ICPC-Eval: Probing the Frontiers of LLM Reasoning with Competitive Programming Contests

NeurIPS 2025 Datasets & Benchmarks Track

Shiyi Xu, Yiwen Hu, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, Ji-Rong Wen

中国人民大學 RENMIN UNIVERSITY OF CHINA

高瓴人工智能学院 Gaoling School of Artificial Intelligence

Motivation: The Rise of Reasoning in LLMs

- Large Language Models (LLMs) have shown exceptional performance in a wide range of tasks.
- Recent models (OpenAI's o1/o3, DeepSeek-R1, Gemini 2.5) demonstrate significantly advanced reasoning capabilities.
- Competitive programming has become a key area for evaluating these advanced abilities, as it requires translating complex mathematical logic into executable code.
- **Core Question:** How do we effectively measure the true reasoning limits of today's best models?

Challenges with Existing Benchmarks

1. Relatively Low Difficulty

- Many LLMs achieve near-perfect scores on benchmarks like HumanEval.
- Platforms like LiveCodeBench and USACO are increasingly being solved by powerful reasoning models, reducing their discriminative power.

2. Lack of Accessibility and Realism

- Benchmarks like CodeElo rely on submissions to Online Judges (OJs) with private test cases, hindering local evaluation.
- The widely used **Pass@K** metric fails to capture the iterative refinement process of real problem-solving.

Our Solution: ICPC-Eval

A Top-Level Competitive Coding Benchmark

1
Challenging

2
Locally Evaluable

3
Better Metric

Our Contributions

- **A challenging benchmark:**
  - Features 118 top-difficulty problems curated from 11 recent International Collegiate Programming Contest (ICPC) events.
  - Ensures a rigorous test of advanced reasoning with minimal risk of data contamination.
- **A novel local evaluation toolkit:**
  - A new test case generation and validation methodology using LLMs to create comprehensive local test suites.
  - Enables robust and accessible offline assessment.
- **An effective test-time scaling evaluation metric, Refine@K:**
  - Measures an LLM's ability to iteratively refine its solutions based on execution feedback.

Contribution 1: A Challenging Benchmark

- **Problem Source:** 11 recent ICPC contests (World Finals, Continent Finals, Regionals).
  - Most contests from late 2024 to minimize contamination risk.
- **Data Curation:**
  - Started with 139 raw problems.
  - Filtered out problems with non-textual images, interactive elements, or no standard solution.
  - Manually developed special judges for 12 problems with complex output requirements.
  - **Final set: 118 high-quality problems.**
- **Problem Distribution:** Problems are tagged across 8 algorithmic domains, often involving multiple advanced topics.

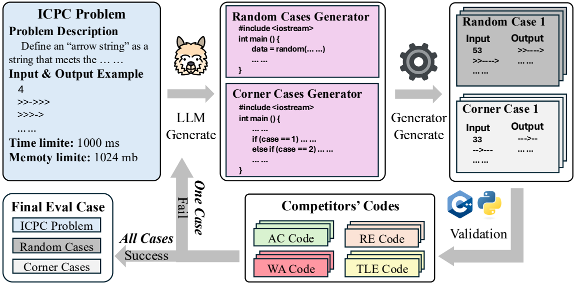
Problem Domain Distribution

Domain	Topic	WFs & CFs	Regionals
Algorithm Basics	Greedy, Divide-and-conquer,	7	27
Computational Geometry	Sweep Line, Rotating Calipers,	6	11
Data Structure	Segment Tree, Binary Search Tree,	6	24
Dynamic Programming	Knapsack, DP on Trees, Bitmask,	11	27
Graph Theory	Dijkstra, Network Flow,	4	22
Mathematics	Combinatorics, Number Theory,	15	33
Search Algorithm	DFS, BFS, Backtracking,	15	20
String Algorithm	KMP, Z-algorithm, Suffix Array,	5	1
All		**31**	**87**

Contribution 2: Local Evaluation Toolkit

Overcoming the reliance on Online Judges.

- Existing difficult benchmarks often lack public test cases.
- Our goal: Create an efficient and accurate local evaluation pipeline.



Test Case Generation and Validation Pipeline

- **Step 1: Synthesize Input Generators**
  - Use an LLM (Gemini 2.5 Pro) to write C++ programs that generate test inputs.
  - Two types of generators:
    - **Random Generator:** Samples uniformly from the data range.
    - **Corner Case Generator:** Creates edge cases and specially structured inputs.
- **Step 2: Generate Outputs**
  - Use a known `Accepted` solution for each problem to generate the correct outputs for the synthesized inputs.
- **Step 3: Rigorous Validation**
  - Validate the generated test cases by ensuring they correctly fail known incorrect programs (e.g., solutions that get `Wrong Answer` or `Time Limit Exceeded` on the OJ).
  - This process ensures our local evaluation has **zero false positives**.

