Algorithm Design Using Large Language Model

Fei Liu, Zhichao Lu, Zhenkun Wang, Qingfu Zhang

Department of Computer Science
City University of Hong Kong

School of System Design and Intelligent Manufacturing
Southern University of Science and Technology

fliu36-c@my.cityu.edu.hk

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

March 2025

Fei Liu is a Postdoctoral Researcher at the Department of Computer Science, City University of Hong Kong, Hong Kong. He received his BSc degree and MSc degrees from Northwestern Polytechnical University, China in 2017 and 2020, and Ph.D. degree from the Department of Computer Science, City University of Hong Kong, Hong Kong in 2025. His main research interests include computational intelligence, optimization, and their applications. More information is available at https://feiliu36.github.io/.

Zhichao Lu is an Assistant Professor at the Department of Computer Science, City University of Hong Kong, Hong Kong. He received bachelor's and Ph.D. degrees in electrical and computer engineering from Michigan State University, East Lansing, MI, USA, in 2013 and 2020, respectively. His current research focuses on the intersections of evolutionary computation, learning, and optimization, notably on developing efficient, reliable, and automated machine learning algorithms and systems. He received the GECCO2019 best paper award. More information is available at https://www.cs.cityu.edu.hk/ zhichalu/.

Zhenkun Wang is an Assistant Professor with the School of System Design and Intelligent Manufacturing, Southern University of Science and Technology, Shenzhen, China. He received the Ph.D. degree in circuits and systems from Xidian University, Xi'an, China, in 2016. From 2017 to 2020, he was a Postdoctoral Research Fellow with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, and with the Department of Computer Science, City University of Hong Kong, Hong Kong. His research interests include evolutionary computation, optimization, machine learning, and their applications. More information is available at https://faculty.sustech.edu.cn/?tagid=wangzk3.

Qingfu Zhang is a Chair Professor at the Department of Computer Science, City University of Hong Kong, Hong Kong. His main research interests include evolutionary computation, optimization, neural network, data analysis, and their applications. Prof. Zhang has been a highly cited researcher in computer science since 2016. More information is available at https://www.cs.cityu.edu.hk/ qzhan7/index.html.

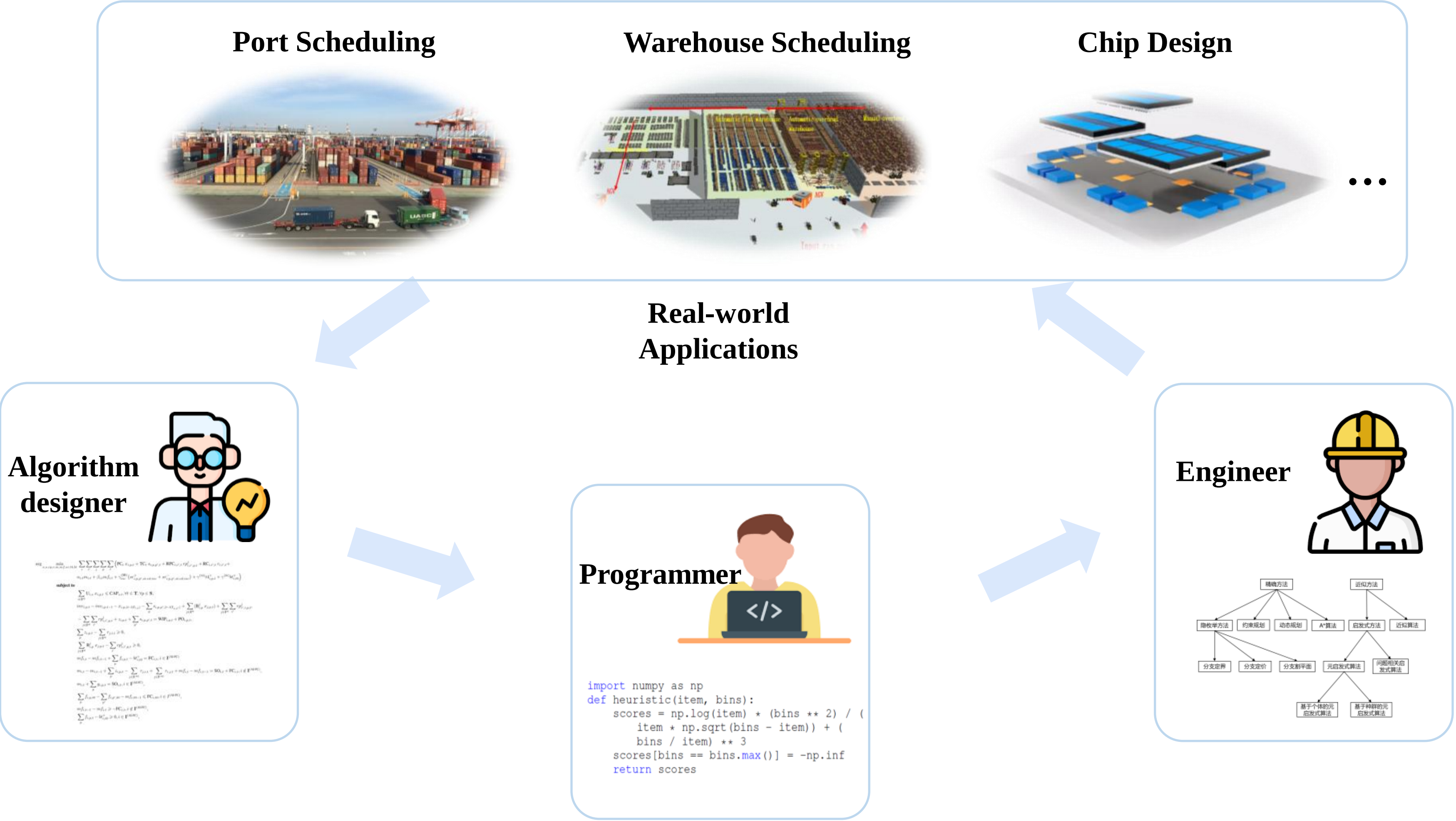
Outlines

- **Part I:** Algorithm Design with Large Language Model (LLM)
- **Part II:** Evolution of Heuristics (EoH)
- **Part III:** Case Study and LLM4AD Platform
- **Part IV:** LLM with/for Multiobjective Optimization

