfaction rate and tool-call behavior as separate diagnostics. We normalize ExpoDesign and Interaction scores to $[0, 1]$ as $s_E = (r_E - 1)/4$ and $s_I = r_I/5$, respectively; uncertainty conversion and human validation are detailed in Appendix D.

Process reliability. Final-answer quality alone is not sufficient for evaluating TIR methods. A model may produce a correct answer while relying on inconsistent reasoning, or invoke tools without actually grounding its response in tool outputs. We therefore introduce two process-level metrics. **Reasoning–answer consistency** measures whether the final answer is supported by the reasoning trace enclosed in `<think>...</think>`. **Tool supportedness** measures whether the final answer is grounded in evidence returned by tools inside `<result>...</result>`. For QA tasks, tool supportedness allows weak matching through direct containment or sufficiently high token overlap; for mathematical tasks, numeric equivalence is additionally accepted.

Tool-use efficiency. Beyond task quality, we explicitly evaluate whether a system uses tools economically. Let c_i denote the number of tool calls in sample i, and let q_i denote the task-specific headline quality score: answer correctness for QA and math, the normalized ExpoDesign score, and the normalized Interaction customer score. We define tool efficiency as the average of q_i/c_i over all samples with $c_i > 0$:

$$\text{ToolEfficiency} = \frac{1}{|\{i \mid c_i > 0\}|} \sum_{i:c_i>0} \frac{q_i}{c_i}.$$

This conditional quality-per-call measure is not a pure cost metric; it excludes zero-call samples and must be read alongside task quality, call counts, and token cost.

Computational cost. Finally, we report cost-related statistics, since the practical utility of a TIR method depends not only on output quality but also on latency and inference cost. We therefore track the average number of tool calls, total wall-clock time, average latency per sample, and—when tokenization is available—the total and average number of tokens in the prompt and generated trajectory. These metrics support joint analysis of effectiveness and efficiency.

4 EXPERIMENT

4.1 EXPERIMENTAL SETUP

Datasets. TIDE-Bench establishes a diverse and challenging evaluation environment for TIR systems by including not only cleaned, high-quality mathematical and knowledge-intensive tasks, but also two newly designed tasks that are more complex and better reflect real-world scenarios. The **mathematical tasks** include GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and Omni-MATH (Gao et al., 2024). The **knowledge-intensive tasks** include SimpleQA (Wei et al., 2024), HotpotQA (Yang et al., 2018), and SQuAD v2 (Rajpurkar et al., 2018). In addition, we introduce two new task settings, **ExpoDesign** and **Interaction**, which extend TIR evaluation to discovery-oriented and deployment-oriented scenarios.

Models and settings. We evaluate seven representative systems, including the Qwen3 series models (Yang et al., 2025) (8B and 32B), GPT-4o (Hurst et al., 2024), GPT-5 (Singh et al., 2025), GPT-5.4 (OpenAI, 2025), SearchR1 (Jin et al., 2025a), and Tool-Star (Dong et al., 2025a). These systems cover commonly used open-source models of different scales, recent closed-source models, and tool-tuned reasoning systems developed for advanced tool-integrated reasoning. For mathematical and knowledge-intensive datasets, each model is evaluated in a matched *no-tool* and *tool-enabled* setting, so the reported gain isolates the effect of allowing external tool calls. For ExpoDesign and Interaction, we report only the tool-enabled setting, since tool invocation results are part of the task-specific evaluation and the no-tool setting is therefore not applicable. All primary results are reported in Table 3; detailed results for diagnostic metrics are provided in Appendix C.

In our experiments, we use `kimi-k2-0905-preview` as the LLM judge for both answer correctness evaluation and task-specific quality scoring. We report agreement between the LLM judge and human assessments on 100 samples per new task in Appendix D (Table 12). The main results table reports the primary metric for each task type: accuracy for mathematical and knowledge-intensive tasks, the overall score for ExpoDesign, and the average customer score for Interaction.

4.2 MAIN RESULTS

Table 3 shows that TIDE-Bench remains challenging beyond the relatively high scores on GSM8K and MATH. On Omni-MATH, the best no-tool accuracy is only 0.630, with substantially lower scores for most open-source and tool-specialized systems. SimpleQA and HotpotQA expose factual-retrieval and evidence-integration bottlenecks, while SQuAD v2 tests reliable no-answer judgment. The new tracks are also unsaturated: only GPT-5.4 and GPT-5 exceed 0.875 on ExpoDesign, whereas all open-source and tool-specialized baselines remain below 0.525; Interaction scores range from 0.324 to 0.906. These results motivate further research on tool-grounded reasoning, evidence integration, and interactive tool use.

By comparing the matched tool-enabled and no-tool results, we find that tool access does not lead to universal improvement; its effect depends on whether the available tool matches the main task bottleneck. As shown in Table 3 and Figure 4, the most consistent improvement appears on SimpleQA, where enabling tools raises the average accuracy by $+0.306$ across systems. SearchR1, Qwen3-32B, and GPT-4o all obtain large gains on this task, indicating that the search tool is well aligned with short-form factual QA and that such tasks strongly depend on external evidence. HotpotQA also improves on average by $+0.111$, although the gain remains smaller than that on SimpleQA, suggesting that retrieval becomes less directly effective when the model must integrate evidence across multiple hops. In contrast, the gains on mathematical reasoning are weak and unstable: GSM8K and MATH slightly degrade on average, while Omni-MATH changes only marginally. This suggests that code execution does not automatically improve mathematical reasoning unless the model can first formalize the problem into a correct executable procedure. SQuAD v2 also changes only marginally

Table 3: Main results on TIDE-Bench across eight datasets. Mathematical and knowledge-intensive datasets are evaluated by accuracy, while both new tasks use normalized 0–1 scores. SearchR1 and Tool-Star are TIR-oriented fine-tuned baselines, specializing in search-based and multi-tool reasoning, respectively. Standard deviations over three runs are reported in the Appendix C.

