CREDIT-BUDGETED ICPC-STYLE CODING: WHEN LLM AGENTS MUST PAY FOR EVERY DECISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Contemporary coding-agent benchmarks applaud "first correct answer," silently assuming infinite tokens, container minutes, and developer patience. In production, every LLM call, test re-run, and rollback incurs hard cost; agents that cannot budget these resources are dead on arrival. We close the gap with USACOArena, an ICPC-inspired arena where agents pay deterministic credits for every prompt, compilation, test, or rollback. A task becomes a cost–benefit negotiation under uncertainty: is a second sample worth 15% of the remaining budget, or should the agent pivot to a cheaper heuristic? Real-time deduction exposes decision profiles hidden from static leaderboards: the tax of over-specialized generators, the ROI of early-exit heuristics, and the compound interest of lightweight scaffolding. Even identically seeded agents diverge in self-play, revealing a rich policy space where the same model oscillates between spendthrift submission sprees and parsimonious exploration. Released as a reproducible benchmark and zero-shot curriculum, USACOArena provides the traces, credit engine, and six state-of-the-art decision logs to catalyze research on coding agents that know when to stop.

1 INTRODUCTION

As Large Language Models (LLMs) master the syntax of correct code, the next frontier is coding agents that decide what to code, how long to persist, and when to walk away—skills that are procedural, not declarative. These meta-decisions—sizing a task, rationing a budget, abandoning a dead-end—cannot be scraped from static repositories; they must be forged in tight feedback loops. Imagine a 500-token cap: does the agent burn 400 on a greedy heuristic that later fails hidden tests, or pause and pivot to dynamic programming? No dataset records the optimal choice; it is revealed only through interaction.

Consequently, the missing ingredient is not more data but a crucible: a cheap, objective, repeatable environment where every decision is punished or rewarded within milliseconds. Competitive programming is that crucible. Each task is a self-contained, perfect-information game: rules, budget, and grader are all disclosed up-front, isolating raw decision-making from the noise of real-world codebases. Pass/fail is binary and costs milliseconds, letting us iterate on policy instead of waiting for cloud builds. The tech industry already uses contests as a proxy for general problem-solving ability; we simply extend the same logic to agents.

Complex software engineering benchmarks (Jimenez et al., 2023; Yang et al., 2024; 2025a; Jain et al., 2025b; Zhou et al., 2025; Badertdinov et al., 2025) remain invaluable, yet they target a different skill stack: long-horizon planning amid missing documentation, legacy APIs, and asymmetric information. Competitive puzzles strip away those confounders, much as chess—simpler than war, yet rich enough to birth modern game-playing AI—so we can measure how an agent reasons under pressure, not merely what it knows.

In this paper, we introduce **USACOArena**, an interactive arena built on the foundations of the ACM International Collegiate Programming Contest (ICPC). We chose the ACM-ICPC format because its all-or-nothing scoring system—where solutions receive points only if they pass all test cases—directly promotes the development of robust, "zero-bug" solutions. This demand for absolute correctness is not just a feature of the game. It is a direct proxy for the reliability required of agents performing high-stakes, real-world tasks.

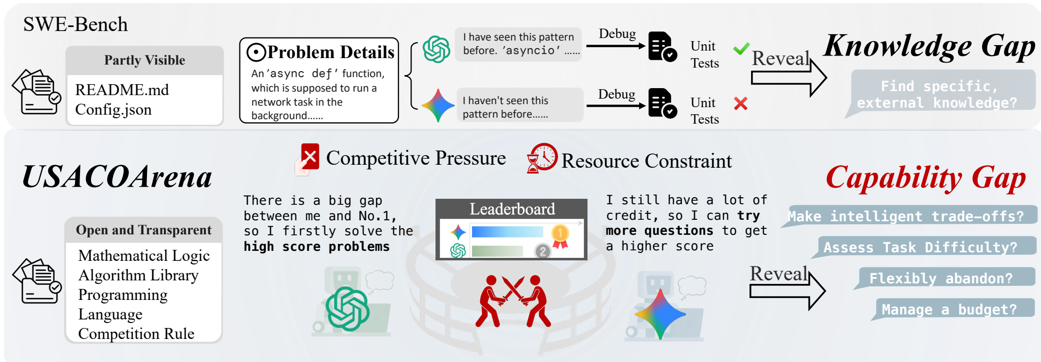


Figure 1: A comparative illustration of two evaluation paradigms for coding agents. **Top:** In an information-asymmetric environment like Software Engineering, where crucial information may be only partly visible, an agent's success can depend on possessing specific, external knowledge. The evaluation primarily reveals a Knowledge Gap. **Bottom:** In our information-symmetric environment, USACOArena, all rules and algorithmic knowledge are transparent. Agents must make decisions under Competitive Pressure and a Resource Constraint. This design isolates the decision-making process itself, revealing a Capability Gap in core skills such as managing a budget, assessing task difficulty, and making intelligent trade-offs.

However, an environment designed for humans cannot be directly used for agents. The key human constraint of **time**, for example, is an unfair metric for LLMs due to network latency. To build a fair and reproducible testbed, we make a key adaptation: we translate time into an agent-native resource called **credit**. Every significant action, from LLM inference to local testing, consumes credits from a finite budget. This design transforms problem-solving into a process of cost-benefit analysis. It compels the agent to reveal its ability to manage resources and make intelligent trade-offs.

We conduct comprehensive experiments in USACOArena that go beyond simple rankings to reveal the decision-making profiles of leading LLM agents. Our analysis uncovers distinct problem-solving approaches, such as Gemini-2.5-pro's aggressive strategy and GPT-5-Codex's conservative one. We further isolate the impact of design choices through controlled "civil war" experiments within the GPT-5 family, finding that code-specialization can enhance reliability at the cost of initiative, while a well-designed agentic framework can boost overall performance. These results demonstrate that USACOArena measures the effectiveness of an agent's approach, not just its raw capability.

Finally, to probe the depth of our environment's decision space, we conduct a series of self-play experiments. By pitting two identical instances of the top-performing agent Gemini-2.5-pro against each other, we remove model capability as a variable and focus purely on the emergent decision-making process. The results show a striking diversity of outcomes; the competitions rarely end in a tie, with each agent discovering different paths through the problem space. This demonstrates that USACOArena is not a simple, deterministic puzzle but a complex environment that elicits varied and path-dependent behaviors. This further highlights the potential of our arena to serve not only as a testbed, but also as a dynamic training ground for cultivating more strategically capable agents.

In summary, our contributions are as follows:

- We introduce a methodology for evaluating coding agents based on their decision-making skills under resource constraints, shifting the focus beyond simple code correctness.
- We present USACOArena, an arena inspired by the ACM-ICPC, featuring a credit-based system designed to rigorously measure agent performance.
- We provide a comprehensive analysis of leading agents, revealing deep differences in their problem-solving approaches that are invisible in static tests.

2 RELATED WORK

Our research is positioned at the intersection of two key areas: the advancement of code generation models and the emergence of LLM-based agents. We introduce a new evaluation paradigm that bridges a critical gap between them by providing a dynamic arena to measure the code correctness and strategic resource management capabilities of agents.

2.1 LLMS FOR CODE AND THE RISE OF STATIC BENCHMARKS

Recent breakthroughs in Large Language Models (LLMs) have significantly advanced automated code generation. Models from OpenAI (OpenAI, 2025b), Anthropic (Anthropic, 2025a;b), and others (Guo et al., 2024; Wei et al., 2024; DeepSeek-AI et al., 2024; Cummins et al., 2024; Liu et al., 2024) have demonstrated powerful coding abilities across various benchmarks. To assess these capabilities, a suite of evaluation benchmarks has been developed. Early benchmarks like HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) establish functional correctness as the primary metric. The scope of evaluation later expands to more complex domains, including algorithmic competition problems (Li et al., 2022; Shi et al., 2024) and real-world software engineering tasks (Jimenez et al., 2023; Li et al., 2025; Liu et al., 2025). Recent efforts have further improved evaluation rigor with live problem sets (Jain et al., 2024; Zheng et al., 2025), robust ranking systems (Quan et al., 2025; Yang et al., 2025b), multiple function calls (Zhuo et al., 2025), language-driven coding (Deng et al., 2025) and interactive debugging (Yuan et al., 2025).

However, these benchmarks share a fundamental limitation: they are static and non-interactive. By focusing exclusively on the correctness of the final code output (the "what"), they fail to measure the strategic decision-making process (the "why"). Factors such as the efficiency of the development process, resource management, and balancing trade-offs under constraints remain unevaluated.

