Annals of the University of Craiova, Mathematics and Computer Science Series
Volume 53(1), 2026, Pages 310–330, DOI: 10.52846/ami.v53i1.2394
ISSN: 1223-6934

Evaluating the Problem-Solving Capabilities of Modern Large Language Models in Competitive Programming

Mirel Coşulschi, Mihai Gabroveanu, Alexandra Vultureanu-Albişi, and Florin Slabu

Abstract. This paper investigates performance on a dataset of 40 competitive programming tasks designed for formal evaluation in educational settings. The primary objective is to analyze how the performance of problem-solving varies between tasks with differing structure, difficulty, and source, using partial acceptance automated test-based scoring. By examining aggregate performance metrics, task-level difficulty, acceptance rates, failure patterns, and discriminative properties, the study characterizes which types of problem are consistently solvable, which remain challenging, and which most effectively differentiate levels of problem-solving capability. The results reveal substantial variation across tasks, highlighting the influence of task difficulty and structure on acceptance outcomes and demonstrating that medium-difficulty problems tend to provide the strongest discrimination. These findings contribute empirical insight into the characteristics of formally designed Olympiad-style programming problems and their suitability for structured performance assessment in educational and competitive programming contexts.

2020 *Mathematics Subject Classification.* Primary 60J05; Secondary 60J20.
Key words and phrases. Competitive programming, Coding tasks, Large language models (LLMs), ChatGPT, Gemini, DeepSeek, Grok, LLM Evaluation.

1. Introduction

The rapid and accelerated development of generative artificial intelligence (AI) tools in recent years has sparked intense debates within the IT community and beyond. Models such as ChatGPT, Gemini and DeepSeek have demonstrated remarkable results in solving tasks such as natural language generation, writing code, or solving domain-specific problems. However, their emergence raises critical questions for both educators and practitioners.

The identified challenges discussed in this paper motivate the following research questions:

RQ1: To what extent are large language models (LLMs) useful in the educational process, particularly for learning and practicing programming concepts?

RQ2: Can such systems assume instructional roles traditionally performed by human educators, and if so, how might this influence the teacher–student dynamic?

RQ3: How reliably can current systems solve programming problems of above-average complexity without human intervention?

RQ4: To what extent can these systems operate as autonomous programmers, independently performing tasks that require advanced reasoning and adaptability?

Received January 11, 2026. Accepted May 25, 2026.

This work addresses these questions by evaluating the performance of several state-of-the-art generative AI models on programming problems of varying difficulty in C and C++. The results highlight clear differences between models and provide an empirical basis for discussing their potential roles in software education and professional development.

Competitive programming problems are a specialized area within the broader category of coding tasks. These problems require a deep understanding and mastery of fundamental algorithms and algorithm development techniques, a deep understanding of data structures, and problem-solving techniques. From the beginning, the research direction was determined by the possibility of an LLM to solve competitive programming problems, leading to the integration of reasoning, coding, and problem-solving models [12].

This work differs from existing research because it focuses on a selection of 40 problems and evaluates the performance of recent LLMs released in late 2024 and 2025. The models included are DeepSeek V3 (December 2024), DeepSeek-R1 (January 2025), Grok 3 (February 2025), ChatGPT o3 (April 2025), ChatGPT o4 (April 2025), Gemini 2.5 Flash (June 2025), DeepSeek V3.1 (August 2025), ChatGPT 5 (August 2025) and Grok 4 (September 2025). Including them is crucial for empirically estimating task difficulty, because in this study difficulty is inferred from aggregate observed performance rather than from predefined complexity categories. By spanning several model families and versions, the evaluation provides a more realistic estimate of how challenging each task is in practice, while reducing the impact of behavior tied to specific architectures or releases. All major competitors have made efforts over the last year toward improving their models' performances for competitive programming: ChatGPT o3 [7], Gemini 2.5 flash, and DeepSeek-R1.

While many studies offer valuable insights into LLM strengths and weaknesses on international platforms, our study focuses on national-level informatics Olympiad problems such as those used in Romanian competitions.

This work extends our previous study [3], which evaluated six top-tier language models on 40 competitive programming tasks spanning grades 7-10. In contrast to the previous analysis, the present study significantly broadens both the set of models and the evaluation methodology. Specifically, we expand the evaluation from six to nine LLMs and introduce a more comprehensive analytical framework that includes formal problem formulation, empirical task difficulty estimation, threshold-based acceptance analysis, robustness and failure-rate assessment, model correlation analysis, task-level dominance evaluation, and comparative visualization across tasks and models. By incorporating these additional dimensions, the extended study provides a deeper and more nuanced assessment of LLM performance on educationally representative, formally designed national Olympiad problems, thereby strengthening and generalizing the conclusions of the original work.

2. Related work

The work by Li et al. [13] explored methods for explaining competitive programming solutions generated by LLMs, highlighting the potential of model-generated explanations to enhance transparency and support learning in programming education.

Tran et al. [24] examined the ability of ChatGPT to solve programming problems with complex contextual requirements, showing that task complexity and contextual dependencies significantly affect the model's performance.

Huang et al. [12] evaluates LLMs' reasoning capacities by solving a number of competitive programming problems on Codeforces[1] [16]. One conclusion revealed a decrease in the performance of the GPT-4 model for problems evaluated at the "first glance". Various prompts or fine-tuning failed to mitigate the effect of dropping performances. The paper concludes that competition-level problems are effective evaluators of LLMs' genuine reasoning abilities.

In [21], the authors introduce CodeElo, a standardized benchmark submitting model solutions directly to Codeforces. The paper evaluates the ratings of 30 existing popular open-source and 3 proprietary LLMs.

Souza et al [23] evaluate five general-purpose Small Language Models (LLama 3.2 3B, Gemma 2 9B, Gemma 3 12B, DeepSeek-R1 14B and PHI-4 14B) on 280 problems from Codeforces. All models were asked to generate Python code. A Small Language Model (SLM) is a light-weight and more efficient alternative for code generation to a LLM.

A study by Dunder et al. [5] focused on evaluating ChatGPT on 127 randomly selected problems from Kattis. The study showed that the model solved only 19 problems correctly, doing well on simple tasks but struggling on more complex ones, highlighting the limits of unsupervised programming in educational settings.

In the study [22], the authors focus on 29 programming problems selected from the Codeforces platform. The generated solutions to the problems were created by ChatGPT-o3-mini and DeepSeek-R1.

Coignion et al [2] studies the performance of code generated by 18 LLMs and evaluates the performance differences of the generated code between these models on problems on the Leetcode platform[2]. The LLMs under evaluation were open-source variants of CodeGen, CodeLlama, Incoder, replit-code, WizardCoder, Santacoder, StarCoder, Codeparrot and GitHub Copilot.

OJBench [27] presents a benchmark tailored to assess LLM performance on competition-grade programming problems sourced from online judge platforms. It comprises tasks that demand algorithmic reasoning, precise treatment of constraints, and end-to-end program implementations capable of passing extensive test suites. The work benchmarks multiple state-of-the-art language models and examines how well they can produce solutions across a spectrum of problem difficulties, underscoring the persistent gap between current LLMs abilities and the demands of advanced algorithmic problem solving.

A recent work, [26], introduce LiveCodeBench Pro[3], a benchmark composed of problems from Codeforces, ICPC and IOI that are continuously updated to reduce the likelihood of data contamination and annotated by Olympiad medalists for algorithmic complexity.

In the paper [4], Dumitran et al. explore the performance of different LLMs in solving problems from the Romanian Olympiad in Informatics at OJI. The research evaluates both closed-source models, such as GPT-4, and open-source models, such

[1] https://codeforces.com/

[2] https://leetcode.com/

[3] https://livecodebenchpro.com/

as CodeLlama and RoMistral, on a dataset of competitive programming challenges covering the period $2002 - 2023$.

