LLM-ProS: Analyzing Large Language Models' Performance in Competitive Problem Solving

Md Sifat Hossain, Anika Tabassum, Md. Fahim Arefin
Department of Computer Science and Engineering,
University of Dhaka, Dhaka, Bangladesh
Email: {*mdsifat-2019217800@cs.du.ac.bd,*
anika-2019417844@cs.du.ac.bd,
fahim@cse.du.ac.bd}

Tarannum Shaila Zaman
Department of Information Systems,
University of Maryland, Baltimore County, Maryland, USA
Email: {*zamant@umbc.edu*}

***Abstract*—The rapid advancement of large language models has opened new avenues for automating complex problem-solving tasks such as algorithmic coding and competitive programming. This paper introduces a novel evaluation technique, LLM-ProS, to assess the performance of state-of-the-art LLMs on International Collegiate Programming Contest (ICPC) problems. Using a curated dataset of 166 World Finals problems from 2011 to 2024, we benchmark the models' reasoning, accuracy, and efficiency. We evaluate the five models-GPT-4o, Mistral Large, Llama-3.1-405B, and the o1 family, consisting of o1-mini and o1-preview, across critical metrics like correctness, resource utilization, and response calibration. Our results reveal significant differences in the models' abilities to generalize, adapt, and solve novel problems. We also investigated the impact of training methodologies, dataset contamination, and chain-of-thought reasoning on model performance. The findings provide new insights into optimizing LLMs for algorithmic tasks, highlighting both strengths and limitations of current models.**

***Index Terms*—Large Language Models, Competitive Programming, ICPC, Performance Evaluation, Chain-of-Thought Reasoning**

I. Introduction

Large language models (LLMs) [1] have revolutionized natural language processing (NLP) [2] and their applications in various technical domains, including algorithmic problem-solving and code generation. Trained on extensive and diverse datasets, these models exhibit remarkable capabilities in understanding and generating code [3]. However, their performance on complex, real-world coding challenges remains an area of active investigation. Competitive programming problems, such as those from the International Collegiate Programming Contest (ICPC) [4], offer a unique opportunity to rigorously evaluate LLMs due to their intricate constraints, computational demands, and emphasis on algorithmic efficiency.

ICPC problems are particularly well-suited for evaluating LLM performance [5] for several reasons. First, these problems require a combination of logical reasoning, algorithmic thinking, and precise implementation, reflecting real-world software engineering challenges. Second, their diverse categories, including graph theory, dynamic programming, and computational geometry, offer a broad spectrum of difficulty levels, testing an LLM's [6] ability to generalize across various problem types. Third, the focus on correctness, efficiency, and edge-case handling aligns with critical evaluation metrics such as runtime performance and memory usage [7]–[9].

Existing works use LLMs at different stages of code generation and testing. For example, Fakhoury et al. [10] improve solutions through test feedback and refine them step-by-step. Liu et al. [7] identify problems like data contamination and over-reliance on memorized patterns, which reduce the reliability of results. A recent study [11] explores how adding syntax and grammar rules during training can enhance the accuracy and robustness of LLMs. Similarly, Coignion et al. [12] analyze the efficiency of code generated by LLMs on Leetcode, revealing performance trends and benchmarks. However, these studies primarily focus on specific aspects of code generation, such as correctness or efficiency, without comprehensively evaluating LLM performance in solving complex, real-world competitive programming problems. To address this gap, we propose an approach, LLM-ProS, to assess the capabilities of advanced LLMs, including OpenAI's o1 family [13], GPT-4o [14], Mistral Large [15], and Llama-3.1-405B [16], in tackling ICPC problems.

To develop LLM-ProS, we collect a curated dataset of 166 ICPC World Finals problems from the years 2011 to 2024, providing a comprehensive benchmark for assessing the reasoning, accuracy, and efficiency of LLMs. The study assessed five state-of-the-art models—GPT-4o, Mistral Large, Llama-3.1-405B, o1-preview and o1-mini-selected for their diverse architectures and training methodologies. Comprehensive data preprocessing involved extracting problem components, standardizing prompts, cleaning text, customizing templates for each model, and validating the preprocessed data. Solutions generated by the LLMs were submitted to Codeforces Gym ICPC contests, receiving automated feedback on correctness and efficiency. All experiments were conducted in a controlled environment, with reproducible submissions and publicly available scripts to ensure consistency and transparency. In summary we make the following contributions:

- We propose a performance analyzer, LLM-ProS, to assess the performance of five different LLMs on new ICPC

problems and examine accuracy trends over time.
- We present an experimental evaluation that identifies factors contributing to variability in LLM performance and highlights the distribution of verdicts across various LLMs for the given problem set.