2.2 THE EMERGENCE OF CODING AGENTS

To move beyond simple code generation, sophisticated agentic frameworks have emerged, featuring diverse architectures, including coder-tester co-evolution (Wang et al., 2025b), generator-verifier pairs (Jain et al., 2025a), policy-critic models (Xie et al., 2025), and retriever-generator pipelines (Wang et al., 2025a), as well as internal reasoning mechanisms such as feedback loops and explanation-driven repair (Jiang et al., 2025; Gehring et al., 2025). Additionally, Multi-Agent Systems like AutoGPT (Significant Gravitas), MetaGPT (Hong et al., 2024), AgentVerse (Chen et al., 2024b), OpenHands (Wang et al., 2024), and others (Ma et al., 2024; Gao et al., 2024; Chen et al., 2024a; Yang et al., 2024; Xia et al., 2024; Aggarwal et al., 2025) have been built on multi-agent platforms employ complex workflows to solve programming tasks.

Despite their architectural complexity, the effectiveness of these agents is still primarily measured using the same static benchmarks. This creates a disconnect: an agent could take a costly and inefficient path, making thousands of attempts to find a solution, but this entire process is invisible in an evaluation that only scores the final, correct output. The strategic competence of these agents remains largely unobserved and unquantified.

2.3 THE MISSING PARADIGM: RESOURCE-AWARENESS EVALUATION

In the domain of Reinforcement Learning, learning from experience in an interactive environment has been the key to achieving superhuman performance. Works such as SWE-rebench (Badertdinov et al., 2025) and R2E-Gym (Jain et al., 2025b) focus on generating large-scale, executable environments from real-world or synthetic data, providing crucial training grounds for agents. Evaluation frameworks like ColBench (Zhou et al., 2025) and TheAgentCompany (Xu et al., 2025) are being developed to improve and measure agent performance on complex, multi-step tasks.

However, these systems are limited by a shared premise: their "environment" primarily serves as an execution sandbox, focusing the evaluation solely on the correctness of the final code artifact. In contrast, USACOArena is designed to evaluate not only an agent's ability to produce correct code but also its capacity for strategic resource management. This is accomplished through a comprehensive scoring and credit model that extends beyond simple correctness checks, forcing agents to make trade-offs that are central to superhuman intelligent behavior.

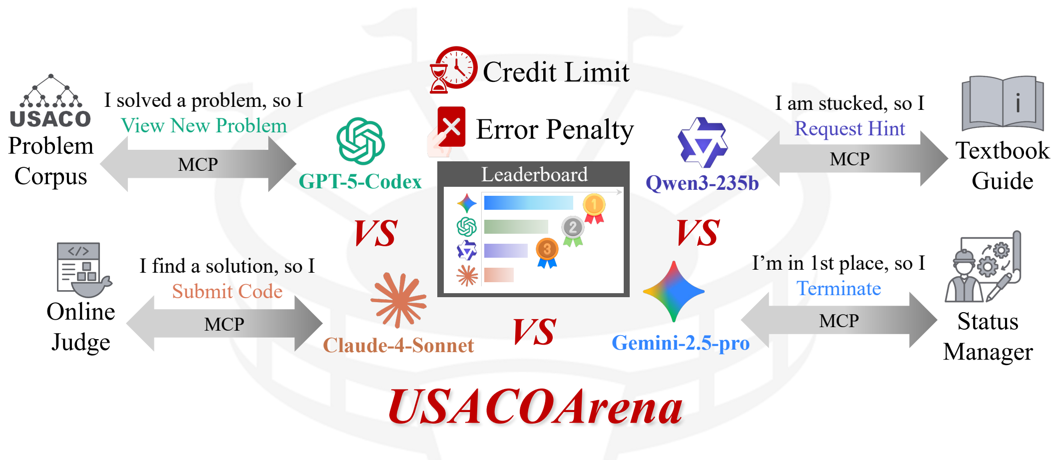


Figure 2: **An overview of the USACOArena environment**, designed to evaluate the strategic decision-making of coding agents. Inspired by ACM-style programming contests, the environment situates multiple competing agents within a formal system governed by a Credit Limit and Error Penalty. Agents interact with the competition system through a standardized communication protocol (MCP). The diagram illustrates the dynamic nature of the competition, where agents must make diverse strategic choices—such as submitting a solution, requesting a hint, or terminating to preserve a high rank—based on their internal state and the real-time Leaderboard.

3 USACOArena: An ACM-Inspired Arena for Coding Agents

This section details the design and construction of USACOArena, an interactive environment engineered to evaluate the strategic decision-making of coding agents. Grounded in the principles of human competitive programming, our methodology moves beyond static correctness checks to create a rigorous and reproducible evaluation framework. We first articulate the foundational design principles of our arena, justifying our adoption and adaptation of the ACM-ICPC competition format and our choice of the USACO problem corpus (Section 3.1). We then describe the specific scoring and credit model that operationalizes resource constraints and strategic trade-offs (Section 3.2). Finally, we detail the system architecture and communication protocol designed to ensure fair and standardized agent interaction (Section 3.3).

3.1 Foundational Design: Adapting the ACM-ICPC Format

To create a meaningful testbed for coding agents, we deliberately model USACOArena on the ACM International Collegiate Programming Contest (ICPC), whose format aligns closely with the demands of real-world software development. The ACM-ICPC's all-or-nothing scoring system and broad problem set reward the rapid development of robust, "zero-bug" solutions—a critical capability for any agent intended for practical application. This philosophy stands in contrast to other paradigms like the IOI, which favor theoretical optimization over immediate correctness. A detailed justification for this design choice is provided in Appendix B.

Operationalizing the ACM-ICPC format for contemporary agents, however, requires two key adaptations. First, given that the extreme difficulty of official ACM-ICPC problems would yield a sparse evaluation signal for current models, we source our problem corpus from the USA Computing Olympiad (USACO). Its tiered difficulty structure (Bronze to Platinum) provides a challenging yet tractable gradient that allows for effective differentiation of agent capabilities. Each competition uses 12 problems to mirror the scale of the ACM-ICPC World Finals, and we draw from the latest season to create a "living benchmark" that mitigates data contamination.

Second, we translate the core human constraint of **time** into a unified, agent-centric resource: **credit**. Physical time is an unreliable metric for API-based LLM agents due to network latency. Instead, each agent receives a fixed credit budget, and all significant actions consume credits. This mechanic transforms the evaluation into a resource management task, compelling agents to make strategic decisions that directly parallel a human's cognitive trade-offs under pressure.

To ensure the competition focuses on strategy rather than basic competence, we establish a qualification standard: an agent must be able to solve at least one Bronze-level problem (details in Section 4.1). Formally, an agent's interaction within the USACOArena environment is a policy π that produces a final result tuple: $(\texttt{Score}, \texttt{Consumed Credit})$. The agent's objective is to maximize its score, subject to the hard constraint that its action costs do not exceed the credit budget:

$$\max_{\pi} \text{Score}(\pi) \quad \text{subject to } C_{\text{action}}(\pi) \leq C_{\text{limit}} \tag{1}$$

3.2 Scoring and Credit Model: Operationalizing ACM-ICPC Rules

To operationalize the principles outlined in Section 3.1, USACOArena employs an explicit model for scores and credits. These mechanics are designed to directly mirror the incentives and pressures of the ACM-ICPC, thereby creating a rigorous evaluation of an agent's strategic competence.

Ranking and Scoring. The ranking system, as illustrated in the middle of Figure 2, is a direct adaptation of the ACM-ICPC rules. Agents are ranked primarily by their total score, with the total consumed credit serving as the tie-breaker, analogous to the number of problems solved and total time in an ACM-ICPC contest. The score itself is a weighted sum of all fully accepted (AC) problems, with higher points awarded for more difficult tiers (e.g., Silver over Bronze). Crucially, and in keeping with the ACM-ICPC's emphasis on correctness, no partial credit is awarded for solutions that fail any test cases.

The Credit Model. The credit system, which models a human's allocation of time, is composed of two distinct components: costs incurred for taking actions and penalties for incorrect submissions.

Action Costs are the resources an agent spends to make progress. Every strategic action has a cost, creating meaningful trade-offs:

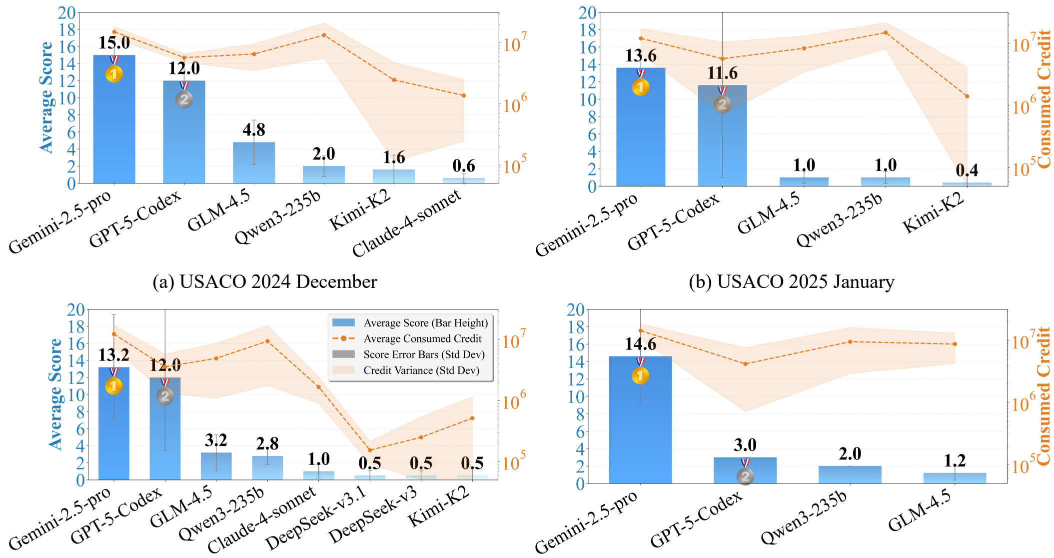
- **LLM Inference Cost:** This is the analogue to a human's "thinking time". The cost is normalized by the model's API price to account for the varying quality of generated tokens (see Appendix F). More powerful, expensive models thus consume more credit, forcing a trade-off between reasoning quality and resource efficiency.
- **Hint and Test Cost:** These costs are proxies for the time a human would spend consulting resources or performing local debugging. They incentivize agents to use these actions judiciously (details in Appendix I).

