Paper

Code

Leaderboard

ALE-Bench: A Benchmark for Long-Horizon Objective-Driven Algorithm Engineering

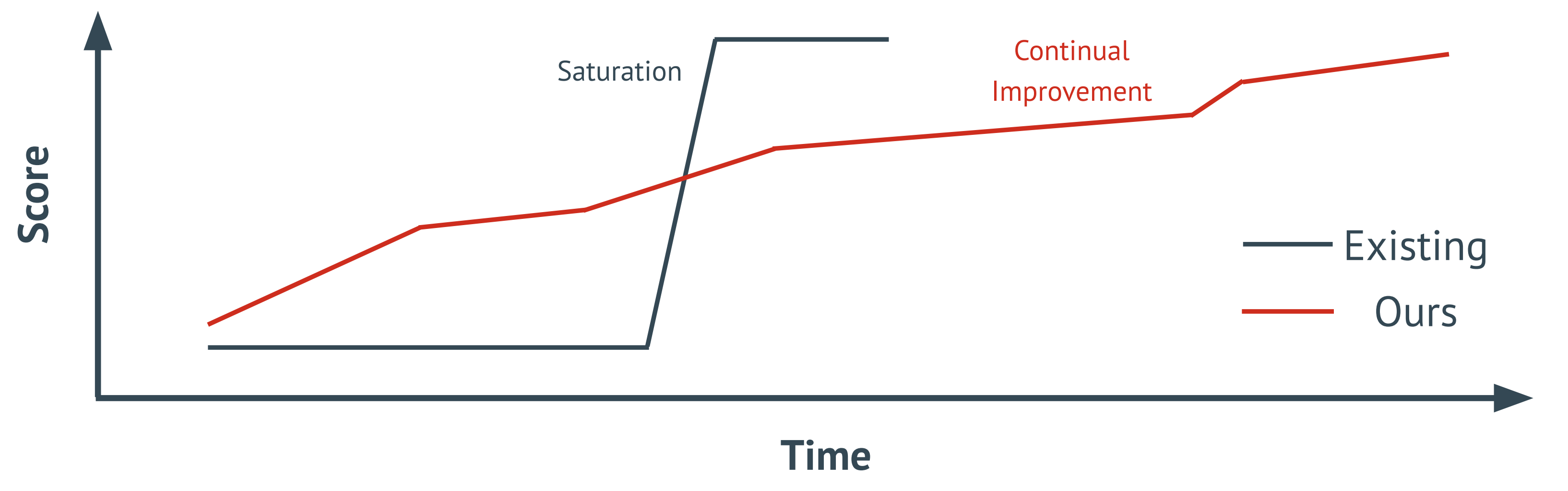
Yuki Imajuku[1], Kohki Horie[1,2], Yoichi Iwata[3], Kensho Aoki[3], Naohiro Takahashi[3], Takuya Akiba[1]
[1]Sakana AI, Japan [2]The University of Tokyo, Japan [3]AtCoder, Japan

sakana.ai AtCoder NEURAL INFORMATION PROCESSING SYSTEMS

Beyond Pass/Fail: A Long-Horizon Coding Benchmark

Existing coding benchmarks [1,2,3] have primarily focused on short-duration, exact-solution contests and current LLMs' performance show signs of saturation [4]. We propose a **new coding benchmark** concentrating on **long-duration, score-based** contests where the objective is to continuously find a "better" solution. The tasks consist of **optimization problems** whose true optima are computationally out of reach (e.g. the underlying problems are NP-hard), collected from AtCoder. **Human participants spend weeks** iteratively refining their programs.

