Frontier-Eng: Benchmarking Self-Evolving Agents on Real-World Engineering with Generative Optimization

Dapeng Jiang, Yizhe Chi, Kaisen Yang, Tianwei Luo, Boshi Zhang, Deyao Hong, Dianqiao Lei, Yifan Zhou, Weiyang Jin, Xiaoyan Fan, Han Hao, Zhe Cao, Qian Houde, Qingle Liu, Bohan Lyu, Bowen Wang, Situ Wang, Youjie Zheng, Bingxiang He, Eren Cai, Calvin Xiao, Qinhuai Na

Navers Lab, Einsia.AI | NeurIPS 2026 · Evaluations and Datasets Track

Einsia

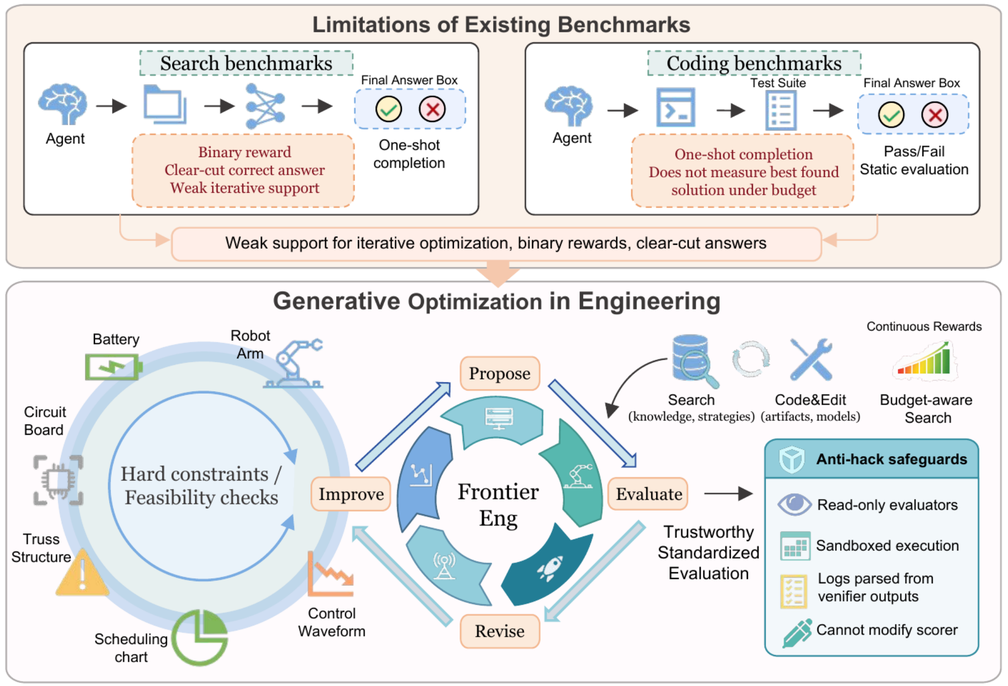
GitHub

Project page

arXiv

Motivation: engineering is iterative improvement

- Value comes from **improving a feasible artifact under hard constraints**: a faster kernel, a lighter truss, a lower-carbon data-center policy.
- Search and coding benchmarks are scored **pass/fail**. They cannot measure how far an agent pushes the best feasible solution within a budget.
- **Generative optimization**: a model proposes artifacts, an external verifier evaluates them, a search process spends the budget.
- Existing work is mostly mathematical, algorithmic or synthetic. We need a **broad, verifier-grounded** test across real engineering.



Binary-reward benchmarks vs. generative optimization with anti-hack safeguards.

Task formulation

Each task is an executable optimization problem $\boldsymbol{\tau = (C, x_0, E)}$

