Automated Algorithm Design with Large Language Models

Zhichao Lu

Department of Computer Science

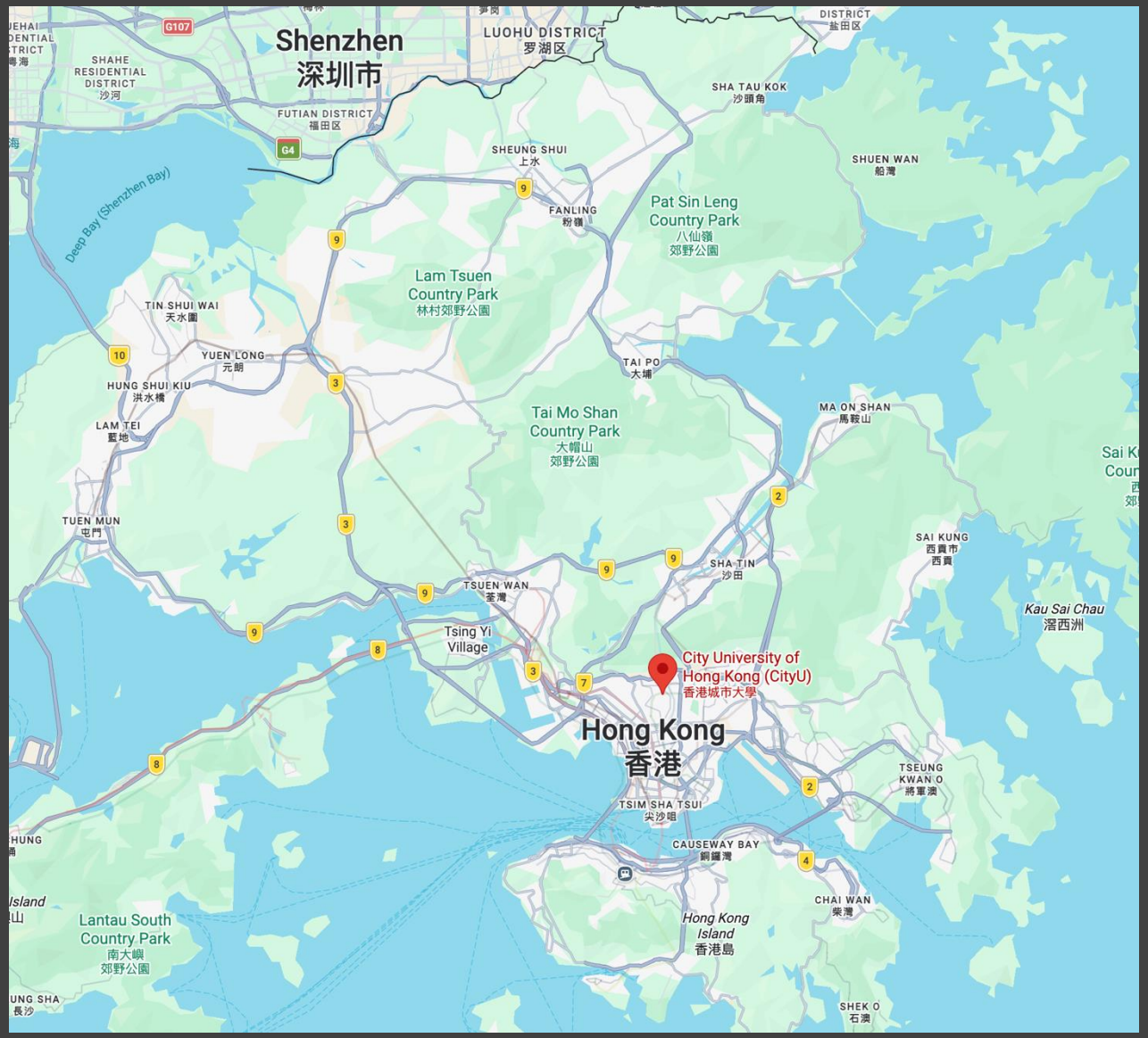
City University of Hong Kong

 zhichao.lu@cityu.edu.hk

Joint work with **CityUHK Optima Group, SUSTech, Huawei, and CATL**

香港城市大學
City University of Hong Kong

>>> City University of Hong Kong (CityUHK)



CS @ CityUHK Statistics

Faculty Members	53
PhD Students	378

>>> Learning and Optimization for Machine Intelligence (Optima) Group

❑ Team

- Prof. Qingfu Zhang and Dr. Zhichao Lu
- 20+ PhD students
- 10+ Postdocs, RAs and visiting scholars/students

❑ Projects

- RGC/ANR,GRF,ITF
- NFSC, MOT
- Guangdong and Shenzhen Governments
- Huawei, CATL, ...

Prof. Qingfu Zhang

Dr. Zhichao Lu

https://optima.cs.cityu.edu.hk/

>>> Today's Agenda

Automated Algorithm Design (AAD)

Mechanism: **Large language models (LLMs)** + **Evolutionary computation (EC)**

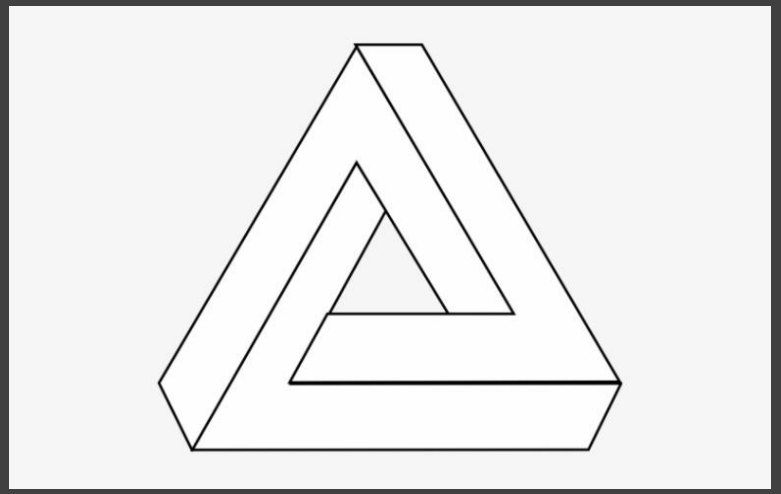
>>> Why Automated Algorithm Design (AAD)?

- **Intelligence**: Ability to generate **knowledge** (theorem, theory, algorithms, stories, movies).
- **Algorithm** = A method in a computer to solve a well-defined problem (not = code).
- **Limitations** of human design:
  - Communication bottleneck btw computer and human brain.
  - Cognitive and physical capacity.
- Significant **impacts** of **AI for AAD**:
  - **Understand how AI generates knowledge**.
    Revolutionize many application fields.

>>> Why So Many Algorithms (No Free Lunch Theorem[1])?

❑ Impossible Triangle for NP-Complete Problem

Any Instance



Optimality **Fast=Polynomial Time**

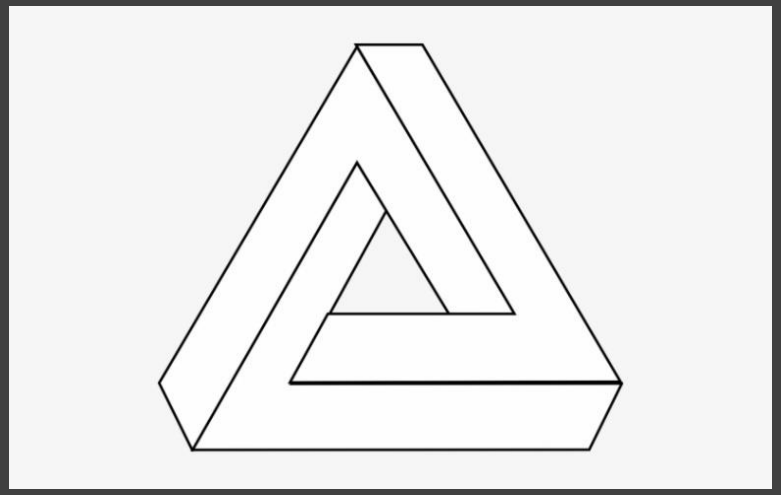
No algorithm can produce the optimal solution to every instance of a NPC problem in polynomial time

[1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," in IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67-82, April 1997

>>> Why So Many Algorithms (No Free Lunch Theorem[1])?