Penalty Costs are incurred for each incorrect submission (e.g., Wrong Answer, Time Limit Exceeded). This mechanism directly mirrors the time penalties in the ACM-ICPC that punish inefficient trial-and-error strategies.

This distinction between action costs and penalties is critical to the competition's dynamics. An agent's session is terminated only if its *action costs* exceed the budget ($C_{\text{action}} = C_{\text{LLM}} + C_{\text{hint}} + C_{\text{test}} \leq C_{\text{limit}}$). However, the final ranking tie-breaker is based on the *total consumed credit*, which includes penalties ($C_{\text{consumed}} = C_{\text{action}} + C_{\text{penalty}}$). This design motivates agents to not only solve problems but to do so with high efficiency and accuracy.

3.3 System Architecture and Communication Protocol

To ensure robust and standardized agent interaction, our system architecture is built around a turn-based communication loop inspired by the Model Context Protocol (MCP) (Anthropic, 2024). At each turn, the USACOArena server transmits the complete competition state—including consumed credit, current score, the public leaderboard, etc. —to the agent in a structured JSON object. The agent then responds with a formatted action, such as `SUBMIT_SOLUTION`. This protocol-driven design is critical for the integrity of our evaluation. It guarantees that every agent operates with identical information and within the same action space, thereby isolating strategic capability as the primary variable under assessment. All code submissions are executed in a sandboxed online judge emulator for security. This standardized, secure architecture makes USACOArena a reproducible and extensible platform, allowing other researchers to easily integrate and evaluate their own agents.



(a) USACO 2024 December

(b) USACO 2025 January

(c) USACO 2025 February

(d) USACO 2025 US Open

Figure 3: **Average agent scores and consumed credit across the four contests of the 2024–2025 USACO season.** Each subplot shows the results for a single contest, with agents sorted by rank. Blue bars represent the average score (left axis), while the orange line indicates the average consumed credit (right axis, log scale). Error bars and the shaded area denote the standard deviation over five independent runs; for clarity, only agents that achieved a non-zero average score are shown. The results reveal a stable and significant performance hierarchy: across all four contests of varying difficulty, Gemini-2.5-pro and GPT-5-Codex consistently rank first and second, respectively.

4 Experiment

Our experimental evaluation demonstrates the utility of USACOArena in characterizing the capabilities of modern coding agents. We first detail the experimental setup, covering the problem corpus, agent construction, and the qualification process (Section 4.1). We then present the main competition results, providing a robust ranking of leading LLM agents and a qualitative analysis of their interactive strategies (Section 4.2). Following this, we probe the behavioral patterns of different agent architectures (Section 4.3) and conclude by exploring emergent behaviors in a self-play context, highlighting USACOArena's potential as a training environment for future reinforcement learning applications (Section 4.4).

4.1 Experimental Setup

All experiments are conducted following the ACM-inspired competition rules detailed in Section 3. Our setup is designed to be rigorous, transparent, and grounded in realistic competitive scenarios.

Problem Corpus and Difficulty Baselining. The evaluation is conducted across the 48 problems from the four contests of the complete 2024–2025 USACO season: the 2024 December, 2025 January, 2025 February, and 2025 US Open contests. To establish a grounded reference for the difficulty of these novel problems, we first conducted a preliminary baselining study with a high-performing agent (Gemini-2.5-pro) across the entire problem set. This analysis, detailed in Appendix C, informs our interpretation of agent performance in the main experiments.

Agent Construction. The participants in our study are LLM-based agents, each constructed by pairing a base LLM with a prompt that outlines the objective rules of the competition. Crucially, the prompt is non-prescriptive; it informs the agent that its goal is to achieve the highest possible rank but provides no explicit strategic guidance. This design ensures that the observed strategies are emergent from the agent's own reasoning capabilities. The full system prompt is provided in Appendix H, and a detailed list of the models used is in Appendix F.

Competitor Qualification. To ensure our experiments primarily test strategic decision-making rather than fundamental coding ability, we establish a qualification standard. An agent must successfully solve the easiest Bronze-level problem from a contest's problem set to qualify for that specific competition. This filtering process yielded a consistent roster of top-tier models for all four contests. The complete results for all contests are detailed in Appendix D.

4.2 MAIN COMPETITION: PERFORMANCE AND STRATEGIC ANALYSIS

To ensure the robustness and generalizability, we evaluate agents across the four distinct contests of the 2024–2025 USACO season. Each contest is run five times, and the results presented in Figure 3 are the average of those runs. For this experiment, we select GPT-5-Codex as the representative for the GPT-5 series, as it is specifically optimized for agentic coding tasks (OpenAI, 2025a).

A Stable Two-Tier Hierarchy. The results reveal a consistent two-tier performance hierarchy across all four contests. Gemini-2.5-pro and GPT-5-Codex invariably secure the first and second ranks, respectively, demonstrating a significant and reliable capability gap between these top-tier agents and the rest of the field, whose performance is far more volatile. For clarity, the figure only displays agents that achieve a non-zero average score in each contest.

In-Depth Analysis of Top-Tier Agents. While the main competition results establish a clear performance hierarchy, they do not fully explain *why* one agent consistently outperforms another. The final scores are merely the outcome of a complex, dynamic decision-making process. To understand the underlying strategic differences that lead to victory, this section provides an in-depth analysis of the two top-performing agents: Gemini-2.5-pro and GPT-5-Codex.

A deeper look at the data reveals a fascinating paradox. As summarized in Table 1, GPT-5-Codex demonstrates a significantly higher peak capability, achieving a maximum score of 29 compared to Gemini-2.5-pro's 19. This confirms its potential to solve more difficult, higher-value problems. However, it is Gemini-2.5-pro that achieves a better average rank and a win rate more than double that of its competitor (68.4% vs. 31.6%). This performance inversion points directly to the decisive role of competitive strategy.

Table 1: Comparison of Agent Profile Metrics for Gemini-2.5-pro and GPT-5-Codex.

Agent	Avg. Rank	Win Rate	Max Score	Min Score
Gemini-2.5-pro	1.3 ± 0.47	70.0%	19	4
GPT-5-Codex	1.7 ± 0.47	30.0%	29	3

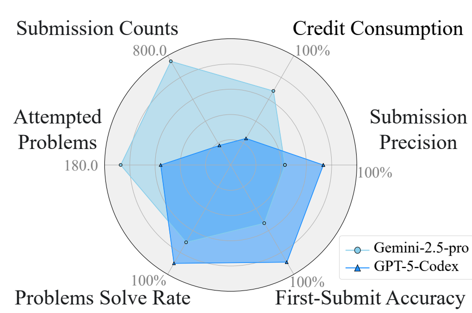


Figure 4: **Strategic Profiles of Top-Tier Agents.** Submission Precision is the percentage of AC submissions out of all submission attempts; Problems Solve Rate is the percentage of AC problems out of all attempted problems; and First-Submit Accuracy is the percentage of problems solved on the first attempt out of all successfully solved problems.

The strategic profiles in Figure 4 explain this outcome, revealing a clear difference. Gemini-2.5-pro's profile is characterized by an aggressive, high-volume strategy. Its plot extends outward on the **Attempted Problems** and **Submission Counts** axes, indicating it attempts more problems to maximize scoring opportunities. This "breadth-first" approach treats credit as a resource to be actively spent in exchange for broader coverage.

In stark contrast, GPT-5-Codex adopts a conservative, "perfectionist" strategy. Its profile is heavily skewed toward near-perfect **First-Submit Accuracy** and **Problems Solve Rate**. This risk-averse approach prioritizes precision, but severely limits its problem coverage, causing it to forego attempts on many potentially solvable problems.

This analysis resolves the performance paradox: Gemini-2.5-pro wins by being a more effective competitor, not necessarily a superior problem-solver. This distinction is best understood through the exploration-exploitation trade-off. Gemini-2.5-pro's strategy is one of aggressive exploration; it attempts many problems to maximize broad coverage and its cumulative score, accepting a lower precision rate as a necessary cost. In contrast, GPT-5-Codex's cautious perfectionism is a form of pure exploitation. Its risk-averse approach limits its attempts to only high-confidence problems, causing it to miss many scoring opportunities and turning its precision into a strategic liability.