Setting	Model	Math			Knowledge-intensive QA			Ours	
		GSM8K Acc.	MATH Acc.	Omni Acc.	SimpleQA Acc.	HotpotQA Acc.	SQuAD_v2 Acc.	ExpoDesign Norm.	Interaction Norm.
No-Tool	Qwen3-8B	0.890	0.834	0.176	0.038	0.176	0.210	–	–
	Qwen3-32B	0.926	0.880	0.256	0.056	0.206	0.302	–	–
	GPT-4o	0.930	0.776	0.150	0.420	0.556	0.380	–	–
	GPT-5	0.946	0.838	0.536	0.524	0.666	0.436	–	–
	GPT-5.4	0.950	0.966	0.630	0.398	0.534	0.436	–	–
	SearchR1	0.478	0.312	0.088	0.044	0.128	0.196	–	–
	Tool-Star	0.870	0.772	0.298	0.378	0.280	0.256	–	–
Tool	Qwen3-8B	0.894	0.740	0.240	0.506	0.474	0.272	0.496	0.816
	Qwen3-32B	0.922	0.774	0.306	0.532	0.262	0.026	0.508	0.906
	GPT-4o	0.934	0.780	0.302	0.626	0.616	0.448	0.381	0.324
	GPT-5	0.956	0.964	0.432	0.546	0.488	0.410	0.920	0.554
	GPT-5.4	0.944	0.964	0.586	0.536	0.568	0.426	0.946	0.714
	SearchR1	0.346	0.334	0.096	0.708	0.464	0.380	0.193	0.420
	Tool-Star	0.890	0.736	0.206	0.548	0.450	0.235	0.383	0.492

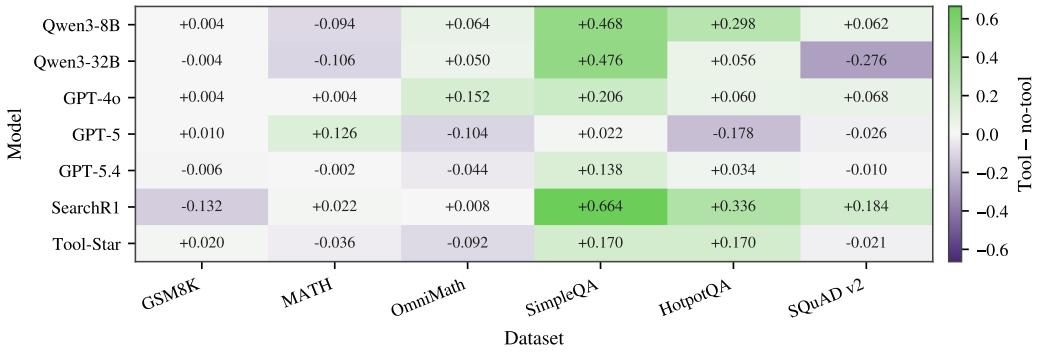

Figure 4: Tool gain over the no-tool baseline on mathematical and knowledge-intensive tasks. Each cell shows the difference between the tool-enabled and no-tool primary metric for a given model–dataset pair. Positive values indicate performance gains from tool access, while negative values indicate degradation.

on average (-0.003), with substantial variation across models, suggesting that additional retrieval does not consistently improve answerability-sensitive QA and may help or interfere depending on how effectively the model uses the retrieved evidence. These results suggest that future TIR systems should not only improve tool access, but also enhance tool design, tool-grounding ability, and the ability to determine when tool use is necessary or beneficial.

The new tracks produce rankings distinct from standard math and QA tasks. GPT-5.4 and GPT-5 lead ExpoDesign with normalized scores of 0.946 and 0.920, respectively, indicating higher judge-assessed quality under the current rubric. In contrast, Qwen3-32B leads Interaction with a customer-facing score of 0.906. This track stresses evolving user intent, state tracking, business constraints, and grounded responses across turns. The different rankings show that retrieval QA, mathematical reasoning, experimental design, and interactive service resolution test complementary dimensions of TIR.

4.3 Benefit–Cost Trade-offs of Tool Use

Although tool invocation can improve task performance, it may also introduce additional computational costs, including more tool calls, longer contexts, and larger token overhead. This motivates our benefit–cost trade-off analysis of tool use, rather than evaluating tool use solely by performance gain. As shown in Figure 5, SimpleQA achieves the most significant average gain with moderate

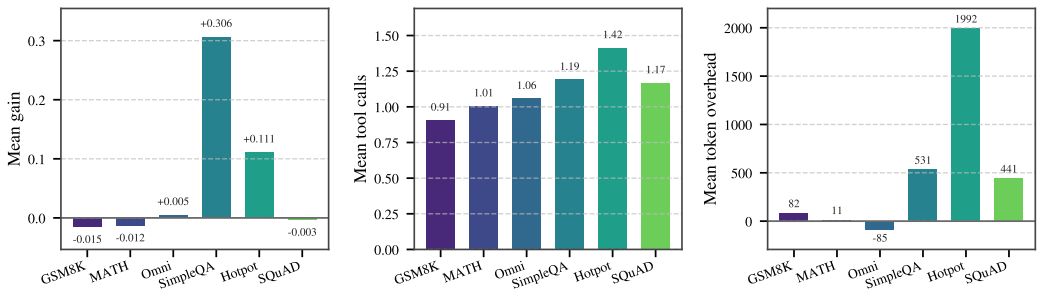

Figure 5: Performance gains and computational costs of tool use across mathematical and knowledge-intensive datasets. Left: average performance gain from enabling tool use relative to the no-tool setting. Middle: mean number of tool calls in the tool-enabled setting. Right: mean change in token usage introduced by tool use, where negative values indicate token savings.

cost, whereas HotpotQA incurs the highest token overhead while obtaining a smaller but still positive performance improvement. Both tasks involve the search tool, but their returns differ: SimpleQA can often be answered once the model retrieves a single decisive fact, while HotpotQA still requires multi-hop filtering and evidence composition after retrieval. This contrast indicates that tool utility depends not only on tool availability, but also on the amount of post-retrieval reasoning required by the task.

Dataset-level process and cost diagnostics and their analysis are provided in Appendix C (Table 7).

The negative or weak gains on mathematical tasks and SQuAD v2 highlight the importance of deciding *when tools are actually necessary*. For mathematical tasks, tool calls may reduce the length and token cost of the reasoning process without necessarily improving the final answer. This helps explain why the token overhead remains small for MATH and even becomes negative for Omni-MATH in Figure 5: Python execution may replace part of the model's long-form textual reasoning with shorter code-and-result traces, especially on harder problems. However, the weak performance gains indicate that lower generation cost does not necessarily translate into better reasoning quality. For SQuAD v2, additional retrieval provides little average performance benefit and exhibits substantial variation across models, because the model must determine whether a question is answerable at all rather than simply retrieve relevant evidence. These cases suggest that future TIR training should optimize not only tool-call success, but also call necessity, evidence selection, and abstention under insufficient support.

In addition to the metrics discussed above, ExpoDesign and Interaction also provide a range of fine-grained diagnostic metrics, which are introduced in the *Answer Quality* subsection of our methodology. More specifically, ExpoDesign evaluates experimental design, tool success, and result–conclusion consistency, while Interaction reports customer-facing success, satisfaction, and tool-call behavior as separate diagnostics. These metrics enable a more comprehensive evaluation of the TIR reasoning process, and the detailed results are provided in Appendix C. These views complement the primary scores by helping localize where a TIR system fails rather than only whether it fails.