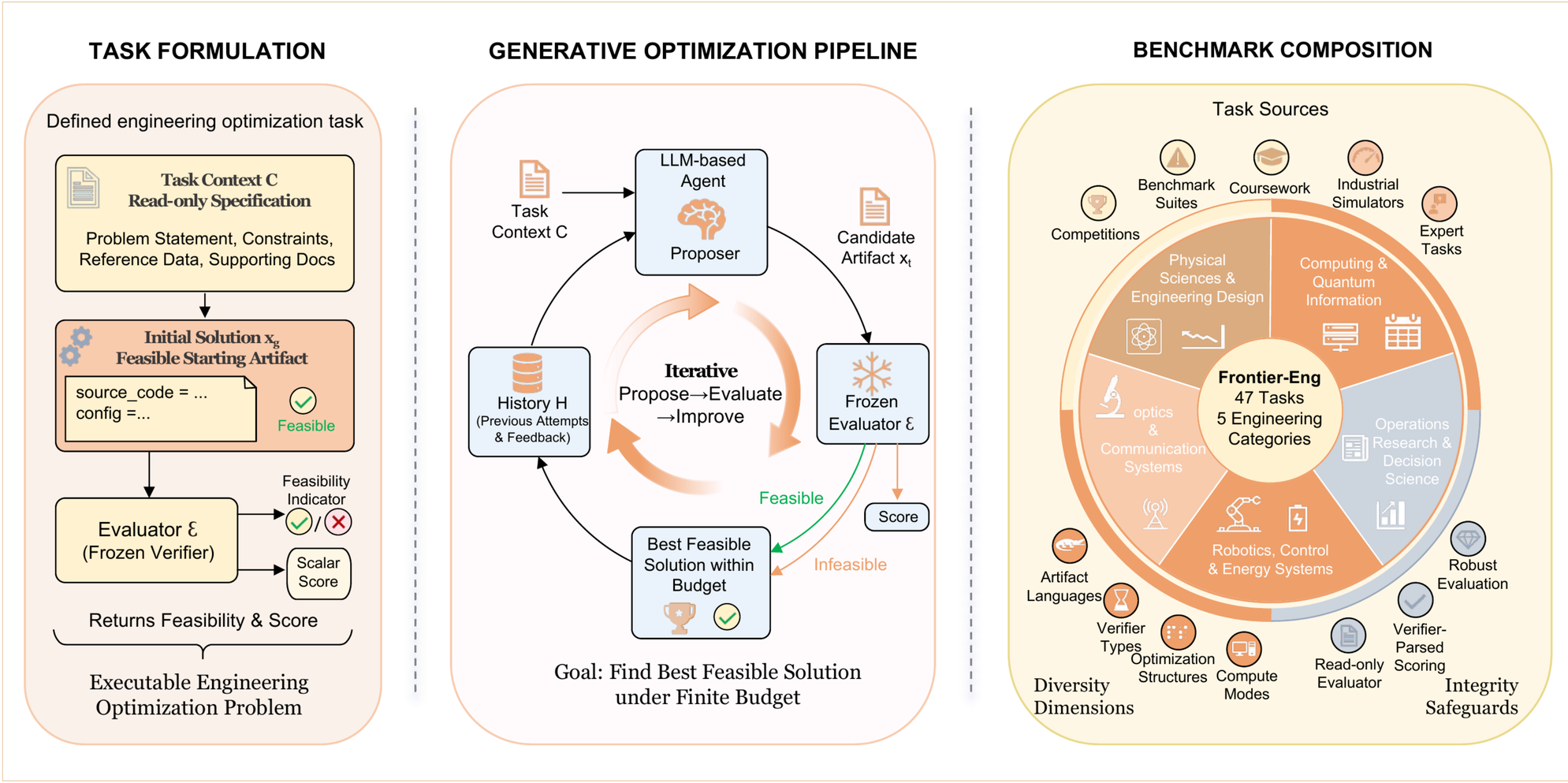
- **C**: read-only context (problem statement, constraints, reference data)
- $\mathbf{x_0}$: a **verified-feasible** starting artifact: Python, C/C++, CUDA/Triton, circuits, configs
- **E**: a frozen evaluator returning feasibility $v \in \{0,1\}$ and a scalar score s
- Feasible *and* infeasible submissions both consume the budget B.

$$\mathbf{x_t = A(C, H_{t-1})}$$
$$(v_t, s_t) = E(x_t)$$

$$\mathbf{s^* = \max s_t}$$
$$\text{over } 0 \le t \le B,\ v_t = 1$$

The proposer and the search strategy (evolutionary, tree search, bandits, self-refine) are decoupled: same start, evaluator and budget for every method.

Frontier-Eng: 47 tasks, 5 engineering categories



Task formulation · iterative propose → evaluate → improve pipeline · benchmark composition

How the tasks are built

Category	Tasks	Representative subfields
Computing & quantum information	10	kernels, systems, crypto, circuits
Operations research & decision science	9	inventory, scheduling, portfolio
Robotics, control & energy systems	8	navigation, control, batteries, data centers
Optics & communication systems	10	adaptive optics, holography, fiber, wireless
Physical sciences & engineering design	10	structures, reactions, aerospace, sensing

Sources: engineering competitions, academic benchmark suites, classic coursework, industrial simulators, expert-written tasks.