Part I
Algorithm Design with Large Language Model

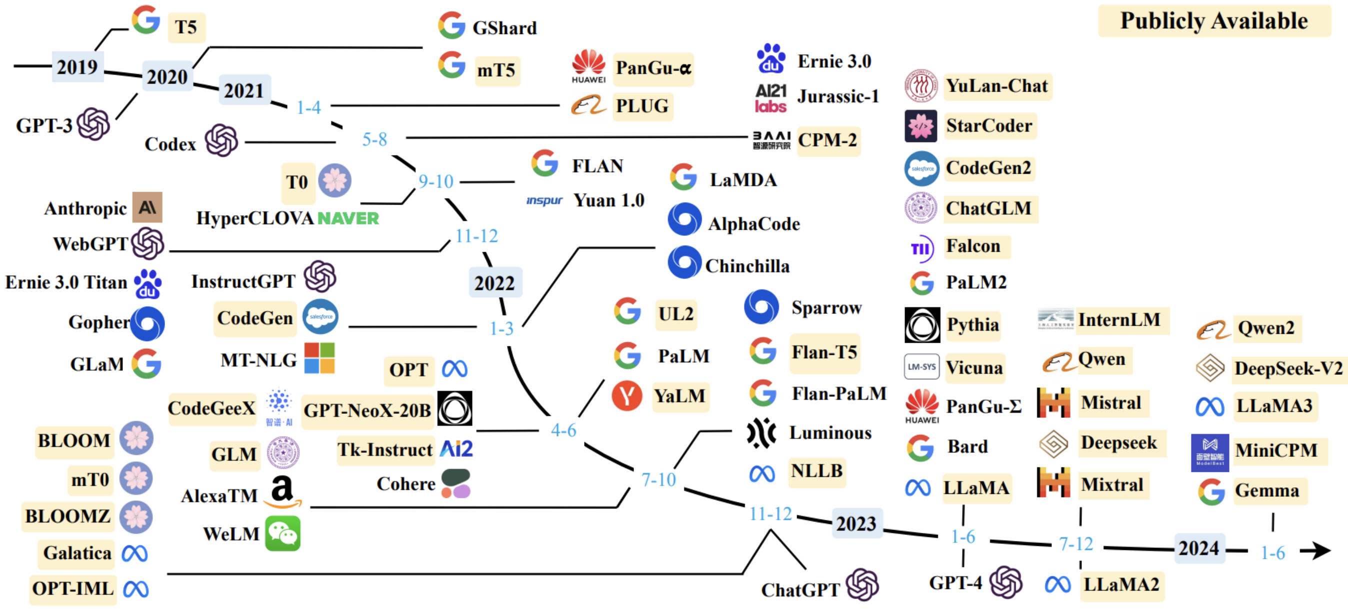
Challenges of Algorithm Design

1. Expertise knowledge
2. Trial-and-error
3. Intensive manpower

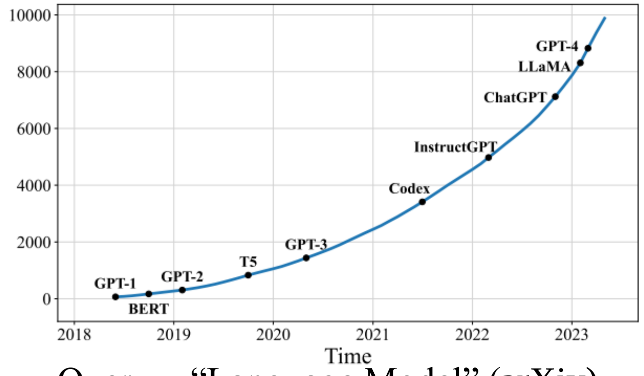


Large Language Models (LLMs)

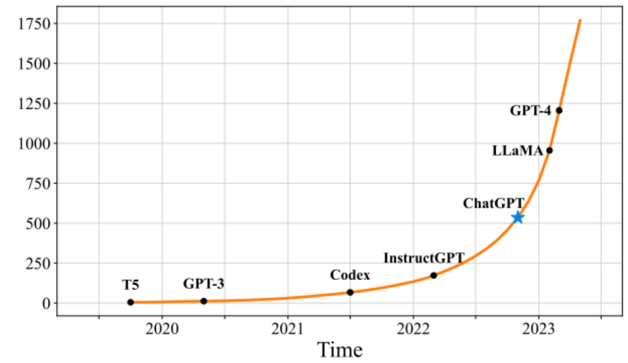
➢ Rapid Increasing of LLM and its research papers in the last four years



A timeline of existing LLMs (having a size larger than 10B)



Query = “Language Model” (arXiv)



Query = “Large Language Model” (arXiv)

- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

Overview on Algorithm Design with LLMs

- Paper collection

Stage I: Data extraction and collection

Date: 2020.1.1 ~ 2024 7.1
Key Words: Title = (LLM OR Large Language Model) AND (Algorithm OR Heuristic OR Search OR Optimization OR Optimizer OR Design OR Function)
Database: Google scholar, Web of Science, Scopus
Results (remove duplication): 850 papers

Stage II: Abstract Scanning

Content: Title and Abstract
Exclusion Criteria: not English, not algorithm design, not using large language model
Remaining Results: 260 papers

Stage III: Full Scanning

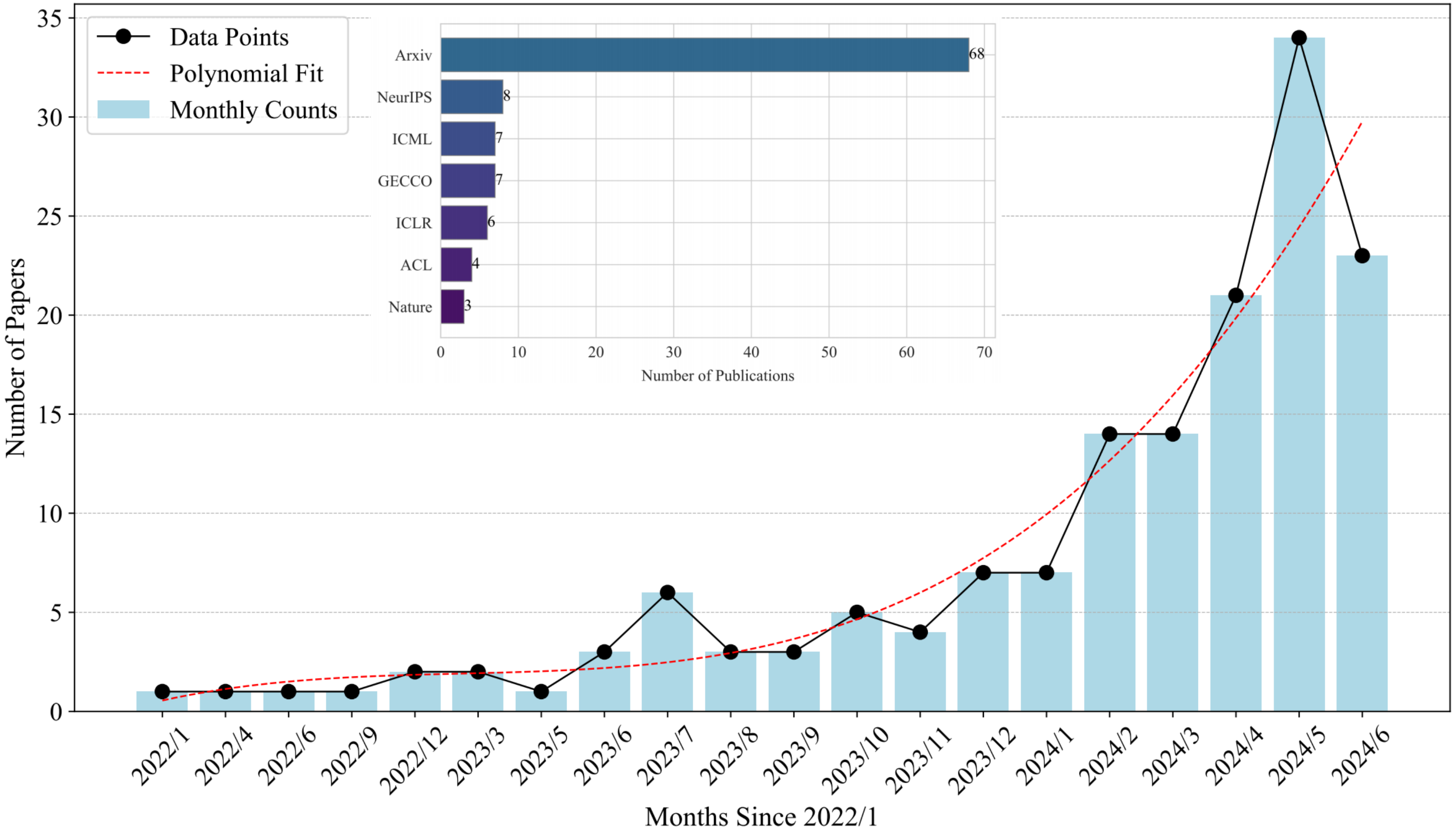
Content: Full paper
Exclusion Criteria: Research relevant to the topic
Remaining Results: 160 papers

