



Home > AI News > [AI Builds Custom Decision-Making Software in About an Hour Instead of Months](#)



CHAT

AI Builds Custom Decision-Making Software in About an Hour Instead of Months

 October 9, 2026  AI News  Comment off

LACE framework is the name of a new AI system from Nanyang Technological University, Singapore (NTU Singapore) that writes the specialised decision-making software behind port schedules, delivery routes and staff rosters. A user describes an operational problem in plain language, and the LACE framework builds, tests and refines a set of algorithms to solve it. The researchers say it does in about an hour, for under US\$10 in computing fees, what normally takes engineers months.

The work was published on 1 October 2026 in [Nature Machine Intelligence](#) and reported by [Tech Xplore](#) on 9 October. On four new port and tugboat problems it reached 97% to 99% of the best result found by any method tested, while five rival AI systems, including Google DeepMind's FunSearch, failed to produce a single workable algorithm.

This article explains what the LACE framework does, how its two stages work, and what the benchmark results show. Using the figure data the team published alongside the paper, we also check the “hour” and “\$10” claims, compare the cost of four different [AI models](#) running the system, and set out who could use it and what still needs a human expert.

Table of contents

- [What the LACE Framework Is](#)
- [How the LACE Framework Works](#)
- [Benchmark Results: LACE Framework vs Other AI Methods](#)
- [The Port and Tugboat Test: LACE Framework on New Problems](#)
- [Does the LACE Framework Really Take an Hour and \\$10?](#)
- [How the LACE Framework Compares With FunSearch and EoH](#)
- [Who Could Use the LACE Framework](#)
- [Limits and Open Questions for the LACE Framework](#)
- [LACE Framework FAQ](#)
- [References](#)

CHAT

What the LACE Framework Is



LACE stands for [Large Language Model](#) (LLM)-driven Algorithm Construction via Complementary Evolution. It belongs to a growing family of systems that use an LLM not to answer a question directly, but to write and improve computer programs that solve a hard problem.

CHAT

The problem it solves

The target is combinatorial optimisation: choosing the best option from an enormous number of possible combinations. It sits under “critical global operations”, the NTU team says, “including maritime port berth scheduling, hospital operating room allocations, supply chain routing and power grid management”. Exact solvers often cannot finish on large real-world instances in the time available, so organisations rely on heuristics: smart rules of thumb that find very good answers quickly.

Writing those heuristics is the bottleneck. As the paper’s abstract puts it, designing them “for each new problem variant requires deep domain knowledge and months of expert iteration”. When a port adds a berth or a hospital changes its shift rules, the software often has to be reworked by specialists.

Who built it

The study was led by assistant professor Yan Ran and research fellow Dr Gong Huatian of NTU's School of Civil and Environmental Engineering, with co-authors from The Hong Kong Polytechnic University, the University of Liverpool and National Taiwan University. It was received by the journal on 8 March 2026, accepted on 25 August and published on 1 October. Funding came from Singapore's A*STAR and Ministry of Education, the Japan Science and Technology Agency and the UK's EPSRC.

The headline numbers

Measure	Result	Source
Average score, 36 classical problems	0.945 vs 0.870 for the best rival AI method	Paper abstract
An LLM (o3-mini) prompted directly, no framework	0.571	Paper abstract
Four new port and tugboat problems	0.97 to 0.99 of the best result found	Paper abstract
Rival LLM methods that produced nothing feasible	Five	Paper abstract
Time per problem	48 to 68 minutes, depending on the AI model	Supplementary figure 5
Computing cost per problem	\$1.59 to \$45.08, depending on the AI model	Supplementary figure 5
Code and data	Open source, MIT licence, 40 problems and 7,109 test instances	GitHub repository

CHAT

How the LACE Framework Works



The key idea is that the LLM should spend its effort on algorithm design, not on rewriting the plumbing of each problem. The [project's GitHub repository](#) describes two stages.

CHAT

Stage one: a verified problem contract

First, LLM agents break the plain-language problem into four modules, which the authors call the I-O-T-H interface. The Input schema defines the data available, the Output schema defines what a valid answer looks like, the Tool library holds a feasibility checker, an objective evaluator and domain helpers, and the Heuristic portfolio is the set of algorithms to be evolved. Each module is admitted only after a smoke test, and only once an LLM-written heuristic can solve through it from end to end.

The abstract calls this “a verified problem contract under which the LLM operates, directing model capacity to high-level algorithmic reasoning rather than problem-specific implementation”. In plain terms, the LACE framework makes the boring, error-prone parts reliable before it asks the model to be clever.