Inclusion rules: self-contained specification · editable artifact · runnable evaluator returning feasibility and score · verified-feasible initial artifact.

Review: automated checks of evaluator execution and interface, then human maintainers verify the formulation and constraints.

Evaluation integrity and protocol

Anti-hack safeguards

- Evaluators, reference data and hidden scripts are **read-only**
- Candidates run in **isolated temp directories** with only declared writable files
- Scores are **parsed from verifier-owned outputs**, not candidate claims
- Randomized seeds, multi-scenario averaging, FEM re-solves, physics simulators

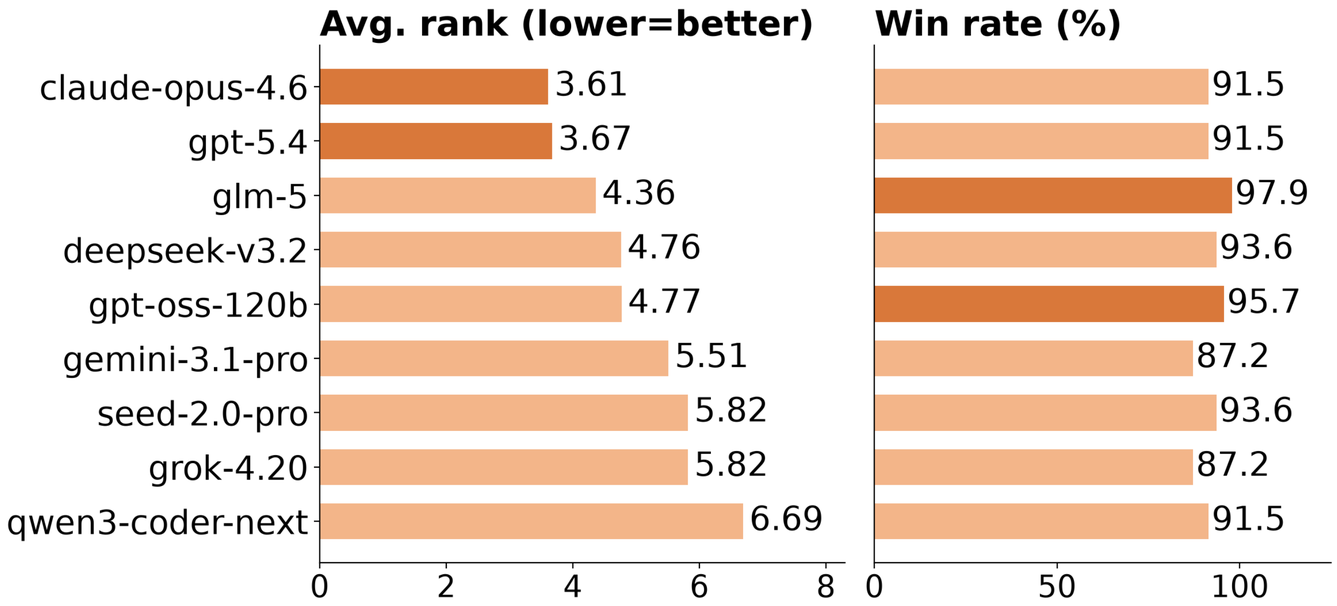
Cross-task protocol

$$R_m = (1/N)\ \Sigma_i\ rank_{i,m}$$ (lower is better)

- Raw units are incompatible, so methods are ranked **within each task**; ties get averaged ranks
- Also reported: **win rate** over the initial solution and **Dolan–Moré performance profiles**