This distinction highlights a key finding of our work. Gemini-2.5-pro's success demonstrates that in a competitive setting, a strategy of broad exploration can outperform a more capable but overly conservative exploitation strategy. This implies that optimal performance in a complex, resource-constrained environment like USACOArena requires more than just raw problem-solving accuracy. For the field to advance, this suggests that developing an agent's decision-making framework—its ability to assess risk and manage resources—is as important as enhancing its core capabilities. True expertise in this domain lies in an agent's ability to dynamically balance the trade-off between exploring all viable opportunities and exploiting the most promising ones.

Impact of Contest Difficulty. The varying difficulty of the contests highlights the extent of this performance gap. While more agents achieve non-zero scores in accessible contests, the most challenging one—the USACO 2025 US Open–showcases a dominant performance by Gemini-2.5-pro. Its score of 14.6 establishes a vast lead over GPT-5-Codex (3.0), suggesting that high-difficulty problems strongly accentuate the top agent's strengths. A detailed breakdown is in Appendix E.

Limitations in Agent Self-Assessment. However, a deeper analysis of agent behavior reveals a widespread deficiency in strategic self-assessment. Lower-ranked agents often mismanage resources by attempting problems far beyond their capabilities, forgoing points on easier tasks. Paradoxically, the top-performing agent exhibits the opposite flaw: despite being capable of solving high-value Platinum problems (see Appendix C), Gemini-2.5-pro consistently defaults to safer, lower-scoring problems. This suggests that even the most advanced agents currently lack sophisticated risk-assessment and strategic planning.

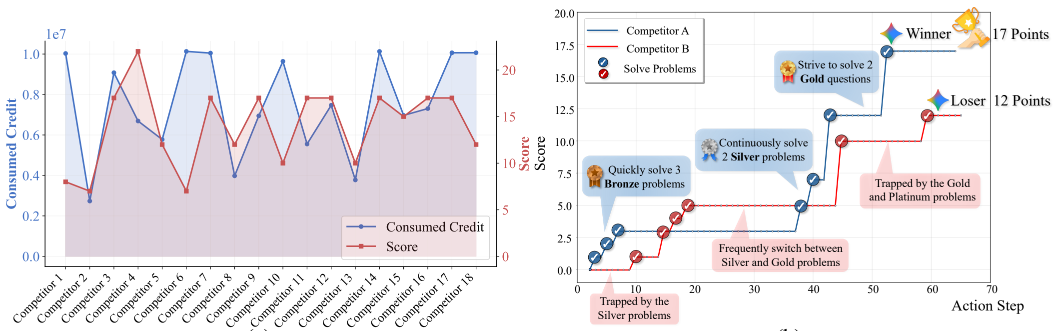
4.3 Probing Agent Architecture: A Case Study on the GPT-5 Family

To isolate the impact of different design choices, we conduct controlled "civil war" experiments within the GPT-5 family. We run two head-to-head matchups: the base GPT-5 against its code-specialized variant, GPT-5-Codex; and GPT-5-Codex against Codex-CLI, a version augmented with an agentic framework. Each matchup is run three times on the challenging US Open contest problem set, with the averaged results reported in Table 2. In our analysis, submission precision is defined as the ratio of correct submissions to the total number of attempts.

Our experiments reveal key trade-offs in agent development. In the first matchup, the specialized GPT-5-Codex adopts a far more cautious approach than its base model. It frequently uses the `TEST` action to ensure high submission precision, but its reluctance to attempt uncertain problems limited its overall score. This suggests that code-specialization enhances reliability, potentially at the cost

Table 2: Performance and Strategy Comparison within the GPT-5 Agent Family. The value in parenthesis indicates the head-to-head win rate for each agent within its matchup.

Agent	Win Rate	Avg. Score	Avg. Credit Consumed	Attempted Problems	Submission Precision (%)
Experiment 1: Generalist vs. Specialist					
GPT-5 (Base)	**100%**	**20.3**	14M	26	12.9%
GPT-5-Codex	0%	8.0	8M	16	68.2%
Experiment 2: Specialist vs. Agentic Framework					
GPT-5-Codex	0%	5	4M	16	44.4%
Codex-CLI	**100%**	**9.5**	15M	21	52.4%



(a) (b)

Figure 5: **Emergent behavioral diversity and strategic divergence in self-play.** (a) Final scores and credit consumed across nine competitions between identical `gemini-2.5-pro` agents, revealing a wide spectrum of outcomes with no trivial correlation between cost and performance. (b) A trajectory analysis of a single match provides a granular explanation, showing how different strategic paths lead to a decisive win-loss result. This demonstrated diversity, resulting from complex path-dependent decisions, validates USACOArena's suitability as a rich training environment.

of problem-solving initiative. The second matchup shows that the agentic framework on Codex-CLI significantly improves performance, achieving a much higher score and win rate while maintaining the same high precision. This demonstrates that a well-designed architecture can directly boost a model's performance, though this may come with higher resource consumption.

4.4 Emergent Behavioral Diversity in Self-Play

To investigate whether a top agent's performance is deterministic, we conduct a series of self-play experiments, pitting two identical instances of `gemini-2.5-pro` against each other. The results reveal a striking diversity of behaviors. As shown in Figure 5(a), the outcomes across 18 competitors are highly variable, rarely ending in a tie, and showing no simple correlation between credit consumed and final score. A trajectory analysis of a single match (Figure 5(b)) provides a granular explanation for this variance. It shows how different, path-dependent strategic choices—such as a methodical, bottom-up approach versus getting stuck on difficult problems—can lead to a decisive win-loss outcome.

These findings yield a key insight: USACOArena is not a simple puzzle but a complex and sensitive environment that can reveal critical inconsistencies in an agent's decision-making. This demonstrated behavioral diversity, where a single policy can produce a wide range of outcomes, is a crucial prerequisite for improvement through learning-based methods. Therefore, our self-play experiments validate USACOArena not only as a robust evaluation testbed but also suggest its potential as a dynamic training ground for future research into cultivating more capable agents.

5 Conclusion

In this work, we introduce USACOArena, an interactive arena designed to measure an agent's ability to make effective decisions under resource constraints. By translating the human constraint of time into an agent-native resource of credit, our environment transforms problem-solving into a process of cost-benefit analysis. Our experiments reveal the deep strategic profiles of leading agents, such as the trade-off between aggressive exploration and conservative precision. Finally, self-play experiments between identical agents confirm that USACOArena is a complex, non-deterministic environment, highlighting its potential not just as a benchmark, but as a training ground for future learning-based approaches to cultivate more strategically capable agents.

REPRODUCIBILITY STATEMENT

We are committed to ensuring the reproducibility of our work. The core methodology of our environment and the complete experimental setup are detailed in Section 3 and Section 4, respectively. The appendix provides exhaustive details necessary for reproduction, including: the full problem corpus from the 2024-2025 USACO season (Appendix C); comprehensive qualification and main competition results (Appendix D and E); a list of all large language models used (Appendix F); all competition hyperparameters (Appendix G); and the exact system prompt provided to the agents (Appendix H). The complete source code for the USACOArena environment, agent wrappers, and evaluation scripts will be made publicly available in an open-source repository upon publication.

REFERENCES

Vaibhav Aggarwal, Ojasv Kamal, Abhinav Japesh, Zhijing Jin, and Bernhard Schölkopf. Dars: Dynamic action re-sampling to enhance coding agent performance by adaptive tree traversal. *arXiv preprint arXiv:2503.14269*, 2025.

Anthropic. Introducing the model context protocol, 2024. URL `https://www.anthropic.com/news/model-context-protocol`. Accessed: 2025-09-21.

Anthropic. Introducing Claude 4. `https://www.anthropic.com/news/claude-4`, 05 2025a.

Anthropic. Claude Opus 4.1. `https://www.anthropic.com/news/claude-opus-4-1`, 08 2025b.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. Swe-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents, 2025. URL `https://arxiv.org/abs/2505.20411`.

Dong Chen, Shaoxin Lin, Muhan Zeng, Daoguang Zan, Jian-Gang Wang, Anton Cheshkov, Jun Sun, Hao Yu, Guoliang Dong, Artem Aliev, Jie Wang, Xiao Cheng, Guangtai Liang, Yuchi Ma, Pan Bian, Tao Xie, and Qianxiang Wang. Coder: Issue resolving with multi-agent and task graphs, 2024a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*, 2024b. URL `https://openreview.net/forum?id=EHg5GDnyq1`.

Chris Cummins, Volker Seeker, Dejan Grubisic, Baptiste Roziere, Jonas Gehring, Gabriel Synnaeve, and Hugh Leather. Meta large language model compiler: Foundation models of compiler optimization, 2024. URL `https://arxiv.org/abs/2407.02524`.