Stage IV: Supplementation

Additional pertinent papers gathered from experience
Final Results: 180 papers

Algorithm Design with LLMs

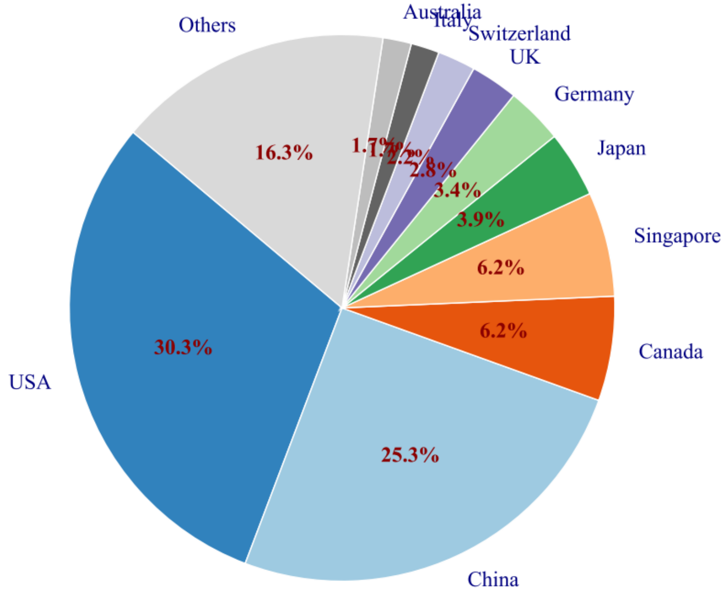
- Number of Publications

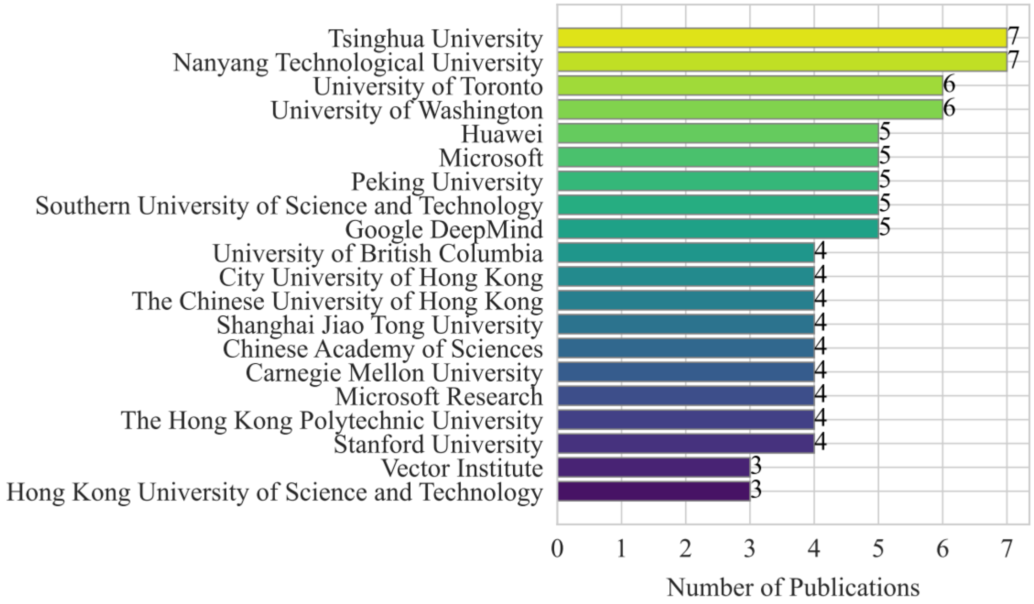


- Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**A Systematic Survey on Large Language Models for Algorithm Design**." *arXiv preprint arXiv:2410.14716* (2024).

Algorithm Design with LLMs

- Country and Institution
  - Top universities: Tsinghua, NTU, and the University of Toronto, alongside major corporations like Huawei, Microsoft, and Google



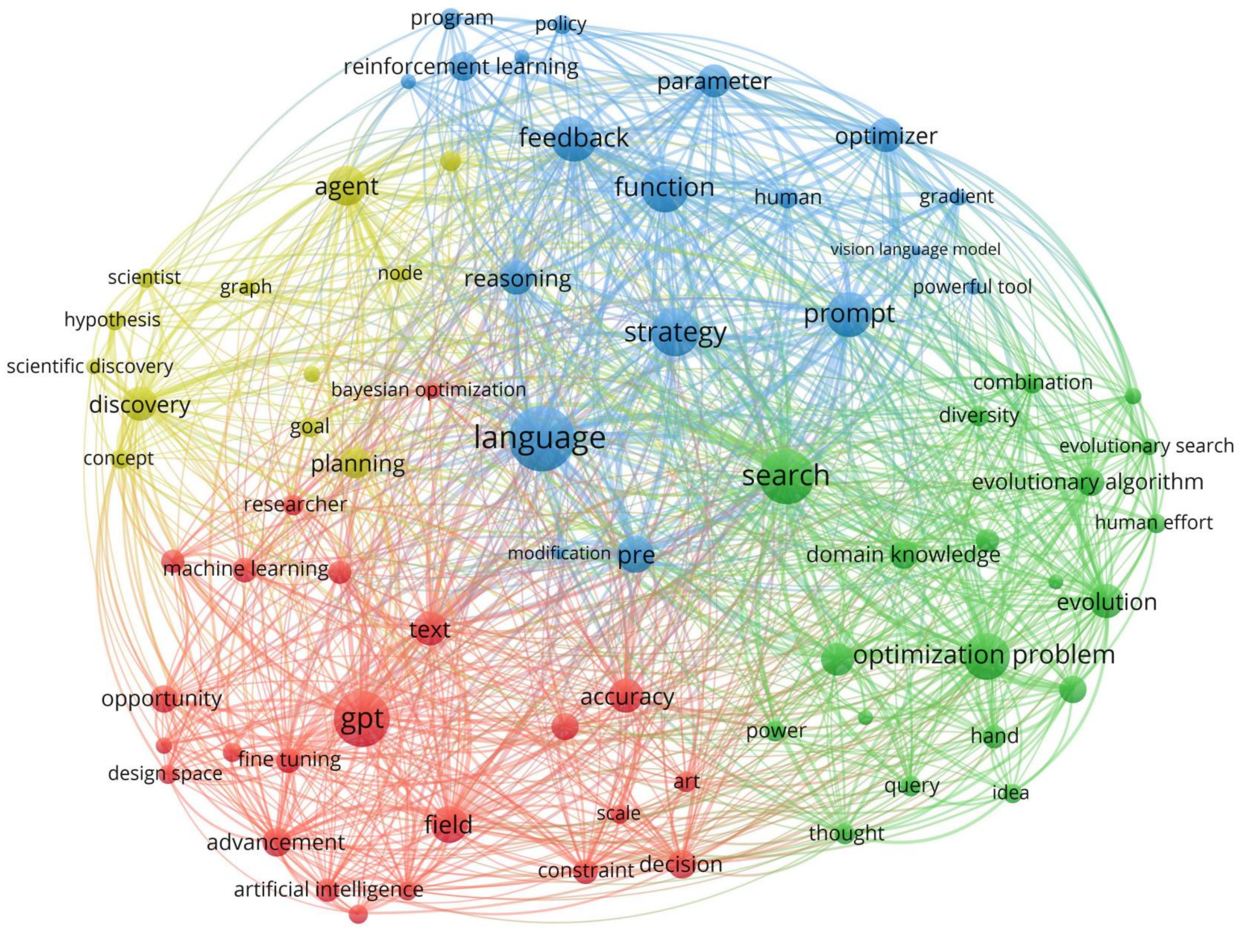


- Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang.“ **A Systematic Survey on Large Language Models for Algorithm Design**." *arXiv preprint arXiv:2410.14716* (2024).