Our findings reveal that o1 models, due to their enhanced Chain of Thought reasoning and calibration, significantly outperform others in terms of accuracy, robustness, and efficiency. Additionally, we identify critical limitations, such as overestimation of confidence by some models and their struggle with complex, high-difficulty problems. Recent studies, such as *The Llama 3 Herd of Models* [17], further highlight the challenges in ensuring consistent performance across diverse technical domains, emphasizing the need for specialized benchmarks. By addressing these issues and leveraging insights from ICPC problem evaluations, this paper lays the foundation for improving LLM design and training methodologies to meet the challenges of technical problem-solving.

II. Background and Motivation

A. Motivation

Despite their remarkable capabilities, the effectiveness of LLMs in tackling complex, real-world coding challenges remains inadequately explored. Traditional benchmarks often fall short in assessing the nuanced reasoning, adaptability, and efficiency required for high-stakes programming tasks.

The International Collegiate Programming Contest (ICPC), the oldest, largest, and most prestigious programming contest in the world, presents a unique and rigorous environment for evaluating LLM performance [18]. ICPC-style problems are characterized by their intricate constraints, diverse problem categories, and emphasis on algorithmic efficiency and edge-case handling. These attributes make them ideal for benchmarking the reasoning capabilities, accuracy, and resource utilization of LLMs in a controlled yet challenging setting [7], [8].

Moreover, the increasing reliance on LLMs for automated code generation in software engineering necessitates a deeper understanding of their strengths and limitations. Evaluating LLMs against competition-level programming problems not only highlights their problem-solving prowess but also identifies critical areas for improvement, such as generalization to unseen problems and efficient resource usage [10], [19]. This study aims to bridge the existing research gaps by providing a comprehensive assessment of state-of-the-art LLMs using a curated set of ICPC World Finals problems.

B. Background

To thoroughly evaluate the performance of LLMs on ICPC-style problems, we selected a diverse range of models representing various architectures, training methodologies, and optimization strategies. The models under consideration include GPT-4o, Mistral Large, Llama-3.1-405B, o1-mini, and o1-preview. Each model was chosen for its distinct characteristics and potential applicability to structured problem-solving tasks.

1) GPT-4o: GPT-4o [12], [17], [20] is a general-purpose language model renowned for its high accuracy in code generation and reasoning tasks. Leveraging its extensive training on diverse datasets, GPT-4o [21] demonstrates strong capabilities in understanding complex problem statements and generating syntactically correct code. However, its performance may be less optimized for structured problem-solving compared to models specifically fine-tuned for reasoning and iterative refinement.

2) Mistral Large: Mistral Large specializes in handling domain-specific tasks with a focus on efficient resource utilization. Designed to balance performance with computational efficiency, Mistral Large offers insights into the trade-offs between model complexity and practical deployment in resource-constrained environments. Its performance on ICPC problems provides valuable data on how domain specialization impacts problem-solving efficacy [11].

3) Llama-3.1-405B: Llama-3.1-405B [12], [22] is a computationally efficient model tailored for general-purpose use. Serving as a benchmark for lightweight LLMs in competitive programming, Llama-3.1-405B facilitates comparisons regarding scalability and efficiency without significant compromises in accuracy. Its performance metrics shed light on the feasibility of deploying smaller models in high-stakes programming scenarios. It is also particularly interesting as the best-performing open source model.

4) OpenAI o1 Family: The o1-mini and o1-preview models [9], [19] from the OpenAI o1 family are specifically fine-tuned for chain-of-thought (CoT) reasoning and iterative refinement processes. These models are engineered to excel in structured, multi-step problem-solving tasks, making them highly suitable for ICPC-style evaluations. Their design emphasizes enhanced reasoning pathways and reduced hallucination rates, enabling more consistent and accurate code generation under complex constraints.

By systematically applying each model to this diverse set of problems, the study evaluates not only the accuracy and robustness of the solutions but also the computational efficiency and adaptability of the models in handling unseen and complex programming tasks. This rigorous evaluation framework provides a holistic view of the current capabilities of LLMs in competitive programming contexts, informing future developments in model design and training methodologies.

III. Methodology

Figure 1 illustrates the overview of our proposed technique, LLM-ProS. LLM-ProS consists of four major steps: data collection, data preprocessing, model testing, and solution generation and submission. The details of each step are discussed in the following subsections.