Time	~1 hour	~few hours	~few days	1 week~
Existing	✗ 16 pass 51 fail	✓ 67 pass 0 fail	Finished as correct Score remained unchanged	
Ours	✓ Score 58.8	✓ Score 77.7	✓ Score 87.6	✓ Score 92.1

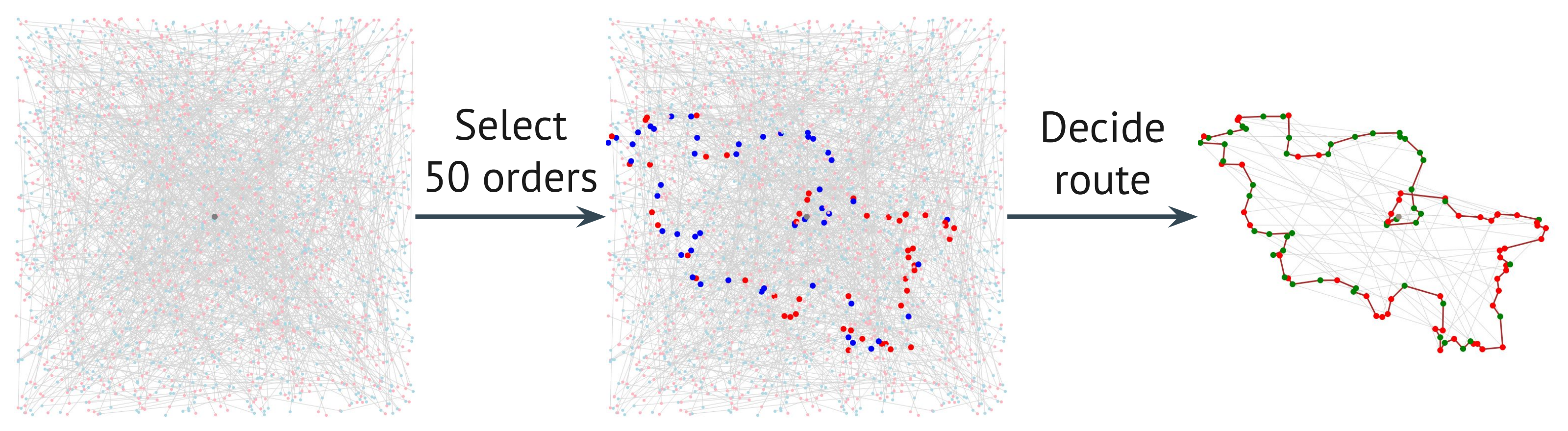


Our benchmark's strengths:

1. **High Real-World Relevance**: Mirroring complex industrial challenges
2. **Direct Comparison with Human Experts**: Reproducing actual contest condition
3. **Measures Advanced AI Reasoning**: Assessing long-horizon reasoning
4. **Future-Proof & Open-Ended**: Continuing even after AI surpass human experts