- Unless P=NP, no polynomial-time algorithm can produce an k-approximately optimal solution to all the instances of the TSP

Any Instance



Approximate Optimality

Fast=Polynomial Time

- k-approximately optimality:

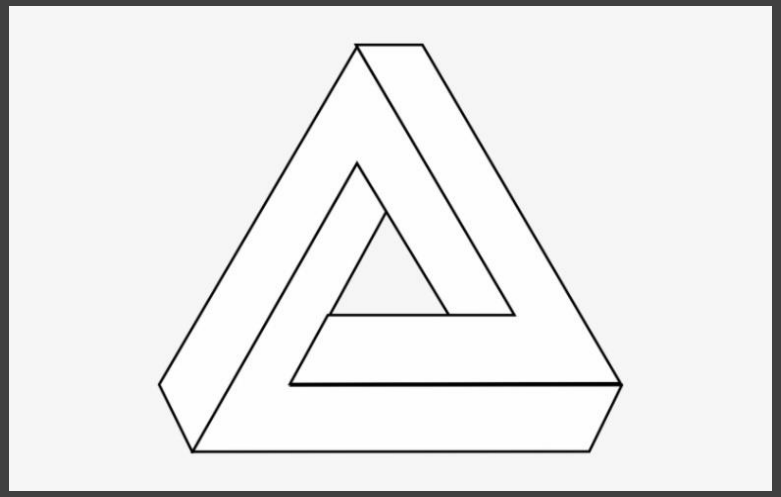
$$\frac{OPT(I)}{A(I)} \leq k$$

***Infeasible to design one fast approximation algorithm for all instances*.**

[1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," in IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67-82, April 1997

>>> Why So Many Algorithms (No Free Lunch Theorem[1])?

Subset **Instance**



Approximate **Optimality**

Fast=Polynomial Time

- ❑ **Feasible Approach: to design algorithms for a subset of instances that**
  - ❑ **are sampled from a distribution,**
  - ❑ **have some specific properties.**

[1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," in IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67-82, April 1997

>>> AAD Prior to LLM Era: Hyperparameter Tuning[1]

Algorithm 1 AdamW Optimizer

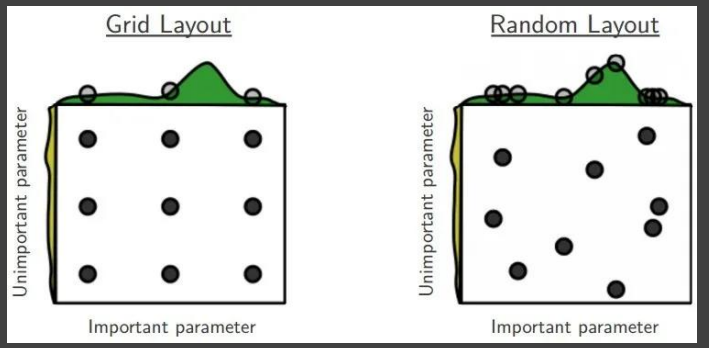
given $\beta_1, \beta_2, \epsilon, \lambda, \eta, f$
initialize $\theta_0, m_0 \leftarrow 0, v_0 \leftarrow 0, t \leftarrow 0$
while θ_t not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
  update EMA of g_t **and** g_t^2
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  bias correction
  $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
  $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
  update model parameters
  $\theta_t \leftarrow \theta_{t-1} - \eta_t (\hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) + \lambda \theta_{t-1})$
end while
return θ_t

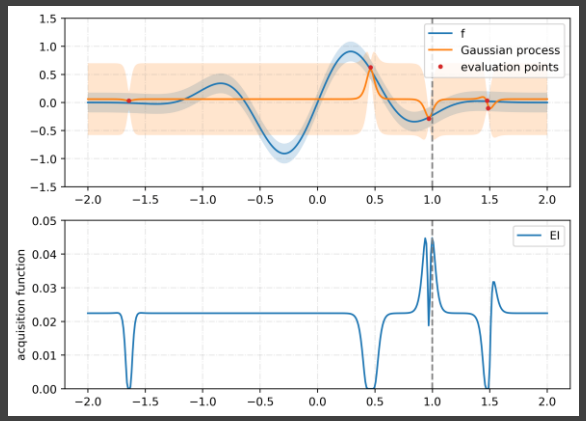
Baseline algorithm

- β_1, β_2: initial decay rates for estimating 1st and 2nd moments.
- η: learning rate.

Identify **key** hyperparameters

Random search[2]





Bayesian optimization[3]

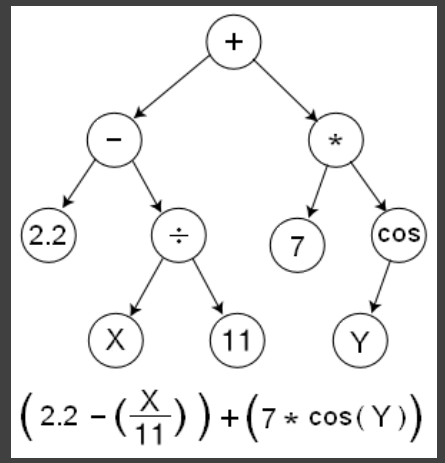
✓ Existing optimization techniques can be readily applied;

× Search a small vicinity around the baseline;

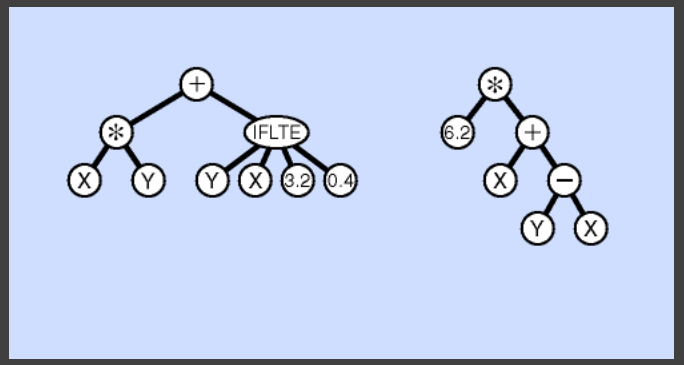
❑ Useful for improving efficacy of existing algorithms rather than designing new algorithms.

[1] Feurer, Matthias, and Frank Hutter. "Hyperparameter optimization." Automated machine learning: Methods, systems, challenges (2019): 3-33.
[2] Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." Journal of machine learning research 13.2 (2012).
[3] Falkner, Stefan, Aaron Klein, and Frank Hutter. "BOHB: Robust and efficient hyperparameter optimization at scale." International conference on machine learning. PMLR, 2018.

>>> AAD Prior to LLM Era: Genetic Programming[1]