A. Data Collection

We scrape a total of 166 competitive programming problems from the ICPC official website [4], specifically from the World Finals editions spanning the years 2011 to 2024 World Finals. We select these years to avoid potential overlap with the

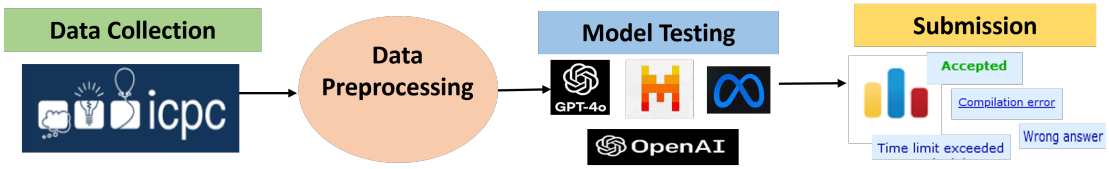

Fig. 1. The Overview of LLM-ProS

training data of major LLMs, ensuring a more accurate evaluation of their performance. Table I shows a sample problem structure that we have collected. Each problem includes comprehensive details such as problem statements, input formats, output formats, sample inputs and outputs, notes, and specified time and memory limits. This dataset provides a diverse and challenging set of tasks that effectively test the reasoning and computational abilities of LLMs.

TABLE I
SAMPLE PROBLEM DATA

Problem Statement	Allied Chute Manufacturers is a company that builds trash chutes. A trash chute ...
Input-Output	The input contains several test cases. Each test case starts with an integer n, representing the number of points in the polygon that models the trash item ($3 \leq n \leq 100$)...

Sample testcase	**Sample Input:**
	`3`
	`0 0`
	`...`
	Sample Output:
	`Case 1: 2.40`
	`Case 2: 14.15`

B. Data Preprocessing

Before generating solutions, we systematically preprocess each of the 166 ICPC World Finals problems to ensure compatibility and consistency across different LLMs. The preprocessing steps include:

Extraction of Problem Components: We break down each problem into its fundamental components, including the problem statement, input format, output format, sample inputs and outputs, notes, and specified time and memory limits. This structured extraction helps us create comprehensive prompts for the models.

Standardized Prompt Formatting: We format the extracted components into a consistent template to maintain uniformity across all prompts. The template includes sections such as "Problem Statement", "Input", "Output", and "Sample Test Cases", ensuring that each prompt provides all the necessary information for the model to generate an accurate solution.

Text Cleaning and Normalization: We clean all textual data to remove extraneous characters, ensure proper encoding, and maintain clarity. This step includes correcting formatting inconsistencies, standardizing terminology, and ensuring that mathematical notations and symbols are appropriately represented.

Template Customization for Each Model: Recognizing the unique characteristics and requirements of each LLM, we slightly adjust the templates to optimize prompt comprehension. For example, we provide models fine-tuned for chain-of-thought reasoning, such as o1-mini and o1-preview, with additional guiding instructions to facilitate step-by-step solution generation.

Validation of Preprocessed Data: We manually review the preprocessed prompts to ensure that all relevant information is accurately captured and that the prompts adhere to the standardized format. This validation step helps us identify and correct any discrepancies or omissions.

C. Model Testing

We evaluate five state-of-the-art LLMs: GPT-4o, Mistral Large, Llama-3.1-405B, and the OpenAI o1 Family, which consists o1-mini and o1-preview. Each model represents a different architecture, training methodology, and reasoning capability. We apply all five models to our preprocessed data. We select these models for their diverse training approaches, which represent a mix of general-purpose and task-optimized LLMs. We test them in a pass@1 setting to assess their ability to generalize to ICPC-style problems [7], [8].

D. Solution Generation and Submission

We generate solutions to the scraped ICPC problems using the selected LLMs through their respective API keys. The generated solutions, which include code implementations aligned with the provided problem statements and constraints, are then submitted to the Codeforces Gym ICPC section contests [4]. Codeforces Gym provides automated feedback on correctness and efficiency, with verdicts such as "Accepted" (AC), "Wrong Answer" (WA), "Time Limit Exceeded" (TLE), "Runtime Error" (RE), and "Compile Error" (CE). Submissions to Codeforces Gym are logged for reproducibility. [23].

IV. EXPERIMENTAL EVALUATION

We use Selenium [24] for the automated downloading of PDFs from the website. For data pre-processing, we utilized PyPDF2 [25] to extract text and re (regular expressions) [26] for cleaning and formatting the extracted problem statements. We evaluate five large language models (LLMs): GPT-4o [12], Mistral Large [17], Llama-3.1-405B [9], and both o1-mini and o1-preview [19]. Solutions [27] generated using their respective APIs [28] are submitted to Codeforces Gym ICPC contests [4]. To evaluate LLM-ProS, we consider four research questions:

RQ1: How do LLMs perform on new ICPC problems compared to those potentially seen during training?

RQ2: What are the trends in accuracy across different categories?
RQ3: What factors contribute to variability in LLM performance?
RQ4: How are verdicts distributed across various LLMs for the given problem set?