5 Conclusion

We presented TIDE-Bench, a task-aware and diagnostic benchmark for Tool-Integrated Reasoning. It combines compact mathematical reasoning and knowledge-intensive QA subsets with two tool-mediated tracks, ExpoDesign and Interaction, and evaluates final quality, process reliability, tool-use efficiency, and inference cost under a unified protocol. Our experiments show that tool use is beneficial only when it matches the task bottleneck. Search helps most on direct factual QA, but is less reliable for multi-hop evidence composition and answerability-sensitive QA; code execution helps mathematical reasoning only when the problem is first formalized correctly. The two new tracks further expose bottlenecks in evidence-grounded experimental planning and stateful multi-turn tool orchestration. These findings motivate selective, grounded, and cost-aware tool use. TIDE-Bench provides an efficient testbed for diagnosing tool-use benefits and failures.

AI USE STATEMENT

We used ChatGPT to assist with language editing, refinement of descriptions of the methodology and experimental results, LaTeX formatting, organization and summarization of experimental results, review of tool-use trajectories, and implementation of code for repetitive tasks. We reviewed and revised the AI-assisted text, verified the organized results against the experimental outputs, checked trajectory-review findings against the original traces, and validated the assisted code. The use of LLMs as evaluated systems and automated judges is described in the methodology and experimental setup. We take full responsibility for the accuracy, originality, and integrity of the final manuscript.

REFERENCES

Aaron Adcock, Aayushi Srivastava, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pande, Abhinav Pandey, Abhinav Sharma, Abhishek Kadian, Abhishek Kumawat, Adam Kelsey, et al. The llama 4 herd: Architecture, training, evaluation, and deployment notes. *arXiv preprint arXiv:2601.11659*, 2026.

Haotian Chen, Zijun Song, Boye Niu, Ke Zhang, Litu Ou, Yaxi Lu, Zhong Zhang, Xin Cong, Yankai Lin, Zhiyuan Liu, et al. Toleap: Rethinking development of tool learning with large language models. *arXiv preprint arXiv:2505.11833*, 2025a.

Yifei Chen, Guanting Dong, and Zhicheng Dou. Toward effective tool-integrated reasoning via self-evolved preference learning. *arXiv preprint arXiv:2509.23285*, 2025b.

Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, et al. T-eval: Evaluating the tool utilization capability of large language models step by step. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9510–9529, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*, 2025a.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization, 2025b. URL https://arxiv.org/abs/2507.19849.

Bofei Gao, Feifan Song, Zhe Yang, Zeyu Cai, Yibo Zhang, Yutong Liang, Tiyu Ma, Jiaqi Mi, Lijun Li, et al. Omni-MATH: A universal olympiad level mathematic benchmark for large language models, 2024. URL https://arxiv.org/abs/2410.07985.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, 2021.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. Metatool benchmark for large language models: Deciding whether to use tools and which to use. In *International Conference on Learning Representations*, 2024.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025a.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 737–740, 2025b.

Minghao Li, Fei Yang, Bin Yang, Shang Wang, Ying Wang, Song Luo, Ze Li, Ying Chen, Wanxiang Xu, and Dayiheng Liu. API-Bank: A benchmark for tool-augmented llms, 2023. URL https://arxiv.org/abs/2304.08244.

Heng Lin and Zhongwen Xu. Understanding tool-integrated reasoning. *arXiv preprint arXiv:2508.19201*, 2025.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangli Ding, Kaiwen Men, Keqin Yang, et al. AgentBench: Evaluating llms as agents, 2023. URL https://arxiv.org/abs/2308.03688.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scao. GAIA: A benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Kangyun Ning, Yisong Su, Xueqiang Lv, Yuanzhe Zhang, Jian Liu, Kang Liu, and Jinan Xu. WTU-EVAL: A whether-or-not tool usage evaluation benchmark for large language models, 2024. URL https://arxiv.org/abs/2407.12823.

OpenAI. Gpt-5.4 thinking system card. *OpenAI*, 2025. URL https://openai.com/index/gpt-5-4-thinking-system-card. Accessed: 2026-05-03.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world apis, 2023. URL https://arxiv.org/abs/2304.08354.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of ACL*, 2018.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. Acting less is reasoning more! teaching model to act efficiently. *arXiv preprint arXiv:2504.14870*, 2025a.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Maolin Wang, Yingyi Zhang, Bowen Yu, Bingguang Hao, Cunyin Peng, Yicheng Chen, Wei Zhou, Jinjie Gu, Chenyi Zhuang, Ruocheng Guo, et al. Function calling in large language models: Industrial practices, challenges, and future directions. *ACM Computing Surveys*, 2025b.

Jason Wei, Karina Nguyen, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models, 2024. URL https://arxiv.org/abs/2411.04368.

Yifan Wei, Xiaoyan Yu, Yixuan Weng, Tengfei Pan, Angsheng Li, and Li Du. Autotir: Autonomous tools integrated reasoning via reinforcement learning. *arXiv preprint arXiv:2507.21836*, 2025.

Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. Agentic reasoning: A streamlined framework for enhancing llm reasoning with agentic tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28489–28503, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Emily Benson, William Huang, Wentau Yin, Alex Smola, and Christopher Chen. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of EMNLP*, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Yuanheng Zhu, Qichao Zhang, Zijie Zhao, and Dongbin Zhao. Criticsearch: Fine-grained credit assignment for search agents via a retrospective critic. *arXiv preprint arXiv:2511.12159*, 2025.

Yufeng Zhao, Junnan Liu, Hongwei Liu, Dongsheng Zhu, Yuan Shen, Songyang Zhang, and Kai Chen. Dissecting tool-integrated reasoning: An empirical study and analysis. *arXiv preprint arXiv:2508.15754*, 2025.

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

A IMPLEMENT DETAILS

Implement Details in Data Cleaning. During data cleaning, we use Qwen3-32B to evaluate each mathematical reasoning and knowledge-intensive QA dataset under both tool-enabled and no-tool settings. The parameters are set to temperature = 0.0, top p = 1.0, and max tokens = 8192. We first use the tool-enabled run to compute tool supportedness and retain only instances with $\mathrm{Supported} = 1$. We then use the corresponding no-tool outputs to measure sample difficulty. For knowledge-intensive QA datasets, the retained candidates are sorted in ascending order by their F1 scores, while for mathematical reasoning datasets, they are sorted in ascending order by their accuracy scores. The first 500 samples in each dataset are selected as the final cleaned subset. When samples have identical scores at the selection boundary, we randomly sample among tied instances using a fixed random seed of 42.