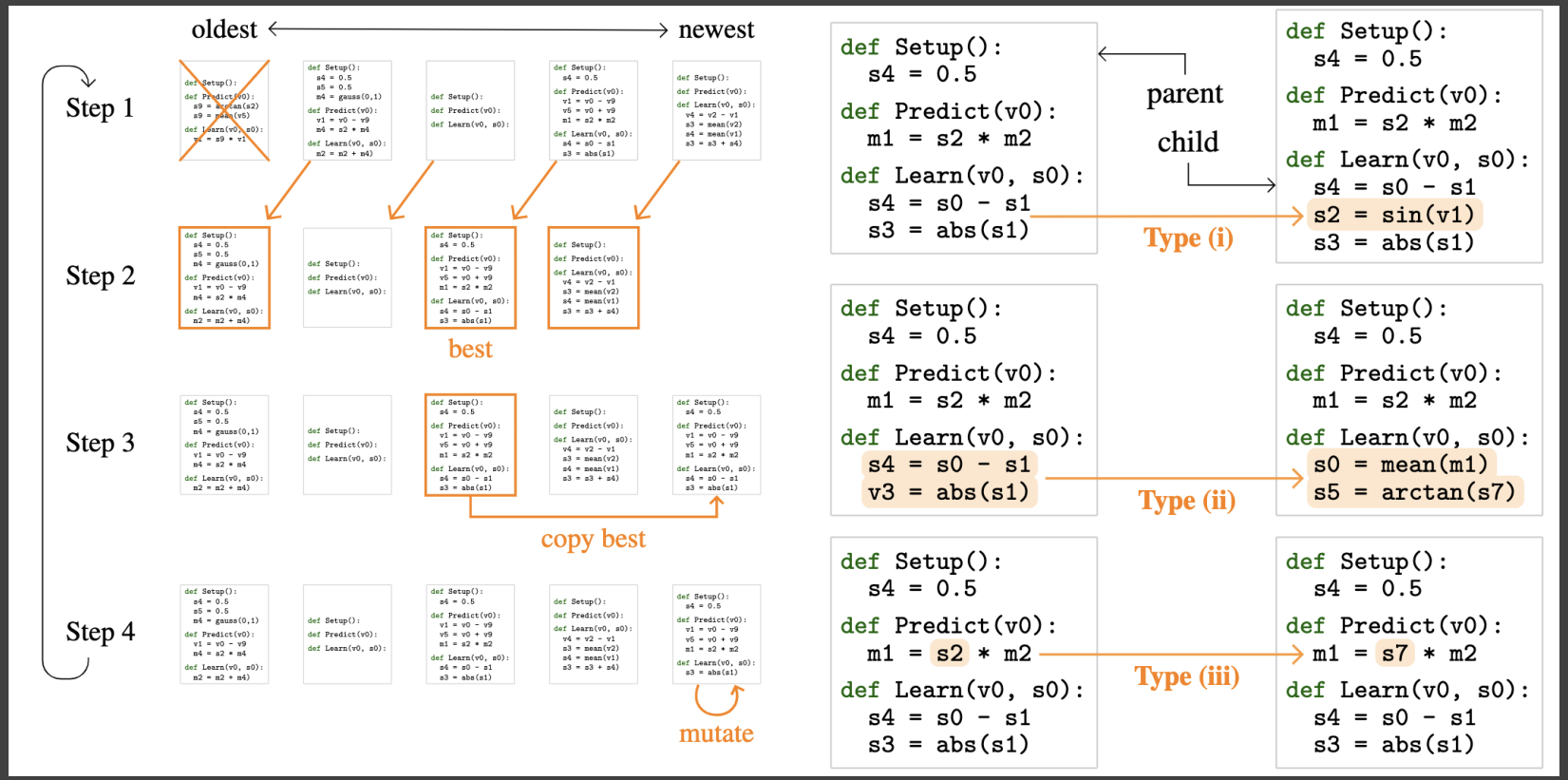


Represent algorithms as *trees*



Better algorithms are created by exchanging subparts between trees

An Application on AutoML[2]



✓ A flexible yet intuitive approach towards AAD;

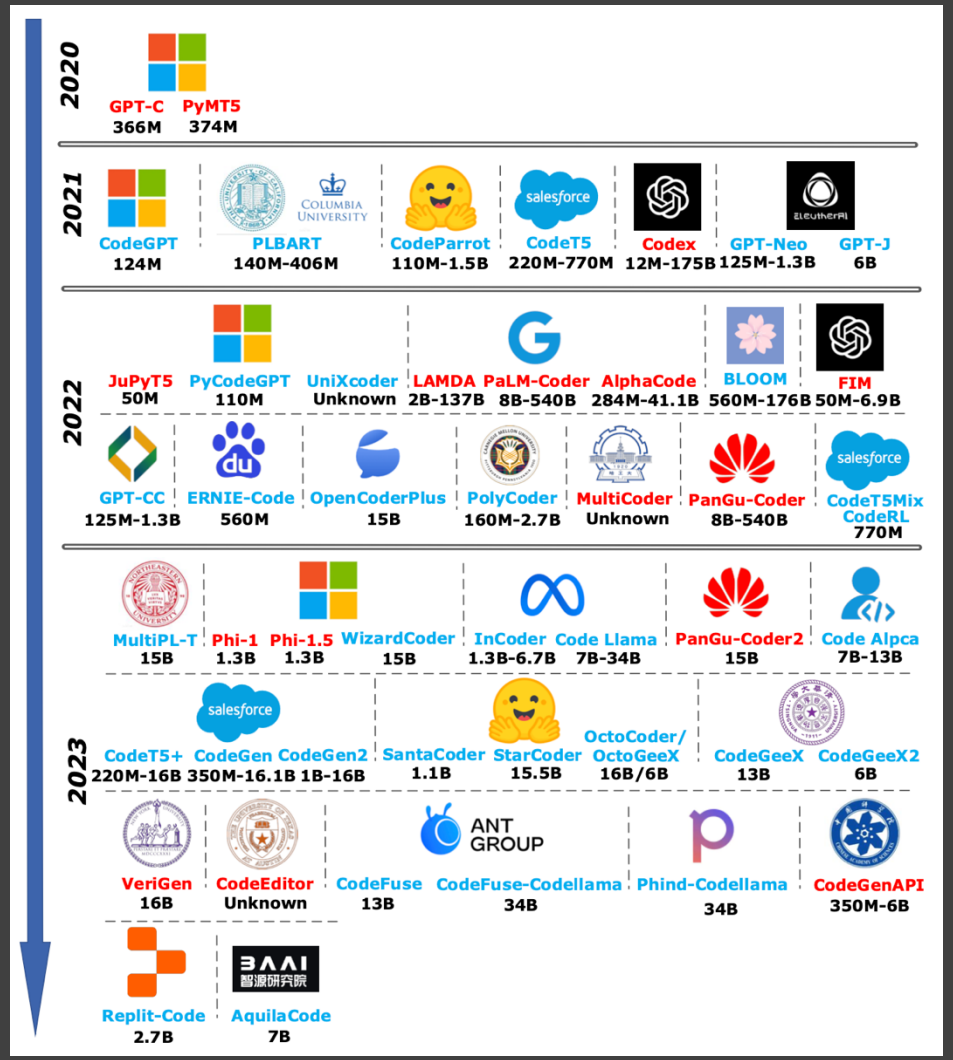
× Pre-define a set of primitives, and low search efficiency;

❑ Despite a long history of existence, GP has yet to gain popularity in the realm of AAD.

[1] Koza, John R. "Genetic programming as a means for programming computers by natural selection." Statistics and computing 4 (1994): 87-112.

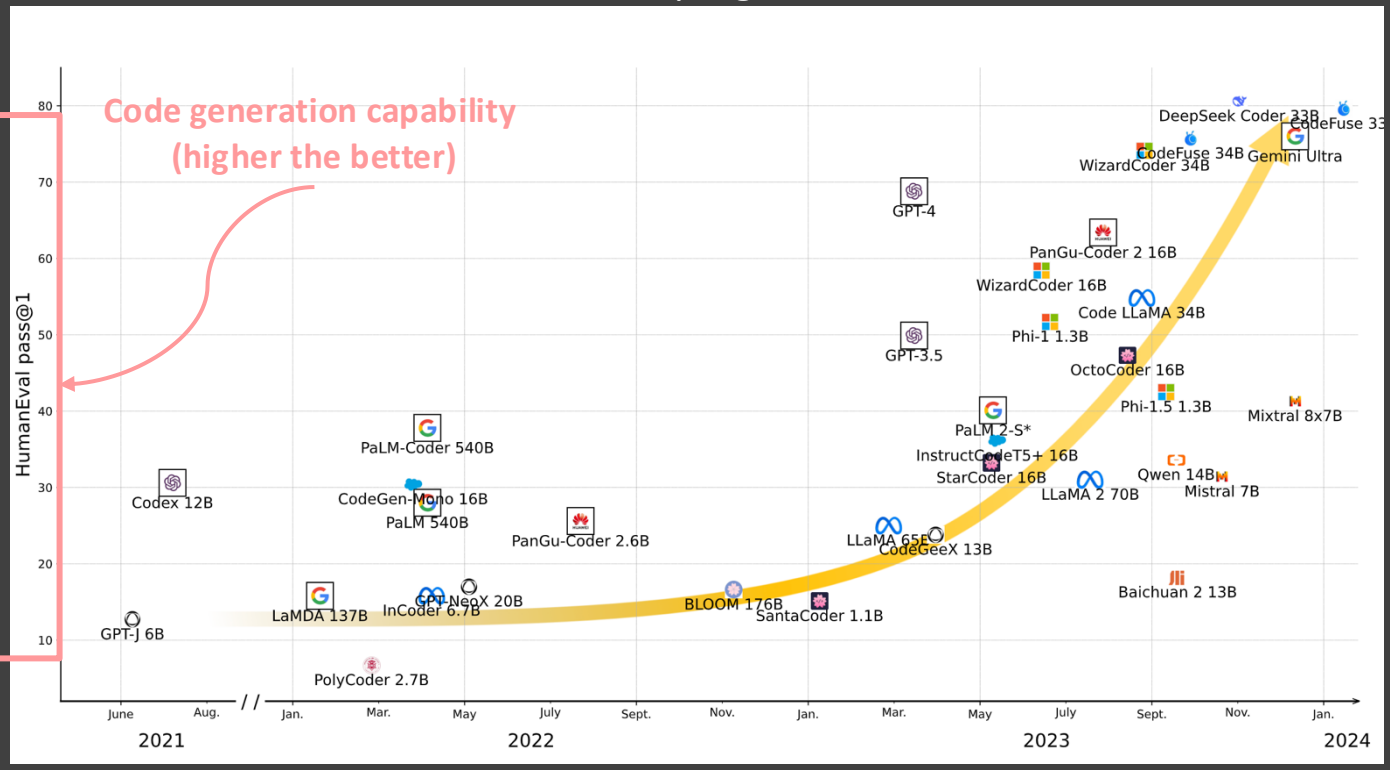
[2] Real, Esteban, et al. "AutoML-Zero: Evolving machine learning algorithms from scratch." International conference on machine learning. PMLR, 2020. (**Humies Gold Award**)