Stage two: complementary evolution

Next, the LACE framework improves the portfolio under a strict runtime budget. Seven operators propose changes: five generate new or improved heuristics, and two repair ones that crashed or produced invalid answers. A mathematical optimisation step (a mixed-integer linear program) then keeps a set of 10 heuristics that, between them, cover every training instance well. The paper-scale settings in the repository use eight evolution rounds and a limit of 10 seconds per instance.

That is the “complementary” in the name. Instead of searching for one algorithm that is best on average, the LACE framework keeps a team of specialists, each strong on different kinds of instance, and uses whichever does best on each case.

Why a portfolio beats one clever algorithm

The team’s ablation data shows how much the team approach adds. On the four new port problems, a portfolio of one heuristic scored 0.912 on average; ten complementary heuristics scored 0.980. The gain was even larger on location and assignment problems, which rose from 0.814 to 0.989.

Score on the four new port problems by portfolio size (supplementary figure 4b)



1 heuristic	0.912
3 heuristics	0.959
5 heuristics	0.970
10 heuristics	0.980

CHAT

Why the tool library matters

The second ablation tested the tool library. With no tools, 36% of generated heuristics failed on the new port problems; with 15 tools, the failure rate fell to 6.7%. For scheduling problems it fell from 22% to 4.1%. Giving the model trusted building blocks, such as a ready-made feasibility checker, stops it from wasting attempts on code that breaks the rules.

Benchmark Results: LACE Framework vs Other AI Methods



CHAT

The main test used 36 classical problems from CO-Bench, an academic benchmark for [AI agents](#) that search for optimisation algorithms. They span scheduling, routing and graphs, packing and cutting, location and assignment, and selection and knapsack problems.

36 classical problems

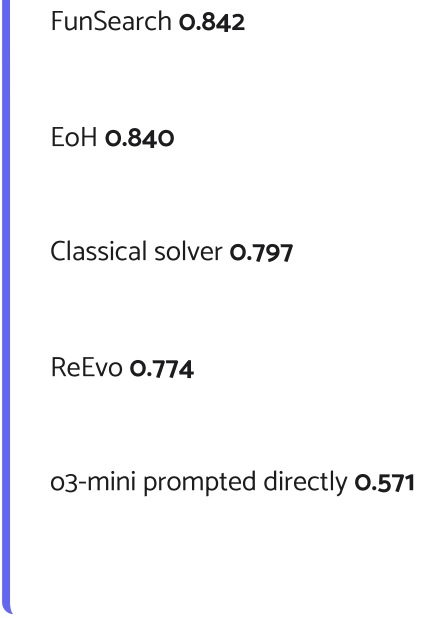
Scores are relative to the best known solution for each instance, so 1.0 means matching it. The interactive supplementary page publishes every method's average.

Average score across 36 CO-Bench problems (supplementary figure 2a)

LACE **0.945**

EoH-S **0.870**

HeurAgenix **0.855**



Method	Score
FunSearch	0.842
EoH	0.840
Classical solver	0.797
ReEvo	0.774
o3-mini prompted directly	0.571