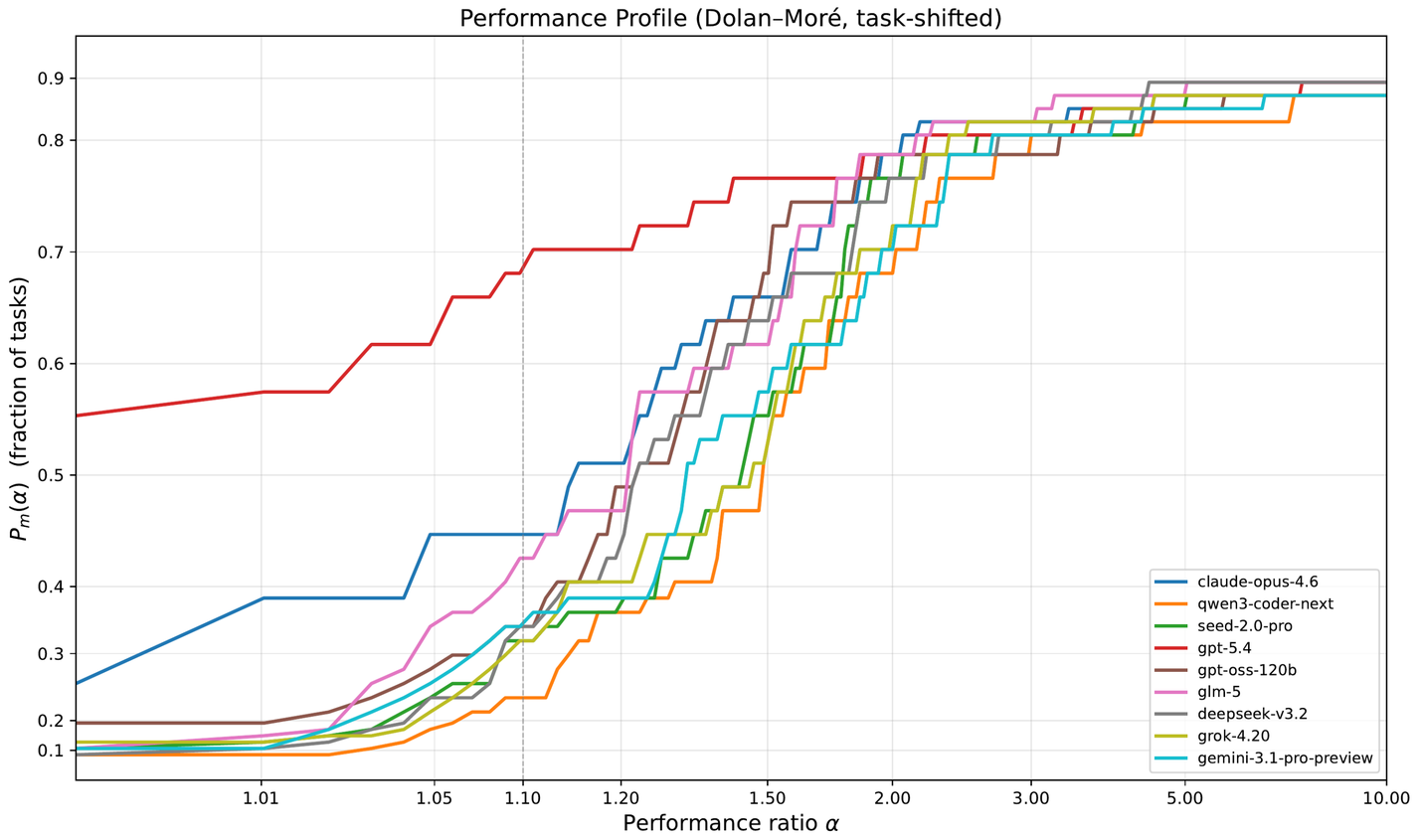
Agents are free to optimize the artifact, but not the measurement process.

Results: nine models under OpenEvolve



- 100 search evaluations per task, identical initial programs, frozen evaluators
- **Claude Opus 4.6 (3.61)** and **GPT-5.4 (3.67)** have the best average rank; the gap is **not statistically significant**
- **GLM-5** and **GPT-OSS-120B** most reliably beat the baseline (97.9% / 95.7%)
- **Peak quality and improvement reliability are different properties**

Performance profile: who is near-best?