>>> Advent of Code LLMs



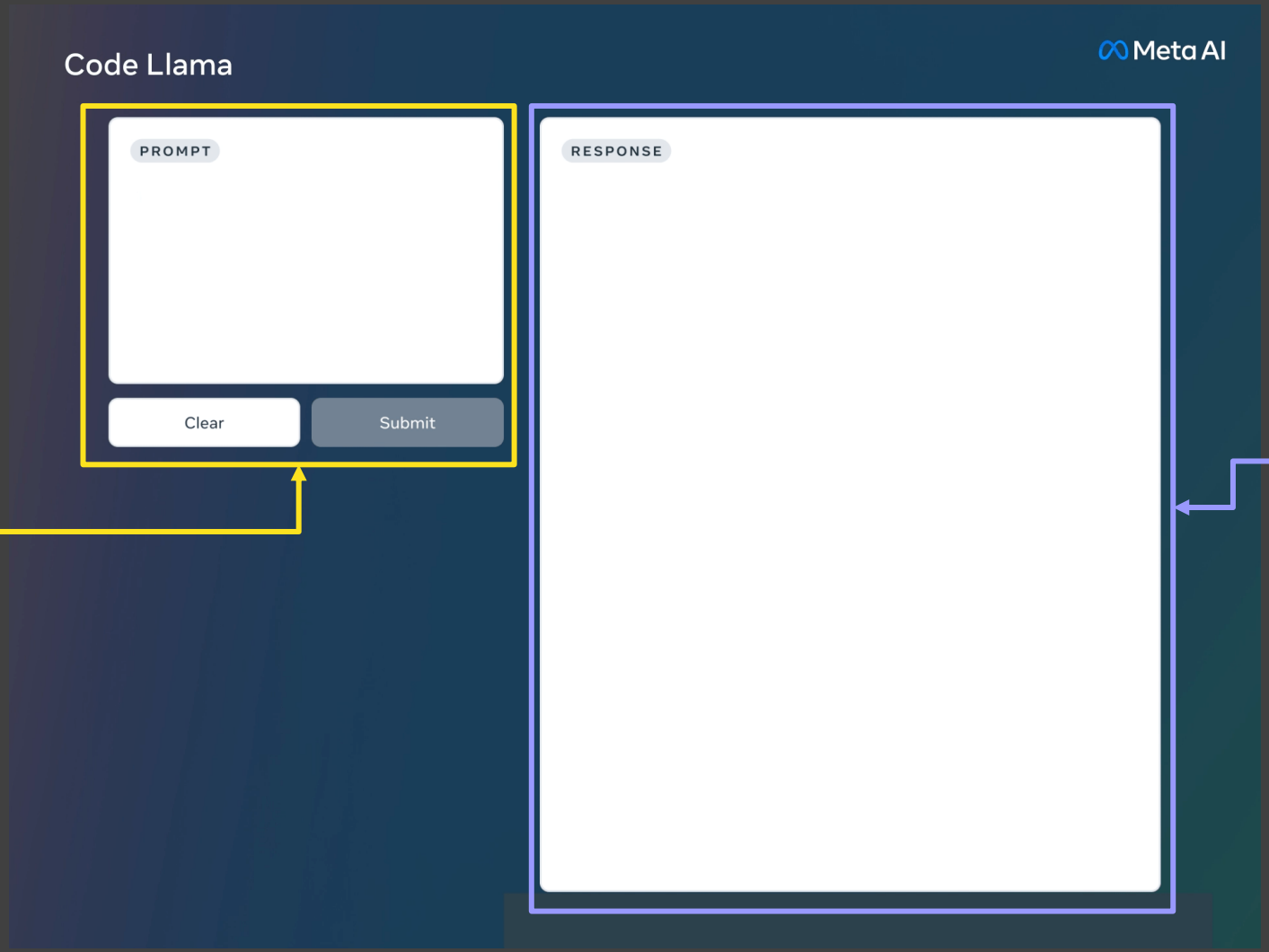
(Source: arXiv 2311.10372)

The timeline of LLMs' progress on Code Generation



(Source: arXiv 2311.07989)

>>> AAD in the Era of Code LLMs



Users simply input desired algorithm specifications

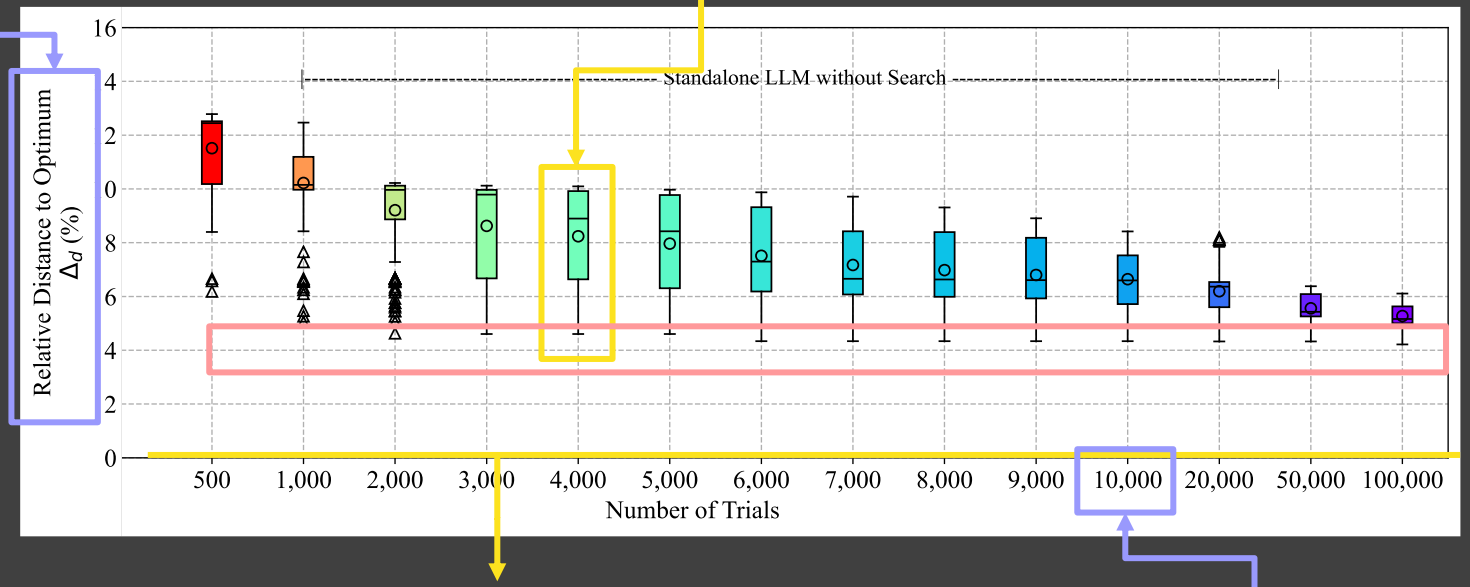
LLMs directly output an algoritnm and evaluate its performance

(Source: https://ai.meta.com/blog/code-llama-large-language-model-coding/)

>>> Are Code LLMs All We Need for AAD?