A. Evaluation Metrics

The performance of each model was assessed using the following metrics:
Accuracy (pass@1): The percentage of problems solved correctly on the first attempt served as the primary metric of success [7], [8].
Error Analysis: Errors were categorized based on verdict types to identify systematic failures in each model's reasoning or implementation [12].
Resource Utilization: Average runtime and memory usage were measured by Codeforces to assess computational efficiency [19].
Temporal Comparison: Models were tested on two subsets of data:

- **Past ICPC Problems**: Problems likely included in the models' training data.
- **Unseen 2024 Problems**: Problems released after the models' training cutoff dates, ensuring evaluation on novel data [8], [12].

B. Results & Analysis

Evaluating large language models (LLMs) on ICPC problems reveals notable differences in accuracy, verdict distribution, and resource utilization across models. These findings offer detailed insights into our research questions, highlighting each model's strengths and limitations.
RQ1: LLMs Performance on Unseen ICPC Problems: Figure 2 presents a heatmap showing the accuracy trends of various LLMs across ICPC World Finals problems from 2011 to 2023 and the unseen 2024 dataset. The results highlight the stark contrast in performance between models optimized for reasoning and general-purpose models when handling challenging unseen problems.

The heatmap reveals that the o1-mini and o1-preview models demonstrate varied accuracy across the years, with both achieving their peak accuracy of 25.0% in 2017. On the unseen 2024 dataset, o1-mini and o1-preview achieved accuracies of 15.4% and 7.7%, respectively, indicating their ability to generalize to new problem sets effectively.

In contrast, GPT-4o, Mistral Large, and Llama-3.1 consistently show 0% accuracy across all years, including the unseen 2024 problems. These results emphasize their limitations in solving complex ICPC World Finals problems. This trend aligns with the hypothesis that general-purpose models lack the structured reasoning pathways necessary to excel in competition-level problem-solving without specific optimization.

Overall, the results confirm that models like o1-mini and o1-preview exhibit superior robustness and adaptability. Their consistent performance across years highlights the effectiveness of their fine-tuning and chain-of-thought reasoning capabilities. Conversely, the failure of other models to achieve any meaningful accuracy underscores the challenges faced by general-purpose LLMs in high-stakes competitive programming tasks.

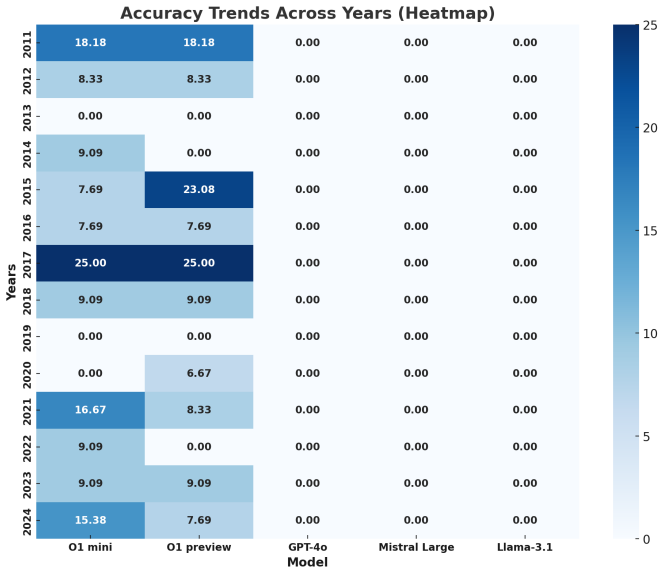

Fig. 2. Accuracy Trends Across Years (Heatmap)

RQ2: Accuracy Trends For Different Categories: The accuracy trends across problem categories provide insights into the strengths and limitations of the evaluated models. Categories like Implementation (5 problems), Graph Theory (3 problems) and Math (4 problems) [29] saw higher overall success rates, particularly for the o1-mini and o1-preview models. These categories test the models' ability to handle structured and logical problem-solving tasks.

In contrast, categories such as Geometry and Greedy, each with only 1 problem, presented challenges across all models, indicating potential difficulties in handling specialized or less frequent problem types. This disparity highlights the need for targeted optimization to improve performance in these areas.

The following chart (Figure 3) illustrates the number of solved problems across different categories:

The findings emphasize that problem category distribution plays a significant role in shaping model performance. The o1 models consistently demonstrated better adaptability across diverse categories compared to GPT-4o, Mistral Large, and Llama-3.1-405B, which struggled to generalize effectively. These trends align with the hypothesis that specialized training, such as chain-of-thought reasoning, enhances model capabilities in handling category-specific complexities.
RQ3: Variabilities in Model Performance: Three key factors—dataset contamination, training methodologies, and reasoning strategies—contribute to variations in model performance.