Studies indicate that LLMs solve a significant portion of basic programming tasks, but performance degrades for algorithmically demanding problems that require compositional reasoning, strict constraints or careful data structure design. Results vary greatly depending on the model and benchmark, *flagship* models tending to outperform smaller models. It is important to mention the significant progress made by the models launched in the last three years since the first official launch of ChatGPT (November 2022): in September 2025, a version of Gemini 2.5 Deep Think solved 10 of the 12 proposed problems at the ICPC World Finals[4].

3. Background

OpenAI is known for its LLMs, such as the GPT series or text-to-image models like DALL-E.

Among the models developed by OpenAI [17], ChatGPT 5 represents a more recent generation of LLMs designed to improve reasoning, code generation, and complex task handling. The model builds on previous architectures by incorporating enhanced reasoning capabilities and improved performance in a wide range of programming and analytical tasks [20]. Compared with earlier versions, ChatGPT 5 demonstrates stronger reliability in structured problem solving and code synthesis, which makes it particularly relevant for evaluating performance in challenges of algorithmic programming and tasks that require advanced logical reasoning.

ChatGPT 4o is a multimodal model that can process and generate any combination of text, audio, image, and video in real time, with response times similar to human conversations. It represents an improvement over previous models, offering improved performance in languages other than English [18].

ChatGPT o3 is a newer and more powerful reasoning model. Launched as a successor to previous models, it is designed to tackle complex problems by performing a series of internal reasoning steps. Unlike models that primarily predict the next word, o3 *thinks* before generating an answer, using tools like web search and code interpretation to verify facts and solve technical problems. This makes it particularly effective for tasks that require precision in fields such as coding, math, and science [19].

Google's Gemini 2.5 Flash is a multimodal LLM designed for fast responses at a low cost while providing solid reasoning capability. It is the first *Flash* model to feature built-in *thinking* capabilities, allowing it to go through its reasoning process before generating an answer. This feature improves accuracy and contextual understanding [8]. The model is suitable for high-volume, low-latency tasks such as chat applications, data mining, and summarization. It accepts a wide range of inputs, including text, images, video, audio, and PDF documents.

xAI is an AI company founded by Elon Musk in March 2023, developing AI products and models, including the Grok chatbot. Grok 3 launches in February 2025. It is a multimodal model featuring enhanced reasoning abilities and has been trained using more computational resources than its predecessor, Grok 2. The model is designed as a strong competitor to other leading AI models, with xAI claiming to outperform them

[4] https://deepmind.google/discover/blog/gemini-achieves-gold-level-performance-at-the$\$-international-collegiate-programming-contest-world-finals/

on several performance tests for math, science and programming [10]. Grok 4, released by xAI in September 2025, represents a further development of the Grok model series. Building on the capabilities introduced in Grok 3, the model aims to improve reasoning accuracy, code generation performance, and robustness in complex analytical tasks. According to xAI, the model achieves improved results on benchmarks related to mathematical reasoning, scientific problem solving, and programming tasks, positioning it as a direct competitor to other state-of-the-art LLMs [9].

DeepSeek V3 is a LLM with a Mixture-of-Experts (MoE) architecture, developed by the Chinese AI company DeepSeek. The model has a total of 671 billion parameters and activates only 37 billion for each token. DeepSeek V3 is freely available through a chat interface, a public API and as open-source model checkpoints on GitHub and Hugging Face [14].

DeepSeek V3.1 represents an updated version of the DeepSeek V3 model. The model introduces improvements in reasoning performance, code generation accuracy, and overall robustness when handling complex analytical tasks. Compared to its predecessor, DeepSeek V3.1 benefits from refinements in training methodology and expanded training data, enabling more reliable performance in a variety of domains, including mathematical reasoning and programming problems [15].

DeepSeek R1 is another model from DeepSeek that has advanced reasoning capabilities and was designed with a reinforcement-learning pipeline. It plays an important role in training and strengthening other models within the DeepSeek ecosystem [11].

The Informatics Olympiad in Romania has a history dating back to the late 1970s. The competition is structured in several stages: school, city, county and, finally, the national phase. The County Olympiad in Informatics (Olimpiada Judeţeană de Informatică - OJI) is the last qualification stage before the national phase of the informatics competition: only a select number of students with best performances from each county qualify for the National Olympiad in Informatics (Olimpiada Naţională de Informatică - ONI). Participants in the county stage are organized into eight distinct classes ($5 - 12$). Approximately 80 students from each class qualify for the national phase, with the best competitor from each county automatically advancing to the final, while the other competitors being selected based on results from a national ranking.

In these last stages, students solve a set of algorithmic problems within a limited time frame (e.g., $3-5$ hours), writing their solutions in C++ and submitting them to an online judge for automatic evaluation. Participating students must demonstrate a good knowledge of algorithms, data structures and programming techniques.

The scoring system for tasks in the Romanian Olympiads of Informatics phases OJI and ONI is based on a partial scoring model. Instead of receiving a simple "correct" or "incorrect" answer for a task (as in the case of ICPC competitions), each problem will be evaluated for a maximum number of points (usually 100 points). These points are then distributed according to the success of the proposed solution in different test cases.

PBInfo[5] is a popular Romanian online platform dedicated to teaching and practicing computer science and algorithms, mainly for high school students. The website was launched in 2010 and contains over 4.800 algorithmic problems organized by high school grade, topic and level of difficulty, plus problems given at national olympiads

[5] https://www.pbinfo.ro/

and competitions. There is an automatic evaluator that provides feedback on submitted solutions. It supports multiple programming languages, including C, C++, Pascal, Python, Java and C#, making it a comprehensive tool for students and educators in the Romanian competitive programming community.

The website is considered the main training platform for high school students learning programming or preparing for the Romanian Baccalaureate in Informatics, offering past exam papers and practical exercises. Also, it is used by students as a means of preparation for the Informatics Olympiad, as well as support for university courses and programming clubs.

A problem can have one of the following labels reflecting the degree of difficulty: *easy*, *medium*, *hard* and *contest-type* problem. The message resulted from the evaluation of a program submitted as a solution to a coding task can have one of the following values: *Compilation error (CE)*, *OK*, *Wrong answer (WA)*, *Time limit exceeded (TLE)*, *Memory exceeded*, *Killed by signal N / Caught fatal signal N / Stopped by signal N*, *Nonzero exit status N / Exited with error status N*[6].

4. Methodology

The models for this study were selected based on their degree of novelty, as well as on various rankings. From OpenAI, we evaluated GPT-5, 4o and o3, and from Google, we evaluated Gemini 2.5 Flash. We also selected Grok 4 and Grok 3 from xAI, as well as the DeepSeek V3, V31 and DeepThink R1 models. In this paper, these models will be referred to as ChatGPT-5, ChatGPT-4o, ChatGPT-o3, Gemini-2.5-flash, Grok-4, Grok-3, DeepSeek-V3, DeepSeek-V31 and DeepThink-R1.