- Dolan–Moré profile on translated within-task scores
- **GPT-5.4** leads the strict near-best regime: best or close to best on the largest share of tasks
- Other models catch up only at looser ratios, so rank alone hides the shape

Search frameworks: model × framework interaction

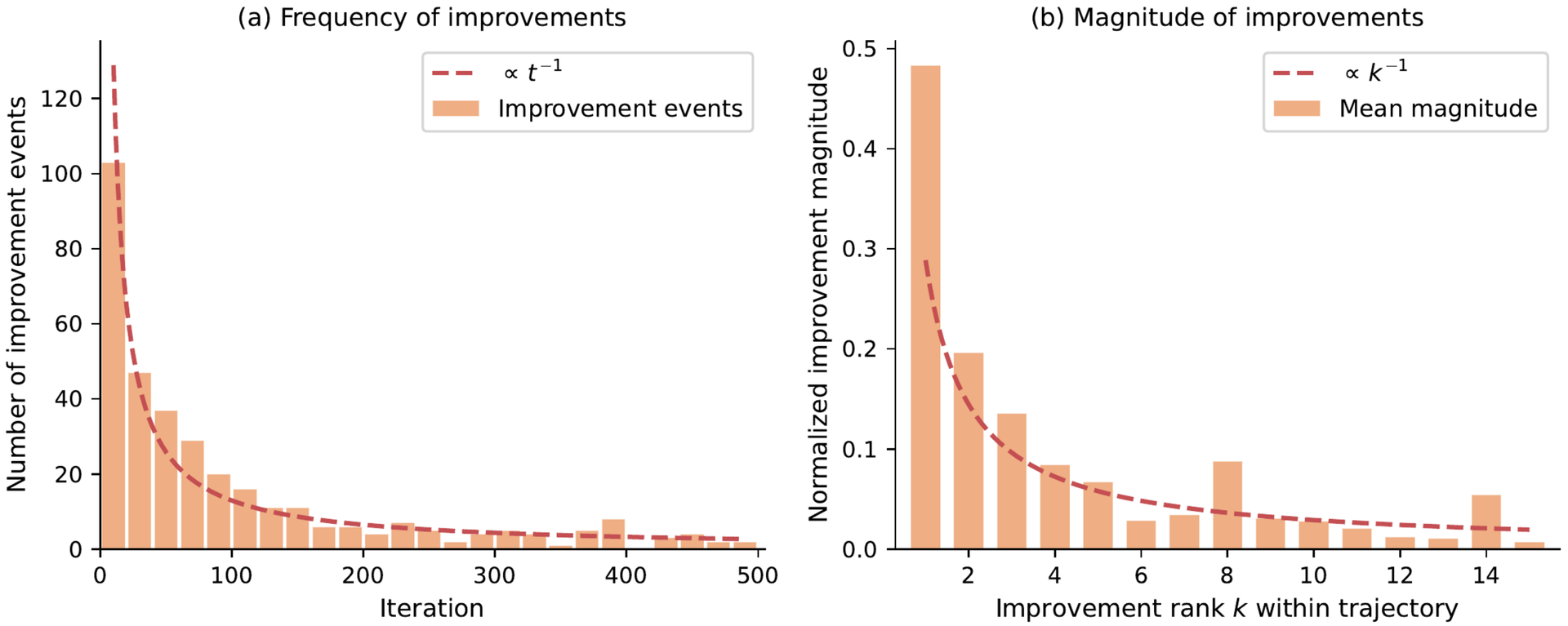
Framework	Opus rank	OSS rank	Wins (Opus/OSS)
ABMCTS	1.43	1.55	26 / 20
OpenEvolve	1.28	1.64	30 / 13
ShinkaEvolve	1.28	1.62	29 / 13

Opus = claude-opus-4.6, OSS = gpt-oss-120b; 47 tasks, lower rank is better.

Model + framework	gain %	early	plateau
Opus + ABMCTS	61.6	0.89	0.65
Opus + OpenEvolve	73.3	0.97	0.51
Opus + ShinkaEvolve	72.9	0.95	0.62
OSS + ABMCTS	50.4	0.99	0.72
OSS + OpenEvolve	58.9	0.99	0.57
OSS + ShinkaEvolve	68.3	0.89	0.45

- Opus beats GPT-OSS under **all three** frameworks
- But the gap changes with the framework: framework quality is **not absolute**
- **ABMCTS**: conservative, early lock-in, long plateaus
- **OpenEvolve**: aggressive structural rewrites, several later improvements
- **ShinkaEvolve**: sustained refinement (shortest plateau for GPT-OSS)
- Evaluate **model–search systems**, not models or algorithms in isolation