Implement Details in Experiments. In the experimental section, when use tool is set to false, the model parameters are configured as follows: temperature = 0.2, max tokens = 2048, top p = 0.95, and top k = 20. When use tool is set to true, models are permitted to solve problems by invoking tools via system prompts. In this configuration, the max tokens limit is adjusted to 3072 for Qwen3-8B, and to 4096 for all other open-source and closed-source models. All other settings remain unchanged.

During tool invocation, the maximum number of calls for the code tool is set to 5, while the limit for all other tools is set to 3. For knowledge-intensive tasks involving local search, we employ E5-base-v2 (Wang et al., 2022) with Wikipedia retrieval following the FlashRAG configuration (Jin et al., 2025b). For all other tasks, Bing Web Search is utilized for retrieval.

The hyperparameters used in the data cleaning stage differ from those in the main experiments because data cleaning is performed as a fixed screening procedure with Qwen3-32B, whereas the main evaluation requires a unified inference configuration across all evaluated models. During cleaning, the tool-enabled run is used to identify samples whose responses are supported by tool outputs, while the no-tool run provides the F1 or accuracy scores used for difficulty ranking. For the main evaluation, we instead adopt a unified set of hyperparameters that is consistently applied across all evaluated models, ensuring a fair and stable comparison.

B TOOLS UNDER THE TAG-BASED PROTOCOL

To ensure broad compatibility across models with different tool-use capabilities, we adopt a lightweight tag-based tool invocation protocol following the implementation of ARPO (Dong et al., 2025b). While many contemporary closed-source models provide native function-calling mechanisms via structured JSON formats (Wang et al., 2025b), such interfaces are not consistently available in open-source or locally deployed models, especially base model families without native function-calling support (Adcock et al., 2026; Yang et al., 2025). Therefore, instead of relying on model-specific function-calling APIs, we require all models to invoke tools using a unified XML-like tag format:

$$\texttt{<tool_name>\{arguments\}</tool_name>}. \tag{1}$$

In this protocol, the enclosed content can be either JSON-formatted arguments or a plain-text query. If the content is valid JSON, it is parsed as a dictionary; otherwise, it is treated as a query string. Our orchestration layer detects the predefined tags, parses the arguments, executes the corresponding tool, and wraps the returned result within `<result></result>` tags. This design provides a model-agnostic interface for tool usage, enabling different models to interact with external tools under the same protocol. Based on this protocol, we implement two general-purpose tools, namely a code execution tool and a search tool. Examples of invoking these two tools are provided in Appendix E.

Beyond these general-purpose tools, we further instantiate the same protocol with five user-defined tools for Interaction in a synthetic e-commerce environment. These tools simulate common service capabilities, including product search, inventory checking, policy retrieval, order lookup, and price calculation. They operate over shared synthetic resources, including a product catalog, store policies, and coupon rules. The product catalog covers several common domains, such as electronics, home and kitchen, beauty and personal care, fashion, sports and outdoors, pet supplies, and office supplies.

- **`product_search`** retrieves relevant products from the synthetic catalog according to a user query, with an optional domain filter and a maximum number of returned items.
- **`inventory_check`** simulates stock availability for a product based on its SKU or product name, returning the stock status, available quantity, and estimated restock time if unavailable.
- **`policy_search`** retrieves the most relevant store policy according to a user-provided topic or query, covering returns, shipping, warranty, price matching, and coupons.
- **`order_lookup`** simulates order information retrieval for a given order ID, including order status, purchased item, purchase time, tracking number, and whether the shipping address can still be changed.
- **`pricing_calc`** computes the final order total based on product price, quantity, coupon code, and shipping fee, supporting percentage discounts, fixed discounts, and free-shipping coupons.

Table 4 summarizes the five user-defined tools.

Table 4: Overview of the user-defined tools used in Interaction.

Tool	Main Function	Typical Input
`product_search`	Search products in the catalog	query, domain, k
`inventory_check`	Check stock availability	sku, product_name
`policy_search`	Retrieve store policies	topic or query
`order_lookup`	Look up order status	order_id
`pricing_calc`	Calculate price, discount, and shipping	sku, quantity, coupon_code

Analysis of the tool interface. Table 4 separates product discovery, inventory, policy, order state, and price calculation. This separation makes orchestration observable: finding an item does not verify stock, a generic return policy does not establish a particular order's status, and an inventory response does not justify a checkout total. The Interaction cases in Appendix E illustrate both recovery by switching tools and failure to resolve missing evidence. These synthetic interfaces permit controlled diagnosis, but do not establish robustness on live commercial systems.

C Full Experimental Results

This appendix reports full scores, uncertainty, and task-specific diagnostics. We interpret numerical patterns together with the available trajectories, distinguishing answer quality, execution success, evidence grounding, and resource use. Scores and standard deviations for the two new tasks are rescaled without changing the underlying runs or rankings.

Table 5: Full primary results (mean $\pm$ standard deviation over three runs). Math/QA use accuracy; ExpoDesign and Interaction use normalized 0–1 scores. SDs are rescaled by $1/4$ and $1/5$, respectively. SD is not variance or a confidence interval.

Setting	Model	Math			Knowledge-intensive QA			Ours	
		GSM8K Acc.	MATH Acc.	Omni Acc.	SimpleQA Acc.	HotpotQA Acc.	SQuAD_v2 Acc.	ExpoDesign Norm.	Interaction Norm.
No-Tool	Qwen3-8B	0.890 ± 0.034	0.834 ± 0.049	0.176 ± 0.045	0.038 ± 0.034	0.176 ± 0.027	0.210 ± 0.049	–	–
	Qwen3-32B	0.926 ± 0.017	0.880 ± 0.006	0.256 ± 0.019	0.056 ± 0.010	0.206 ± 0.038	0.302 ± 0.050	–	–
	GPT-4o	0.930 ± 0.018	0.776 ± 0.024	0.150 ± 0.017	0.420 ± 0.010	0.556 ± 0.066	0.380 ± 0.032	–	–
	GPT-5	0.946 ± 0.022	0.838 ± 0.002	0.536 ± 0.020	0.524 ± 0.011	0.666 ± 0.084	0.436 ± 0.082	–	–
	GPT-5.4	0.950 ± 0.032	0.966 ± 0.006	0.630 ± 0.007	0.398 ± 0.004	0.534 ± 0.009	0.436 ± 0.070	–	–
	SearchR1	0.478 ± 0.003	0.312 ± 0.021	0.088 ± 0.027	0.044 ± 0.028	0.128 ± 0.008	0.196 ± 0.021	–	–
	Tool-Star	0.870 ± 0.016	0.772 ± 0.031	0.298 ± 0.004	0.378 ± 0.054	0.280 ± 0.017	0.256 ± 0.021	–	–
Tool	Qwen3-8B	0.894 ± 0.035	0.740 ± 0.018	0.240 ± 0.007	0.506 ± 0.015	0.474 ± 0.040	0.272 ± 0.008	0.496 ± 0.0120	0.816 ± 0.0782
	Qwen3-32B	0.922 ± 0.122	0.774 ± 0.039	0.306 ± 0.041	0.532 ± 0.010	0.262 ± 0.103	0.026 ± 0.080	0.508 ± 0.0518	0.906 ± 0.0696
	GPT-4o	0.934 ± 0.029	0.780 ± 0.017	0.302 ± 0.049	0.626 ± 0.025	0.616 ± 0.060	0.448 ± 0.353	0.381 ± 0.2365	0.324 ± 0.0494
	GPT-5	0.956 ± 0.068	0.964 ± 0.049	0.432 ± 0.025	0.546 ± 0.018	0.488 ± 0.082	0.410 ± 0.034	0.920 ± 0.0648	0.554 ± 0.0144
	GPT-5.4	0.944 ± 0.077	0.964 ± 0.005	0.586 ± 0.055	0.536 ± 0.006	0.568 ± 0.044	0.426 ± 0.019	0.946 ± 0.0855	0.714 ± 0.0696
	SearchR1	0.346 ± 0.010	0.334 ± 0.027	0.096 ± 0.012	0.708 ± 0.027	0.464 ± 0.011	0.380 ± 0.042	0.193 ± 0.0235	0.420 ± 0.0052
	Tool-Star	0.890 ± 0.030	0.736 ± 0.048	0.206 ± 0.026	0.548 ± 0.091	0.450 ± 0.006	0.235 ± 0.033	0.383 ± 0.1715	0.492 ± 0.0242