4.1. Study settings. Each model was presented with the same prompt template with the following structure:

```
Write a complete C/C++ program that solves the following problem.
Ensure proper handling of edge cases. Problem description:
[....]
```

Because sometimes the generated code contained technical details characteristics to C++ 17 or 14, for the source code of generated solution sent to the platform we obtained the "Compile Error" message. In these situations, we have used a slightly modified version of the previous prompt template, in order to obtain a compilable code:

```
Write a complete C++ 11 program that solves the following problem.
Ensure proper handling of edge cases. Problem description:
[....]
```

The experimental approach relied on the single-shot nature of the responses generated by the LLMs. Primarily, no additional prompts or iterative refinement steps were used. However, in about 15% of cases, the code generated on the first attempt, whether in C or C++, did not meet the PBInfo platform requirements for the current compiler[7]. In these cases, we made another attempt by asking the LLM to generate

[6] where `signal N` is the POSIX/Unix signal whose numeric code is N

[7] The information regarding the compile line for C source code is
`gcc -Wall -O2 -static source_code_file.c -lm`,
while the compile line for C++ source code is
`g++ -std=c++11 -Wall -O2 -static source_code_file.cpp -lm`.

code that met the specified requirements. If the response was still not acceptable, we did not send any further requests. This explains why in a few cases, the generated source code resulted in a "Compiler Error" message.

As a result, any inaccuracies or incomplete solutions in the initial output were not corrected. While this design choice ensures that the results reflect the models' immediate problem-solving capabilities, it also introduces potential inefficiencies. In practical scenarios, LLM-assisted programming often involves multiple interactions in which the user guides the model toward a correct or optimized solution.

Yeh et al. [25] investigated how interactive features can bridge the gap between novice and experienced programmers. They demonstrated that guided interactivity can improve the quality of solutions and learner engagement.

4.2. Coding tasks. The 40 coding tasks were randomly selected from PBInfo in June 2025. Before starting to evaluate the LLMs on the selected problems, we manually solved most of them. This will provide the evaluation team with a deeper understanding of the approaches and common mistakes of the solutions proposed by the LLMs.

The selected problems are part of the *contest-type* category. More specifically, the selected problems are part of the problems given at the county (OJI) or national (ONI) stage of the Romanian National Olympiad in Informatics, for grades $7 - 10$.

The main reason for selecting problems mainly from the 7th to 9th-grade level is that the competition curriculum for those grades includes simple data structures like arrays and matrices, fundamental algorithms such as sorting and searching, and basic algorithm design methods like greedy programming or dynamic programming. The curriculum at this level does not require knowledge of advanced data structures or advanced algorithms (e.g. specific graph algorithms).

From this point of view, the study was designed to evaluate how LLMs generate creative solutions with a limited set of data structures, excluding advanced ones such as balanced search trees, trie trees or suffix arrays.

The majority of LLMs, including ChatGPT, Gemini, Grok and DeepSeek, tend to favor the use of Python as the suggested programming language for solving computer science problems. Their answers often implicitly use Python unless explicitly stated otherwise.

Despite this, we used C++ as the programming language for the problems solutions. The majority of competitive programming competitions, including ACM ICPC and IOI, prefer the use of C++ over Python.

4.3. Analysis Framework and Data Assumptions. Let $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$ denote a finite set of competitive programming tasks, where $|\mathcal{T}| = N$ and each task is defined by a problem statement, a set of input–output specifications, and an official test suite used for automated evaluation. Let $\mathcal{M} = \{m_1, m_2, \ldots, m_M\}$ denote a set of LLMs evaluated on these tasks, where $|\mathcal{M}| = M$.

For each task $t_i \in \mathcal{T}$ and model $m_j \in \mathcal{M}$, the model is required to generate a complete C/C++ solution that satisfies the problem constraints and is evaluated against the corresponding test cases. The performance of model m_j on task t_i is quantified by a score

$$x_{i,j} \in [0, 100],$$

representing the proportion of test cases accepted by the evaluation system for each generated solution.

The evaluation results are represented by the score matrix

$$X = [x_{ij}] \in \mathbb{R}^{N \times M}, \tag{1}$$

where x_{ij} denotes the score obtained by model m_j on task t_i. Based on this matrix, the analysis aims to (i) compare the overall performance of individual models across the task set, (ii) assess model robustness and reliability in the presence of partial or complete failures, and (iii) characterize task properties such as empirical difficulty and discriminative capacity using aggregated model outcomes.

The overall performance of the model μ_j is summarized using the arithmetic mean score computed across the tasks:

$$\mu_j = \frac{1}{N} \sum_{i=1}^{N} x_{ij}, \qquad j = \overline{1, M}. \tag{2}$$

This metric provides a first-order comparison of the average effectiveness of the models. Beyond mean performance, the distribution of scores obtained by each model across tasks is analyzed using quartile-based summaries. These distributions characterize variability, stability, and the presence of extreme outcomes and allow for comparison of models with similar averages but differing consistency.

Task difficulty d_i, is defined empirically based on observed performance rather than theoretical complexity. For each task t_i, difficulty is quantified as the average score across all evaluated models:

$$d_i = \frac{1}{M} \sum_{j=1}^{M} x_{ij}, \qquad i = \overline{1, N}. \tag{3}$$

The distribution of d_i is analyzed to characterize the benchmark in terms of relative task difficulty.

To assess task discrimination, each model m_j is assigned an overall ability rank based on its mean performance on all tasks. The discriminative power of task t_i is then quantified using the Spearman rank correlation between the vector of model ability ranks and the task-specific score vector [1]:

$$\delta_i = \rho_S(\mathrm{rank}(\boldsymbol{\mu}), \mathbf{x}_i)\,, \tag{4}$$

where

$$\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_M],$$
$$\mathbf{x}_{i\cdot} = [x_{i1}, x_{i2}, \ldots, x_{iM}].$$

Higher values of δ_i indicate that the task t_i consistently assigns higher scores to stronger models and lower scores to weaker ones, while values close to zero indicate limited discriminative capacity.

To evaluate practical task-solving capability, a threshold-based acceptance criterion is introduced.

For a fixed acceptance threshold score $\tau = 80$, an acceptance indicator is defined such that task t_i is considered solved by model m_j whenever

$$\mathbb{I}_{ij}(\tau) = \begin{cases} 1, & \text{if } x_{ij} \geq \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The number of solved tasks for each model m_j is then computed as:

$$S_j(\tau) = \sum_{i=1}^{N} \mathbb{I}_{ij}(\tau) \tag{6}$$

This metric provides an acceptance-oriented measure of model performance.

The robustness of a model is assessed by the rate of complete failures. A failure is defined as a zero-score outcome. To maintain consistency with the indicator notation introduced above, the failure rate of model m_j is computed as:

$$F_j = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_0(x_{ij}) \tag{7}$$

where

$$\mathbb{I}_0(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{otherwise.} \end{cases}$$

This metric quantifies the proportion of catastrophic failures, corresponding to tasks for which the generated solution receives a score of zero.

The similarity in model behavior is quantified using Pearson correlation coefficients between model-specific score vectors defined as:

$$\rho_{jk} = \frac{\sum_{i=1}^{N}(x_{ij} - \mu_j)(x_{ik} - \mu_k)}{\sqrt{\sum_{i=1}^{N}(x_{ij} - \mu_j)^2}\sqrt{\sum_{i=1}^{N}(x_{ik} - \mu_k)^2}}. \tag{8}$$

where $j, k \in \{1, \ldots, M\}$ denote model indices. High correlation indicates similar success and failure patterns across tasks, while low correlation suggests complementary or divergent behavior.

The score matrix X is visualized as a task–model heatmap to facilitate comparative analysis. The tasks are ordered according to their empirical difficulty d_i, allowing visual inspection of performance trends from harder to easier tasks and highlighting systematic differences between models.