Performance gap to best-known algorithms (Lower the better)

Performance of Top-5 algorithms designed by LLMs



Performance of best-known algorithms

# of independent trials allowed for LLMs

- AAD Problems:
  - Admissible Set (Mathematics)
  - Online Bin Packing
  - Traveling Salesman
- Choice of LLM:

OpenAI GPT-3.5 Turbo

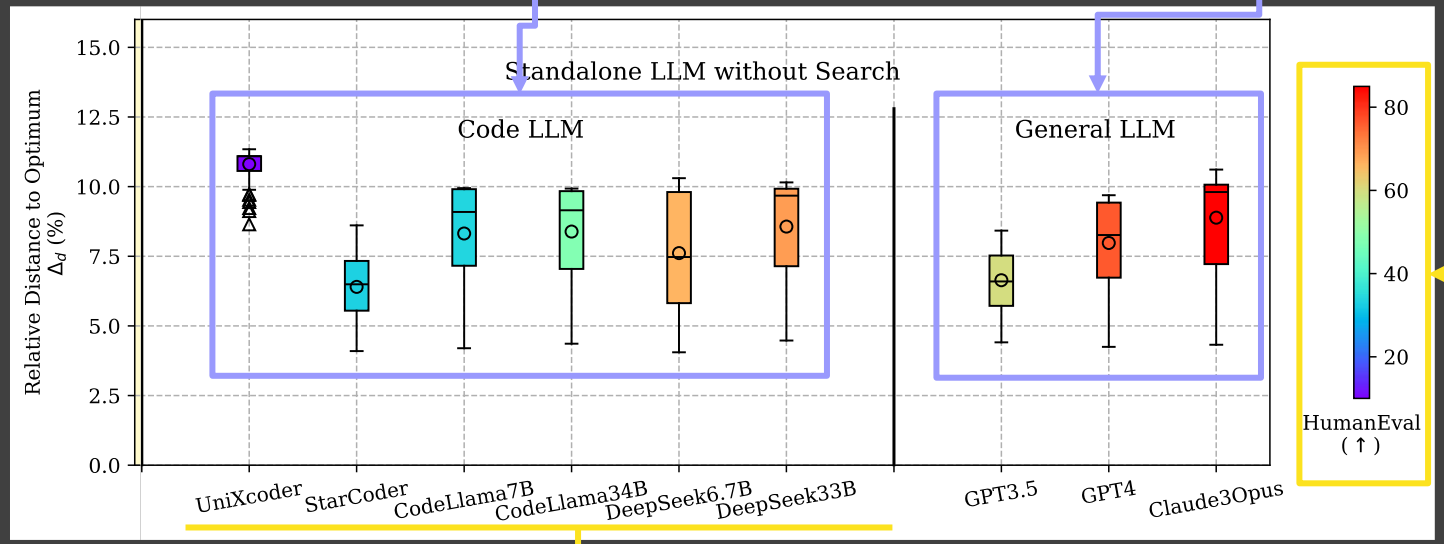
∗ *Simply letting a LLM try many times is insufficient for AAD*

[1] Rui Zhang, Fei Liu, Xi Lin, Zhenkun Wang, Zhichao Lu, Qingfu Zhang. "Understanding the Importance of Evolutionary Search in Automated Heuristic Design with Large Language Models.", **PPSN 2024 (Ours)**

>>> Are Code LLMs All We Need for AAD?

LLMs fine-tuned on Codes

General purpose LLMs



Ascending order in # of model parameters

Performance on regular code generation task (Higher/more red the better)

- AAD Problems:
  - Admissible Set (Mathematics)
  - Online Bin Packing
  - Traveling Salesman
- Query Budget: 10,000

∗ *Simply importing more capable LLMs is also insufficient for AAD.*

[1] Rui Zhang, Fei Liu, Xi Lin, Zhenkun Wang, Zhichao Lu, Qingfu Zhang. "Understanding the Importance of Evolutionary Search in Automated Heuristic Design with Large Language Models.", **PPSN 2024 (Ours)**

>>> Are Code LLMs All We Need for AAD? -- NO

- The inherent generative capability of **LLMs alone is insufficient for AAD**, which holds true under:
  - Increased query budget
  - More capable models

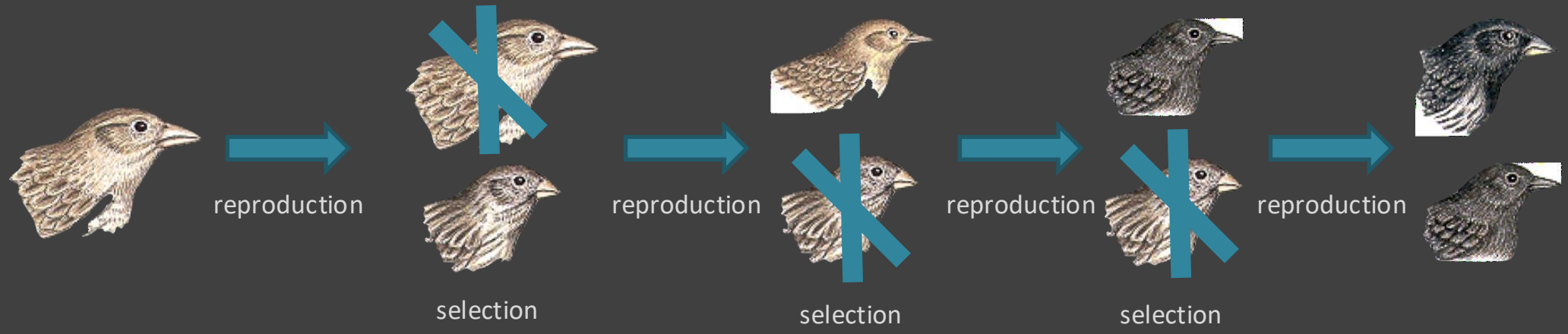
- Suggest the necessity of coupling LLMs with a search strategy for AAD:

Evolutionary Computation

[1] Rui Zhang, Fei Liu, Xi Lin, Zhenkun Wang, Zhichao Lu, Qingfu Zhang. "Understanding the Importance of Evolutionary Search in Automated Heuristic Design with Large Language Models.", **PPSN 2024 (Ours)**

>>> Evolutionary Computation (EC)

Computational paradigm simulating the evolutionary processes in nature, widely used for solving complex modeling and optimization problems

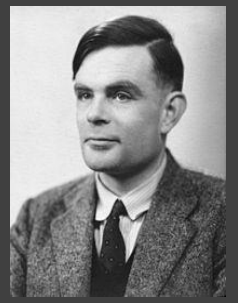


In 1950, Turing described how evolution might be used for

building intelligent machine

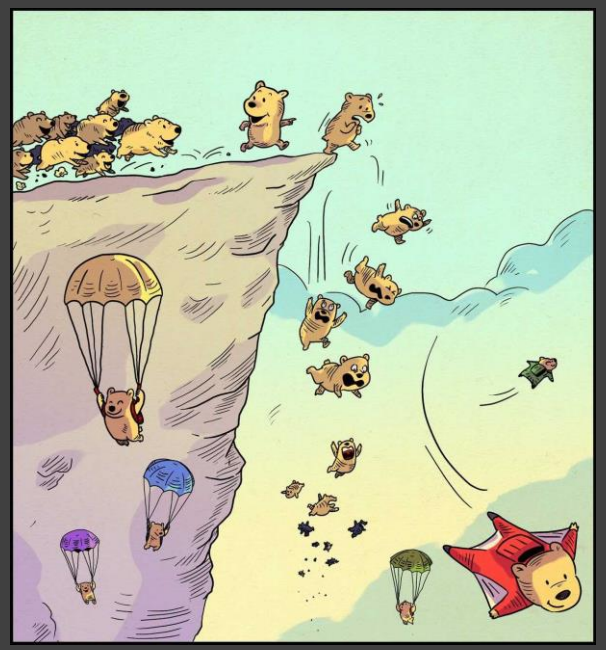
"Structure of the child machine = Hereditary material
Changes of the child machine = Mutations
Judgment of the experimenter = Natural selection"

[A. M. Turing. Computing machinery and intelligence. Mind 49: 433-460, 1950.]



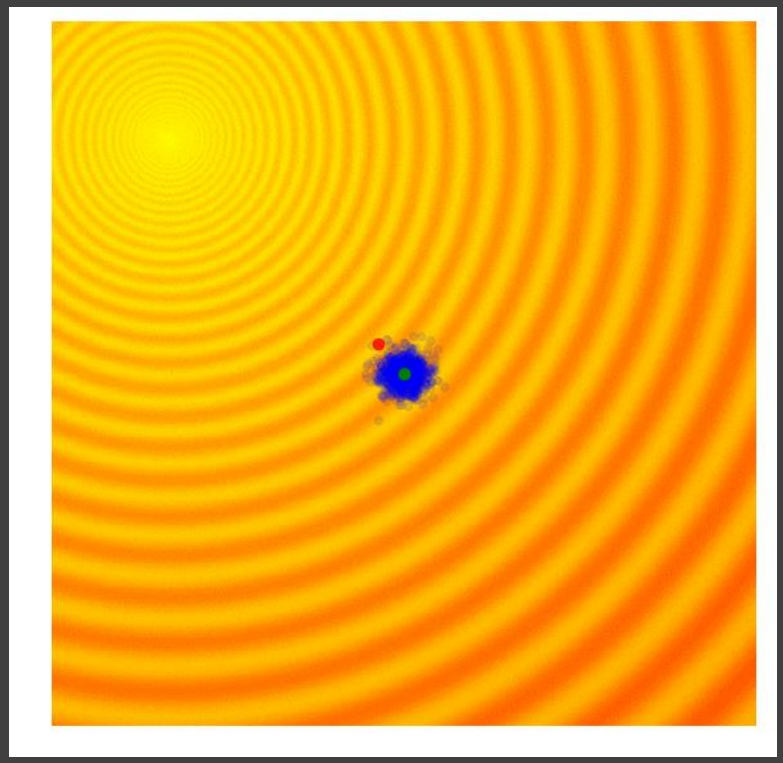
>>> Why EC: General and Flexible

EC harnesses the principle of survival of the fittest, reflecting nature's way of evolving and adapting to environments over time, bringing diversity, complexity, and robustness.