DeepSeek-AI, Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y. Wu, Yukun Li, Huazuo Gao, Shirong Ma, Wangding Zeng, Xiao Bi, Zihui Gu, Hanwei Xu, Damai Dai, Kai Dong, Liyue Zhang, Yishi Piao, Zhibin Gou, Zhenda Xie, Zhewen Hao, Bingxuan Wang, Junxiao Song, Deli Chen, Xin Xie, Kang Guan, Yuxiang You, Aixin Liu, Qiushi Du, Wenjun Gao, Xuan Lu, Qinyu Chen, Yaohui Wang, Chengqi Deng, Jiashi Li, Chenggang Zhao, Chong Ruan, Fuli Luo, and Wenfeng Liang. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence, 2024. URL `https://arxiv.org/abs/2406.11931`.

Le Deng, Zhonghao Jiang, Jialun Cao, Michael Pradel, and Zhongxin Liu. Nocode-bench: A benchmark for evaluating natural language-driven feature addition, 2025. URL https://arxiv.org/abs/2507.18130.

Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, Liuyi Yao, Hongyi Peng, Ze Yu Zhang, Lin Zhu, Chen Cheng, Hongzhu Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. Agentscope: A flexible yet robust multi-agent platform. *CoRR*, abs/2402.14034, 2024.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Taco Cohen, and Gabriel Synnaeve. RLEF: Grounding code LLMs in execution feedback with reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=PzSG5nKe1q.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming – the rise of code intelligence, 2024. URL https://arxiv.org/abs/2401.14196.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2024. URL https://arxiv.org/abs/2308.00352.

Arnav Kumar Jain, Gonzalo Gonzalez-Pumariega, Wayne Chen, Alexander M Rush, Wenting Zhao, and Sanjiban Choudhury. Multi-turn code generation through single-step rewards. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=aJeLhLcsh0.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Naman Jain, Jaskirat Singh, Manish Shetty, Tianjun Zhang, Liang Zheng, Koushik Sen, and Ion Stoica. R2e-gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=7evvwwdo3z.

Nan Jiang, Xiaopeng Li, Shiqi Wang, Qiang Zhou, Soneya Binta Hossain, Baishakhi Ray, Varun Kumar, Xiaofei Ma, and Anoop Deoras. Ledex: training llms to better self-debug and explain code. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. Curran Associates Inc. ISBN 9798331314385.

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

Bowen Li, Wenhan Wu, Ziwei Tang, Lin Shi, John Yang, Jinyang Li, Shunyu Yao, Chen Qian, Binyuan Hui, Qicheng Zhang, Zhiyin Yu, He Du, Ping Yang, Dahua Lin, Chao Peng, and Kai Chen. Prompting large language models to tackle the full software development lifecycle: A case study. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 7511–7531, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.502/.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and LINGMING ZHANG. Evaluating language models for efficient code generation. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=IBCBMeAhmC.

Kaiyuan Liu, Youcheng Pan, Yang Xiang, Daojing He, Jing Li, Yexing Du, and Tianrun Gao. ProjectEval: A benchmark for programming agents automated evaluation on project-level code generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20205–20221, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1036. URL https://aclanthology.org/2025.findings-acl.1036/.

Yingwei Ma, Qingping Yang, Rongyu Cao, Binhua Li, Fei Huang, and Yongbin Li. Alibaba lingmaagent: Improving automated issue resolution via comprehensive repository exploration. *arXiv preprint arXiv:2406.01422*, 2024.

OpenAI. Gpt-5-codex, 2025a. URL https://openai.com/index/introducing-upgrades-to-codex/. Accessed: 2025-09-23.

OpenAI. Introducing GPT-5. https://openai.com/index/introducing-gpt-5, 08 2025b.

Shanghaoran Quan, Jiaxi Yang, Bowen Yu, Bo Zheng, Dayiheng Liu, An Yang, Xuancheng Ren, Bofei Gao, Yibo Miao, Yunlong Feng, et al. Codeelo: Benchmarking competition-level code generation of llms with human-comparable elo ratings. *arXiv preprint arXiv:2501.01257*, 2025.

Quan Shi, Michael Tang, Karthik Narasimhan, and Shunyu Yao. Can language models solve olympiad programming? *arXiv preprint arXiv:2404.10952*, 2024.

Significant Gravitas. AutoGPT. URL https://github.com/Significant-Gravitas/AutoGPT.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An Open Platform for AI Software Developers as Generalist Agents, 2024. URL https://arxiv.org/abs/2407.16741.

Yanlin Wang, Yanli Wang, Daya Guo, Jiachi Chen, Ruikai Zhang, Yuchi Ma, and Zibin Zheng. RLCoder: Reinforcement Learning for Repository-Level Code Completion . In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pp. 1140–1152, Los Alamitos, CA, USA, May 2025a. IEEE Computer Society. doi: 10.1109/ICSE55347.2025.00014. URL https://doi.ieeecomputersociety.org/10.1109/ICSE55347.2025.00014.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Co-evolving llm coder and unit tester via reinforcement learning, 2025b. URL https://arxiv.org/abs/2506.03136.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with OSS-instruct. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 52632–52657. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/wei24h.html.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint*, 2024.

Zhihui Xie, Jie chen, Liyu Chen, Weichao Mao, Jingjing Xu, and Lingpeng Kong. Teaching language models to critique via reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=UVoxPlv5E1.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks, 2025. URL https://arxiv.org/abs/2412.14161.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

John Yang, Carlos E Jimenez, Alex L Zhang, Kilian Lieret, Joyce Yang, Xindi Wu, Ori Press, Niklas Muennighoff, Gabriel Synnaeve, Karthik R Narasimhan, Diyi Yang, Sida Wang, and Ofir Press. Swe-bench multimodal: Do ai systems generalize to visual software domains? In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=riTiq3i21b.

Lei Yang, Renren Jin, Ling Shi, Jianxiang Peng, Yue Chen, and Deyi Xiong. Probench: Benchmarking large language models in competitive programming. *arXiv preprint arXiv:2502.20868*, 2025b.

Xingdi Yuan, Morgane M Moss, Charbel El Feghali, Chinmay Singh, Darya Moldavskaya, Drew MacPhee, Lucas Caccia, Matheus Pereira, Minseon Kim, Alessandro Sordoni, and Marc-Alexandre Côté. debug-gym: A text-based environment for interactive debugging, 2025. URL https://arxiv.org/abs/2503.21557.

Zihan Zheng, Zerui Cheng, Zeyu Shen, Shang Zhou, Kaiyuan Liu, Hansen He, Dongruixuan Li, Stanley Wei, Hangyi Hao, Jianzhu Yao, et al. Livecodebench pro: How do olympiad medalists judge llms in competitive programming? *arXiv preprint arXiv:2506.11928*, 2025.

Yifei Zhou, Song Jiang, Yuandong Tian, Jason Weston, Sergey Levine, Sainbayar Sukhbaatar, and Xian Li. Sweet-rl: Training multi-turn llm agents on collaborative reasoning tasks, 2025. URL https://arxiv.org/abs/2503.15478.

Terry Yue Zhuo, Vu Minh Chien, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen GONG, James Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, David Lo, Binyuan Hui, Niklas Muennighoff, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro Von Werra. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=YrycTjllL0.

A Statement on LLM Usage

Throughout the preparation of this manuscript, we utilized a large language model (Google's Gemini) as a collaborative writing assistant. The model's contributions were primarily focused on the articulation and presentation of our research. Specific tasks included refining the paper's narrative structure and logical flow, condensing and polishing sentences for clarity and impact, suggesting alternative terminology to strengthen key arguments, and providing feedback on the design and captions of figures. The core scientific contributions—including the initial research ideation, the design and implementation of the USACOArena environment, the execution of experiments, and the analysis of the results—were conceived and conducted entirely by the human authors. The role of the LLM was that of a writing partner and editor. We are disclosing its use here to maintain full transparency regarding our research and writing process.

B Foundations in Competitive Programming Standards

The design of USACOArena is deeply rooted in the well-established standards of human competitive programming. This appendix provides a detailed rationale for our two foundational design choices: (1) the adoption of a competitive programming format as the evaluation paradigm, and (2) the specific selection of the ACM-ICPC ruleset over other formats, such as the IOI. These choices are crucial for creating an evaluation framework that measures the capabilities most relevant to the practical application of coding agents in real-world software engineering contexts.

B.1 The Rationale for a Competitive Evaluation Paradigm

Current static benchmarks for code generation, such as HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), primarily evaluate an agent's ability to produce functionally correct code for isolated, well-defined problems. While valuable, this static pass/fail approach fails to capture the dynamic, resource-constrained decision-making process that defines true autonomous agency. It assesses *what* an agent can produce, but not *how* or *why* it arrives at a solution.

For decades, programming competitions have served as the de facto standard for assessing human intelligence in computational problem-solving. They provide a holistic evaluation by creating an environment where participants must:

- **Manage Finite Resources:** Operate under strict constraints (e.g., time, computational resources), forcing strategic allocation of effort.
- **Prioritize Tasks:** Analyze a set of diverse problems, assess their difficulty, and strategically decide the order in which to tackle them.
- **Balance Trade-offs:** Make critical decisions between solution optimality, implementation speed, and correctness risk.