Algorithm Design with LLMs

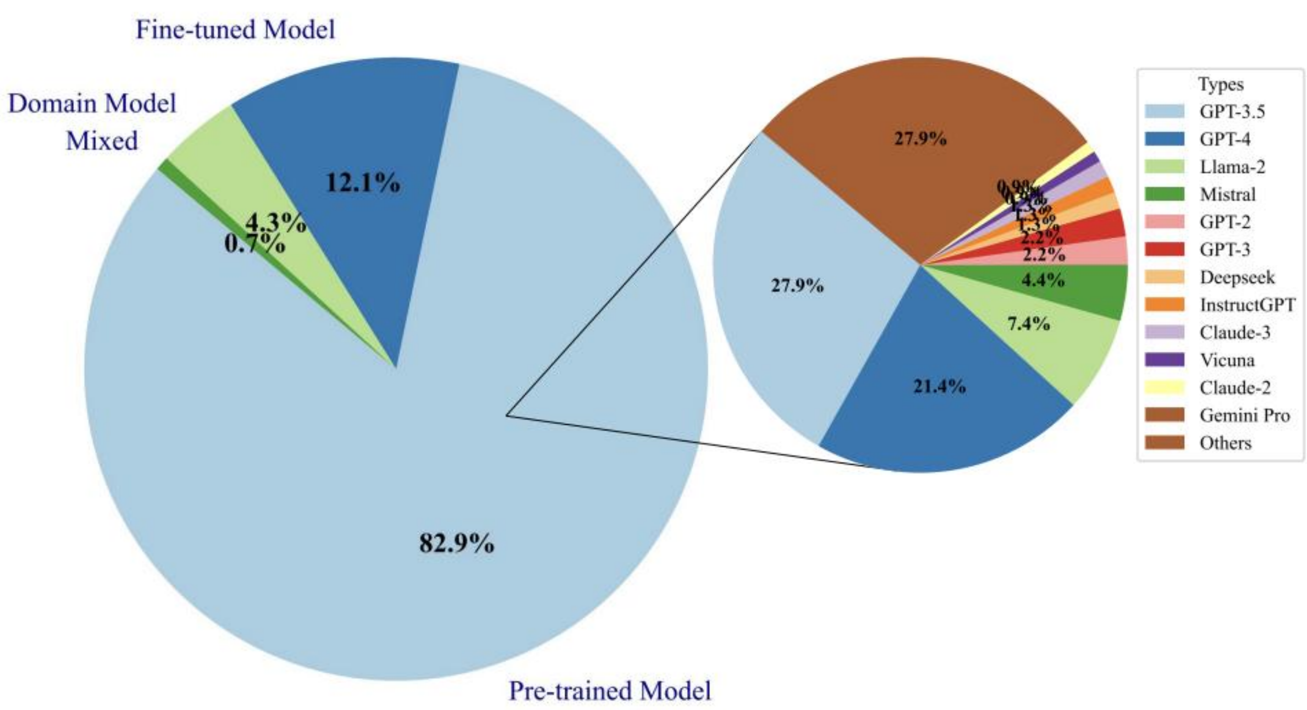
Word Cloud

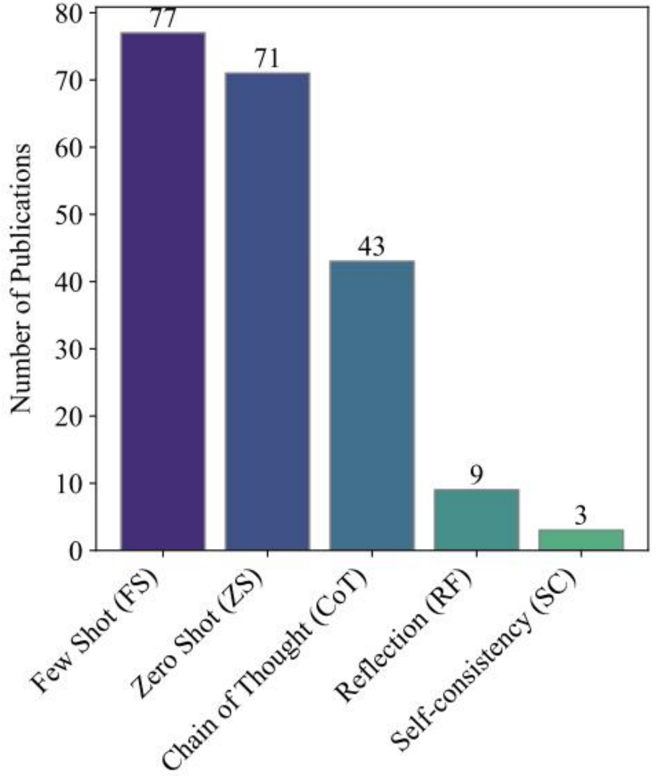
- **Blue Cluster** features the keyword "**Language**", includes other frequently used terms such as "strategy", "reasoning", "prompt", and "function".
- **Red Cluster** is centered around "**GPT**". This cluster also contains terms like "fine-tuning", "text", and "accuracy", which are crucial in model training and inference.
- **Green Cluster** focuses on **search and optimization**, encompassing terms such as "evolutionary algorithm", "combination", and "diversity".
- **Yellow Cluster** emphasizes the role of **LLMs in scientific discovery**, highlighting keywords such as "scientist", "hypothesis", and "concept".



- Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**A Systematic Survey on Large Language Models for Algorithm Design**." *arXiv preprint arXiv:2410.14716* (2024).

LLMs and Prompt Strategies

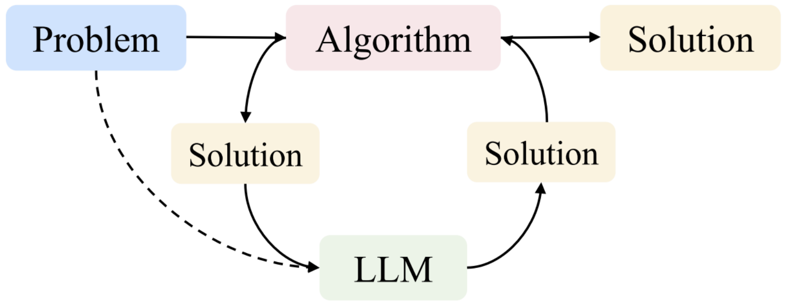




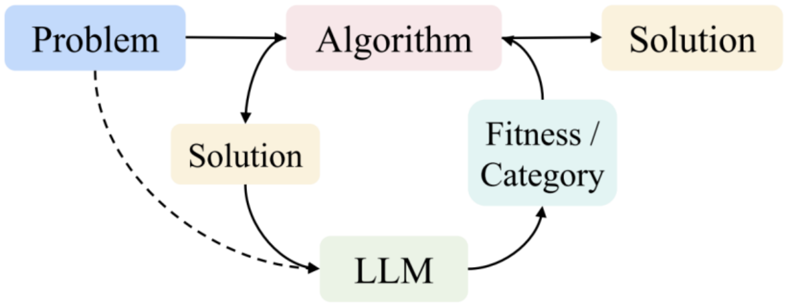
- Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**A Systematic Survey on Large Language Models for Algorithm Design**." *arXiv preprint arXiv:2410.14716* (2024).

Algorithm Design with LLMs

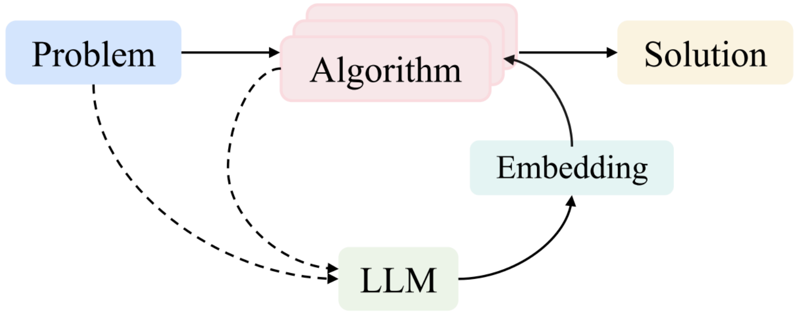
- LLM Roles



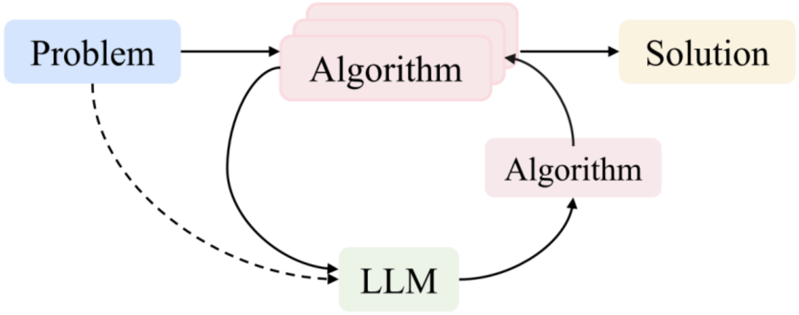
(a) Large Language Models as Optimizers (LLMaO)



(b) Large Language Models as Predictors (LLMaP)



(c) Large Language Models as Extractors (LLMaE)

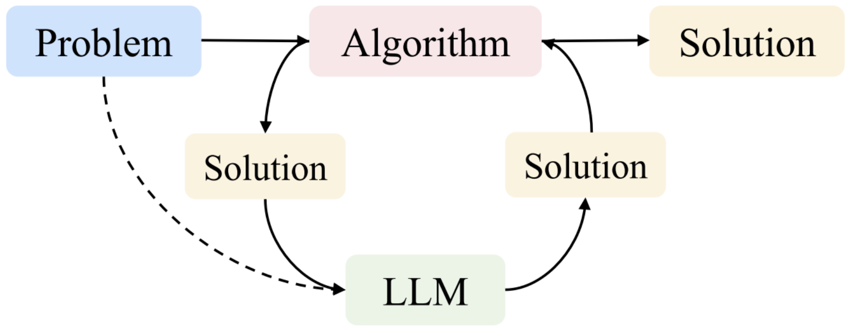


(d) Large Language Models as Designers (LLMaD)

- Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**A Systematic Survey on Large Language Models for Algorithm Design**." *arXiv preprint arXiv:2410.14716* (2024).