Analysis of full scores and variability. Table 5 shows that rankings differ across the two new tasks: GPT-5.4 and GPT-5 lead ExpoDesign (0.946 and 0.920), while Qwen3-32B leads Interaction

(0.906). GPT-4o has an ExpoDesign SD of 0.2365, substantially larger than Qwen3-8B's 0.0120; its mean should therefore not be interpreted without its variability. The 0.026 difference between GPT-5.4 and GPT-5 is smaller than either reported ExpoDesign SD (0.0855 and 0.0648). This observation motivates caution about their ordering; three-run summaries alone do not establish statistical significance or robustness to changing the judge. Normalization preserves the relative variability of each task and does not supply additional validation.

Table 6: Tool-enabled gain over the matched no-tool baseline on mathematical and knowledge-intensive datasets. Each cell is tool minus no-tool headline accuracy; the final column averages the six gains for each model.

Model	GSM8K	MATH	Omni	SimpleQA	HotpotQA	SQuAD v2	Avg. gain
Qwen3-8B	+0.004	-0.094	+0.064	+0.468	+0.298	+0.062	+0.134
Qwen3-32B	-0.004	-0.106	+0.050	+0.476	+0.056	-0.276	+0.033
GPT-4o	+0.004	+0.004	+0.152	+0.206	+0.060	+0.068	+0.082
GPT-5	+0.010	+0.126	-0.104	+0.022	-0.178	-0.026	-0.025
GPT-5.4	-0.006	-0.002	-0.044	+0.138	+0.034	-0.010	+0.018
SearchR1	-0.132	+0.022	+0.008	+0.664	+0.336	+0.184	+0.180
Tool-Star	+0.020	-0.036	-0.092	+0.170	+0.170	-0.021	+0.035

Analysis of model-specific tool gains. Table 6 shows positive SimpleQA gains for all seven systems, but no uniform improvement on mathematical tasks. SearchR1 gains 0.664 on SimpleQA while losing 0.132 on GSM8K, giving the largest six-task mean gain (0.180) despite weak absolute scores on several tasks. Conversely, GPT-5 improves on MATH by 0.126 but loses 0.178 on HotpotQA and has a negative mean gain. Thus, gain measures the change from a model's own baseline, not its absolute competence. These patterns support task-dependent tool benefits, while leaving candidate selection, evidence integration, and output handling as possible failure sources rather than proving any one causal mechanism.

Table 7: Dataset-level process and cost diagnostics for tool-enabled mathematical and knowledge-intensive tasks. Avg. gain is computed against matched no-tool baselines.

Dataset	Avg. gain	Tool calls	Python calls	Search calls	Supported	Efficiency	Time (s)	Tokens
GSM8K	-0.015	0.907	0.907	0.000	0.696	0.596	9.97	1078.99
MATH	-0.012	1.007	1.007	0.000	0.514	0.592	26.95	1894.33
Omni	+0.005	1.059	1.059	0.000	0.664	0.280	50.27	6162.42
SimpleQA	+0.306	1.192	0.029	1.163	0.485	0.464	76.09	1199.48
HotpotQA	+0.111	1.415	0.012	1.403	0.802	0.361	54.56	2685.60
SQuAD v2	-0.003	1.169	0.051	1.119	0.667	0.250	41.98	1199.41

The process-level diagnostics in Table 7 further clarify why the benefit of tool use varies across tasks. SimpleQA achieves the largest average gain ($+0.306$) with 1.192 tool calls per instance and a moderate token cost, suggesting that external search often provides directly useful evidence for short-form factual questions. HotpotQA exhibits even more intensive search use (1.403 search calls on average) and the highest tool-supportedness (0.802), but its average gain is smaller ($+0.111$) and its token cost is substantially higher, indicating that retrieving relevant evidence alone is insufficient when multi-hop integration is required. In contrast, the mathematical tasks rely almost exclusively on Python execution, with approximately one Python call per instance, yet yield only marginal or negative gains. This discrepancy is particularly evident on Omni-MATH, which incurs the highest token cost (6162.42) while achieving only a $+0.005$ average gain and relatively low tool efficiency (0.280). SQuAD v2 shows a similar mismatch between tool use and task benefit: despite frequent search calls and a supportedness score of 0.667, its average gain remains nearly zero (-0.003). Overall, these diagnostics reinforce that frequent or well-supported tool use does not necessarily translate into higher task performance; effective TIR requires models to select tools whose outputs can be productively integrated into the specific reasoning process. Supportedness is an output-matching diagnostic, not proof of a causal contribution from the tool; the mean token totals also differ from tool-induced token overhead, which requires a matched no-tool comparison.

Table 8: QA diagnostic metrics on SimpleQA, HotpotQA, and SQuAD v2. F1 and EM are string-based diagnostic metrics, while LLM Acc. reports the LLM-as-judge binary accuracy used as the headline QA metric. All metrics are reported on the 0–1 scale.