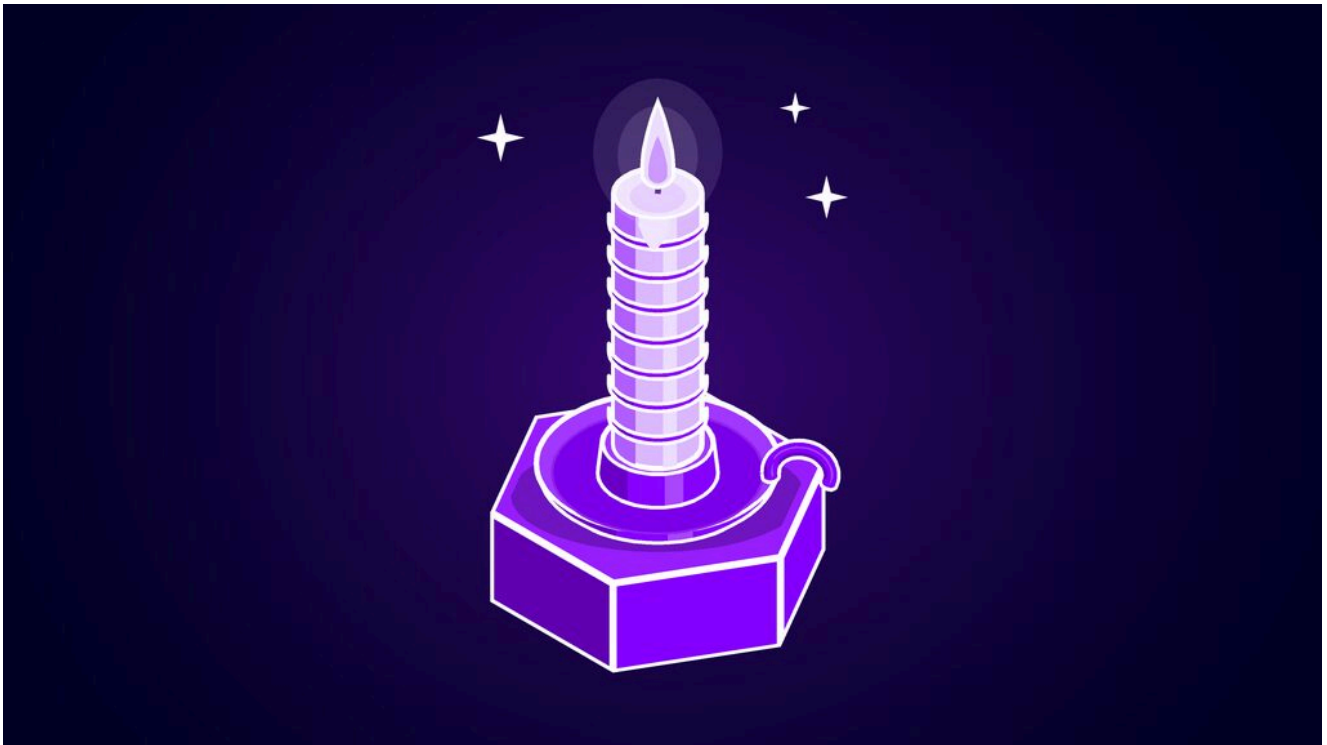
The valid rate tells a similar story. The LACE framework produced a valid solution on 94.4% of instances, against 61.1% for EoH-S, the strongest earlier LLM method. By problem family, it scored 0.990 on location and assignment, 0.982 on packing and cutting, 0.954 on scheduling, 0.894 on selection and knapsack, and 0.890 on routing and graphs.

The framework, not the model

The most important comparison is the bottom bar. Asking an LLM (here, OpenAI's o3-mini) to write solver directly scored 0.571. The LACE framework scored 0.945. The abstract says this locates "the gain in the framework rather than the model". Lead author Yan made the same point to Tech Xplore: "clever design of an AI framework, rather than just building larger AI models, can bridge this gap."

The Port and Tugboat Test: LACE Framework on New Problems

CHAT



Benchmarks can flatter a system if similar problems appeared in a model's [training data](#). So the team built four “structurally new” maritime problems for which no published optimum exists.

CHAT

Four problems no method had seen

The first is a port scheduling problem: each arriving vessel needs an inbound tug tow, a berth and an outbound tow in sequence, using berths and tugs of different sizes, and the port may leave some vessels unserved at a heavy penalty. The other three are tugboat routing and scheduling variants: a single base, multiple bases, and tugs that can change speed. Each was run with five random seeds, and scores are relative to the best result any method found on each instance.

LACE framework vs Gurobi and metaheuristics

The rivals included Gurobi, a leading commercial optimisation solver, with time limits of one minute and 30 minutes; six classic metaheuristics, such as genetic algorithms and ant colony optimisation; and six LLM-based systems. Five of the LLM systems (EoH-S, FunSearch, EoH, Greedy Refine and ReEvo) failed to produce any feasible algorithm.

Method (mean of 5 seeds)	Port scheduling	Tug routing	Multi-base tugs	Variable-speed tugs
LACE	0.981	0.990	0.978	0.969
Genetic algorithm	0.963	0.852	0.797	0.571
Ant colony	0.957	0.852	0.800	0.582
Gurobi, 30 minutes	0.627	0.637	0.597	0.402
Gurobi, 1 minute	0.527	0.434	0.494	0.283
HeurAgenix (LLM-based)	0.580	0.380	0.540	0.230
Five other LLM methods	No feasible algorithm	No feasible algorithm	No feasible algorithm	No feasible algorithm

Our analysis: the gap grows with difficulty

CHAT

Averaged across the four problems, the LACE framework scored 0.980, the genetic algorithm 0.79 Gurobi with 30 minutes 0.566 and HeurAgenix 0.433 (our averages of the published means). On the easiest problem, port scheduling, the best metaheuristic came within two points of LACE. On the hardest, variable-speed tugs, the best metaheuristic reached 0.582 against LACE's 0.969, a gap of 39 points. As problems pick up more interacting rules, hand-tuned general methods fall away and problem-specific heuristics matter more. That is exactly where months of expert work used to go.