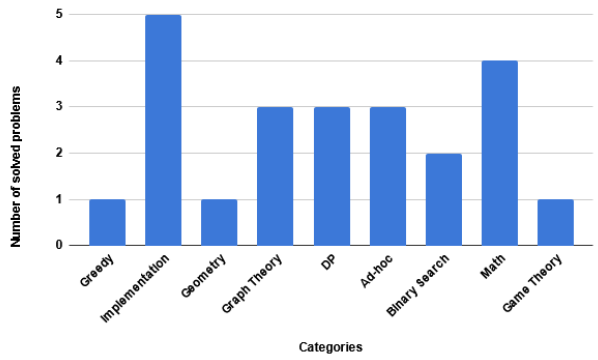

Fig. 3. Number of solved problems across categories.

1) Dataset Contamination: One major variability in LLM performance stems from dataset contamination. As shown in Figure 2, the accuracy of o1-mini and o1-preview on non-contaminated data (2024) is significantly lower than their highest accuracy on potentially contaminated data from earlier years. For example, o1-preview achieves 25.0% accuracy in 2017, whereas its accuracy on the unseen 2024 problems drops to 7.7%. Similarly, o1-mini's accuracy drops from 25.0% in 2017 to 15.4% in 2024. This disparity suggests that data contamination likely contributes to inflated performance metrics, as models may rely on memorized patterns rather than genuine reasoning for problems included in their training data. These results highlight the need for contamination-free datasets to accurately evaluate reasoning and generalization capabilities [7], [12].

2) Training Methodologies: The training methodologies adopted by different models significantly influence their performance variability. As highlighted in Figure 2 and Figure 4, o1-mini and o1-preview consistently outperform general-purpose models such as GPT-4o and Llama-3.1-405B. This success is attributed to their specialized fine-tuning, which emphasizes chain-of-thought (CoT) reasoning and iterative refinement.

Unlike general-purpose models, the o1-mini and o1-preview models are trained to simulate step-by-step problem-solving, enabling them to break down complex ICPC problems into manageable sub-tasks [30]. For example, the verdict distribution analysis in Table II shows that o1-mini achieves 16 Accepted (AC) solutions and 10 Compile Errors (CE), compared to GPT-4o, which recorded 0 ACs and 41 CEs. These results highlight how tailored training methodologies enhance the reasoning pathways of o1 models, directly translating to higher success rates on unseen, complex tasks [11], [19].

3) Reasoning Strategies: The reasoning strategies employed by LLMs also play a crucial role in their performance variability. Models in the o1 family leverage advanced CoT prompting, enabling logical, step-by-step reasoning that aligns with the structured problem-solving nature of ICPC challenges. This is evident in their ability to handle edge cases and adhere to strict computational constraints, as shown in Figure 4, where o1-mini achieves the highest percentage of "Accepted" (AC) verdicts at 9.64%.

In contrast, general-purpose models like GPT-4o and Llama-3.1-405B often lack CoT-specific training, resulting in a reliance on pattern recognition and memorized data. This limitation leads to higher error rates on unseen problems and significantly reduced adaptability, as reflected in their 0% accuracy across all years [12].

These findings reinforce the importance of specialized training methodologies and advanced reasoning strategies in achieving consistent and robust performance on competitive programming tasks. The combination of contamination-free datasets, tailored training, and effective reasoning strategies enables models like o1-mini and o1-preview to outperform general-purpose alternatives.

RQ4: Verdict Distribution Analysis: Verdict distributions highlight differences in error frequencies and success rates across models. As illustrated in Figure 4, o1-preview and o1-mini had the highest proportions of "Accepted" (AC) verdicts, with 9.64% for o1-mini and 9.04% for o1-preview. In contrast, models like GPT-4o showed predominantly error states such as "Compile Error" (CE) and "Wrong Answer" (WA). Table II shows that o1-mini and o1-preview achieved 16 and 15 Accepted (AC) solutions, respectively, while GPT-4o, Mistral Large, and Llama-3.1-405B recorded 0 ACs.

For error states, GPT-4o had the highest percentage of Compile Errors (CE) at 24.7%, followed by Llama-3.1 at 19.88% and Mistral Large at 18.07%. "Wrong Answer" (WA) verdicts were the most frequent error state across all models, with o1-mini recording 124 WA (74.70%) and GPT-4o recording 55 WA (33.25%).

These results emphasize the superiority of the o1 models, particularly their ability to achieve higher success rates and fewer Compile Errors compared to general-purpose models. However, the dominance of "Wrong Answer" verdicts across all models highlights the challenges faced in generating accurate solutions for competitive programming tasks. Figure 4 and Table II summarize these findings.