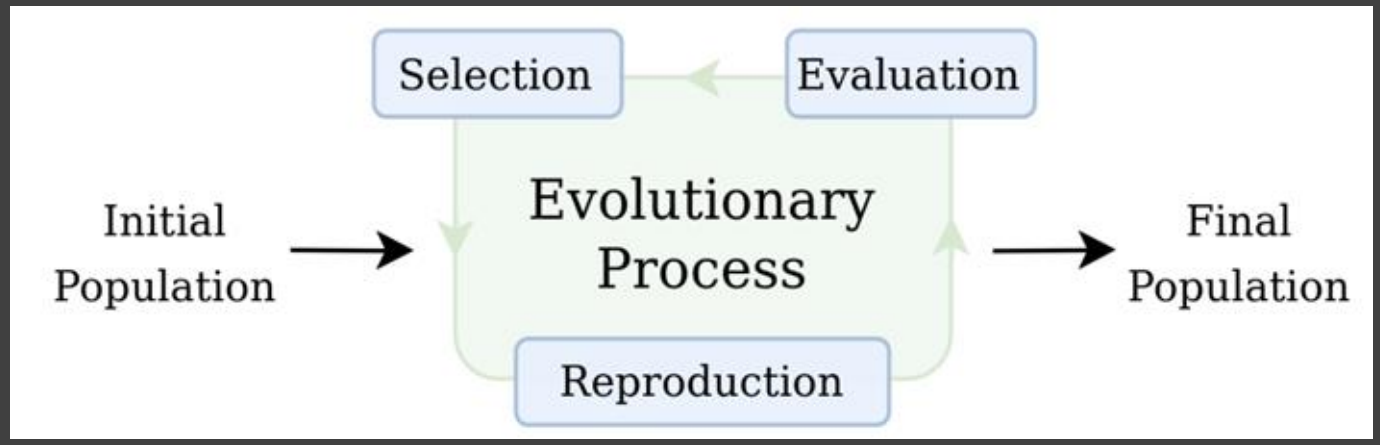
(Source: https://blog.otoro.net/2017/10/29/visual-evolution-strategies/)

Survival of the Fittest



>>> EC Framework

- Simple, yet highly modular:
  - ★ Genetic Algorithms
  - ★ Genetic Programming
  - ★ Evolution Strategies
  - ★ Particle Swarm Optimization
  - ★ Ant Colony Optimization
  - ★ ….



>>> EC in Engineering Design

High-speed train head design


Series 700

evolve


Series N700

save 19% energy

Technological overview of the next generation Shinkansen high-speed train Series N700

M. Ueno[1], S. Usui[1], H. Tanaka[1], A. Watanabe[2]

[1]*Central Japan Railway Company, Tokyo, Japan,* [2]*West Japan Railway Company, Osaka, Japan*

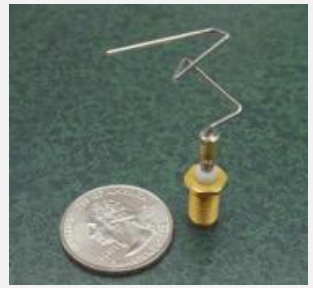
waves and other issues related to environmental compatibility such as external noise. To combat this, an aero double-wing-type has been adopted for nose shape (Fig. 3). This nose shape, which boasts the most appropriate aerodynamic performance, has been newly developed for railway rolling stock using the latest analytical technique (i.e. genetic algorithms) used to develop the main wings of airplanes. The shape resembles a bird in flight, suggesting a feeling of boldness and speed.

On the Tokaido Shinkansen line, Series N700 cars save 19% energy than Series 700 cars, despite a 30% increase in the output of their traction equipment for higher-speed operation (Fig. 4).

This is a result of adopting the aerodynamically excellent nose shape, reduced running

Antenna design

evolve



38% efficiency

93% efficiency

Computer-Automated Evolution of an X-Band Antenna for NASA's Space Technology 5 Mission

Gregory. S. Hornby Gregory.S.Hornby@nasa.gov
University Affiliated Research Center, NASA Ames Research Park, UC Santa Cruz at Moffett Field, California, 94035

Jason D. Lohn Jason.Lohn@sv.cmu.edu
Carnegie Mellon University, NASA Ames Research Park and Moffett Field, California 94035

this, different combinations of the two evolved antennas and the QHA were tried on the ST5 mock-up and measured in an anechoic chamber. With two QHAs 38% efficiency was achieved, using a QHA with an evolved antenna resulted in 80% efficiency, and using two evolved antennas resulted in 93% efficiency. Here "efficiency" means how

>>> EC in Science

The Nobel Prize in Chemistry 2018

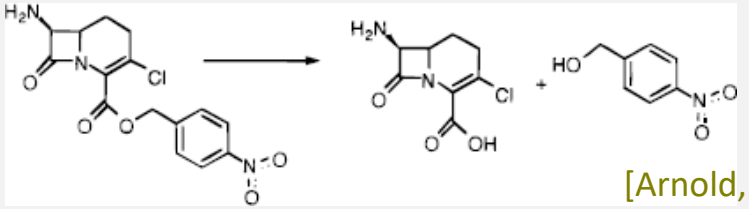
© Nobel Media AB. Photo: A. Mahmoud
Frances H. Arnold
Prize share: 1/2

© Nobel Media AB. Photo: A. Mahmoud
George P. Smith
Prize share: 1/4

© Nobel Media AB. Photo: A. Mahmoud
Sir Gregory P. Winter
Prize share: 1/4

The Nobel Prize in Chemistry 2018 was divided, one half awarded to Frances H. Arnold "for the directed evolution of enzymes", the other half jointly to George P. Smith and Sir Gregory P. Winter "for the phage display of peptides and antibodies."

Protein design

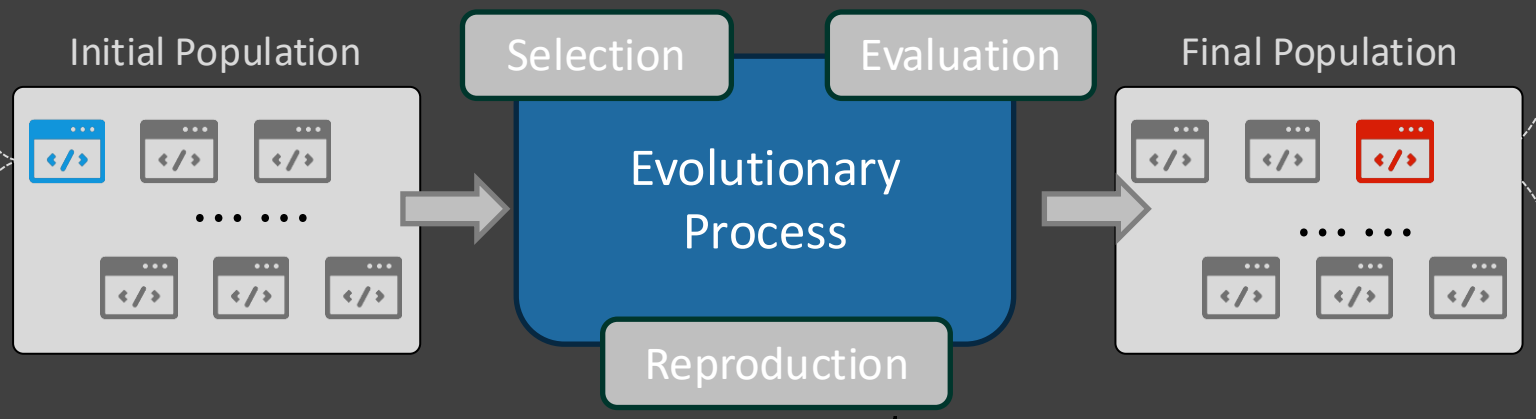


[Arnold, 1998]