Large Language Models as Optimizers (LLMaO)

- **LLMs are used as optimizer to suggest new solutions, human-designed algorithm**
  - **Pros:** LLM excels at learning and handling complicated patterns, integrating preference
  - **Cons:** Lack generalization and interpretability



You are given a list of points with coordinates below: (0): (-4, 5), (1): (17, 76), (2): (-9, 0), (3): (-31, -86), (4): (53, -35), (5): (26, 91), (6): (65, -33), (7): (26, 86), (8): (-13, -70), (9): (13, 79), (10): (-73, -86), (11): (-45, 93), (12): (74, 24), (13): (67, -42), (14): (87, 51), (15): (83, 94), (16): (-7, 52), (17): (-89, 47), (18): (0, -38), (19): (61, 58).
Below are some previous traces and their lengths. The traces are arranged in descending order based on their lengths, where lower values are better.

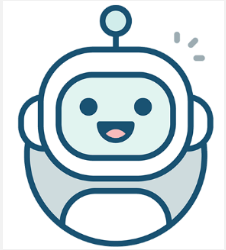
<trace> 0,13,3,16,19,2,17,5,4,7,18,8,1,9,6,14,11,15,10,12 </trace>
length:
2254

<trace> 0,18,4,11,9,7,14,17,12,15,10,5,19,3,13,16,1,6,8,2 </trace>
length:
2017

<trace> 0,11,4,13,6,10,8,17,12,15,3,5,19,2,1,18,14,7,16,9 </trace>
length:
1953

<trace> 0,10,4,18,6,8,7,16,14,11,2,15,9,1,5,19,13,12,17,3 </trace>
length:
1840

Give me a new trace that is different from all traces above, and has a length lower than any of the above. The trace should traverse all points exactly once. The trace should start with <trace> and end with </trace>.



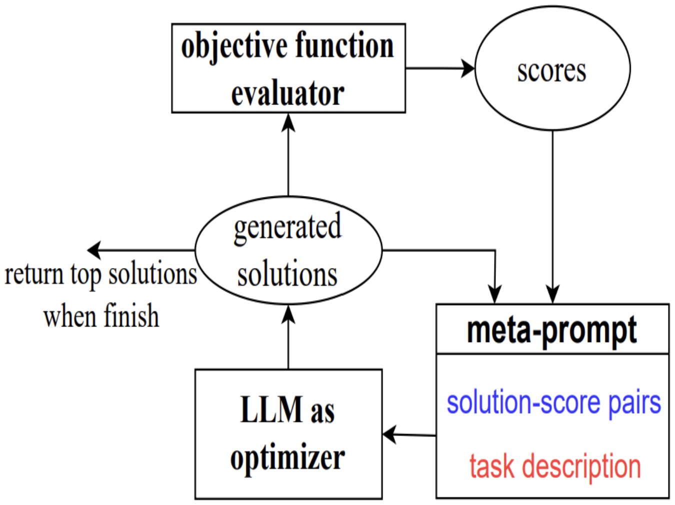
LLM

To create a new trace with a length lower than the previous ones, we need to carefully select the order of points to minimize the total Euclidean distance traveled. Here's a new trace that is different from the ones provided and aims to achieve a shorter length:

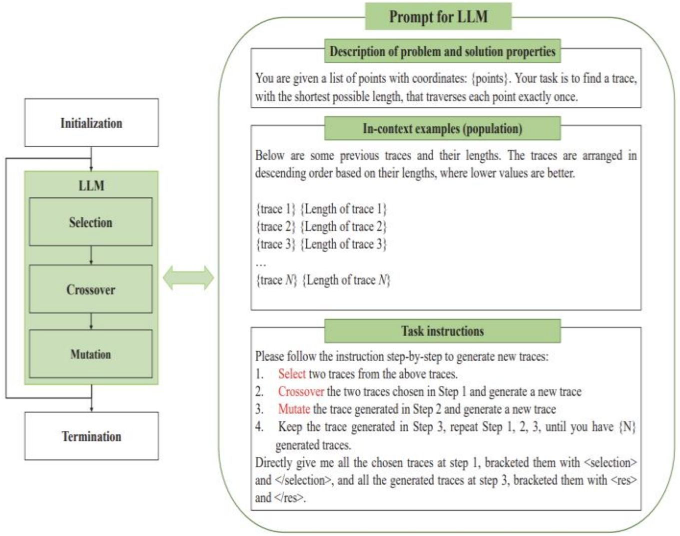
<trace>0,16,2,18,8,3,10,13,6,4,19,7,5,9,1,14,12,15,17,11</trace>

This trace is designed to minimize the distance by connecting points that are closer to each other in sequence.

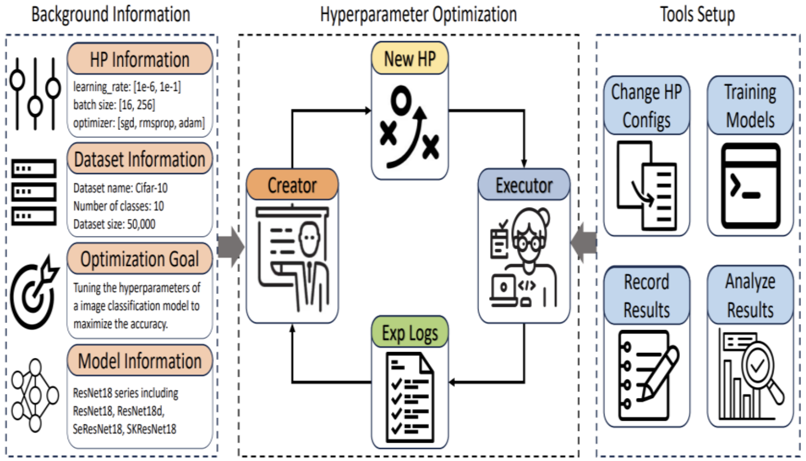
Large Language Models as Optimizers (LLMaO)



LLM as optimizers [1]



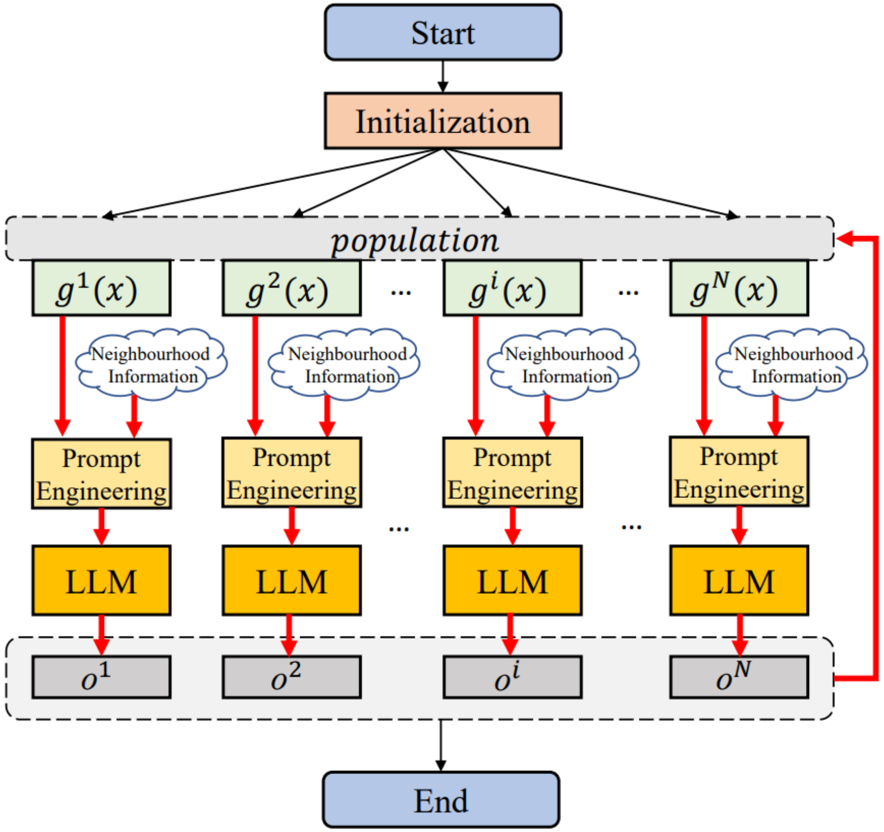
EA optimizers [2]