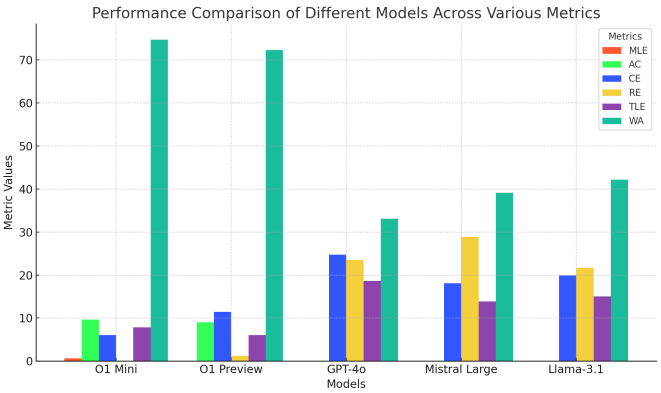

Fig. 4. Verdict Distribution Across Models

These findings are consistent with [19], which highlights the superior chain-of-thought reasoning capabilities of o1 models,

TABLE II
VERDICT DISTRIBUTION ACROSS MODELS (COUNTS)

Model	AC	CE	RE	TLE	WA	MLE
o1-mini	16	10	0	13	124	1
o1-preview	15	19	2	10	120	0
GPT-4o	0	41	39	31	55	0
Mistral-Large	0	30	48	23	65	0
Llama-3.1	0	33	36	25	70	0

enabling them to handle complex problem constraints more effectively.

C. Key Insights

The comprehensive evaluation highlights several key takeaways:

1) **o1 Models Superior Performance**: o1-mini and o1-preview consistently outperform other LLMs in accuracy, verdict distribution, and resource efficiency, demonstrating their advanced reasoning and calibration.
2) **Impact of Training Methodologies**: Models with specialized training for chain-of-thought reasoning exhibit greater robustness and adaptability to unseen problems.
3) **Necessity for Contamination-Free Benchmarks**: The significant performance drop in general-purpose models on unseen data underscores the importance of uncontaminated benchmarks to accurately assess model generalization [8], [11].
4) **Resource Efficiency**: o1 models not only achieve higher accuracy but also demonstrate superior computational efficiency, making them more suitable for resource-constrained environments.

These insights set the stage for future research in training methodologies and evaluation frameworks for LLMs in technical problem-solving.

V. THREATS TO VALIDITY

In this section, we discuss four types of threats, similar to prior research [31].

A. Threats to Internal Validity

Data contamination poses a major threat to our study's internal validity. Some of the ICPC problems we used might already exist in the evaluated LLMs' training data. We select problems from years (2011-2024) that likely do not overlap with the training cutoffs of popular models, but ensuring no overlap is challenging. If any problem appears in the training data, they could artificially inflate performance scores, especially for models exposed to similar problem statements [32]. Therefore, we evaluated the models on ICPC 2024's problems within 24 hours of the contest to ensure none of the models were trained on that data.

B. Threats to External Validity

The study focuses on ICPC World Finals problems, which may limit the generalizability of the findings. While ICPC problems are diverse and complex, they represent only a subset of the programming challenges faced in broader software engineering and real-world application development. As a result, the performance of LLMs on ICPC problems may not directly translate to other programming tasks, potentially limiting the applicability of our conclusions to different domains [7], [8], [22]. To mitigate this bias, we use a diverse set of 166 ICPC World Finals problems, covering various categories and difficulty levels, to strengthen the robustness of the evaluation.

C. Threats to Construct Validity

We test the models in a zero-shot setting without additional fine-tuning or iterative interactions. This approach may not fully leverage the models' capabilities, as fine-tuning or interactive prompting could improve performance on specific tasks. Additionally, differences in how each model interprets prompts or handles specific problem components may introduce variability not fully accounted for in our study. These constraints could limit the ability to generalize the findings to scenarios where models are fine-tuned or interactively guided during problem-solving [10], [17], [19]. To mitigate this bias, we document all experimental procedures and provide access to the scripts used for data preprocessing and solution evaluation to ensure reproducibility and transparency.

D. Threats to Conclusion Validity

Our evaluation relies exclusively on the Codeforces Gym platform for submitting solutions and determining verdicts. This creates a dependency on the platform's specific execution environment and judging criteria. Variations in compiler versions, runtime environments, or hidden test cases used by the platform can affect the consistency and reliability of the verdicts. Additionally, platform-specific nuances in error reporting or resource measurement may influence the evaluation of resource utilization metrics [11], [12]. To minimize this bias, we ensure that all submissions are made under the same conditions within Codeforces Gym, maintaining consistency in verdict determination and resource measurement.

VI. RELATED WORK

Advancements in large language models (LLMs) have spurred significant interest in evaluating their capabilities across various domains. In competitive programming, several studies emphasize the importance of rigorous benchmarks and highlight challenges such as reasoning ability, data contamination, and evaluation methodologies.