Does the LACE Framework Really Take an Hour and \$10?



The headline claims come from the NTU team’s own description: “about an hour for each problem and at a cost of under US\$10 in computation fees per problem”. The supplementary page lets us check them, because it reports the LACE framework run with four different AI models across all 4c problems.

CHAT

The four AI models

AI model driving LACE	Average score	Tokens	Cost	Wall-clock time
Gemini 3 Flash Preview	0.951	3.03M	\$1.59	48 min
OpenAI o3 Mini	0.961	3.86M	\$9.21	59 min
DeepSeek V4 Pro	0.810	3.50M	\$2.07	68 min
Claude Sonnet 4.6	0.981	7.33M	\$45.08	67 min

The figures are per-problem averages. On that basis, the “about an hour” claim holds for every model (48 to 68 minutes), and the “under \$10” claim holds for three of the four. Claude Sonnet 4.6 delivered the best score, 0.981, but at \$45.08 a problem, about 28 times the cost of Gemini 3 Flash

(\$45.08 divided by \$1.59) for three more points. For most organisations, the cheaper models will be the sensible starting point, with a stronger model kept for problems where the last few per cent are worth money.

Where the hour goes

The supplementary page also breaks the time down. With Gemini 3 Flash, generating heuristics took 10 minutes, repairing them 4 minutes, selecting the portfolio 4 minutes, and evaluating candidates on training instances 30 minutes. Evaluation was the largest share for every model, between 30 and 39 minutes.

Where a 48-minute LACE run goes with Gemini 3 Flash (supplementary figure 5b)

Evaluating heuristics on training instances **30 min**

Generating heuristics **10 min**

Repairing failed heuristics **4 min**

Selecting the portfolio **4 min**

CHAT

Bar widths are each step's share of the 48 minutes (30 divided by 48 is 62.5%). The practical point: most of the hour is ordinary computing, running candidate algorithms against test data, not the AI model thinking. Faster machines or fewer training instances would shorten a LACE framework run, while a cheaper model mainly cuts the bill.

What “months” compares against

The comparison with months of work comes from the authors. “Traditional software designs can take months and cost hundreds of thousands,” Gong told Tech Xplore. That figure is not broken down in the paper, and the LACE framework still needs someone to describe the problem, supply realistic instances and check the results. But even allowing generously for that work, the shift from months of specialist coding to an hour of computing and a review is large.

How the LACE Framework Compares With FunSearch and EoH

LACE builds on several years of research into LLM-guided algorithm discovery. The best known is Google DeepMind’s [FunSearch](#), published in Nature in 2023, which paired an LLM with an evaluator to discover new mathematical constructions. Others include Evolution of Heuristics (EoH), ReEvo and HeurAgenix, plus AlphaEvolve, DeepMind’s later coding agent. Related work has even used LLM-driven search to discover new [reinforcement learning](#) algorithms.

System	Approach	CO-Bench score in the LACE paper
FunSearch (2023)	LLM proposes program changes; an evaluator keeps the best	0.842
EoH (2024)	Evolves ideas and code together	0.840
ReEvo (2024)	Adds reflection on why heuristics worked	0.774
HeurAgenix (2025)	Agent that evolves and selects heuristics	0.855
EoH-S (2026)	Evolves a set of heuristics	0.870
LACE (2026)	Verified contract plus complementary portfolio under a time budget	0.945

CHAT

The difference in the LACE framework is less about a smarter search and more about structure. Earlier systems usually assume someone has already written the scaffolding for each problem. LACE generates and checks that scaffolding itself, which is why it could tackle the new port problems when most rivals could not start.

Who Could Use the LACE Framework

Yan argues that “by turning months of manual coding into a roughly one-hour automated process, we are making high-performance operational optimization accessible to organizations of all sizes”. The team names regional logistics hubs, public health care institutions, utility providers and smaller research teams that “lack multimillion-dollar algorithm engineering budgets”.