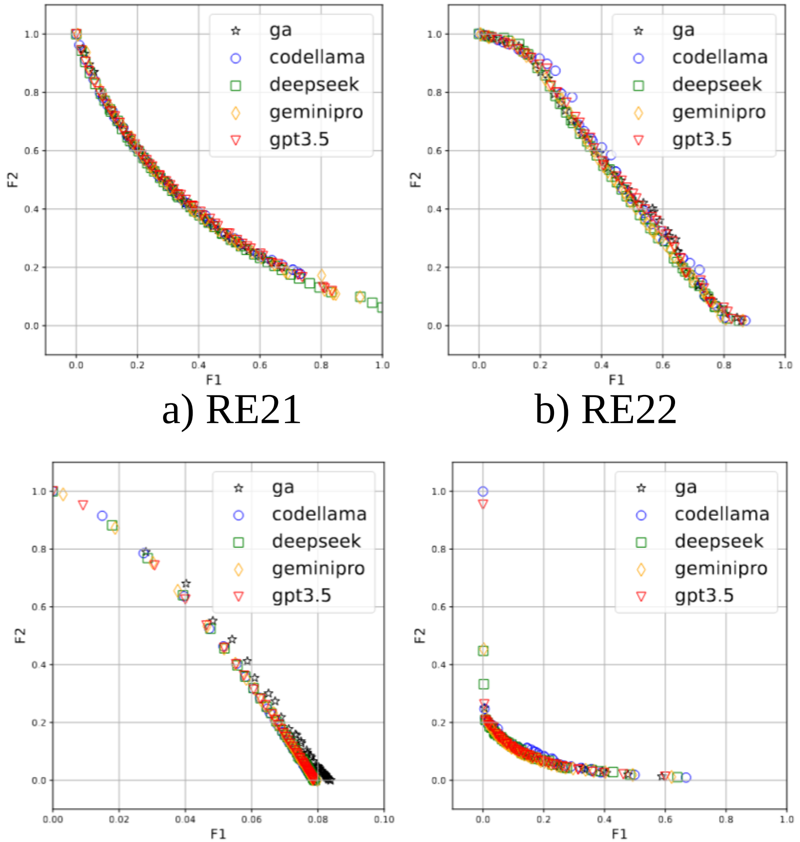


Hyperparameter optimizer [3]

1. Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. **Large Language Models as Optimizers**. ICLR 2024.
2. Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. 2024. **Large language models as evolutionary optimizers**. CEC 2024.
3. Siyi Liu, Chen Gao, and Yong Li. **Large Language Model Agent for Hyper-Parameter Optimization**. arXiv preprint arXiv:2402.01881 (2024).

Large Language Models as Optimizers (LLMaO)





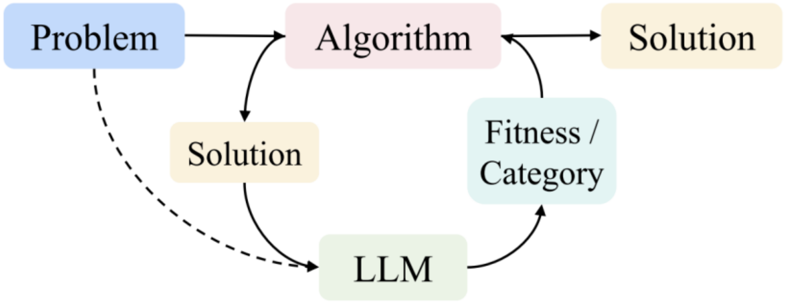
a) RE21 b) RE22

c) RE23 d) RE24

- Fei Liu, Xi Lin, Zhenkun Wang, Shunyu Yao, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. **Large Language Model for Multi-objective Evolutionary Optimization**. EMO 2025.

Large Language Models as Predictors (LLMaP)

- **LLMs are used as predictors predict a solution's outcomes or responses**
  - **Pros:** LLMs excel at processing and generating human-like response, reduce the computational load and time required in model training
  - **Cons:** Can be costly, the effectiveness depends the knowledge on target task



You are tasked with evaluating each object based on its numerical attributes to determine its category as 'better' or 'worse'. These attributes derive from a black box function's decision space, with the assessment of the label based on the post-mapping function values. Your role involves discerning the internal variable relationships of the black box function from provided historical data, moving beyond mere statistical analyses like calculating means and variances.

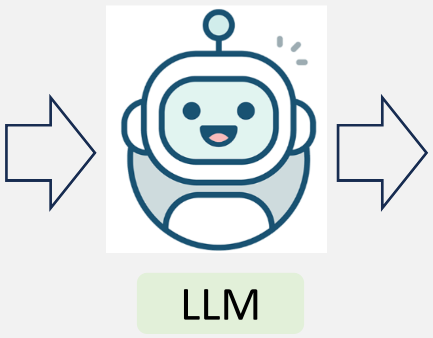
Procedure:

1. Identify patterns in how attributes are categorized.
2. Apply these patterns to assess new objects, determining whether its category is better or worse.
3. Respond using JSON format, e.g. {'Class': 'result'}

Historical Examples: Features: ⟨0.555, 0.881, . . . , 0.491⟩, Class: better Features: ⟨0.593, 0.515, . . . , 0.456⟩, Class: worse ... Features: ⟨0.253, 0.747, . . . , 0.475⟩, Class: better

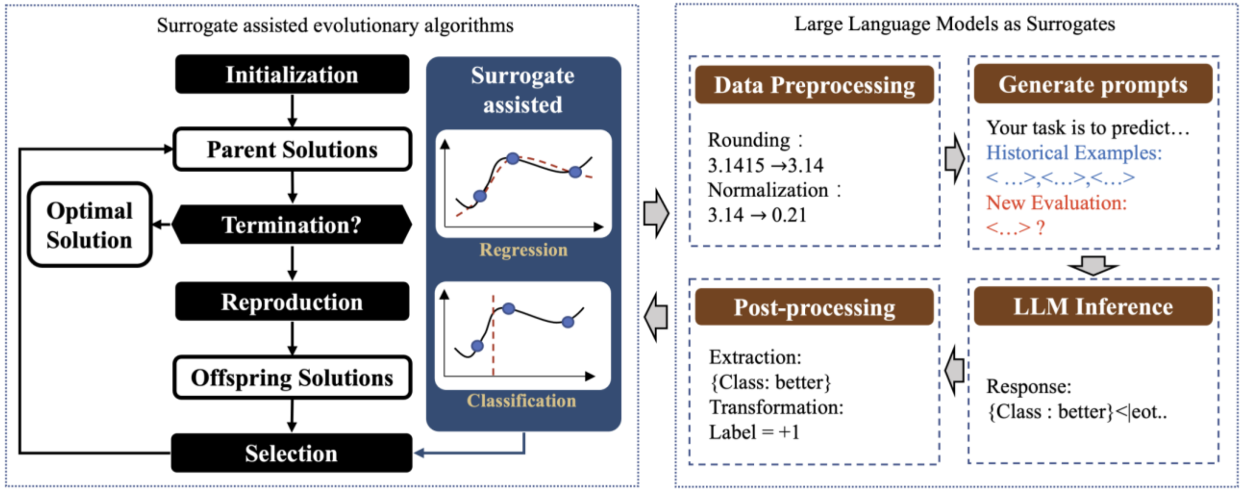
New Evaluation: ⟨0.189, 0.917, . . . , 0.443⟩ better or worse?