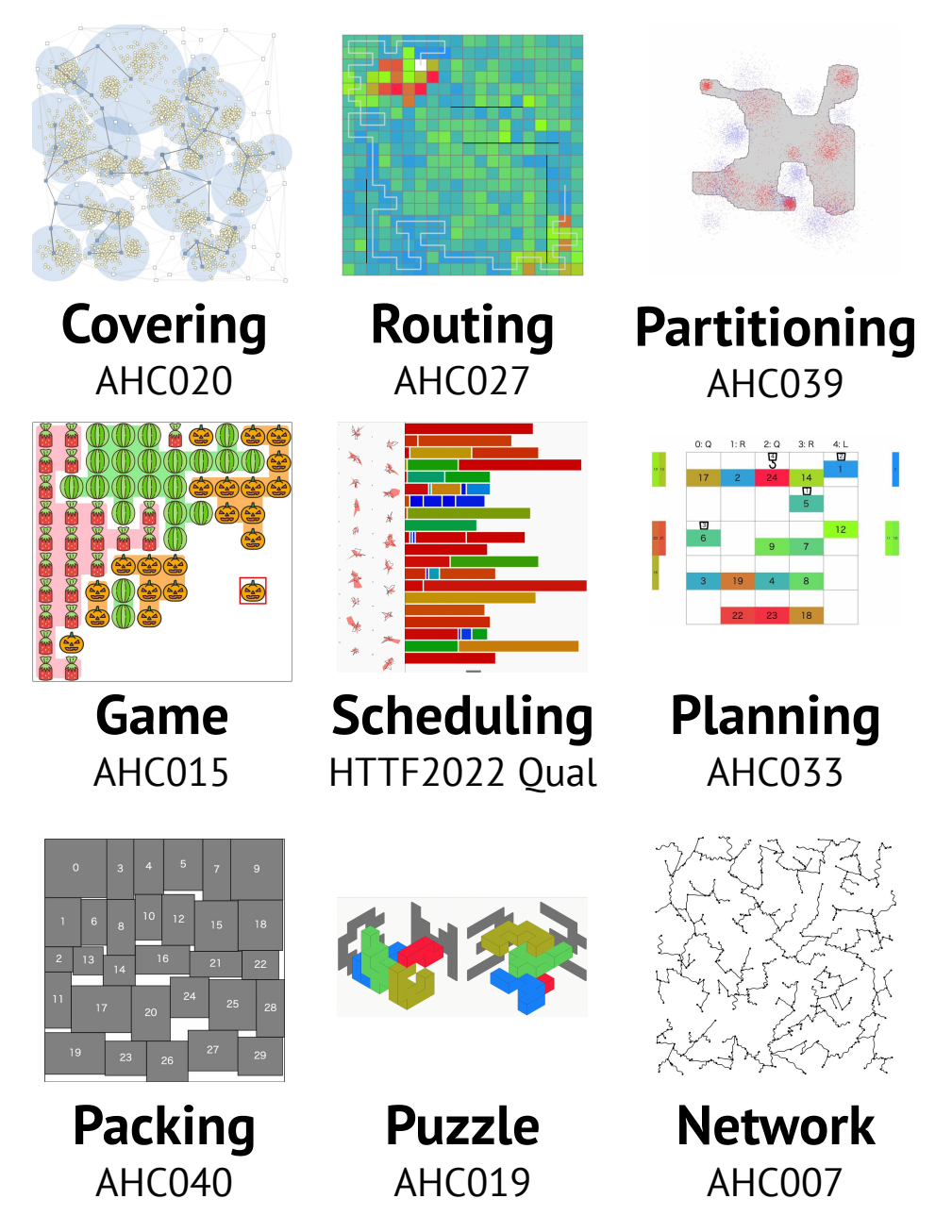
Task Example

Write a program that, given **a large collection of pickup-delivery pairs** on a 2D grid, chooses a prescribed number of requests and outputs **a depot-to-depot tour** that visits the pickup location of each selected request before its corresponding drop-off. The score is the total length of the route; **the shorter, the better.**
CPU time limit: 2 seconds / CPU memory limit: 1024 MiB