By situating agents in a competitive arena, we move beyond measuring mere correctness and begin to quantify these crucial agentic abilities. The competitive format transforms the evaluation from a simple test of knowledge into a rigorous assessment of strategy, efficiency, and performance under pressure, providing a far richer and more meaningful signal of an agent's true capabilities.

B.2 Why ACM-ICPC over IOI? A Framework for Engineering-Aligned Evaluation

While both the ACM International Collegiate Programming Contest (ICPC) and the International Olympiad in Informatics (IOI) are premier competitions, their underlying philosophies and mechanics are tailored to measure different skills. We deliberately model USACOArena on the ACM-ICPC because its format is substantially more aligned with the values and demands of professional software engineering. The goal is to evaluate coding agents as potential engineering collaborators, not as pure research tools. This alignment is evident across several key dimensions.

Evaluation Paradigm: Breadth and Pragmatism vs. Depth and Originality. The ACM-ICPC is fundamentally a test of **problem-solving breadth and rapid implementation**. A typical contest features a large number of problems (8–13) to be solved within a tight five-hour window. This structure rewards a broad, practical knowledge of standard algorithms and data structures, and the ability to quickly recognize a problem pattern and apply a known, reliable solution. This directly parallels the day-to-day reality of a software engineer, who must efficiently address a wide variety of tasks, from implementing new features to fixing bugs, by applying the right tool for the job.

In contrast, the IOI is a test of **algorithmic depth and creative invention**. With only a few (typically three) highly complex problems per day, it challenges participants to devise novel or highly optimized algorithms, often pushing the boundaries of known techniques. This format is more akin to academic research or work in a specialized R&D lab. For an agent intended to serve as a general-purpose coding assistant, the broad-based, pragmatic skill set measured by the ICPC is a more relevant benchmark.

Scoring Philosophy: The Imperative of Zero-Bug Correctness. A defining feature of the ACM-ICPC is its **all-or-nothing scoring system**. A submission earns credit if and only if it passes every single hidden test case. A solution that fails on a single edge case is equivalent to one that fails completely. This binary outcome brutally enforces the concept of **robustness and correctness**, which is the bedrock of reliable software engineering. In a production environment, code with a "99% pass rate" is simply buggy code, and a single critical failure can have catastrophic consequences. The ICPC format, therefore, directly measures an agent's ability to produce *zero-bug* solutions.

Furthermore, the ICPC's penalty system—which adds a fixed time penalty for each incorrect submission on a problem that is eventually solved—explicitly disincentivizes a careless trial-and-error approach. It rewards careful planning, local testing, and a deep consideration of edge cases before submission, all of which are hallmarks of a disciplined engineering process.

The IOI, conversely, employs a **partial-credit system**, awarding points based on the number of test cases a solution correctly handles. This is excellent for measuring incremental progress and rewarding clever heuristics that solve a subset of the problem. However, it does not instill the same absolute imperative for correctness that the ICPC format does. For an agent destined for real-world deployment, the ICPC's unforgiving standard of correctness is a far more meaningful and critical measure of its reliability.

Alignment with Engineering Values: Efficiency and Delivery over Theoretical Optimality. The intense time pressure and large problem set of the ACM-ICPC naturally encourage competitors to find the **simplest, most direct path to a correct solution**. The goal is not necessarily to write the most theoretically optimal or elegant code, but to write *correct and efficient enough* code to pass within the given constraints, and to do so quickly. This mindset perfectly mirrors the agile, delivery-focused nature of modern software development, where delivering a working, maintainable feature on schedule is paramount, and premature optimization is a well-known anti-pattern.

The IOI's focus on a small number of extremely difficult problems, in contrast, incentivizes the pursuit of **theoretical optimality**. The challenge often lies in shaving off logarithmic factors in complexity or designing a complex algorithm that precisely meets stringent time and memory limits. While an exceptional display of algorithmic prowess, this is often a form of over-engineering in a typical software development context. An agent that rapidly delivers a correct $O(N \log N)$ solution is often more valuable than one that spends immense resources to discover a complex $O(N)$ solution, especially when the former is sufficient for the task at hand.

In summary, by adopting the ACM-ICPC format, we are explicitly choosing to evaluate coding agents against a set of criteria that prioritize the core values of software engineering: broad applicability, rigorous correctness, and the efficient delivery of robust solutions under constraints. This makes USACOArena not just a test of algorithmic knowledge, but a direct measure of an agent's potential as a practical and reliable engineering tool.

C Problem Corpus and Difficulty Baselining

Corpus Philosophy The problem corpus for USACOArena is not a fixed, static set. It is a living collection that mirrors the official USACO contest schedule. For each of the four contests in a USACO season, we adopt the official 12-problem set—three problems each for Bronze, Silver, Gold, and Platinum levels—as the basis for a distinct USACOArena competition. This approach ensures that our benchmark stays current with the evolving difficulty and style of competitive programming problems and avoids any potential bias from manual problem curation.

Difficulty Baselining Study To provide an empirical baseline of difficulty for the novel problems in the 2024-2025 season, we conduct an analysis using a top-tier agent, Gemini-2.5-pro. Each of the 48 problems is run with ample resources to assess its inherent solvability by a state-of-the-art model. The results of this study, presented in Table 3, are not used to select or filter problems, but rather to provide a grounded reference for analyzing agent performance in the main experiments. For example, this data helps us understand when an agent fails on a problem that is known to be solvable, indicating a potential strategic failure rather than a fundamental capability gap.

Table 3: Difficulty baselining results for the 48 problems of the USACO 2024-2025 season, using Gemini-2.5-pro. This data provides a grounded reference for expected problem difficulty in our main experiments.

Problem ID	Level	Result	Test Cases	Cons. Credit	LLM Calls	Sub.
1526	platinum	WA	1/20	10167024	48	37
1525	platinum	WA	1/15	10306989	45	36

Continued on next page

Table 3: – continued from previous page

Problem ID	Level	Result	Test Cases	Cons. Credit	LLM Calls	Submissions
1524	platinum	TLE	1/16	10018138	63	49
1523	gold	TLE	1/20	10151386	57	47
1522	gold	AC	15/15	1446214	6	1
1521	gold	AC	25/25	2818862	18	8
1520	silver	AC	17/17	2936761	17	14
1519	silver	AC	12/12	7132837	36	27
1518	silver	WA	0/18	10131801	55	45
1517	bronze	AC	11/11	510085	3	1
1516	bronze	AC	11/11	204672	2	1
1515	bronze	AC	12/12	137433	3	1
1502	platinum	WA	0/20	10164077	45	21
1501	platinum	WA	1/19	10035673	55	42
1500	platinum	WA	1/13	10065047	54	38
1499	gold	AC	18/18	252597	2	1
1498	gold	TLE	1/20	10012120	52	45
1497	gold	AC	21/21	2939562	10	3
1496	silver	AC	12/12	583964	4	2
1495	silver	AC	18/18	628954	4	3
1494	silver	TLE	1/18	10099233	53	35
1493	bronze	AC	13/13	214167	2	1
1492	bronze	AC	11/11	201263	2	1
1491	bronze	AC	16/16	104275	2	1
1478	platinum	WA	2/19	10175416	45	38
1477	platinum	TLE	1/14	10031850	46	28
1476	platinum	AC	23	735288	4	3
1475	gold	WA	4/18	10129632	50	39
1474	gold	AC	23/23	2377487	12	10
1473	gold	AC	16/16	1723281	8	2
1472	silver	AC	15/15	1368787	10	8
1471	silver	AC	16/16	411089	3	1
1470	silver	AC	23/23	420372	3	1
1469	bronze	AC	13/13	153411	2	1
1468	bronze	AC	11/11	420033	4	3
1467	bronze	AC	12/12	472262	4	2
1454	platinum	WA	1/20	10146203	70	26
1453	platinum	TLE	1/18	10067438	84	19
1452	platinum	AC	15/15	571806	3	2
1451	gold	AC	16/16	1151155	6	4
1450	gold	AC	23/23	4026173	26	13
1449	gold	AC	20/20	1119907	6	5
1448	silver	AC	13/13	2734345	11	2
1447	silver	AC	11/11	2035808	8	1
1446	silver	AC	11/11	1649536	8	5
1445	bronze	AC	13/13	373173	3	2
1444	bronze	AC	16/16	102214	3	1
1443	bronze	AC	13/13	311834	2	1
1430	platinum	TLE	4/24	10106758	55	45
1429	platinum	TLE	1/22	10170320	56	41
1428	platinum	AC	25/24	7526267	40	32
1427	gold	AC	20/20	1213337	7	4
1426	gold	AC	20/20	479578	3	1
1425	gold	AC	23/23	374396	3	2
1424	silver	AC	16/16	528027	3	1
1423	silver	AC	15/15	951701	5	2

Continued on next page

Table 3: – continued from previous page

Problem ID	Level	Result	Test Cases	Cons. Credit	LLM Calls	Submissions
1422	silver	WA	1/21	10032808	74	44
1421	bronze	TLE	2/11	10098563	66	51
1420	bronze	AC	11/11	173543	2	1
1419	bronze	AC	26/26	221140	2	1

D Participant Qualification Results

To ensure that the agents evaluated in our main experimentspossess a baseline of functional competency, we implemented a qualification stage for each of the four USACO contests. The core requirement was for an agent to successfully solve the single easiest Bronze-level problem from the respective contest set. Only agents that achieved an 'Accepted' (AC) status on this prerequisite task were included in the full, multi-problem competitive runs analyzed in the main paper.