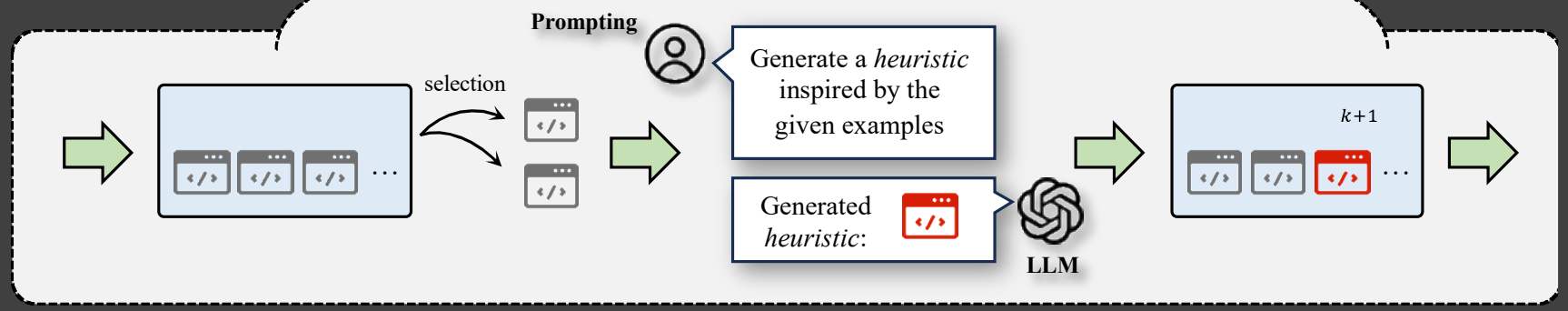
"Evolution—the adaption of species to different environments —has created an enormous diversity of life. Frances Arnold has used the same principles – genetic change and selection – to develop proteins that solve humankind's chemical problems. In 1993, Arnold conducted the first directed evolution of enzymes, which are proteins that catalyze chemical reactions. The uses of her results include more environmentally friendly manufacturing of chemical substances, such as pharmaceuticals, and the production of renewable fuels."

>>> LLM-based Evolutionary Program Search (EPS)[1,2,3]

```python
def initial_algorithm(
    item: float,
    bins: list) -> list:
    """Returns priority with
    which we want to add item
    to each bin.
    Args:
        item: Size of item to
    be added to the bin.
        bins: List of
    capacities for each bin.
    Return:
        List of the same size
    as bins with priority
    scores for each bin.
    """
    scores = 0.0
    return scores
```




```python
def final_algorithm(
    item: float,
    bins: list) -> list:
    """Returns priority with which
    we want to add item to each bin.
    Args:
        item: Size of item to be
    added to the bin.
        bins: List of capacities for
     each bin.
    Return:
        List of the same size as
    bins with priority scores for
    each bin.
    """
    d = bins - item
    e = np.exp(d)
    r = np.sqrt(d)
    u = r * (1 - d / bins)
    adj = np.where(d > 3*item, u +
    0.8, u + 0.3)
    scores = bins / e*(e + 0.7)
    scores = utility + adj
    return scores
```



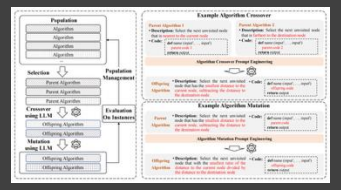
1. Algorithm is modeled as executable computer **programs;**
2. New algorithms are created via **LLMs**

[1] Liu, Fei, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. "Algorithm evolution using large language model." arXiv preprint arXiv:2311.15249 (2023).
[2] Romera-Paredes, Bernardino, et al. "Mathematical discoveries from program search with large language models." Nature 625.7995 (2024): 468-475.
[3] Zhang, Rui, et al. "Understanding the importance of evolutionary search in automated heuristic design with large language models." PPSN, 2024.

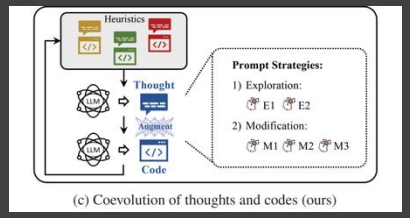
>>> Recent LLM-EPS Progress: Algorithms

Ours

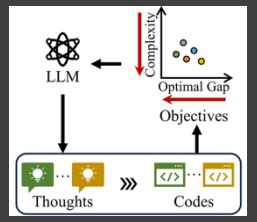
2023-11-26 AEL: Algorithm Evolution using LLM



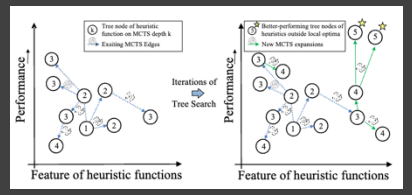
2024-01 EoH: Evolution of Heuristics … Using LLM (ICML 2024 Oral)



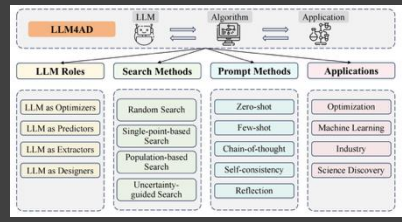
2024-09 MEoH: Multi-objective Evolution of Heuristic using LLM (AAAI 2025 Oral)



2025-05 MCTS-AHD: MCTS for AAD w/ LLMs (ICML 2025)

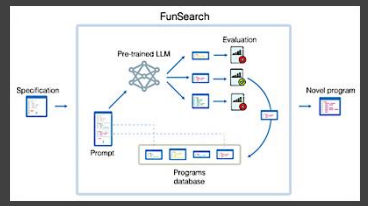


2026-01 A Systematic Survey on AAD w/ LLMs (ACM CSUR)

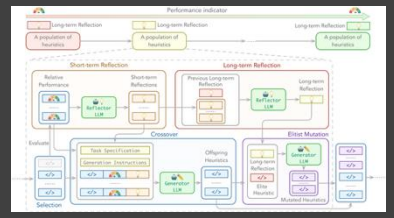


Others

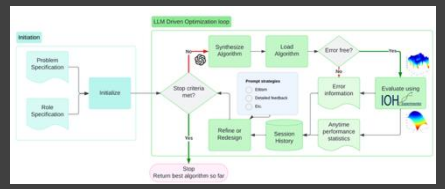
2023-12-14 FunSearch (Nature, Google DeepMind)



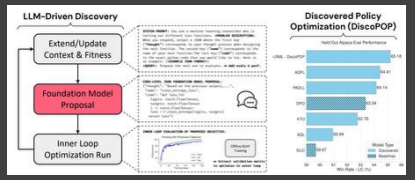
2024-02 ReEvo (NeurIPS 2024)



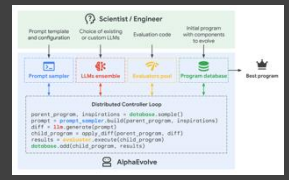
2024-05 LLaMEA (TEVC, Leiden University)



2024-06 DiscoPOP (NeurIPS 2025)



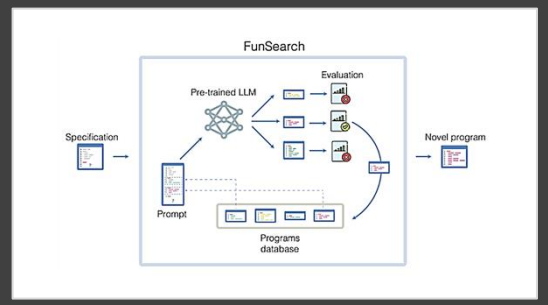
2025-05 AlphaEvolve (Google DeepMind)



- Our European colleagues & we held Dagstuhl Seminar 26191 — "Large Language Models for Evolutionary Computation" (May 2026).