Note: Respond in Json with the format {'Class': 'result'} only

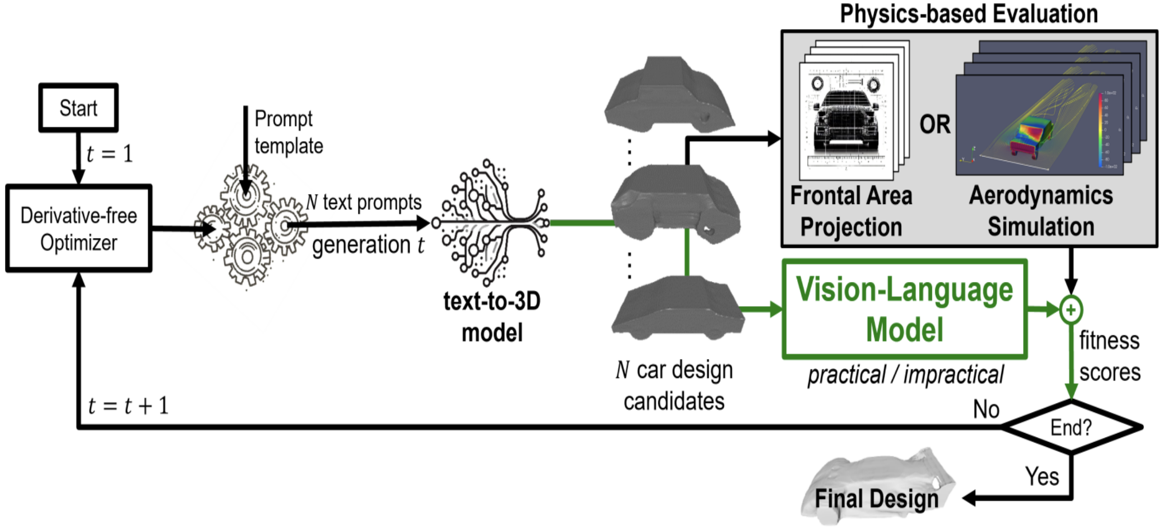


{'Class': 'better'}

Large Language Models as Predictors (LLMaP)



LLMs as predictors (regression and classification) for EA [1]

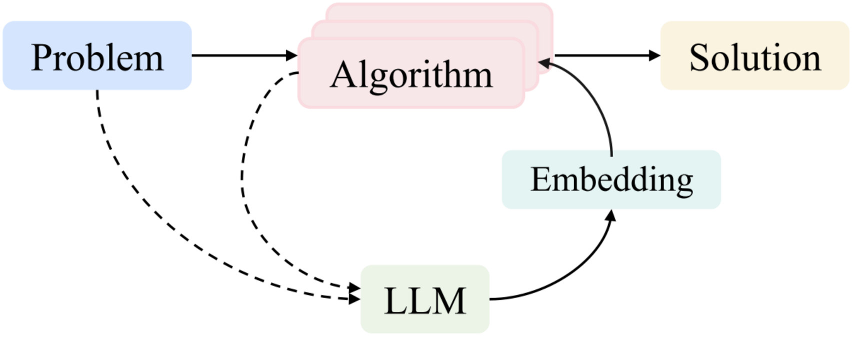


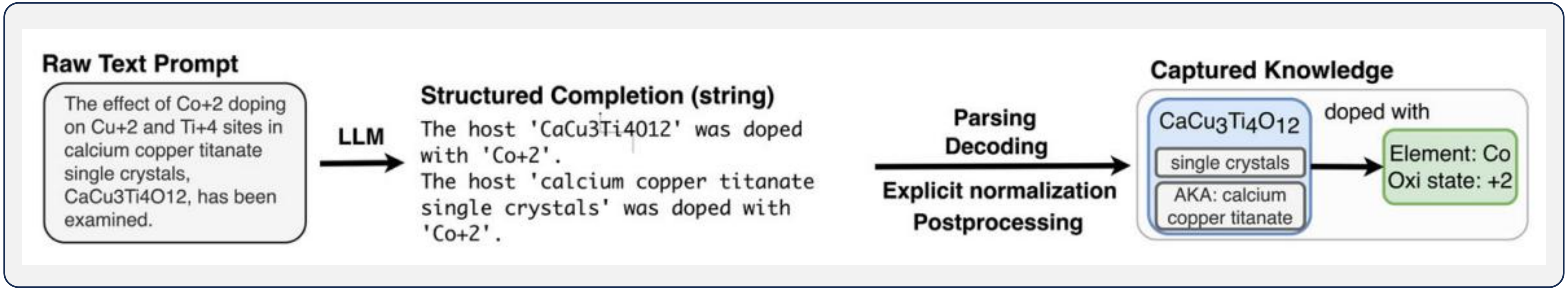
MLLMs as car shape score predictor [2]

1. Hao Hao, Xiaoqun Zhang, and Aimin Zhou. **Large Language Models as Surrogate Models in Evolutionary Algorithms: A Preliminary Study**. Swarm and Evolutionary Computation 2024.
2. Melvin Wong, Thiago Rios, Stefan Menzel, and Yew Soon Ong. **Generative AI-based Prompt Evolution Engineering Design Optimization With Vision-Language Model.** arXiv preprint arXiv:2406.09143 2024.

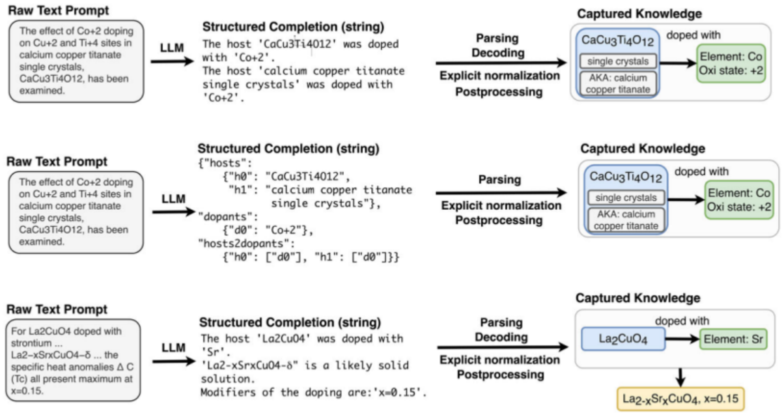
Large Language Models as Extractors (LLMaE)

- **LLMs are employed to extract features or specific knowledge from target problem and/or algorithms to enhance problem-solving**
  - **Pros:** LLMs excel at extract high-level features and comprehends text and code
  - **Cons:** Can be costly, the effectiveness depends the knowledge on target task

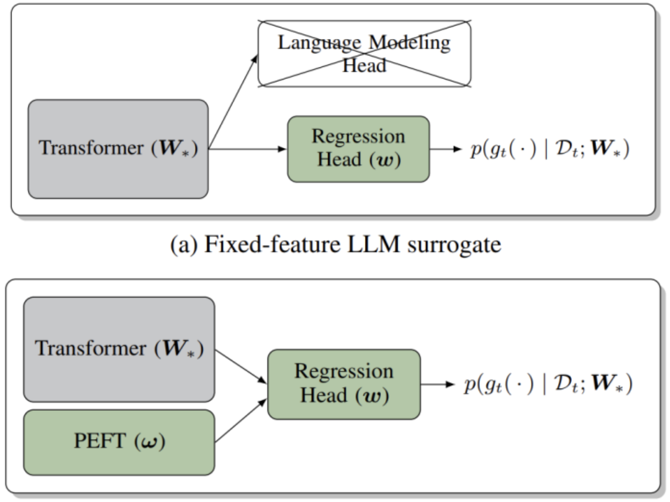




Large Language Models as Extractors (LLMaE)