A. Competition-Level Problems as LLM Evaluators

Programming challenges from platforms like Codeforces and the International Collegiate Programming Contest (ICPC) offer unique evaluation benchmarks for LLMs due to their complexity and diversity. These problems require a deep understanding of algorithms, mathematics, and reasoning,

making them ideal for assessing LLM capabilities. Performance on unseen problems often drops significantly, indicating limitations in reasoning and generalization [7], [22]. These challenges underscore the need for reliable benchmarks and techniques to enhance reasoning in LLMs.

B. Evaluation Methodologies

Traditional methods for evaluating code generation have faced criticism due to issues like "context leakage" and "evolving-ignored" problems. Benchmarks such as those discussed in [8] better simulate real-world scenarios by considering the evolving nature of software development. These approaches reduce inflated performance metrics caused by unrealistic evaluation settings, providing a more accurate reflection of an LLM's problem-solving capabilities.

C. Code Refinement and Interactivity

Interactive code refinement and test-driven workflows have been shown to improve the quality of LLM-generated solutions. Iterative problem-solving techniques, combined with clear feedback, enhance LLM performance, particularly on complex programming challenges [10], [11].

D. Model-Specific Insights

Studies focusing on specific models, such as the o1 family, emphasize their advanced chain-of-thought reasoning and robust handling of competitive programming problems [19]. These models outperform others in minimizing hallucinations and achieving consistent performance across diverse tasks, further validating their efficacy for high-stakes evaluations. Similarly, Mistral 7B demonstrates optimized performance for efficiency, highlighting the importance of model design for resource-constrained environments [17].

E. Challenges in Code Evaluation

Issues like data contamination, overfitting, and reliance on pretraining data limit the generalization of LLMs. Research on the calibration and correctness of LLM-generated code highlights the importance of confidence scores and error analysis in ensuring reliable outputs, particularly in scenarios requiring high precision [33].

F. Performance Comparisons

Comparative studies of models like GPT-4o, Llama-3.1, and o1 systems reveal stark differences in accuracy and resource efficiency [9], [12]. These comparisons underscore the importance of both training data and architectural design in achieving superior results on competitive programming problems.

By synthesizing these findings, our work contributes to the growing body of research by evaluating multiple LLMs on ICPC-style problems. This expands on previous studies by combining problem-solving performance with detailed error and resource utilization analysis.

VII. Conclusion

This study underscores the efficacy of competition-level programming problems, specifically those from the International Collegiate Programming Contest (ICPC), as robust benchmarks for evaluating large language models (LLMs). Utilizing a curated dataset of 166 ICPC World Finals problems from 2011 to 2024, we systematically assessed state-of-the-art LLMs, including OpenAI's o1 family, GPT-4o, Mistral Large, and Llama-3.1-405B. Our findings reveal that the o1 models, particularly o1-mini and o1-preview, significantly outperform others in terms of accuracy, robustness, and computational efficiency, owing to their advanced chain-of-thought (CoT) reasoning and diverse training methodologies. In contrast, models such as GPT-4o, Mistral Large, and Llama-3.1-405B demonstrate limited generalization abilities and higher error rates on unseen 2024 problems, highlighting their reliance on pretraining data and the need for improved generalization skills [7], [33]. Additionally, the analysis of verdict distributions indicates that o1 models consistently achieve higher proportions of "Accepted" (AC) verdicts while minimizing errors like "Compile Error" (CE) and "Time Limit Exceeded" (TLE), in contrast to other models which show higher frequencies of these errors. These insights emphasize the importance of developing contamination-free benchmarks and enhancing reasoning capabilities through advanced training methodologies like CoT and iterative refinement [10], [19]. Overall, LLM-ProS contributes to a deeper understanding of LLM performance in real-world problem-solving scenarios and paves the way for further advancements in LLM design and evaluation, ensuring that they can meet the complex demands of technical problem-solving tasks.

References

[1] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, "A survey of large language models," https://arxiv.org/abs/2303.18223, mar 31 2023.

[2] K. R. Chowdhary, "Natural language processing," https://link.springer.com/chapter/10.1007/978-81-322-3972-7_19, jan 1 2020.

[3] C. et al., "Evaluating large language models trained on code," https://arxiv.org/abs/2107.03374, jul 7 2021.

[4] "The ICPC International Collegiate Programming Contest," https://icpc.global/worldfinals/past-problems.

[5] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, and C. Sutton, "Program synthesis with large language models," https://arxiv.org/abs/2108.07732, aug 16 2021, [Online; accessed 2025-01-26].

[6] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, and C. Zhang, "Quantifying memorization across neural language models," https://arxiv.org/abs/2202.07646, feb 15 2022.

[7] Y. Huang, Z. Lin, X. Liu, Y. Gong, S. Lu, F. Lei, Y. Liang, Y. Shen, C. Lin, N. Duan *et al.*, "Competition-level problems are effective llm evaluators," *arXiv preprint arXiv:2312.02143*, 2023.