Task-level dominance is evaluated by identifying, for each task, the model achieving the highest score:

$$w_i = \arg \max_{j \in \{1,\ldots,M\}} x_{ij}. \tag{9}$$

where w_i denotes the index of the model obtaining the maximum score on task t_i. Thus, for each task, the analysis identifies the best-performing model among all evaluated LLMs. Aggregating these outcomes yields the total number of task wins obtained by each model, providing a dominance-based comparison that is independent of the magnitude of the score:

$$W_j = \sum_{i=1}^{N} \mathbb{I}_j(w_i), \tag{10}$$

where W_j denotes the number of tasks for which model m_j achieves the highest score among all evaluated models, and

$$\mathbb{I}_j(w_i) = \begin{cases} 1, & \text{if model } m_j \text{ achieves the highest score on task } t_i, \\ 0, & \text{otherwise.} \end{cases}$$

5. Results and Data analysis

5.1. Model Performance Evaluation. Table 1 presents the summary of the results for the C++ solutions of the 40 evaluated problems. The label `#Solved` represents the number of problems for which a model's solution successfully obtained a maximum value of 100. The `Avg Score` for a specific model is calculated as the average of the scores it received on all 40 coding tasks.

The *average score for a particular problem* is computed by taking the average of the scores from the solutions generated by all nine models under consideration.

TABLE 1. Overview of the number of accepted solutions and the average score for each model across all 40 selected programming tasks in C++

Model	C++ Solutions	
	`#Solved Tasks`	`Mean Score`
ChatGPT-5	33	93.90
ChatGPT-4o	7	47.85
ChatGPT-o3	22	74.95
DeepSeek-V3	0	21.30
DeepSeek-V31	10	51.85
DeepThink-R1	2	34.80
Gemini-2.5-flash	16	72.68
Grok-4	21	81.83
Grok-3	1	25.53

For the 40 programming tasks evaluated, model performance showed substantial variation in both the number of problems solved and the average scores (see Table 2). *ChatGPT-5* obtained the highest number of solved problems, with 33 fully solved tasks out of the 40 evaluated problems. Among the remaining models, *ChatGPT-o3* solved 22 tasks and *Grok-4* solved 21, followed by *Gemini-2.5-flash* with 16 solved problems and *DeepSeek-V31* with 10. In contrast, the models with the fewest solved problems are *DeepSeek-V3* (0), *DeepThink-R1* (2) and *Grok-3* (1).

The average scores of the models range from 21.30 (*DeepSeek-V3*) to 93.90 (*ChatGPT-5*). *ChatGPT-5* achieved the highest average score (93.90), followed by *Grok-4* (81.83), *ChatGPT-o3* (74.95), and *Gemini-2.5-flash* (72.68). Moderate average scores were obtained by *DeepSeek-V31* (51.85) and *ChatGPT-4o* (47.85), while *DeepThink-R1* (34.80), *Grok-3* (25.53), and *DeepSeek-V3* (21.30) recorded substantially lower averages.

Overall, the distribution of average scores reveals a substantial performance gap between the strongest and weakest models, with a difference of over 72 points between the highest and lowest averages. The consistency of problem solving also varied: models with high scores tended to maintain stable results on most tasks, while models with lower scores frequently produced zero or near-zero results on more difficult problems.

The results suggest that advanced LLMs can perform very well on a wide range of competitive programming problems. The descriptive statistics reveal a clear stratification of performance among the evaluated models. The mean score across all models is 56.08, which remains substantially lower than the maximum average score of 93.90 achieved by *ChatGPT-5*.

TABLE 2. Performance comparison across all 40 selected programming tasks. Scores represent the percentage of accepted solutions obtained by each model.

No	Problem	Source	ChatGPT 4o	ChatGPT o3	ChatGPT 5	Gemini 2.5 flash	Grok 3	Grok 4	DeepSeek V3	DeepSeek V31	DeepThink R1	Avg. Score
1	Trafalet	OJI 2024, 9th gr.	97	100	100	71	0	88	54	100	91	77.89
2	Macarie	OJI 2024, 9th gr.	100	100	100	100	0	100	75	100	75	83.33
3	Santinele	OJI 2024, 9th gr.	98	100	100	100	20	100	34	100	62	79.33
4	Cufar	OJI 2018, 9th gr.	100	100	100	100	15	58	54	100	71	77.56
5	Magicl	ONI 2015, 8th gr.	46	100	100	100	47	49	0	100	47	65.44
6	Restaurare	ONI 2015, 8th gr.	0	100	100	100	40	5	0	0	0	38.33
7	Antidivizor	ONI 2025, 9th gr.	0	80	100	100	0	0	0	54	80	46.00
8	Poseidon	OJI 2024, 10th gr.	0	100	100	95	100	100	45	100	100	82.22
9	Kmajo	ONI 2024, 8th gr.	27	0	100	90	34	50	13	90	27	47.89
10	Teleportor	OJI 2025, 9th gr.	18	7	100	100	0	100	0	52	0	41.89
11	Bile	ONI 2021, 8th gr.	52	88	84	52	52	92	24	20	52	57.33
12	Tema	OJI 2023, 8th gr.	85	0	100	60	57	100	61	60	0	58.11
13	Hibrid	OJI 2023, 8th gr.	90	0	100	100	60	100	35	60	55	66.67
14	Parking	OJI 2024, 7th gr.	0	100	100	68	0	84	48	32	0	48.00
15	Patratele	OJI 2022, 7th gr.	68	98	100	61	0	71	30	61	0	54.33
16	Eras	ONI 2024, 9th gr.	40	90	100	90	0	90	0	70	0	53.33
17	Spirala3	ONI 2024, 9th gr.	0	0	100	0	0	70	0	0	0	18.89
18	Joc	OJI 2025, 8th gr.	40	0	100	100	40	100	0	21	0	44.56
19	Teren	OJI 2025, 7th gr.	85	85	80	80	15	85	15	40	80	62.78
20	Mandms	ONI 2024, 8th gr.	30	0	42	25	30	25	0	0	0	16.89
21	Castel	ONI 2023, 8th gr.	4	100	100	100	44	100	32	59	36	63.89
22	Nisip	ONI 2023, 9th gr.	0	0	0	0	0	100	0	0	0	11.11
23	Partitura	ONI 2023, 9th gr.	55	100	100	100	20	100	20	5	100	66.67
24	Strips	OJI 2022, 8th gr.	0	100	100	0	24	100	0	40	40	44.89
25	Mun	OJI 2024, 8th gr.	70	85	100	40	40	100	70	85	40	70.00
26	Vemus	ONI 2019, 8th gr.	100	100	100	30	0	100	15	10	10	51.67
27	Succes	ONI 2025, playoff	30	95	78	30	30	68	30	30	33	47.11
28	Vnoroc	ONI 2025, playoff	48	100	84	80	32	98	44	42	0	58.67
29	Carita	ONI 2021, playoff	84	88	100	84	0	100	24	84	0	62.67
30	Fuziune	ONI 2023, playoff	40	82	88	46	40	76	28	0	40	48.89
31	Tombola	ONI 2019, playoff	25	100	100	60	0	100	0	30	35	50.00
32	UDP	ONI 2023, playoff	21	0	100	61	10	100	0	64	14	41.11
33	Bug	ONI 2022, playoff	17	100	100	17	21	24	0	13	13	33.89
34	Inno	ONI 2021, playoff	0	100	100	100	0	100	0	32	0	48.00
35	Intergalactic	ONI 2021, playoff	100	100	100	100	22	67	26	100	58	74.78
36	Criptografie	ONI 2019, 8th gr.	100	100	100	100	76	100	65	100	71	90.22
37	Lumini	ONI 2019, 8th gr.	44	100	100	100	0	73	0	100	70	65.22
38	Pro3	ONI 2019, 9th gr.	100	100	100	72	72	100	0	20	92	72.89
39	Fibofrac	ONI 2019, 9th gr.	0	100	100	100	0	100	0	0	0	44.44
40	Gene	ONI 2018, 8th gr.	100	100	100	95	80	100	10	100	0	76.11

This indicates that only a small subset of models consistently produces high-quality solutions across the evaluated tasks. The median score (51.85) is slightly lower than

the mean, suggesting a mildly left-skewed distribution in which a few strong models raise the overall average, while several models obtain considerably lower performance levels. Overall, the results highlight a clear separation between high-performing models and those that struggle to produce reliable solutions.