Contribution 3: A Better Metric for Test-time Scaling

- **The Problem with Pass@K:**
  - Samples N independent code completions.
  - Doesn't reflect how reasoning models (or humans) solve problems: through **iterative refinement** based on feedback.
  - In a real ICPC contest, teams submit an average of **1.95 attempts per solved problem**.

- **Our Proposed Metric: Refine@K**
  - Simulates a real competition environment.
  - Measures if a model can solve a problem within a budget of K attempts.
  - The model receives specific execution feedback after each failed attempt and is prompted to repair its code.

How Refine@K Works

- **Attempt 1:** Model receives the problem description and generates a solution.
  - $\mathit{Response}_1 = \mathrm{LLM}(\mathit{Problem})$
- **If Attempt $i-1$ Fails:** The model receives its previous code and targeted feedback.
  - *Feedback type depends on the error:*
    - Compilation Error: Full compiler message.
    - Sample Case Failure: Wrong output vs. expected output.
    - Hidden Case Failure: Just the error type (e.g., `Wrong Answer`, `Time Limit Exceeded`).
- **Attempt i ($1 < i \leq K$):** Model generates a revised solution.
  - $\mathit{Response}_i = \mathrm{LLM}(\mathit{Problem}, \mathit{Response}_{i-1}, \mathit{Feedback}_{i-1})$
- A problem is considered ”solved” if any attempt within the K budget passes all test cases.

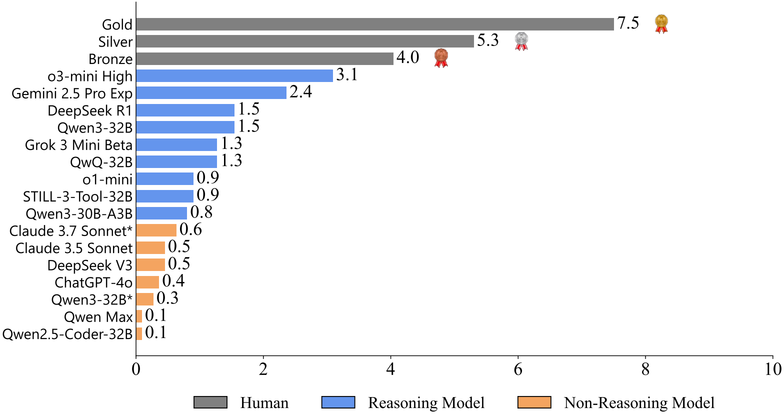
Experimental Setup

- **Models Evaluated (15 SOTA LLMs):**
  - **Reasoning Models:** o3-mini High, Gemini 2.5 Pro Exp, DeepSeek R1, Grok 3 Mini, etc.
  - **Hybrid-reasoning Models:** Qwen3-32B, Claude 3.7 Sonnet, etc.
  - **Non-reasoning Models:** ChatGPT-4o, Claude 3.5 Sonnet, DeepSeek V3, etc.

- **Evaluation Details:**
  - Primary Metric: **Refine@5**.
  - Compiler: GNU GCC 14 with C++23 standard.
  - Hardware: Intel Xeon Platinum 8160 CPU.

Main Results: A Significant Gap Remains

Even top AI models lag behind human ICPC medalists.



This

Main Results: Model Performance (Refine@5)

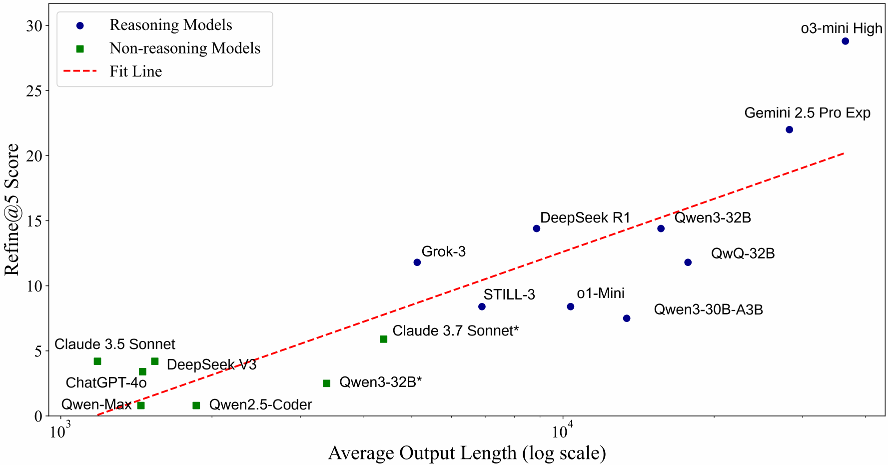
tableo3-mini High leads, and reasoning models significantly outperform non-reasoning ones. Full results are in the paper.