[8] D. Zheng, Y. Wang, E. Shi, R. Zhang, Y. Ma, H. Zhang, and Z. Zheng, "Towards more realistic evaluation of LLM-based code generation: An experimental study and beyond," https://arxiv.org/abs/2406.06918, jun 11 2024.

[9] M. A. Research, "The llama 3 herd of models," https://arxiv.org/abs/2407.21783, jul 31 2024.

[10] S. Fakhoury, A. Naik, G. Sakkas, S. Chakraborty, and S. K. Lahiri, "Llm-Based test-driven interactive code generation: User study and empirical evaluation," https://arxiv.org/abs/2404.10100, apr 15 2024.

[11] Y. Zhang, R. Wang, X. Li, and J. Sun, "Grammar-aware large language models for code generation," https://arxiv.org/abs/2410.21276, oct 25 2024.

[12] T. Coignion, C. Quinton, and R. Rouvoy, "A Performance Study of LLM-Generated Code on Leetcode," https://doi.org/10.1145/3661167.3661221, jun 2024.

[13] O. Research, "Evaluation of OpenAI o1: Opportunities and Challenges of AGI," https://arxiv.org/abs/2409.18486, sep 27 2024.

[14] R. Islam and O. M. Moushi, "Gpt-4o: The cutting-edge advancement in multimodal llm," 07 2024.

[15] C. Munley, A. Jarmusch, and S. Chandrasekaran, "Llm4vv: Developing llm-driven testsuite for compiler validation," *Future Generation Computer Systems*, vol. 160, pp. 1–13, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X24002449

[16] A. Deroy and S. Maity, "Code generation and algorithmic problem solving using llama 3.1 405b," https://arxiv.org/abs/2409.19027, sep 26 2024.

[17] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," https://arxiv.org/abs/2310.06825, oct 10 2023.

[18] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating large language models trained on code," https://arxiv.org/abs/2107.03374, jul 7 2021.

[19] "Openai," https://openai.com/research/o1-system-card.

[20] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang, "Sparks of Artificial General Intelligence: Early experiments with GPT-4," https://arxiv.org/abs/2303.12712, mar 22 2023.

[21] "Extending the frontier of chatgpt: Code generation and debugging," https://ieeexplore.ieee.org/abstract/document/10698405.

[22] D.-A. et al., "Empirical evaluation of large language models in stem," https://arxiv.org/abs/2303.08774, mar 15 2023.

[23] "Gym," https://codeforces.com/gyms.

[24] "Automating functional tests using Selenium," https://ieeexplore.ieee.org/abstract/document/1667589.

[25] J. Li, P. Wang, L. Jia, R. Mao, Q. Li, Y. He, Y. Sun, and P. Zhao, "Design and implementation of an automated PDF drawing statistics tool based on Python," in *Sixth International Conference on Computer Information Science and Application Technology (CISAT 2023)*, H. Dong and S. Jia, Eds., vol. 12800, International Society for Optics and Photonics. SPIE, 2023, p. 128006R. [Online]. Available: https://doi.org/10.1117/12.3003927

[26] L. Karttunen, J.-P. Chanod, G. Grefenstette, and A. Schille, "Regular expressions for language engineering," *Natural Language Engineering*, vol. 2, no. 4, pp. 305–328, 1996.

[27] sifat-hossain-niloy, anika-tabassum, "Github - Sifat-hossain-niloy/LLMs-Performance-in-ICPC-Problems," https://github.com/sifat-hossain-niloy/LLMs-Performance-in-ICPC-Problems.git, [Online; accessed 2025-01-24].

[28] "Openai API," https://openai.com/index/openai-api.

[29] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," https://arxiv.org/abs/2110.14168, oct 27 2021.

[30] A. Patel, S. Bhattamishra, and N. Goyal, "Are NLP Models really able to Solve Simple Math Word Problems?" https://arxiv.org/abs/2103.07191, mar 12 2021.

[31] S. R. Pyreddy and T. S. Zaman, "Emoxpt: Analyzing emotional variances in human comments and llm-generated responses," *arXiv preprint arXiv:2501.06597*, 2025.

[32] Y. Xu, H. Su, C. Xing, B. Mi, Q. Liu, W. Shi, B. Hui, F. Zhou, Y. Liu, T. Xie, Z. Cheng, S. Zhao, L. Kong, B. Wang, C. Xiong, and T. Yu, "Lemur: Harmonizing natural language and code for language agents," https://arxiv.org/abs/2310.06830, oct 10 2023.

[33] C. Spiess, D. Gros, K. S. Pai, M. Pradel, M. R. I. Rabin, A. Alipour, S. Jha, P. Devanbu, and T. Ahmed, "Calibration and correctness of language models for code," https://arxiv.org/abs/2402.02047, feb 3 2024.