The following tables (Table 4 through Table 7) provide the detailed performance results for this qualification task across the four contests. These results not only justify our selection of participants for the main analysis but also offer a preliminary glimpse into the vast performance disparities among the models. Even on these relatively simple entry-level problems, we observe significant variance in resource consumption (Consumed Credit) and efficiency (Submissions), foreshadowing the more complex strategic differences analyzed in the main text.

Table 4: Representative qualification results for the USACO 2024 December Contest. Agents are required to solve the easiest Bronze problem 1445. Cons. Credit means Consumed Credit.

Model	Result	Cons. Credit	LLM Calls	Submissions	Qualified
Gemini-2.5-Pro	AC	373,173	3	2	Yes
GPT-5-Codex	AC	26,986	3	1	Yes
Claude-4-Sonnet	AC	57,405	3	2	Yes
DeepSeek-V3.1	AC	69,514	19	7	Yes
DeepSeek-V3	AC	536,880	130	47	Yes
Kimi-K2-0905	AC	449,065	35	17	Yes
Qwen3-235B	AC	495,102	49	7	Yes
GLM-4.5	AC	7,725	2	1	Yes

Table 5: Representative qualification results for the USACO 2025 January Contest. To qualify, agents were required to solve at least one of the three available Bronze problems (1467, 1468, 1469).

Model	Result	Cons. Credit	LLM Calls	Submissions	Qualified
Gemini-2.5-Pro	AC	472,262	4	2	Yes
GPT-5-Codex	AC	177,127	9	1	Yes
	WA	10,035,455	168	90	
Claude-4-Sonnet	TLE	10,037,109	508	232	No
	TLE	10,013,656	227	127	
DeepSeek-V3.1	AC	497,429	121	46	Yes
Kimi-K2-0905	AC	2,605,011	196	89	Yes
Qwen3-235B	AC	99,574	10	5	Yes
GLM-4.5	AC	245,624	29	5	Yes

Table 6: Representative qualification results for the USACO 2025 February Contest. Agents are required to solve the easiest Bronze problem 1491. Cons. Credit means Consumed Credit.

Model	**Result**	**Cons. Credit**	**LLM Calls**	**Submissions**	**Qualified**
Gemini-2.5-Pro	AC	104,275	2	1	Yes
GPT-5-Codex	AC	19,739	3	1	Yes
Claude-4-Sonnet	AC	57,486	3	2	Yes
DeepSeek-V3.1	AC	14,048	5	1	Yes
DeepSeek-V3	AC	6,918	4	1	Yes
Kimi-K2-0905	AC	213,684	18	9	Yes
Qwen3-235B	AC	122,844	12	3	Yes
GLM-4.5	AC	7,474	2	1	Yes

Table 7: Representative qualification results for the USACO 2025 US Open Contest. Agents are required to solve the easiest Bronze problem 1515. Cons. Credit means Consumed Credit.

Model	**Result**	**Cons. Credit**	**LLM Calls**	**Submissions**	**Qualified**
Gemini-2.5-Pro	AC	137433	3	1	Yes
GPT-5-Codex	AC	37583	4	1	Yes
Claude-4-sonnet	AC	813746	20	9	Yes
DeepSeek-V3.1	AC	1153921	257	47	Yes
DeepSeek-V3	AC	239910	55	17	Yes
Kimi-K2	AC	1366068	95	23	Yes
Qwen3-235B	AC	308213	24	4	Yes
GLM-4.5	AC	245624	29	5	Yes

E Detailed Results of Main Competition

This section provides the detailed results from our main experiment, where each qualified agent competed in the full 12-problem USACOArena contest. To account for the stochastic nature of agent performance and potential variations in LLM API responses, each agent completed the competition five times. The results presented in the main paper are the average of these five runs.

Table 8 presents the aggregated performance metrics for each agent, including the average rank, score, and consumed credit, along with their standard deviations. We also provide a breakdown of the average credit consumption across the main categories—LLM inference, hints, and penalties—to offer deeper insight into each agent's prevailing strategy. The agents are sorted by their final average rank, determined first by average rank and then by average score and consumed credit.

Table 8: Aggregated results from the main experiment, averaged over 5 runs across four contests. The data shows each agent's final rank, score, and credit consumption, reflecting their strategic priorities. Values are presented as mean $\pm$ standard deviation.

Model	**Avg. Rank**	**Avg. Score**	**Avg. Consumed Credit**	**Inference Credit**	**Hint Credit**	**Penalty Credit**
Gemini-2.5-pro	1.30 ± 0.47	14.00 ± 3.88	13,762,787 $\pm$ 4.3M	13.76M $\pm$ 4.3M	2.5K $\pm$ 2.5K	4.1K $\pm$ 1.9K
GPT-5-Codex	1.70 ± 0.47	9.39 ± 7.59	4,707,464 $\pm$ 3.1M	4.71M $\pm$ 3.1M	1.1K $\pm$ 2.0K	0.3K $\pm$ 0.3K
Qwen3-235b	4.00 ± 1.59	1.61 ± 0.92	11,732,391 $\pm$ 6.9M	11.17M $\pm$ 6.5M	560.2K $\pm$ 375.7K	3.6K $\pm$ 2.6K
GLM-4.5	4.35 ± 1.57	2.33 ± 2.28	7,249,215 $\pm$ 4.0M	7.06M $\pm$ 3.9M	161.7K $\pm$ 93.2K	22.7K $\pm$ 16.8K
DeepSeek-V3	5.70 ± 1.13	0.11 ± 0.32	194,050 $\pm$ 0.2M	0.17M $\pm$ 0.2M	19.8K $\pm$ 24.9K	1.2K $\pm$ 1.3K
DeepSeek-V3.1	6.00 ± 1.30	0.06 ± 0.24	253,013 $\pm$ 0.4M	0.23M $\pm$ 0.4M	21.9K $\pm$ 32.7K	1.4K $\pm$ 2.7K
Kimi-K2-0905	6.00 ± 1.45	0.72 ± 1.18	1,337,561 $\pm$ 2.0M	1.32M $\pm$ 2.0M	10.0K $\pm$ 17.2K	3.0K $\pm$ 4.2K
Claude-4-sonnet	6.95 ± 1.36	0.39 ± 0.61	1,285,766 $\pm$ 0.8M	1.28M $\pm$ 0.8M	1.4K $\pm$ 1.7K	1.1K $\pm$ 0.6K

The raw, run-by-run data for each agent, including detailed action logs and final scores for each of the five trials, are available in the supplementary material for full reproducibility.

The aggregated results highlight key strategic differences. For example, while GPT-5 and Gemini-2.5-pro are the clear top performers, GPT-5 consistently consumes less credit across all categories, indicating a more efficient problem-solving process. The credit breakdown also reveals that lowertier models often accumulate significant penalty credit without a corresponding increase in score, suggesting a tendency towards inefficient trial-and-error strategies.

F Large Language Model Details

Our evaluation leverages a diverse suite of Large Language Models (LLMs) to ensure a comprehensive analysis of agent capabilities within the USACOArena. Table 9 provides a detailed breakdown of the models employed in our study, including their provider and associated costs. All pricing data, specified in U.S. dollars per million input and output tokens respectively, was retrieved from Artificial Analysis[1] in September 2025. This selection represents a cross-section of the contemporary LLM landscape, encompassing models with varied architectures, parameter scales, and economic costs, thereby facilitating a robust and multifaceted analysis of agent performance.

Table 9: Specifications of Large Language Models used in our evaluation. Costs are denoted in USD per million tokens.

Provider	Model	Input Cost	Output Cost
OpenAI	`GPT-5-2025-08-07`	$1.25	$10.00
	`GPT-5-Codex`	$1.25	$10.00
Google	`Gemini 2.5 Pro`	$1.25	$10.00
Anthropic	`Claude-Sonnet-4-20250514`	$3.00	$15.00
DeepSeek	`DeepSeek-v3`	$0.27	$1.10
	`DeepSeek-v3.1`	$0.27	$1.10
Alibaba Cloud	`Qwen3-235B-A22B-Instruct-2507`	$0.70	$2.80
Moonshot AI	`Kimi-K2-0905`	$1.00	$2.75
Zhipu AI	`GLM-4.5`	$0.59	$2.19

[1] `https://artificialanalysis.ai`

G USACOArena Hyperparameters

The main experiments conducted in this study utilize a standardized default configuration for the USACOArena environment. This configuration, which is highly customizable to facilitate diverse research questions, is detailed in Table 10.