Model	Type	Full Score (%)	Math (%)	DP (%)	DS (%)
o3-mini High	**Reasoning**	**28.8**	**29.2**	**21.1**	**33.3**
Gemini 2.5 Pro Exp	Reasoning	22.0	22.9	13.2	30.0
DeepSeek R1	Reasoning	14.4	8.3	10.5	23.3
Qwen3-32B	Hybrid	14.4	10.4	10.5	20.0
Claude 3.5 Sonnet	Non-reasoning	4.2	6.3	0.0	3.3
DeepSeek V3	Non-reasoning	4.2	2.1	0.0	6.7
ChatGPT-4o	Non-reasoning	3.4	4.2	0.0	3.3

- **Key Insight:** Execution feedback effectively elicits the reasoning and reflection capabilities of advanced models.

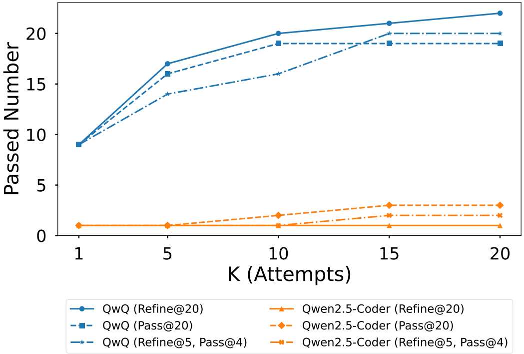
Analysis: Refine@K Scales with Output Length

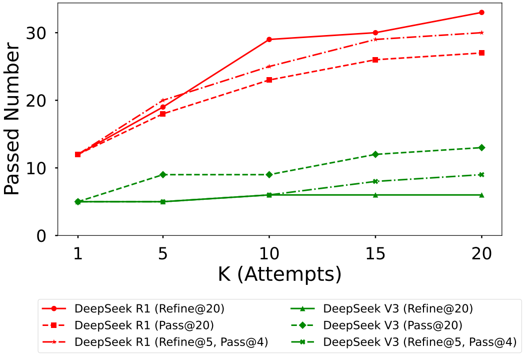
Longer ”thinking” (Chain-of-Thought) correlates with better performance.



Analysis: Refine@K vs. Pass@K

Refine@K is a more suitable metric for *reasoning* models.





- Reasoning Models (QwQ, DeepSeek R1): Performance scales much better with Refine@K, as they can leverage feedback to improve.
- Non-reasoning Models (Qwen-Coder, DeepSeek V3): Perform better with Pass@K (simple resampling). They struggle to benefit from feedback and may even be hindered by it.

Comparison with Other Benchmarks

ICPC-Eval is significantly more challenging. table

Model	ICPC-Eval (%) (Refine@5)	LiveCodeBench (%) (Pass@1)	CodeElo (Rating)
o3-mini High	**28.8**	67.4	-
Gemini 2.5 Pro Exp	22.0	**67.8**	2001
DeepSeek R1	14.4	64.3	**2029**
Grok 3 Mini Beta	11.8	66.7	-
Claude 3.5 Sonnet	4.2	36.4	710

Conclusion

- We introduced **ICPC-Eval**, a new benchmark to probe the frontiers of LLM reasoning with challenging competitive programming problems.
- Our results show that even SOTA models have a substantial gap to top human performance, highlighting the benchmark's rigor.
- We provide a **robust local evaluation pipeline**, removing the dependency on online judges and enabling broader research.
- We proposed **Refine@K**, a metric that more faithfully captures the iterative, feedback-driven problem-solving process of advanced reasoning models.

Limitations and Future Work

- **Limitations:**
  - **Scope:** Current version focuses on 11 recent contests.
  - **Language:** Primarily C++, aligned with ICPC practice.
  - **Modality:** Excludes problems requiring image understanding or interaction.

- **Future Work:**
  - Periodically refresh ICPC-Eval with new contests.
  - Extend evaluation to more programming languages (Python, Java).
  - Explore multimodal and interactive reasoning tasks.
  - Support evaluation of tool-augmented agents (e.g., with debuggers).