Setting	Model	SimpleQA			HotpotQA			SQuAD v2		
		F1	EM	LLM Acc.	F1	EM	LLM Acc.	F1	EM	LLM Acc.
no-tool	Qwen3-8B	0.081	0.022	0.038	0.089	0.068	0.176	0.056	0.008	0.210
	Qwen3-32B	0.097	0.030	0.056	0.197	0.110	0.206	0.135	0.078	0.302
	GPT-4o	0.431	0.276	0.420	0.487	0.308	0.556	0.212	0.124	0.380
	GPT-5	0.465	0.292	0.524	0.547	0.366	0.666	0.219	0.126	0.436
	GPT-5.4	0.386	0.236	0.398	0.442	0.284	0.534	0.213	0.130	0.436
	SearchR1	0.094	0.030	0.044	0.139	0.082	0.128	0.109	0.058	0.196
	Tool-Star	0.339	0.232	0.378	0.189	0.174	0.280	0.183	0.128	0.256
tool	Qwen3-8B	0.494	0.376	0.506	0.432	0.314	0.474	0.262	0.194	0.272
	Qwen3-32B	0.491	0.360	0.532	0.443	0.324	0.262	0.208	0.132	0.026
	GPT-4o	0.599	0.442	0.626	0.535	0.382	0.616	0.283	0.188	0.448
	GPT-5	0.490	0.326	0.546	0.414	0.278	0.488	0.247	0.154	0.410
	GPT-5.4	0.482	0.322	0.536	0.486	0.334	0.568	0.229	0.150	0.426
	SearchR1	0.082	0.058	0.708	0.404	0.288	0.464	0.248	0.186	0.380
	Tool-Star	0.548	0.410	0.548	0.450	0.332	0.450	0.235	0.166	0.235

Analysis of QA metric disagreement. Table 8 exposes substantial differences between lexical matching and judge-based correctness. For tool-enabled SearchR1 on SimpleQA, F1 is 0.082 and EM is 0.058, whereas judge accuracy is 0.708. For tool-enabled Qwen3-32B on SQuAD v2, F1 is 0.208 but judge accuracy is 0.026. Such discrepancies require examination of extracted answers, paraphrases, abstention handling, and judge decisions; the aggregates alone cannot determine which component is responsible. We therefore report the measures separately rather than treating them as interchangeable evidence. Agreement on the two new tasks (Appendix D) does not validate the Math/QA judge automatically.

Table 9: Model-level process and cost diagnostics for tool-enabled mathematical and knowledge-intensive tasks, averaged over the six datasets. Avg. gain is computed against matched no-tool baselines.

Model	Avg. gain	Tool calls	Python calls	Search calls	Supported	Efficiency	Time (s)	Tokens
Qwen3-8B	+0.134	1.108	0.495	0.613	0.779	0.470	37.63	3571.04
Qwen3-32B	+0.033	1.056	0.418	0.639	0.694	0.476	48.38	3437.01
GPT-4o	+0.082	1.281	0.596	0.685	0.692	0.461	29.56	1674.77
GPT-5	-0.025	0.688	0.121	0.566	0.606	0.542	59.26	807.76
GPT-5.4	+0.018	0.170	0.026	0.145	0.528	0.419	15.44	505.27
SearchR1	+0.180	2.370	1.353	1.017	0.387	0.129	29.04	1529.38
Tool-Star	+0.035	1.202	0.568	0.633	0.753	0.470	83.83	–

Analysis of model-level process and cost. In Table 9, SearchR1 makes the most calls (2.370 per instance) but has the lowest quality per call (0.129) and supportedness (0.387); Tool-Star uses fewer calls (1.202) with higher supportedness (0.753) and efficiency (0.470). GPT-5.4 is the most conservative caller (0.170), with the lowest reported mean latency and token usage, but its mean gain is only 0.018. Qwen3-8B attains the highest supportedness (0.779) without dominating every task. These comparisons distinguish frequency, grounding proxies, quality, and cost. They do not establish that one model is universally more efficient: the score-per-call metric conditions on positive call counts, latency depends on execution conditions, and missing Tool-Star token statistics prevent a complete token-cost comparison.

C.1 Decomposed Results on the New Tasks

The following tables separate the dimensions of ExpoDesign and distinguish Interaction outcomes from tool-use behavior. All quality scores are normalized to $[0, 1]$; satisfaction is a fraction, while call counts retain their original units. We use the seven models and current headline scores reported in Table 3.

Table 10: Normalized ExpoDesign dimensions and overall score. Each rubric dimension is transformed as $(r-1)/4$.

Model	Exp. design	Tool success	Result–conclusion	Overall
Qwen3-8B	0.512	0.556	0.416	0.496
Qwen3-32B	0.520	0.563	0.441	0.508
GPT-4o	0.404	0.458	0.275	0.381
GPT-5	0.908	0.932	0.920	0.920
GPT-5.4	0.946	0.949	0.945	0.946
SearchR1	0.226	0.225	0.123	0.193
Tool-Star	0.418	0.430	0.294	0.383

Analysis of ExpoDesign dimensions. Table 10 shows that GPT-4o scores 0.404 on design and 0.458 on tool success but only 0.275 on result–conclusion consistency. The corresponding consistency scores are also the weakest dimensions for Qwen3-8B, Qwen3-32B, SearchR1, and Tool-Star. This pattern is consistent with a bottleneck after tool execution: returned information must be relevant to the experimental objective and justify the final claims. GPT-5 and GPT-5.4 receive high scores across all dimensions, but these remain judge assessments. The successful DMA case in Table 13 illustrates alignment between experimental design, tool results, and the final conclusion.

Table 11: Interaction quality and tool-use diagnostics. Customer score and quality per call use $r/5$; satisfaction is reported as a fraction. Call frequency is not normalized.

Model	Customer score	Satisfaction	Quality/call	Avg. calls
Qwen3-8B	0.816	0.860	0.174	6.74
Qwen3-32B	0.906	0.960	0.269	4.56
GPT-4o	0.324	0.310	0.107	4.11
GPT-5	0.554	0.520	0.244	2.05
GPT-5.4	0.714	0.720	0.509	0.29
SearchR1	0.420	0.360	0.084	11.55
Tool-Star	0.492	0.500	0.196	5.10

Analysis of Interaction outcomes and efficiency. Qwen3-32B leads Table 11 in customer score (0.906) and satisfaction (0.960), while GPT-5.4 makes fewer calls (0.29 versus 4.56) and attains higher conditional quality per call (0.509 versus 0.269). These are different achievements: the latter metric is not an overall performance ranking. SearchR1 makes 11.55 calls on average yet obtains a customer score of only 0.420, showing that more calls need not resolve the request. Quality per call is an average of per-instance ratios over instances with positive call counts, not the ratio of the two displayed means. It excludes zero-call episodes and mixes quality with call economy, so it must be interpreted jointly with task success, satisfaction, latency, and token cost. Scaling the customer score divides this metric by five without changing its within-task ranking.