We selected the top 20 problems based on the average scores from the solutions generated by the nine LLMs. We then asked the same nine LLMs to create solutions for these problems, but this time in C. We instructed the models to provide pure C source code, and we have since evaluated these new solutions.

Table 3 presents a comparative analysis of the results obtained by the evaluated models when solving the top 20 coding tasks in both C and C++. For most models, the number of fully solved problems remains relatively similar across the two programming languages. For example, *Gemini-2.5-flash* solved 13 problems with the maximum score in C and 10 in C++, while *Grok-4* solved 12 problems in both languages. Similarly, *ChatGPT-o3* solved 16 tasks in C and 14 in C++, and *ChatGPT-5* solved 18 tasks in C and 17 in C++.

TABLE 3. Overview of the number of accepted solutions and the average score for each model for top 20 selected programming tasks in both C and C++.

Model	C++ Solutions		C Solutions	
	#Solved Tasks	Mean Score	#Solved Tasks	Mean Score
ChatGPT-5	17	97.40	18	98.25
ChatGPT-4o	6	72.90	5	57.70
ChatGPT-o3	14	87.30	16	92.90
DeepSeek-V3	0	34.40	0	24.25
DeepSeek-V31	10	73.75	6	68.75
DeepThink-R1	1	55.00	3	51.30
Gemini-2.5-flash	10	86.45	13	90.00
Grok-3	1	37.60	0	34.75
Grok-4	12	90.50	12	88.37

In terms of average performance, most models achieved slightly higher scores when generating solutions in C compared with C++. For instance, *ChatGPT-5* obtained an average score of 98.25 in C compared with 97.40 in C++, while *ChatGPT-o3* achieved 92.90 in C and 87.30 in C++. Similarly, *Gemini-2.5-flash* recorded average scores of 90.00 in C and 86.45 in C++. An exception is *ChatGPT-4o*, which performed better in C++ (72.90) than in C (57.70).

Overall, the aggregate average score across all evaluated models is slightly higher in C++ (70.59) than in C (67.36), indicating that the overall performance differences between the two programming languages remain relatively small. These results suggest that model effectiveness is largely consistent across both languages when solving algorithmic programming tasks.

5.2. Task-Level Performance Analysis. Table 4 reports the empirical task difficulty d_i and discrimination values δ_i computed according to Eq. (3) and Eq. (4), respectively.

The results reveal a wide range of task difficulty levels. Problems such as *Criptografie*, *Macarie*, and *Poseidon* exhibit high average difficulty scores d_i, indicating

that they are relatively easy for most evaluated models. In contrast, problems such as *Mandms*, *Nisip*, and *Spirala3* obtain very low values of d_i, indicating substantially higher empirical difficulty.

TABLE 4. Empirical difficulty and discriminative capacity of the evaluated programming tasks. Difficulty is defined as the average score across models, while the difficulty level categorizes tasks into Easy (≥ 60), Medium (30–60), and Hard (< 30). Discrimination represents the Spearman correlation between task scores and model ability rankings.

Problem	`Difficulty`	`Difficulty Level`	`Discrimination`
Eras	53.33	Medium (30–60)	0.966
Tombola	50.00	Medium (30–60)	0.928
Cartita	62.67	Easy (≥60)	0.923
Inno	48.00	Medium (30–60)	0.913
Santinele	79.33	Easy (≥60)	0.895
Patratele	54.33	Medium (30–60)	0.874
Fibofrac	44.44	Medium (30–60)	0.866
Teleportor	41.89	Medium (30–60)	0.845
Castel	63.89	Easy (≥60)	0.836
Criptografie	90.22	Easy (≥60)	0.822
Macarie	83.33	Easy (≥60)	0.807
Lumini	65.22	Easy (≥60)	0.804
Fuziune	48.89	Medium (30–60)	0.797
Vnoroc	58.67	Medium (30–60)	0.783
Parking	48.00	Medium (30–60)	0.783
Magic1	65.44	Easy (≥60)	0.769
Bug	33.89	Medium (30–60)	0.760
Gene	76.11	Easy (≥60)	0.749
Spirala3	18.89	Hard (<30)	0.730
Mun	70.00	Easy (≥60)	0.721
Venus	51.67	Medium (30–60)	0.717
Strips	44.89	Medium (30–60)	0.710
Bile	57.33	Medium (30–60)	0.705
Succes	47.11	Medium (30–60)	0.694
Pro3	72.89	Easy (≥60)	0.682
UDP	41.11	Medium (30–60)	0.681
Intergalactic	74.78	Easy (≥60)	0.676
Partitura	66.67	Easy (≥60)	0.633
Teren	62.78	Easy (≥60)	0.633
Restaurare	38.33	Medium (30–60)	0.630
Trafalet	77.89	Easy (≥60)	0.610
Joc	44.56	Medium (30–60)	0.589
Cufar	77.56	Easy (≥60)	0.584
Hibrid	66.67	Easy (≥60)	0.553
Antidivizor	46.00	Medium (30–60)	0.527
Kmajo	47.89	Medium (30–60)	0.504
Nisip	11.11	Hard (<30)	0.411
Tema	58.11	Medium (30–60)	0.405
Poseidon	82.22	Easy (≥60)	0.396
Mandms	16.89	Hard (<30)	0.316

The discrimination values δ_i further highlight tasks that effectively separate strong and weak models according to the ranking-based criterion defined in Eq. (4). Problems such as *Eras*, *Tombola*, and *Cartita* exhibit the highest discrimination scores, indicating that they strongly correlate with overall model ability and therefore serve as informative benchmarks for evaluating problem-solving performance. Conversely, tasks with lower discrimination values provide less separation between models and therefore contribute less to distinguishing relative system performance.

Table 5 compares the average performance of model groups on tasks originating from the OJI and ONI competitions. The results indicate that models generally achieve higher scores on OJI tasks than on ONI tasks. This pattern suggests that ONI problems tend to be more challenging, reflecting their role as national-level competition tasks.

TABLE 5. Average scores obtained by model groups on tasks originating from the OJI and ONI competitions.

Source Type	DeepSeek	DeepThink	Google	OpenAI	xAI
OJI	52.57	43.86	76.79	76.33	59.18
ONI	27.96	29.92	70.46	70.03	50.71

Across both sources, models developed by Google and OpenAI achieve the highest average scores, exceeding 70 points in both categories. In contrast, the DeepSeek and DeepThink models obtain substantially lower averages, particularly on ONI tasks, where their performance drops below 30 points. The xAI models occupy an intermediate position, outperforming the DeepSeek and DeepThink groups while remaining below the leading systems. Overall, these results highlight a consistent performance gap between the strongest model families and those with more limited problem-solving capabilities.

Table 6 presents the average performance of model groups across different task levels. The results show clear variation depending on the difficulty level associated with the competition grade.

TABLE 6. Mean scores obtained by model groups across different task difficulty levels.