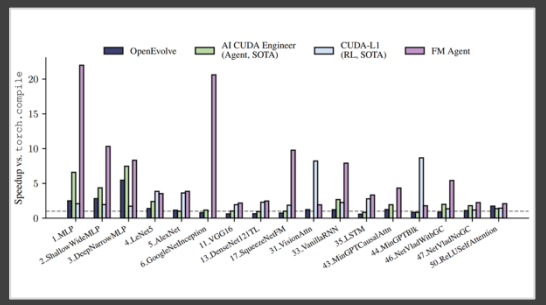
Ours — AEL: Liu et al., "Algorithm Evolution using Large Language Model," arXiv 2023 · EoH: Liu et al., "Evolution of Heuristics," ICML 2024 · MEoH: Yao et al., "Multi-objective Evolution of Heuristics," AAAI 2025 · MCTS-AHD: Zheng et al., "MCTS for Automatic Heuristic Design," ICML 2025 · Survey: Liu et al., "A Systematic Survey on LLMs for Algorithm Design," ACM Computing Surveys 2026
Others — FunSearch: Romera-Paredes et al., "Mathematical discoveries from program search with LLMs," Nature 2024 · ReEvo: Ye et al., "LLMs as Hyper-Heuristics with Reflective Evolution," NeurIPS 2024 · LLaMEA: van Stein & Bäck, "LLaMEA: LLM Evolutionary Algorithm for Automatically Generating Metaheuristics," IEEE TEVC 2024 · DiscoPOP: Lu et al., "Discovering Preference Optimization Algorithms with and for LLMs," NeurIPS 2025 · AlphaEvolve: Novikov et al., "AlphaEvolve: A Coding Agent for Scientific and Algorithmic Discovery," 2025

>>> Recent LLM-EPS Progress: Applications



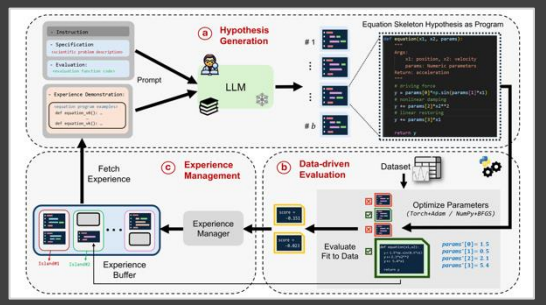
Mathematics

FunSearch: new cap-set bound (Nature ’24)
AlphaEvolve: beats Strassen 4×4 (’25)
extremal combinatorics · records



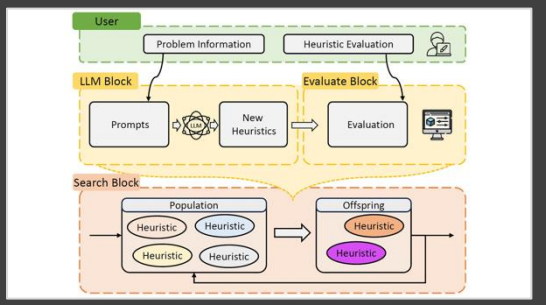
Engineering & Systems

AlphaEvolve: 23% Gemini kernel (’25)
FM Agent: 20× GPU kernels (Baidu ’25)
scheduling · circuits · compilers



Science

LLM-SR: beats SOTA symbolic reg. (’25)
AI Scientist: auto research (Sakana ’24)
equations · materials · biology



Operations Research

EoH: heuristics beat hand-crafted (’24)
ReEvo: reflective hyper-heuristics (’24)
LLaMEA · MEoH: black-box & multi-obj.

Industrial Adoption — Commercial Production

Google DeepMind
AlphaEvolve
In production — Borg scheduling, TPU circuit design, Gemini kernel speedups.

Sakana AI
ShinkaEvolve · AI Scientist
Sample-efficient program evolution; fully-automated scientific discovery.

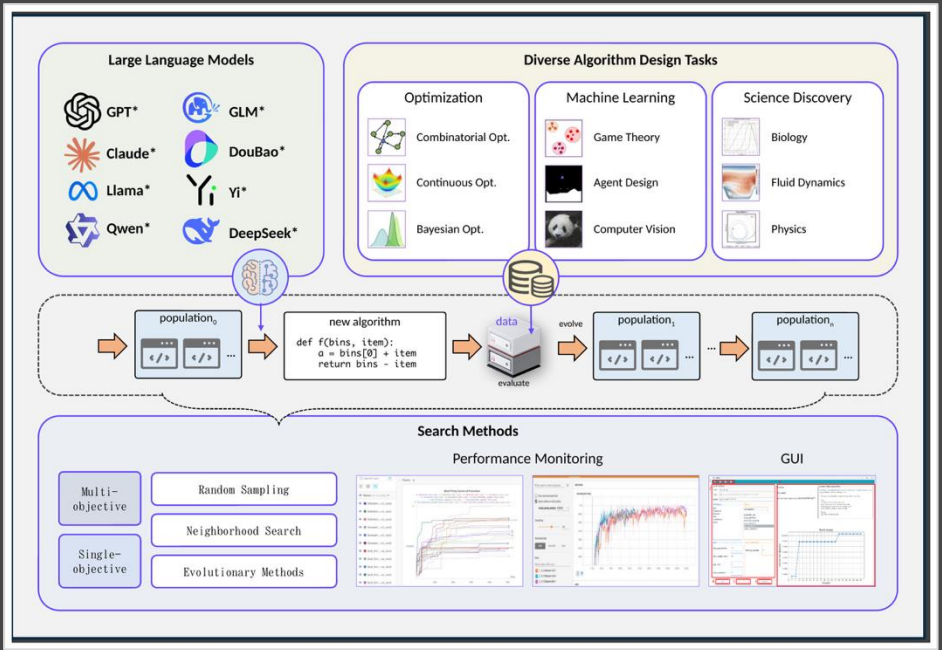
Baidu AI Cloud
FM Agent
LLM + large-scale evolutionary search — 20× GPU kernels, MLE-Bench, math SOTA.

FunSearch: Nature 2024 · AlphaEvolve: DeepMind 2025 · LLM-SR: ICLR 2025 · EoH: ICML 2024 · ReEvo: NeurIPS 2024 · AI Scientist: Sakana 2024 · FM Agent: Baidu 2025 (arXiv:2510.26144) · ShinkaEvolve: Sakana 2025 · Survey (LLM4AD): ACM CSUR 2025 (arXiv:2410.14716)

>>> LLM4AD & LLM4AD_Next: Getting Started with LLM-EPS

You don't have to start from scratch — two platforms, one for each kind of user

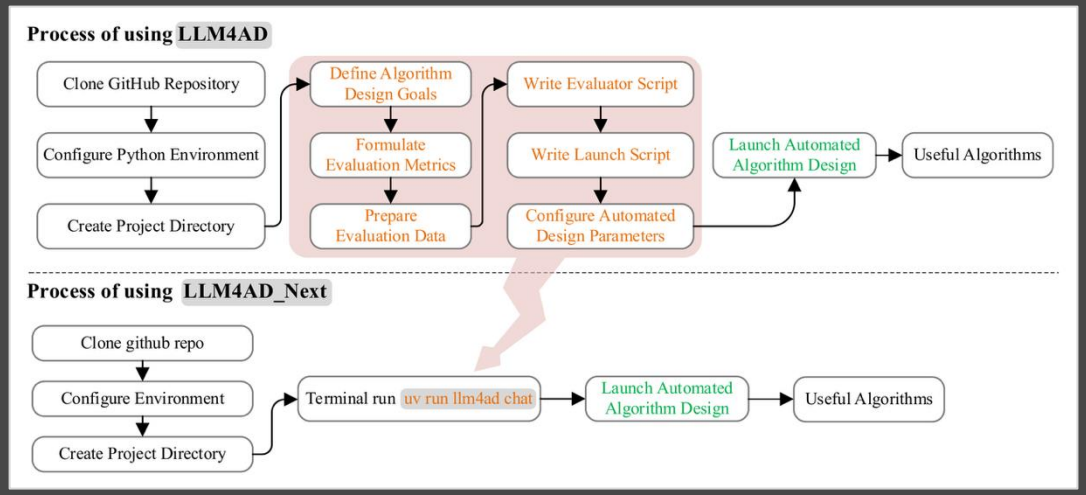
LLM4AD — for researchers



- replicate & rigorously compare methods on one unified platform
- 27+ papers of tasks & benchmarks out of the box

Best fit: academics & advanced DIY developers

LLM4AD_Next — for practitioners



- zero-config — a chat agent builds a runnable pipeline in minutes
- evolves your existing codebase via the built-in Recommender

Best fit: beginners & application engineers

Try it: github.com/Optima-CityU/llm4ad · llm4ad-doc.readthedocs.io · **www.llm4ad.com**

future LLM4AD_Next iterations will fully inherit LLM4AD's strengths

Liu, Fei, et al. "LLM4AD: A Platform for Algorithm Design with Large Language Model." arXiv 2024.

>>> Acknowledgement

With sincere gratitude to my collaborators and students —

City University of Hong Kong — Optima Group

Prof. Qingfu Zhang · **Rui Zhang** (PhD student) · **Qinglong Hu** (PhD student) · **Ruoyu Zhao** (MPhil student) · **Jiaqi Wu** (MPhil student) · **Chenyu Zhu** (Undergraduate student)

Academic collaborators

Prof. Zhenkun Wang (SUSTech) · **Prof. Xi Lin** (Xi'an Jiaotong University) · **Dr. Fei Liu** (ETH Zürich)

Industry collaborators

Dr. Luziwei Leng (Huawei) · **Dr. Guoyang Xie** (CATL) · **Dr. Liyong Lin** (CATL)

>>> THANKS!!

Questions ?