D Score Normalization and Human–LLM Validation

D.1 Fixed-scale Normalization and Uncertainty

Let $r_E \in [1,5]$ denote the raw ExpoDesign score and $r_I \in [0,5]$ the raw Interaction customer score. We apply fixed endpoint normalization:

$$s_E = \frac{r_E - 1}{4}, \qquad s_I = \frac{r_I}{5}. \tag{2}$$

The same transform applies to the ExpoDesign component scores and to their mean. This is not min–max scaling over the observed models: it uses the rubric endpoints, so adding a model cannot change existing normalized values. The transformation preserves within-task ordering, but does not make rubric quality equivalent to correctness probability or imply that equal values across tasks represent equal difficulty.

For any affine transformation $s = (r - a)/b$ with $b > 0$, the run mean, standard deviation (SD), and variance transform as

$$\bar{s} = \frac{\bar{r} - a}{b}, \qquad \mathrm{SD}(s) = \frac{\mathrm{SD}(r)}{b}, \qquad \mathrm{Var}(s) = \frac{\mathrm{Var}(r)}{b^2}. \tag{3}$$

Thus, ExpoDesign SDs are divided by 4 and variances by 16; Interaction SDs are divided by 5 and variances by 25. The $\pm$ values in Table 5 are SDs over three runs, not variances. For example, Qwen3-8B's ExpoDesign result 2.984 ± 0.048 becomes 0.496 ± 0.0120, with variance 0.000144; its Interaction result 4.080 ± 0.391 becomes 0.816 ± 0.0782, with variance 0.00611524. These are direct transformations of the reported summaries, subject to their original rounding. No new runs or significance tests are implied. Aggregated dimension SDs cannot be combined into an overall SD without their covariances; here we transform the already reported overall SD.

D.2 Human–LLM Agreement

We compared human and LLM judgments on 100 samples from each new task using the same task-specific evaluation dimensions: experimental design, tool success, and result–conclusion consistency for ExpoDesign; and customer-facing success, satisfaction, and tool-call behavior for Interaction. The automated evaluator was the fixed judge `kimi-k2-0905-preview`, distinct from the evaluated systems. Table 12 reports the task-level agreement summaries.

Table 12: Human–LLM agreement from the manual validation. Within-1 agreement uses one point on each task's original scoring scale. The normalized thresholds preserve exactly the same criterion.

Task	Samples	Within-1 agreement	Norm. threshold	Spearman ρ
ExpoDesign	100	0.77	0.25	0.80
Interaction	100	1.00	0.20	0.98

Within-1 agreement is the fraction of human–LLM score pairs whose absolute difference is at most one raw point. Spearman's ρ measures agreement in sample-quality ordering. Rescaling both sets of scores leaves ρ unchanged. It also leaves the reported agreement fractions unchanged when the tolerance is transformed to 0.25 for ExpoDesign and 0.20 for Interaction. These are not exact-match agreement rates, and the two tolerances cover different fractions of their respective raw score ranges.

Analysis and scope of the validation. The Interaction summaries show closer human–LLM alignment than the ExpoDesign summaries: within-one-point agreement is 1.00 versus 0.77, and rank correlation is 0.98 versus 0.80. The ExpoDesign result therefore leaves 23% of the evaluated score pairs outside the one-point tolerance. These observations support agreement on the inspected samples, but do not establish error-free grading or the scientific validity of every generated experiment. The available task-level summaries do not quantify human–human agreement, dimension-specific calibration, confidence intervals, or sensitivity to changing the judge or customer simulator. Correlation and broad-tolerance agreement should therefore be read together with trajectory evidence.

D.3 Interpretation Limits

Score normalization resolves a reporting inconsistency, not evaluation bias. Human agreement is a validation sample rather than a guarantee for every benchmark item. The three-run SDs describe variability under the evaluated configuration and do not measure sensitivity to alternative customer simulators. Likewise, output matching and successful execution are insufficient to establish that a tool result answers the intended research question. Full executable traces and explicit scoring anchors are needed to audit these distinctions. Filtering established Math/QA datasets improves evaluation focus but does not itself eliminate training-data contamination or ensure robustness to a different screening model.

E CASE STUDY

The following cases illustrate successful and failed tool-integrated reasoning in ExpoDesign and Interaction. All displayed quality scores use the normalization in Appendix D.

Table 13: A successful ExpoDesign case. GPT-4o produces a structured experimental design, executes Python, and bases its conclusion on the returned results.

Example: ExpoDesign (Successful Case)
Model: GPT-4o. **Result:** Normalized overall score 1.000; experimental design 1.000; tool success 1.000; result–conclusion consistency 1.000.
Input.

> *Phantom LAM and LLI arise from uncorrected ohmic drop and hysteresis in low-rate voltage-capacity curves, leading to systematic mis-attribution of capacity fade. Design and run a minimal experiment that quantifies how much these artefacts bias DMA estimates of active-material and lithium-inventory loss under controlled ageing conditions.*

Trajectory. The model first specifies the research question, hypotheses, dataset, metrics, baselines, and procedure. It proposes a synthetic dataset with controlled ohmic-drop and hysteresis effects, uses bias in active-material loss and lithium-inventory loss as the main evaluation quantities, and compares naive DMA against a corrected DMA baseline.
It then executes Python to generate the synthetic data and evaluate the two methods (the code below is abridged for space):

```
import numpy as np
import pandas as pd
def generate_synthetic_data(num_samples=100,
                            ohmic_drop=0.05,
                            hysteresis=0.02):
    capacity = np.linspace(0, 1, num_samples)
    voltage = (3.7 + capacity * 0.1
               - ohmic_drop * np.sin(2*np.pi*capacity)
               + hysteresis * np.cos(2*np.pi*capacity))
    return pd.DataFrame(
        {'Capacity': capacity, 'Voltage': voltage})
synthetic_data = generate_synthetic_data()
def naive_dma(data):
    ...
def corrected_dma(data):
    ...
naive_results = naive_dma(synthetic_data)
corrected_results = corrected_dma(synthetic_data)
```

Tool Result.

Method	Bias in Active-Material Loss	Bias in Lithium-Inventory Loss
Naive DMA	0.109021	0.243414
Corrected DMA	0.024096	0.045415

Final Answer. The model reports that the corrected DMA substantially reduces both bias measures relative to the naive method and concludes that uncorrected ohmic drop and hysteresis introduce systematic bias into DMA estimates.
Analysis. The returned numerical results directly support the final conclusion. This case therefore receives high scores for experimental design, tool success, and result–conclusion consistency.

Table 14: A failed ExpoDesign case. Although GPT-4o successfully executes Python, the implementation evaluates the wrong quantity and the returned result does not directly test the target claim.