Level	DeepSeek	DeepThink	Google	OpenAI	xAI
7th gr.	37.67	26.67	69.67	79.56	42.50
8th gr.	39.00	29.87	72.80	70.82	60.60
9th gr.	34.92	47.58	77.75	71.81	47.21
10th gr.	72.50	100.00	95.00	66.67	100.00
playoff	30.39	21.44	64.22	73.33	49.33

For lower grade levels (7th gr. and 8th gr.), OpenAI and Google models achieve the highest average scores, indicating strong performance on introductory algorithmic problems. At the 9th gr. level, Google models slightly outperform the other groups, while OpenAI models maintain relatively stable performance across all levels. The 10th gr. tasks produce particularly high scores for some models, with DeepThink and xAI reaching maximum or near-maximum averages. However, this result should be interpreted cautiously, as the number of tasks at this level may be limited. The

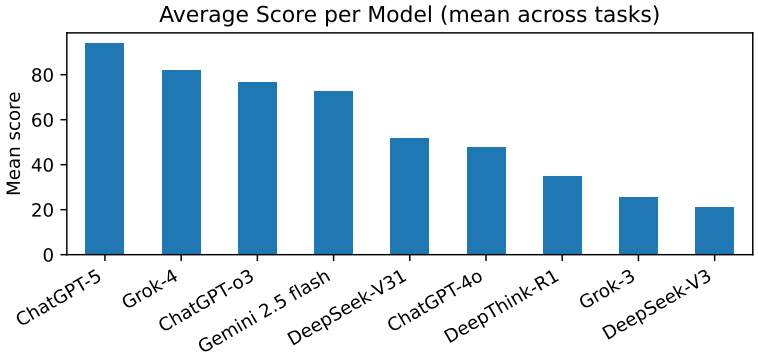

FIGURE 1. Average score obtained by each evaluated model across all programming tasks.

playoff tasks, which represent playoff-level problems, appear more difficult for most systems. All model groups show a noticeable decrease in average performance at this stage, suggesting that these tasks require more advanced reasoning and algorithmic design capabilities.

5.3. Visual Analysis of Model Performance. To complement the tabular results, several visualizations are used to analyze model behavior across the evaluated tasks. These plots illustrate differences in average performance, solution reliability, failure patterns, and task dominance among the evaluated models.

Figure 1 presents the mean score metric μ_j defined in Eq. (2) for each evaluated model across all tasks. The results show a clear hierarchy of performance. *ChatGPT-5* achieves the highest average score, followed by *Grok-4*, *ChatGPT-o3*, and *Gemini-2.5-flash*. These models consistently produce higher-quality solutions across tasks. In contrast, *DeepSeek-V3*, *Grok-3*, and *DeepThink-R1* obtain considerably lower average scores, indicating more limited success in solving the evaluated programming problems.

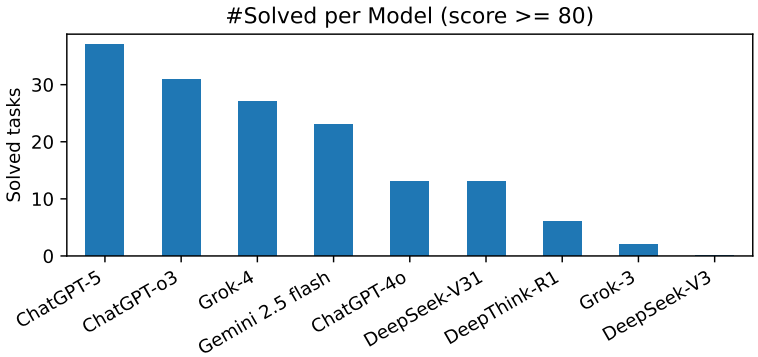

FIGURE 2. The number of tasks solved with high accuracy per model.

Figure 2 reports the number of solved tasks $S_j(\tau)$ defined in Eq. (6), using the acceptance criterion specified in Eq. (5) with threshold $\tau \geq 80$. The figure confirms the dominance of the strongest models, with *ChatGPT-5* solving the largest number of tasks, followed by *ChatGPT-o3* and *Grok-4*. Models such as *DeepSeek-V3* and *Grok-3* solve very few tasks at this threshold, reflecting weaker problem-solving capability on competitive programming tasks.

Model robustness is further examined through the failure-rate metric F_j defined in Eq. (7), shown in Figure 3. The plot measures the proportion of tasks where a model produced a score of zero, indicating a completely incorrect or unusable solution. *DeepSeek-V3*, *DeepThink-R1*, and *Grok-3* exhibit the highest failure rates, suggesting limited reliability on more challenging tasks. In contrast, models such as *ChatGPT-5*, *Grok-4*, and *Gemini-2.5-flash* show substantially lower failure rates, indicating more stable performance across the benchmark.

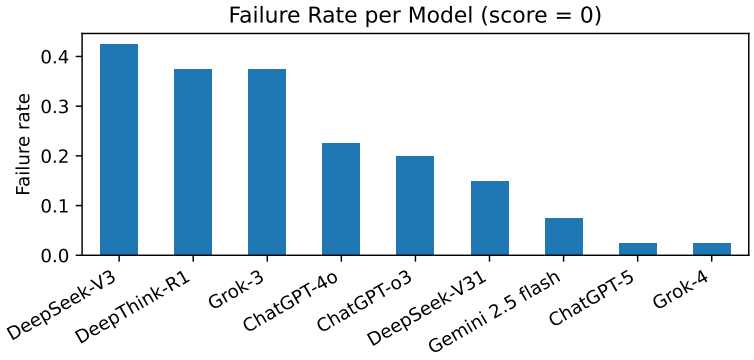

FIGURE 3. Failure rate of each evaluated model, measured as the proportion of tasks for which the generated solution obtained a score of zero.

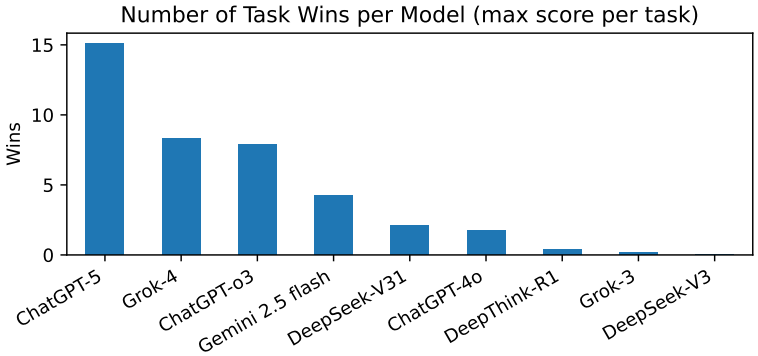

FIGURE 4. The number of task wins per model, where a win is defined as achieving the highest score among all models evaluated for a given programming task.

Finally, Figure 4 illustrates the task-winning metric W_j defined in Eq. (10), where task winners are identified according to the criterion defined in Eq. (9). This metric highlights task-level dominance by counting how frequently each model achieves the highest score among all evaluated systems. The results show that *ChatGPT-5* achieves the largest number of task wins, followed by *Grok-4* and *ChatGPT-o3*, indicating that these models most frequently attain the maximum score on individual tasks. The remaining models record substantially fewer wins, suggesting lower task-level competitiveness.

Figure 5 presents the correlation matrix derived from the Pearson similarity coefficient ρ_{jk} defined in Eq. (8). The visualization captures similarities in model behavior across the evaluated programming tasks. High correlation values indicate that two models exhibit similar success and failure patterns across tasks, whereas lower values suggest divergent problem-solving behavior.

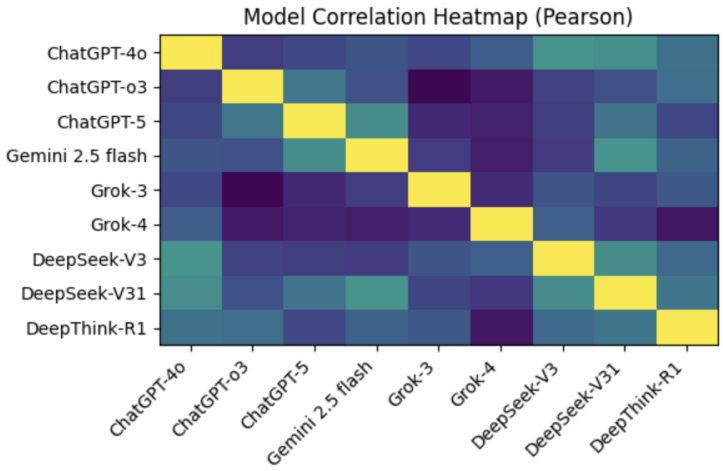

FIGURE 5. Heatmap visualization of the Pearson correlation coefficients ρ_{jk} computed according to Eq. (8).