Table 10: USACOArena Competition Configuration Parameters

Parameter	Description	Default Value
Basic Setup		
max_credits_per_participant	Maximum credits per participant	20,000,000
Scoring System		
bronze_score	Points for Bronze problems	1
silver_score	Points for Silver problems	2
gold_score	Points for Gold problems	5
platinum_score	Points for Platinum problems	10
LLM Inference		
agent_temperature	Model generation temperature	0.7
Hint Request Costs		
level_0_hint	Strategy hints cost	500
level_1_hint	Textbook knowledge cost	1,000
level_2_hint	Knowledge-specific content cost	1,000
level_3_hint	Similar problems cost	1,500
level_4_hint	Example problems cost	1,500
Test Code Costs		
test_code	Base cost per test request	10
Penalty System		
WA_penalty	Penalty for Wrong Answer	100
RE_penalty	Penalty for Runtime Error	100
CE_penalty	Penalty for Compile Error	100
TLE_penalty	Penalty for Time Limit Exceeded	100
MLE_penalty	Penalty for Memory Limit Exceeded	100
Problem Distribution		
total_problems	Total number of problems	12
bronze_problems	Bronze difficulty count	3
silver_problems	Silver difficulty count	3
gold_problems	Gold difficulty count	3
platinum_problems	Platinum difficulty count	3

H Prompt for USACOArena Evaluation

The following box details the complete prompt structure provided to agents in the USACOArena competition. The prompt is designed as a purely objective specification of the environment. It comprehensively delineates the foundational components of the competition: the governing rules, the format for communicating game state, the complete set of available actions, and the structure of action results. Crucially, the prompt deliberately refrains from offering any strategic guidance or heuristics. This ensures that all observed strategies are properties of the agent's autonomous decision-making process, rather than a reflection of guidance embedded in the instructions

An Example Prompt for Evaluating Agentic LLMs in USACOArena

System Prompt

You are a competitive programming agent participating in a coding competition. You will receive the current state of the competition and results of your previous actions. Your goal is to solve as many problems as possible (achieve 'Accepted' status).
Your final ranking is determined first by your total score. The score is a weighted sum of the problems you solve, with harder problems (e.g., Platinum) being worth more than easier ones (e.g., Bronze). If scores are tied, the agent with the lower total of (*actual consumed credit* + *penalties*) ranks higher.
You start with a limited credit budget, and many actions consume credit. **You will be terminated from the competition when your actual consumed credit reaches the limit.**
Credit is consumed in three main ways:

1. **LLM Inference:** Generating responses, which consumes credit based on the number of tokens you use.
2. **Purchasing Hints:** Using hints to help solve problems.
3. **Testing Code:** Running your code against test cases before final submission.

IMPORTANT:

- Penalties from wrong submissions affect your ranking tie-breaker but do **NOT** count toward termination.
- In this competition, solving problems is much more important than minimizing the consumed credit. So you should try your best to solve as many problems as possible.

Please respond with a JSON object containing `'action'` and `'parameters'` fields.

User Prompt

Competition Rules

- **Credit System:**
  - Each participant starts with a total of **20,000,000** credit limit.
  - Credit is consumed by three main sources: LLM Inference, Purchasing Hints, and Testing Code.
  - Your participation ends when your **actual consumed credit** reaches the limit.
- **Scoring Rules:**
  - Your **Final Score** is the sum of points from all problems you solve completely.
  - No partial credit is awarded.
  - Points are weighted by difficulty: Bronze (1), Silver (2), Gold (5), Platinum (10).
- **Penalties:** A penalty of 100 points is incurred for CE, MLE, RE, TLE, and WA submissions.
- **Ranking and Tie-Breaking:** Rank is determined by Final Score. Ties are broken by the lower (Actual Consumed Credit + Penalties).
- **Programming Languages:** C++17, Java, and Python3 are available.

Your Status

- **Name:** `<agent_name>`
- **Consumed Credit:** `<consumed_credit>`
- **Solved Problems:** `<solved_list>`
- **Current Score:** `<score>`
- **Penalty:** `<penalty>`

Available Problems

- `<problem_id_1>`
- `<problem_id_2>`
- ...

Current Rankings

```
1. <Agent 1>: Score <S1>, Credit+Penalty: <C1> [ACTIVE]
2. <Agent 2>: Score <S2>, Credit+Penalty: <C2> [TERMINATED]
...
```

AVAILABLE ACTIONS

1. **VIEW_PROBLEM:** View problem details.
2. **GET_HINT:** Get a hint for a problem (consumes credit). Levels 0-4 are available.
3. **SUBMIT_SOLUTION:** Submit a solution.
4. **TEST_CODE:** Test code with custom test cases (consumes credit).
5. **TERMINATE:** End participation.

RESPONSE FORMAT

Please respond using the following JSON format:

```
{
  "action": "<action_name>",
  "parameters": {
    // Fill in parameters according to the action type
  }
}
```

I THE FIVE-TIERED HINT SYSTEM IN USACOARENA

To rigorously evaluate an agent's ability to make strategic decisions under resource constraints, we engineered a sophisticated five-tiered hint system within USACOArena. This system is not merely a help feature; it functions as an economic model where information is a commodity with varying costs and utilities. Agents must perform a cost-benefit analysis to decide if, when, and what type of hint to purchase. This design allows us to observe and quantify an agent's resource management and problem-solving strategies.

Level 0: Strategic Guidance (500 Credit) This foundational hint provides high-level, static information about competitive programming.

- **Function:** It delivers a pre-compiled document containing the core philosophy of competitive programming, a comprehensive debugging checklist, and general contest strategies (e.g., time management). The contents are derived from USACO Guide[2].
- **Mechanism:** The system retrieves the full content from a static JSON file (`/dataset/corpuses/USACO_strategy.json`). The API call is parameter-free (`{"hint_level": 0}`).
- **Strategic Purpose:** This low-cost hint is intended for the early stages of a competition, allowing an agent to establish a baseline understanding of the meta-game without spending significant resources.

Level 1: Problem-Specific Textbook Content (1,000 Credit) This hint offers theoretical knowledge directly relevant to a specific problem.

- **Function:** It provides a concise, relevant excerpt from a competitive programming textbook that explains the theoretical concepts or algorithms needed for a given problem.
- **Mechanism:** Upon receiving a `problem_id`, the system automatically extracts key algorithmic and data structure terms from the problem description. It then employs a BM25 search algorithm to find the most relevant section in a 2.8MB textbook corpus which is derived from Algorithms for Competitive Programming[3]. The top result is returned, truncated to 1,000 characters.
- **Strategic Purpose:** This is for agents that can identify a knowledge gap related to a problem but do not know the name of the required algorithm. It tests the agent's ability to recognize when it needs theoretical grounding.

[2]https://usaco.guide

[3]https://cp-algorithms.com

Level 2: Knowledge-Targeted Textbook Content (1,000 Credit) Similar to Level 1, this hint also retrieves textbook content, but with a key difference in agent interaction.

- **Function:** It provides a detailed explanation of a specific algorithm or data structure explicitly named by the agent.
- **Mechanism:** Instead of a `problem_id`, the agent must provide a `hint_knowledge` keyword (e.g., "segment tree"). The system uses this keyword directly in its BM25 search against the same textbook corpus.
- **Strategic Purpose:** This hint is for a more advanced scenario where an agent correctly identifies the required algorithm by name but needs to learn its implementation details. It tests an agent's self-awareness of its specific knowledge deficits.

Level 3: Similar Problem Retrieval (1,500 Credit) This high-cost hint provides a concrete, solved example of a similar problem.

- **Function:** It returns a full problem description, along with a complete, vetted solution and explanation, for a problem that is semantically similar to the one the agent is currently working on.
- **Mechanism:** The system uses the current `problem_id`'s text (description and samples) as a query for a BM25 search against the entire USACO problem library derived from USACO Guide[4], excluding problems from the current competition. The most similar problem is returned.
- **Strategic Purpose:** This is a powerful tool for agents that are completely stuck on the problem-solving approach. Its high cost forces the agent to consider whether viewing a direct analogy is worth the significant credit expenditure.

Level 4: Curated Example Problems (1,500 Credit) This is the most targeted hint, designed to provide practice on a specific topic at a specific difficulty.

- **Function:** It retrieves a complete example problem (description, solution, complexity analysis) that matches both a user-specified difficulty level and a knowledge keyword.
- **Mechanism:** The system filters the entire USACO problem library based on both `problem_difficulty` (e.g., "Bronze") and `hint_knowledge` (e.g., "complete search") tags provided by the agent.
- **Strategic Purpose:** This hint allows an agent to request a targeted exercise, simulating a human's process of looking for practice problems. It tests the agent's ability to formulate a precise learning objective.

[4] `https://usaco.guide`