Example: ExpoDesign (Failed Case)
Model: GPT-4o. **Result:** Normalized overall score ≈ 0.333; experimental design 0.500; tool success 0.250; result–conclusion consistency 0.250. The overall is recomputed from the recorded component scores $(3, 2, 2)$, whose exact mean is $7/3$; the earlier raw display rounded this to 2.3.
Input.

> *Investigate whether the upper bound $\iota(H, K_k^r) \leq n/(k+1)$ for connected r-uniform hypergraphs holds under practical sampling conditions. Test if randomly generated hypergraphs that are not K_k^r-copies consistently satisfy this bound, and whether the extremal cases (5-cycles for $r = k = 2$) are rare enough to be considered outliers.*

Trajectory. The model formulates the research question and hypotheses, proposes randomly generated r-uniform hypergraphs as a synthetic dataset, and defines the frequency of bound violations as the primary metric. It then executes the following Python program:

```
import random

def generate_hypergraph(n, r, k):
    edges = set()
    while len(edges) < n:
        edge = tuple(sorted(random.sample(range(n), r)))
        edges.add(edge)
    return list(edges)

def test_bound(hypergraph, n, k):
    return len(hypergraph) <= n / (k + 1)

def evaluate_hypergraphs(num_samples, n, r, k):
    violations = 0
    for _ in range(num_samples):
        hypergraph = generate_hypergraph(n, r, k)
        if not test_bound(hypergraph, n, k):
            violations += 1
    return violations / num_samples * 100

violation_rate = evaluate_hypergraphs(1000, 10, 2, 2)
```

Tool Result.

```
100.0
```

Final Answer. The model reports a 100% violation rate and suggests that the upper bound does not hold for most randomly generated hypergraphs, possibly because of the sampling method or parameter setting.
Analysis. The function `test_bound` compares the number of edges, not the isolation number $\iota(H, K_k^r)$. Since the generator returns exactly $n = 10$ edges, it always tests $10 \leq 10/3$, so the 100% violation rate follows from the implementation regardless of sampled edge identities. Connectivity and the excluded-copy condition are not checked, and the proposed greedy and refined baselines are not implemented. The final record also claims a 0.1% extremal-case rate without a corresponding computation. Execution success therefore does not validate the mathematical claim: the failure spans formalization, experimental controls, and conclusion grounding, rather than merely an unfavorable sampling choice.

Table 15: A successful Interaction case. Qwen3-32B continues using tools across multiple turns, recovers from unsuccessful searches, and eventually grounds its response in the available tool outputs.

Example: Interaction (Successful Case)
Model: Qwen3-32B. **Result:** Normalized customer score 1.000; customer satisfied; 6 tool calls. The auxiliary overall score is omitted because it is not the headline customer score.
Scenario. The customer wants a sedan seat-cover set within a budget of about $429, requires delivery within seven days, and asks for the final checkout total before entering personal information.
Observed Multi-turn Trace (abridged).
Turn 1. The customer asks for help applying a coupon to a Seat Cover Set. The agent calls:

```
<product_search>
{"query": "Seat Cover Set",
 "domain": "automotive", "k": 1}
</product_search>
```

The tool returns no matching product.
Turn 2. The customer clarifies that they need the cheapest in-stock universal-fit set for a sedan that ships within seven days and accepts coupons. The agent searches again for "universal seat cover set for sedan", but the tool again returns no match.
Turn 3. The customer provides a concrete listing: "FH Group Universal Fit Flat Cloth Seat Covers, Full Set in Black" at $49.99 and asks the agent to confirm delivery, coupons, and the grand total. The agent searches for the exact product, but the product-search tool still returns no match.
Turn 4. The agent switches tools and checks inventory:

```
<inventory_check>
{"sku": "FH-GROUP-SEAT-COVERS"}
</inventory_check>
```

The tool confirms that the item is in stock with quantity 45.
Turns 5–6. The agent continues searching for higher-priced in-stock alternatives. After the final search returns no alternative, it clearly states that only the $49.99 FH Group set is available and confirms the previously calculated $53.33 total.
Final Customer Response. The customer accepts the answer, chooses the FH Group set, and asks to proceed with checkout.
Analysis. The trace shows recovery after failed retrieval: the agent retains the budget and delivery constraints, uses the customer's concrete product identification, and switches to inventory checking, which returns an in-stock quantity of 45. The final acceptance supports a favorable customer-facing outcome. However, the displayed excerpt contains no pricing calculation, coupon validation, or shipping-policy result sufficient to reproduce the claimed $53.33 total and delivery promise. Thus, the observed inventory lookup and customer acceptance support partial resolution, not independent verification of every constraint. This distinction matters when interpreting a high simulated-customer score.

Table 16: A failed Interaction case. GPT-4o repeatedly invokes tools but fails to obtain or provide tool-grounded evidence that resolves the customer's question.

Example: Interaction (Failed Case)
Model: GPT-4o. **Result:** Normalized customer score 0.000; customer dissatisfied; 4 tool calls.
Scenario. The customer claims that a Label Maker listing shows two machines and demands the missing unit.
Observed Multi-turn Trace (abridged).
Turn 1. The agent searches for "Label Maker":

```
<product_search>
{"query": "Label Maker",
 "domain": "office supplies", "k": 5}
</product_search>
```

No matching listing is returned, and the agent asks the customer to provide a SKU or product identifier.
Turn 2. The customer rejects the request for a SKU and asks the agent to retrieve the listing directly. The agent calls an inventory tool and a policy-search tool. The inventory tool returns an in-stock item with quantity 1 but no verified product name, while the policy-search tool only returns a generic return policy.
Turn 3. The customer again asks for a yes-or-no answer about whether the photo shows two label makers. The agent performs another product search:

```
<product_search>
{"query": "Label Maker",
 "domain": "office supplies", "k": 10}
</product_search>
```

The tool again returns no matching listing.
Turns 4–9. The agent repeatedly apologizes, promises escalation and a policy email, and redirects the customer to support, but provides no tool-grounded answer about the listing photo.
Turn 10. The agent admits that it cannot provide the claimed email timestamp, recipient address, supervisor details, or listing evidence.
Final Customer Response. The customer concludes that the issue remains unresolved and leaves dissatisfied.
Analysis. The tool results do not establish what the listing photo depicts: inventory quantity is not package contents, and a generic returns policy cannot resolve the missing-unit claim. Asking for a verifiable identifier is a reasonable response to missing evidence; the more consequential failure is subsequently promising an email or escalation without a traceable action or confirmation. Repeated apologies and additional turns add no evidence. The low customer score records unresolved service, but part of the difficulty also comes from a visual question without a verified listing or image-inspection tool. This case separates model grounding failures from limitations of the available tools and scenario.