What you need to try it

The [supplementary information page](#) and repository are unusually complete. They include the 40 problems, 7,109 test instances, all 400 evolved heuristics, the operator prompts and a Colab notebook. Shipped portfolios can be scored with no API key. A full run needs Python 3.10 or later, an OpenRouter API key and a problem description with sample instances. The default model is Gemini 3 Flash Preview, and no commercial solver is required. The team reports that a clean run of all 40 problems made 659 LLM calls with none failing, and 39 of 40 problems reached full feasibility on held-out tests.

Where it fits in a business

The best candidates are recurring decisions with clear rules and a measurable goal: delivery routing, shift rostering, warehouse slotting, production sequencing, maintenance scheduling and room or bed allocation. Many UK firms solve these today with spreadsheets or a vendor’s fixed rules. If the operating rules change often, the ability to regenerate a tailored solver in an hour is the real prize. Our report on [AI drilling optimisation](#) shows the same pattern in a different industry, and teams building this in-house may want help with [ML model development](#).

CHAT

What still needs a human expert

Someone still has to state the problem correctly, decide what counts as a good answer, and supply instances that reflect reality. A heuristic that looks excellent on test data can still break a rule nobody wrote down. The generated code also needs review before it controls anything safety-critical, and the usual cybersecurity checks apply to any code produced by an AI model. Our article on the [decision model layer for AI agents](#) covers the governance side of automated decisions.

Limits and Open Questions for the LACE Framework

The results are strong, but they come with caveats that a buyer or researcher should keep in mind.

Relative scores

Scores are measured against the best known or best found solution, not against a proven optimum. On the four new problems, “best found” means the best of the methods tested, so a 0.98 says LACE nearly matched the strongest competitor, not that its answers are perfect.

Problems designed by the authors

The four new maritime problems were built by the team, which has deep expertise in port logistics. Independent groups testing the LACE framework on their own messy, real-world problems will be the real test. The open code makes that possible.

Model and cost changes

Costs depend on model prices, which change quickly, and DeepSeek V4 Pro’s lower score (0.810) shows that results are not identical across models. Anyone budgeting for the LACE framework should test two or three models on their own problem before choosing one.

CHAT

LACE Framework FAQ

What does LACE stand for?

Large Language Model-driven Algorithm Construction via Complementary Evolution.

Who developed the LACE framework?

Researchers at Nanyang Technological University, Singapore, led by assistant professor Yan Ran and Dr Gong Huatian, with co-authors from Hong Kong, Liverpool and Taiwan.

Is the LACE framework free to use?

The code is open source under the MIT licence. You pay for the AI model calls through OpenRouter, which the team measured at \$1.59 to \$45.08 per problem depending on the model.

Does it replace optimisation experts?

No. It automates the slow work of writing and tuning heuristics, but people still need to define the problem, provide data and check the results.

What problems can the LACE framework solve?

Combinatorial optimisation problems such as scheduling, routing, packing, facility location and knapsack selection. The paper tested 40 of them, including four new port and tugboat problems.

References

[Large language models discover complementary heuristics for combinatorial optimization \(Nature Machine Intelligence\)](#)

[AI builds custom decision-making software in about an hour instead of months \(Tech Xplore\)](#)

[LACE reference implementation \(GitHub\)](#)

[LACE Supplementary Information \(interactive results\)](#)

[LACE code and data archive \(Zenodo\)](#)

[CO-Bench: Benchmarking Language Model Agents in Algorithm Search for Combinatorial Optimization \(arXiv\)](#)

[Mathematical discoveries from program search with large language models \(Nature\)](#)

Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model (arXiv)

ReEvo: Large Language Models as Hyper-Heuristics with Reflective Evolution (arXiv)

HeurAgenix: Leveraging LLMs for Solving Complex Combinatorial Optimization Challenges (arXiv)

AlphaEvolve: A coding agent for scientific and algorithmic discovery (arXiv)

Get expert help with this

Want this delivered by our team instead of doing it yourself? Here is how we can help.

Managed IT Services

End-to-end patch management, monitoring and 24/7 support for production workloads.

Cloud Adoption

Migrate, modernise and run securely on AWS, Azure or GCP with our cloud architects.

DevOps Engineering

CI/CD, infrastructure-as-code and SRE practices that ship safer, faster releases.

Digital Transformation

Strategy + delivery across people, process and technology – measurable outcomes.

IT Outsourcing

Augment or fully outsource your IT function with a UK-based, ITIL-aligned team.

Chester IT Support

On-site and remote IT support for Chester, Wirral and the wider North West.

[Talk to a specialist](#) · [About Progressive Robot](#) · [Browse all tutorials](#)

CHAT