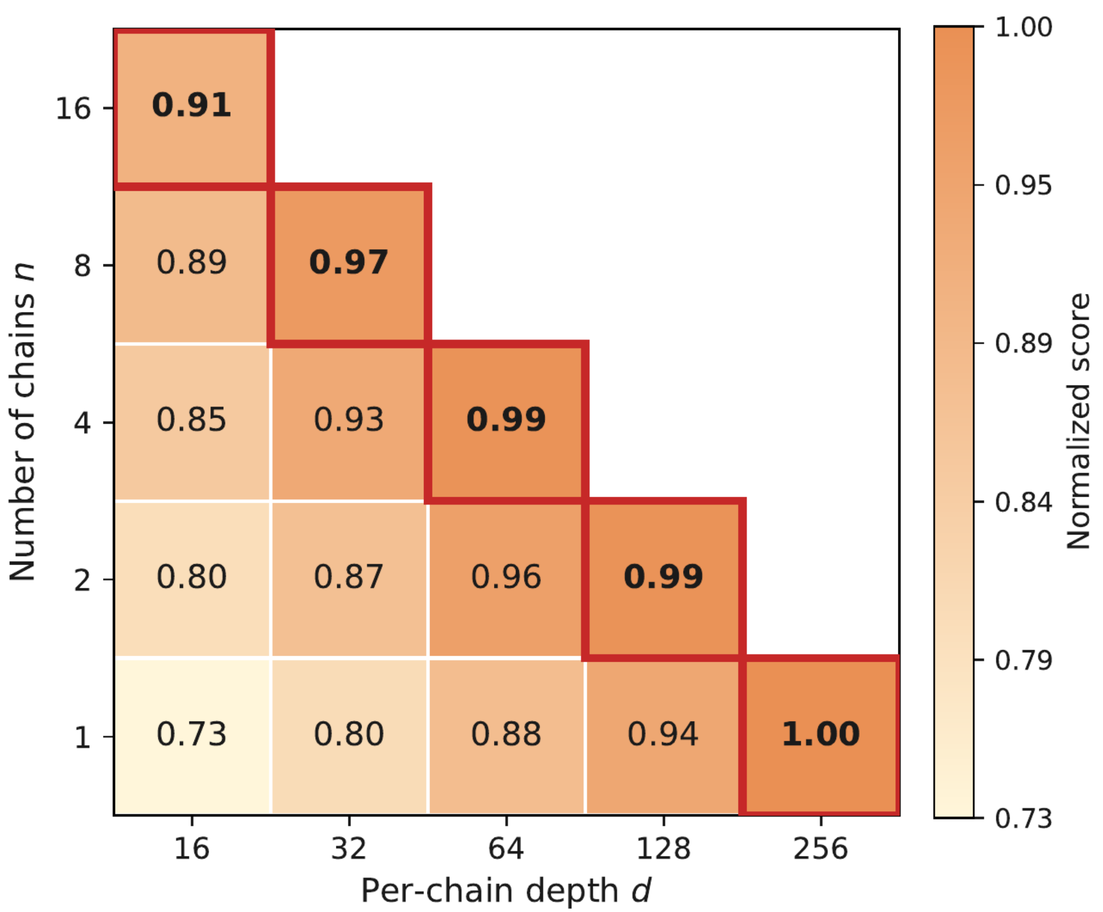
Long-horizon dynamics: diminishing returns



GPT-OSS-120B + OpenEvolve, 500 iterations, all 47 tasks

Improvement events become rarer as $\propto \mathbf{t^{-1}}$; the k-th improvement shrinks as $\propto \mathbf{k^{-1}}$ (both $R^2 > 0.83$). Median 7 improvements per task. Most progress is early, but late improvements still decide final margins.