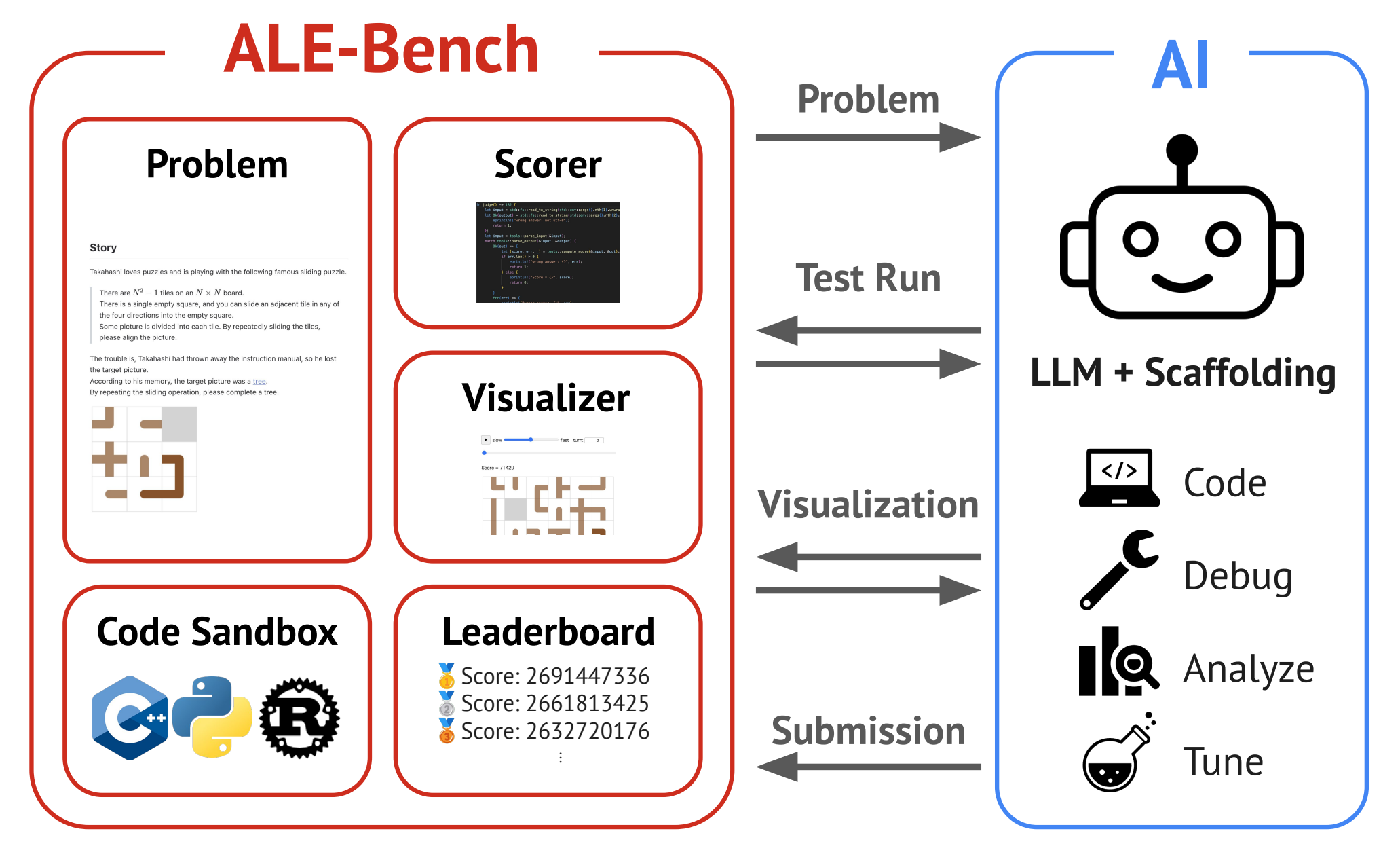


ALE-Bench

Problems



Framework



ALE-Bench includes...

- **40 real AtCoder Heuristic Contest (AHC) problems**
  - Statements in Markdown & Rust program evaluating the code
  - Web-based / Rust visualizer that displays the behavior of the code
- **Ready-to-use reproducible benchmarking framework**
  - Key actions can be performed via Python-based API
  - Thoroughly reproduced Docker image and computational environment
    Example Code:

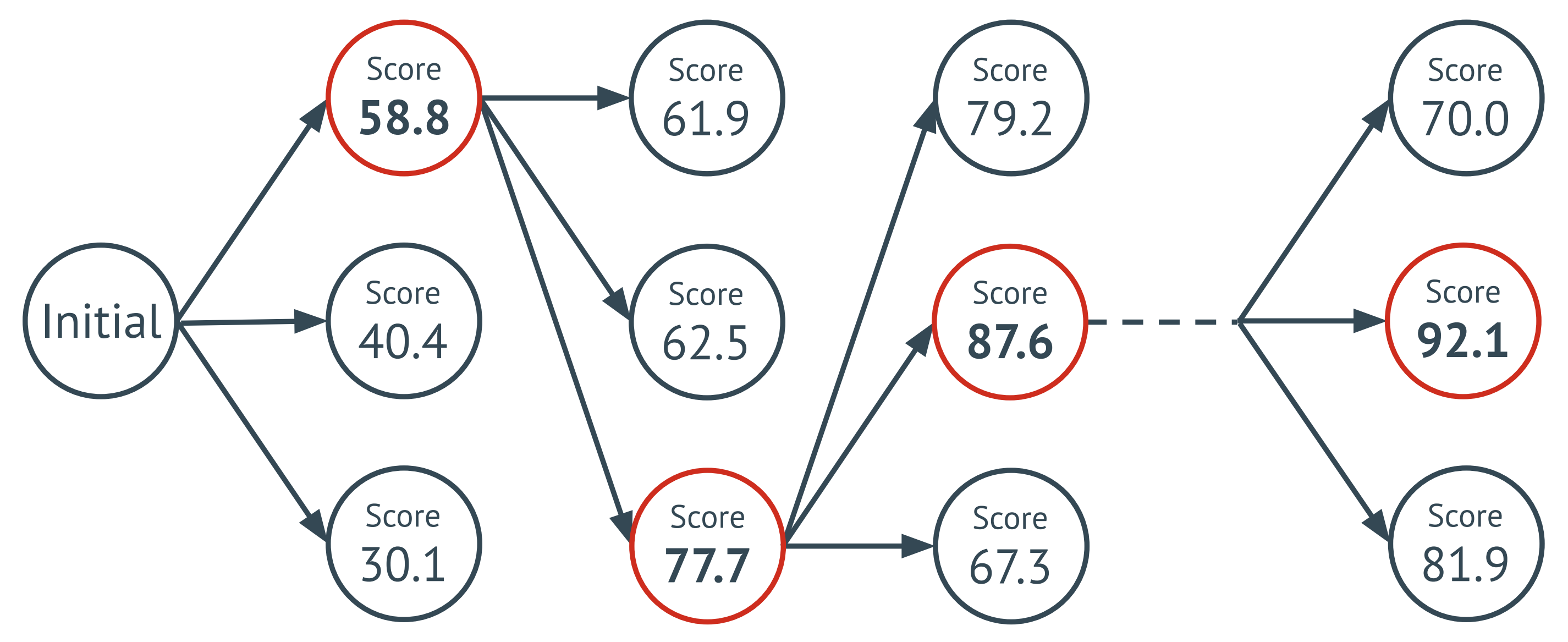
```
import ale_bench
session = ale_bench.start("ahc001")
code = ai_agent.generate_cpp23_answer(session.problem)
result_details, rank, perf = session.private_eval(code, "cpp23")
```

Visit our GitHub repository!

ALE-Agent

We develop a specialized prototype designed as a strong baseline. This agent incorporates established techniques like...