Several notable patterns emerge from the correlation structure. Strong positive correlations are observed between *ChatGPT-5* and *Gemini-2.5-flash*, as well as between *DeepSeek-V3* and *DeepSeek-V31*, suggesting comparable performance trends across tasks. The results indicate that models from related families tend to exhibit more similar behavioral characteristics.

In contrast, models such as *Grok-3* and *DeepThink-R1* display weaker correlations with several high-performing systems, indicating more distinct performance patterns and less consistent agreement in task-level behavior.

Figure 6 presents a heatmap visualization of the score matrix X defined in Eq. (1). In this representation, rows correspond to individual tasks while columns represent the evaluated models. Color intensity reflects the obtained score, allowing patterns of success and failure to be easily identified across the benchmark.

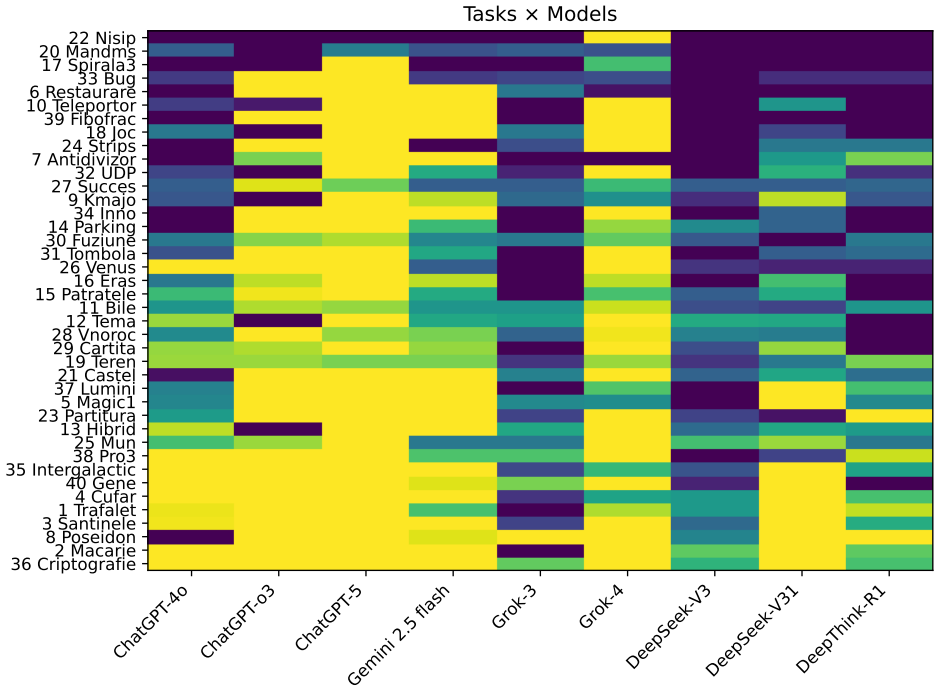

FIGURE 6. Heatmap visualization of the score matrix X, where rows correspond to programming tasks and columns correspond to evaluated models.

Several observations emerge from this visualization. Tasks such as *Criptografie*, *Macarie*, *Poseidon*, and *Santinele* exhibit high values of the empirical accessibility metric d_i, indicating that they are relatively easy for most models. In contrast, tasks appearing in the upper region of the heatmap, including *Nisip*, *Mandms*, *Spirala3*, and *Bug*, show substantially lower values of d_i, across many models, suggesting that they represent more challenging problems within the dataset.

The heatmap also highlights clear differences in model capabilities. *ChatGPT-5*, *ChatGPT-o3*, and *Grok-4* display consistently high scores across a large portion of the tasks, forming visible vertical bands of higher values. *Gemini-2.5-flash* also performs strongly on many problems but exhibits slightly greater variability. In contrast, models such as *DeepSeek-V3*, *Grok-3*, and *DeepThink-R1* show larger regions of low scores, indicating a higher frequency of partial or unsuccessful solutions.

Overall, this visualization provides an intuitive overview of task difficulty and model consistency. It reveals both clusters of easier tasks that are broadly solvable and a set of more difficult problems that clearly differentiate the capabilities of the evaluated models.

Figure 7 illustrates the mean performance metric μ_j computed separately across task-level categories. The visualization highlights several important trends in model performance.

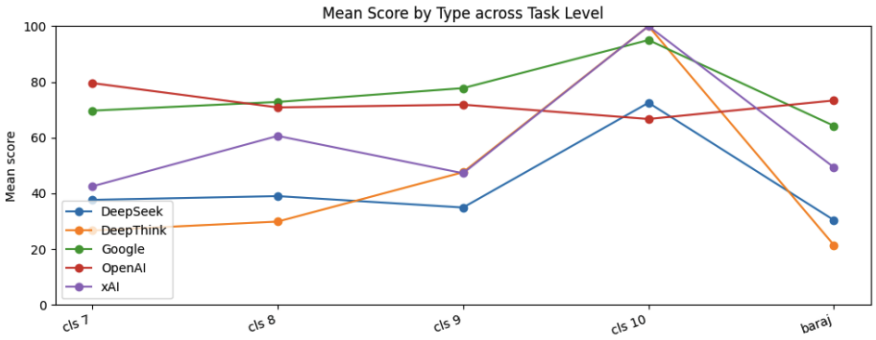

FIGURE 7. Mean score obtained by each model group across different task levels.

First, Google and OpenAI models maintain consistently high scores across most levels, demonstrating stable performance across tasks of varying difficulty. Second, DeepSeek models generally achieve lower scores across all levels, with only moderate improvements for higher-grade tasks. DeepThink models show greater variability, including a sharp peak at the cls 10 level but significantly lower performance on other levels.

The xAI models exhibit intermediate behavior, achieving competitive scores at higher levels while remaining below the leading groups in most cases. Finally, the playoff tasks appear to represent the most challenging category overall, as nearly all model groups experience a noticeable decline in performance at this level.

Together, these results confirm that task level significantly influences model success and that stronger models maintain more consistent performance across increasing levels of algorithmic difficulty.

6. Conclusions and Future Work

By correlating performance data with broader questions of capability and applicability, this paper contributes to an informed understanding of the current state of AI-based generative tools and their possible evolution from supplementary aids to collaborative partners or even replacements in certain areas of human expertise.

This paper evaluated the performance of nine state-of-the-art language models (*ChatGPT-5*, *ChatGPT-4o*, *ChatGPT-o3*, *Gemini-2.5-flash*, *Grok-4*, *Grok-3*, *DeepSeek-V3*, *DeepSeek-V31* and *DeepThink-R1*) on a set of 40 competitive programming problems selected from the Romanian National Olympiad in Informatics mainly for grades $7-9$ (there is only one problem for the 10th grade).

The results reveal a clear stratification in model performance. *ChatGPT-5* achieved the highest average score (93.90) and the largest number of completely solved problems, followed by *Grok-4* (81.83) and *ChatGPT-o3* (74.95). *Gemini-2.5-flash* also demonstrated strong performance with an average score of 72.68. A second performance tier includes *DeepSeek-V31* (51.85) and *ChatGPT-4o* (47.85), while *DeepThink-R1*, *Grok-3*, and *DeepSeek-V3* obtained substantially lower average scores.