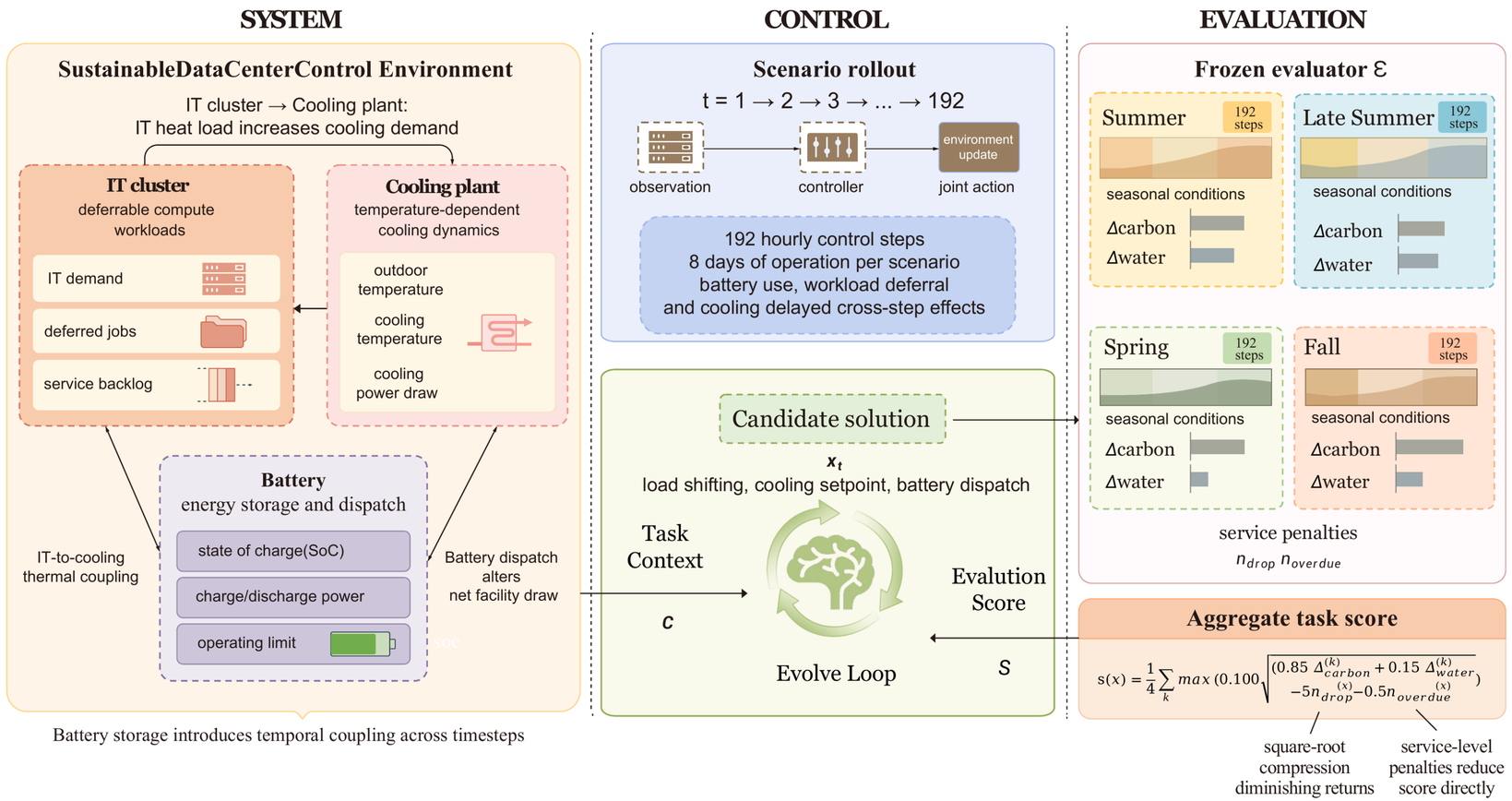
Depth vs. width under a fixed budget



Normalized best score, B = n × d ≤ 256, 10-task v1-lite subset

- Split a fixed budget into *n* independent chains of depth *d*
- Along the equal-budget diagonal the score falls from **1.00 to 0.91** as n goes from 1 to 16
- Wider search adds shallow diversity but discards accepted edits, evaluator feedback and partial repairs
- On the tested tasks, **preserving and exploiting search context** beats generating more independent candidates

Case study: SustainableDataCenterControl



- Start from a feasible no-op controller; coordinate **load shifting, cooling and battery dispatch**
- 4 seasonal / carbon scenarios × 192 hourly steps; score = carbon and water reductions minus service penalties
- Many controllers are valid code but operationally poor (deadline violations, one-season exploits)
- Late gains come from small repairs: clipping actions, battery thresholds, less load shifting

Take-aways, limitations and links

Take-aways

- Judge engineering agents by the **best feasible artifact** within budget, not one-shot completion
- Agents already improve artifacts, but gains are **uneven across domains**
- Early candidate quality, search context and model–framework fit all matter
- Multi-objective trade-offs and brittle simulators remain open

Limitations

- Some tasks use reduced-order simulators or emulators; single-agent protocol only
- Equal evaluation budgets do not equalize compute or API cost
- Public tasks may overlap with training data

GitHub

github.com/Einsia/Frontier-Engineering

Project page

lab.einsia.ai/frontier-eng

arXiv

arXiv:2604.12290