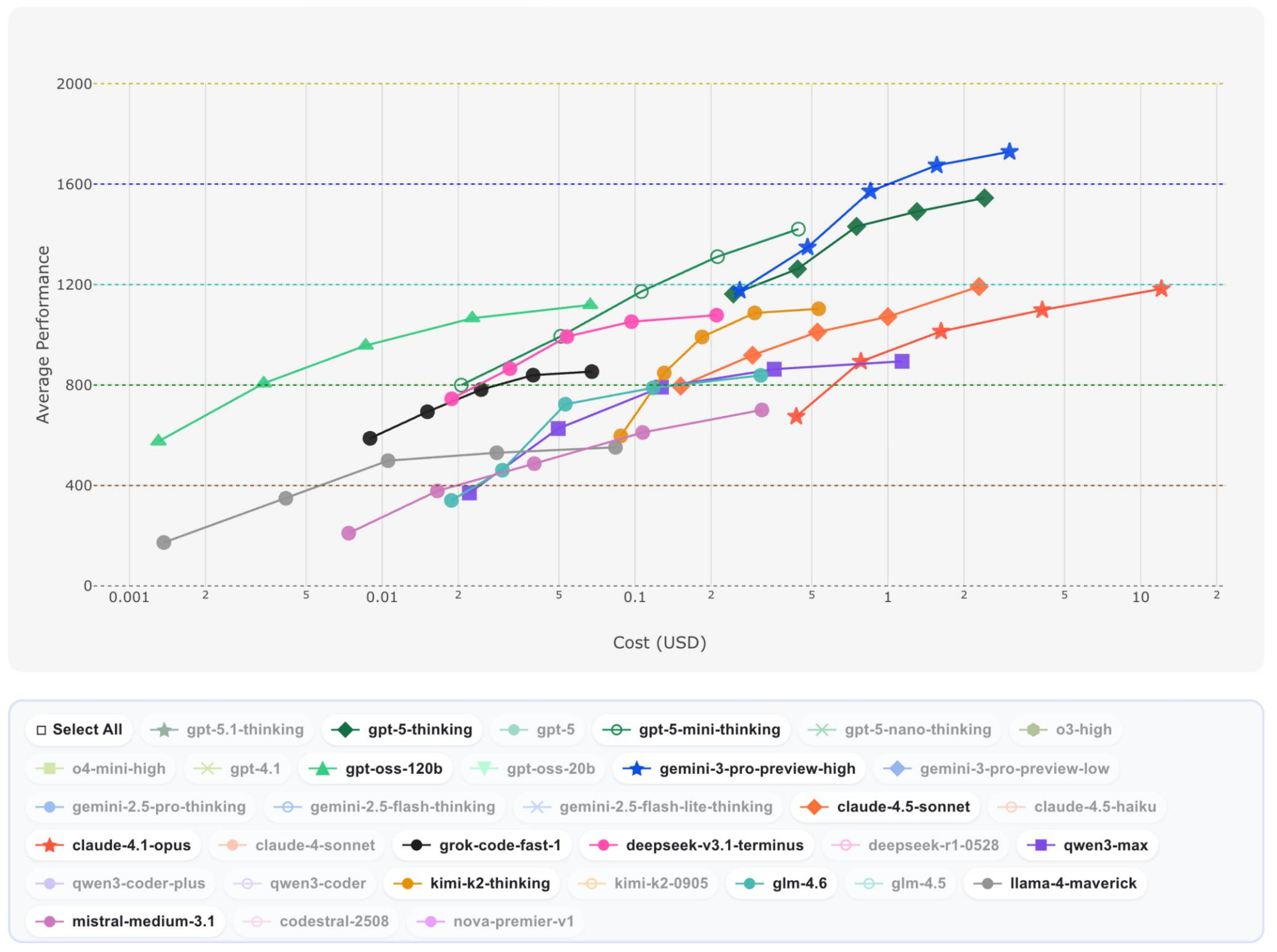
1. **Prompting with Domain Knowledge**: Injecting expert knowledge about standard techniques in algorithm engineering (e.g. simulated annealing)
2. **Inference Time Scaling via Tree Search**: Employing a best-first-search-based algorithm to generate and refine answer candidates using an LLM



Result

Performance by Model

We regularly evaluate upcoming LLMs and update the leaderboard page. Please see the leaderboard website for more information.



Each point on each line represents self-refine x1 (zero-shot), x2, x4, x8, x16 performance for each model.

Performance by Agent

Agent (Gemini 2.5 Pro)	Avg. Performance	LLM API Cost	
Self Refinement	1198 (Top 54.1%)	$11.10 / Problem	$0.157 / API Call
OpenHands [5]	903 (Top 82.8%)	$3.25 / Problem	$0.134 / API Call
ALE-Agent	1879 (Top 6.8%)	$100.33 / Problem	$0.113 / API Call

Analysis in Paper

- ✓ **Long-Horizon Reasoning vs. Brute-force Exploration**: long-horizon reasoning
- ✓ **Contamination and Plagiarism Check**: No significant trend was observed
- ✓ **Performance Difference across Code Languages**: C++20 > Rust > Python 3
- ✓ **Algorithms Where LLMs Excel in Implementation**: Simulated Annealing
- ✓ **Real-time Competition Participation**: ALE-Agent achieved top 2% in AHC047

References

1. Dan Hendrycks, et al. (2021). "Measuring coding challenge competence with APPS." In: NeurIPS Track on Datasets and Benchmarks.
2. Yujia Li, et al. (2022). "Competition-level code generation with AlphaCode." In: Science, 378(6624). pp. 1092–1097.
3. Naman Jain, et al. (2025). "LiveCodeBench: Holistic and contamination free evaluation of large language models for code." In: ICLR.
4. OpenAI. (2025). "Introducing openai o3 and o4-mini." https://openai.com/index/introducing-o3-and-o4-mini/
5. Xingyao Wang, et al. (2025). "OpenHands: An open platform for AI software developers as generalist agents." In: ICLR.