The distribution of scores indicates a noticeable performance gap between the strongest and weakest systems, with a difference of more than 70 points between the highest and lowest averages. Finally, the comparison between C and C++ implementations does not reveal substantial performance differences, as the aggregate average scores differ by only a few points across the two programming languages.

From an educational perspective, the results suggest that advanced LLMs can serve as effective tools for intermediate-level competitive programming, providing correct and well-structured solutions for a large part of problems. However, their current limitations prevent them from completely replacing human instruction.

The dataset was restricted to 40 problems in a specific difficulty range and the evaluation followed a single-shot protocol without iterative refinement. Future work may involve expanding the dataset by incorporating a broader range of programming problems from additional competitions and sources, as well as increasing the diversity of algorithmic topics and difficulty levels represented in the benchmark. Another promising direction is the exploration of more advanced prompting strategies, including multi-turn interactions and chain-of-thought reasoning, which could provide deeper insight into how LLMs approach complex algorithmic problems. Furthermore, future studies could investigate additional evaluation dimensions, such as solution efficiency, code quality, and reasoning transparency, in order to obtain a more comprehensive understanding of the capabilities and limitations of LLMs in competitive programming contexts.

Acknowledgment

The work of M. Coşulschi and M. Gabroveanu have been supported by a grant of the Romanian Ministry of Research, Innovation and Digitalization (MCID), project number 22 - Nonlinear Differential Systems in Applied Sciences, within PNRR-III-C9-2022-I8.

References

[1] K. Ali Abd Al-Hameed, Spearman's correlation coefficient in statistical analysis, *International Journal of Nonlinear Analysis and Applications* **13** (2022), no. 1, 3249-3255.

[2] T. Coignion, C. Quinton, R. Rouvoy, A Performance Study of LLM-Generated Code on Leetcode, *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering - EASE'24*, ACM, 79–89, 2024.

[3] M. Cosulschi, M. Gabroveanu, F. Slabu, The problem-solving capabilities of modern LLMs in the competitive programming context, In: *Proceedings of 2025 Balkan Conference on Informatics (BCI'25)*, Tirana, Albania.

[4] A.M. Dumitran, A.C. Badea, S.-G. Muscalu, Evaluating the Performance of Large Language Models in Competitive Programming: A Multi-Year, Multi-Grade Analysis, In: *Proceedings of 2024 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, IEEE Explore, 1–7, 2024.

[5] N. Dunder, S. Lundborg, J. Wong, O. Viberg, Kattis vs ChatGPT: Assessment and Evaluation of Programming Tasks in the Age of Artificial Intelligence, In: *14th International Conference on Learning Analytics and Knowledge (LAK 2024)*, ACM, 821–827, 2024.

[6] S.E. Elhambakhsh, Evaluating ChatGPT-3's efficacy in solving coding tasks: implications for academic integrity in English language assessments, *Language Testing in Asia* **15** (2025), 37.

[7] OpenAI: A. El-Kishky et al, Competitive programming with large reasoning models, arXiv.2502.06807, 2025.

[8] Google, Start building with Gemini 2.5 Flash, 2025. [Online]. Available: https://developers.googleblog.com/en/start-building-with-gemini-25-flash/.

[9] xAI, Grok 4, 2025. [Online]. Available: https://x.ai/news/grok-4.
[10] xAI, Grok 3 Beta - The Age of Reasoning Agents, 2025. [Online]. Available: https://x.ai/news/grok-3.
[11] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, arXiv preprint, arXiv:2501.12948, 2025.
[12] Y. Huang, Z. Lin, X. Liu, Y. Gong, S. Lu, F. Lei, Y. Liang, Y. Shen, C. Lin, N. Duan, W. Chen, Competition-Level Problems are Effective LLM Evaluators, In: *Findings of the Association for Computational Linguistics: ACL 2024*, 13526–13544, 2024.
[13] J. Li, S. Tworkowski, Y. Wu, R. Mooney, Explaining Competitive-Level Programming Solutions using LLMs, 10.48550/arXiv.2307.05337, 2023.
[14] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu,, et al, Deepseek-v3 technical report, arXiv preprint, arXiv:2412.19437, 2024.
[15] DeepSeek-V3.1 deepseek, [Online]. Available: https://huggingface.co/deepseek-ai/DeepSeek-V3.1.
[16] M. Mirzayanov, O. Pavlova, P. Mavrin, R. Melnikov, A. Plotnikov, V. Parfenov, A. Stankevich, Codeforces as an Educational Platform for Learning Programming in Digitalization, *Olympiads in Informatics Jurnal* **14** (2020), 133–142.
[17] OpenAI: GPT-4 Technical Report, arXiv preprint, arXiv:2303.08774, 2023.
[18] OpenAI: Hello GPT-4o, OpenAI, [Online]. Available: https://openai.com/index/hello-gpt-4o.
[19] OpenAI: Introducing OpenAI o3 and o4-mini, 2025. [Online]. Available: https://openai.com/index/introducing-o3-and-o4-mini/.
[20] OpenAI: GPT-5 is here, 2025. [Online]. Available: https://openai.com/gpt-5/.
[21] S. Quan, Y. Jiaxi, Y. Bowen, Z. Bo, et al, CodeElo: Benchmarking Competition-level Code Generation of LLMs with Human-comparable Elo Ratings, arXiv preprint, arXiv:2501.01257, 2025.
[22] R. Shakya, F. Vadiee, M. Khalil, A Showdown of ChatGPT vs DeepSeek in Solving Programming Tasks, *2025 International Conference on New Trends in Computing Sciences (ICTCS)*, Amman, Jordan, 413–418, 2025.
[23] D. Souza, R. Gheyi, L. Albuquerque, G. Soares, M. Ribeiro, Code Generation with Small Language Models: A Deep Evaluation on Codeforces, arXiv preprint, arXiv:2504.07343, 2025.
[24] N.D. Tran, J. May, N. Ho, L.B. Ngo, Exploring ChatGPT's Ability to Solve Programming Problems with Complex Context, *Journal of Computing Sciences in Colleges* **38** (2023), no. 3, 195–209.
[25] T.Y. Yeh, K. Tran, G. Gao, T. Yu, W.O. Fong, T.Y. Chen, Bridging novice programmers and LLMs with interactivity, In: P*roc. 56th ACM Tech. Symp. Comput. Sci. Educ.* (SIGCSETS 2025), 1295–1301, 2025.
[26] Z. Zheng, Z. Cheng, Z. Shen, S. Zhou, K. Liu, H. He, D. Li, S. Wei, H. Hao, J. Yao, P. Sheng, LiveCodeBench Pro: How Do Olympiad Medalists Judge LLMs in Competitive Programming?, arXiv preprint, arXiv:2506.11928, 2025.
[27] Z. Wang, Y. Liu, Y. Wang, W. He, B. Gao, M. Diao, Y. Chen, K. Fu, F. Sung, Z. Yang, T. Liu, Ojbench: A competition level code benchmark for large language models, arXiv preprint, arXiv:2506.16395, 2025.

(Mirel Coşulschi, Mihai Gabroveanu, Florin Slabu) Department of Computer Science, Faculty of Sciences, University of Craiova, 13 A.I. Cuza Street, Craiova, 200585, Romania
E-mail address: mirelc@central.ucv.ro, mihaiug@central.ucv.ro, florin.slabu@inf.ucv.ro

(Alexandra Vultureanu-Albişi) Department of Computers and Information Technology, Faculty of Automatics, Computers and Electronics, University of Craiova, Bvd. Decebal 107, Craiova, 200440, Romania
E-mail address: alexandra.vultureanu@edu.ucv.ro