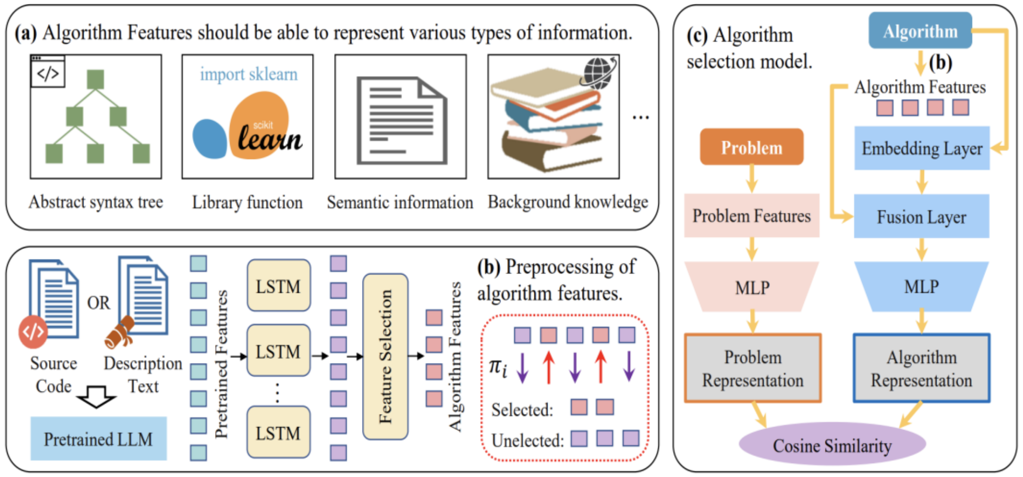
Extractor for Scientific Text



(a) Fixed-feature LLM surrogate

(b) Adaptive-feature LLM surrogate

Extractor for Bayesian Opt.

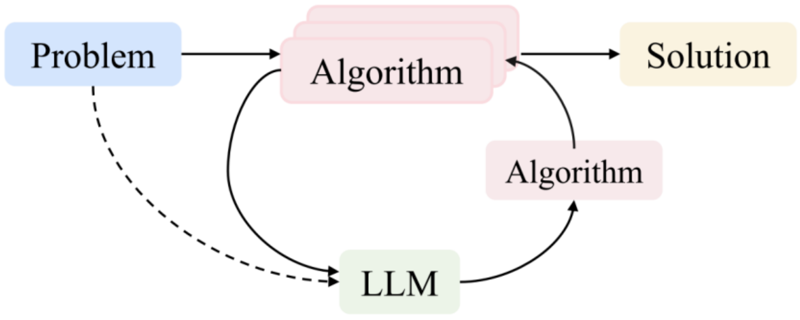


Extractor for Algorithm Selection

1. Dagdelen, John, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S. Rosen, Gerbrand Ceder, Kristin A. Persson, and Anubhav Jain. **Structured information extraction from scientific text with large language models.** Nature Communications 2024.
2. Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and Geoff Pleiss. **A Sober Look at LLMs for Material Discovery: Are They Actually Good for Bayesian Optimization Over Molecules?** ICML 2024.
3. Xingyu Wu, Yan Zhong, Jibin Wu, Bingbing Jiang, Kay Chen Tan, et al. **Large language model-enhanced algorithm selection: towards comprehensive algorithm representation.** IJCAI 2024.

Large Language Models as Designers (LLMaD)

- **LLMs are employed to directly create algorithms or specific components**
  - LLMs excel at code generation, text comprehension, and reasoning
  - Algorithm generation instead of parameter tuning or simple function design
  - New insights and very competitive results



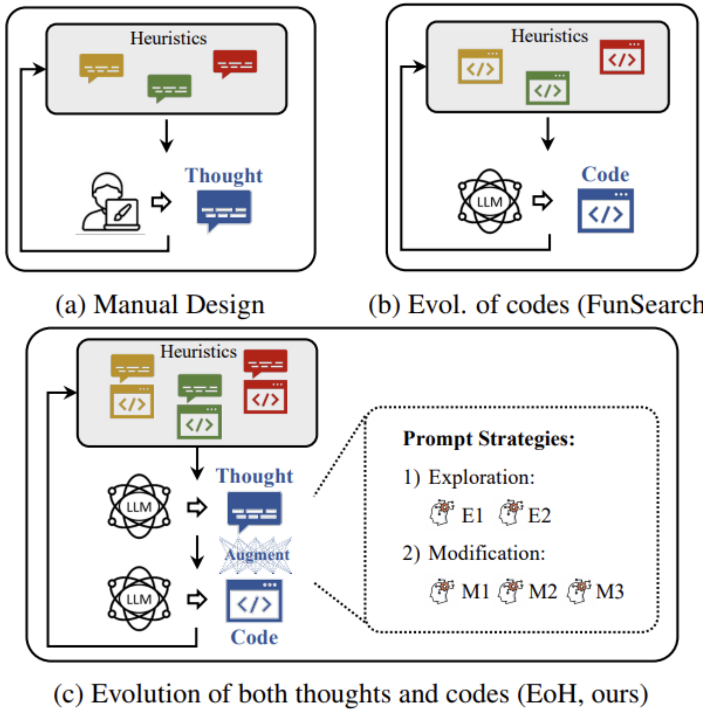
I have some algorithms with their code. The first algorithm and the corresponding code is:
<Algorithm description>: …
<Code>: …
The second algorithm and the corresponding code is
<Algorithm description>: …
<Code>: …
(more …)
Please help me create a new algorithm that motivated by the given algorithms. Please provide a brief description of the new algorithm and its corresponding code. The description must using the following template …

Algorithm Description:
This enhanced algorithm considers the current node's distance to the other unvisited nodes and the mean distance, but also incorporates the standard deviation of these distances into the scoring system, giving a higher weight to close clusters of nodes. It also includes a conditional statement that chooses the closest node when the minimum calculated score exceeds a specified threshold, ensuring that nodes far from the rest are not chosen prematurely.

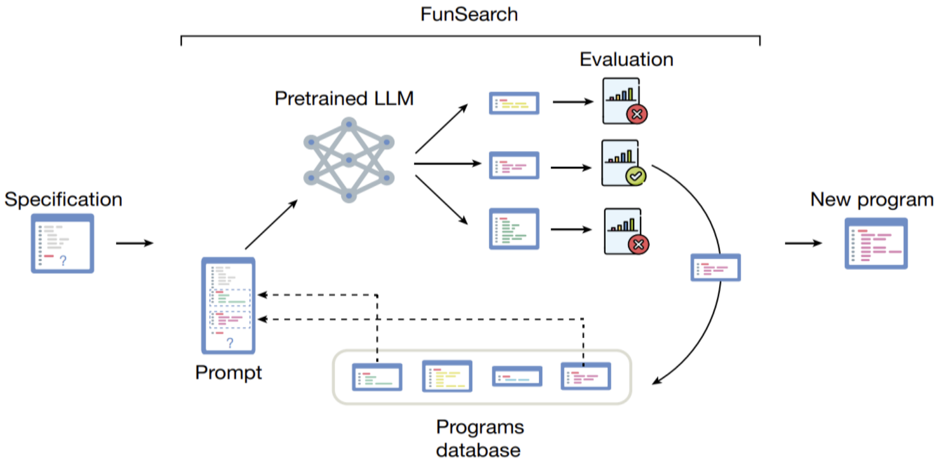
Code Block:
```
import numpy as np
def select_next_node(current_node, destination_node, unvisited_nodes, distance_matrix,
                threshold=0.7):
    scores = {}
    for node in unvisited_nodes:
        all_distances = [distance_matrix[node][i] for i in unvisited_nodes if i != node]
        average_distance_to_unvisited = np.mean(all_distances)
        std_dev_distance_to_unvisited = np.std(all_distances)

        score = 0.4 * distance_matrix[current_node][node] - 0.3 *
                average_distance_to_unvisited + 0.2 * std_dev_distance_to_unvisited - 0.1 *
                distance_matrix[destination_node][node]
        scores[node] = score
    if min(scores.values()) > threshold:
        next_node = min(unvisited_nodes, key=lambda node:
                distance_matrix[current_node][node])
    else:
        next_node = min(scores, key=scores.get)
    return next_node
```

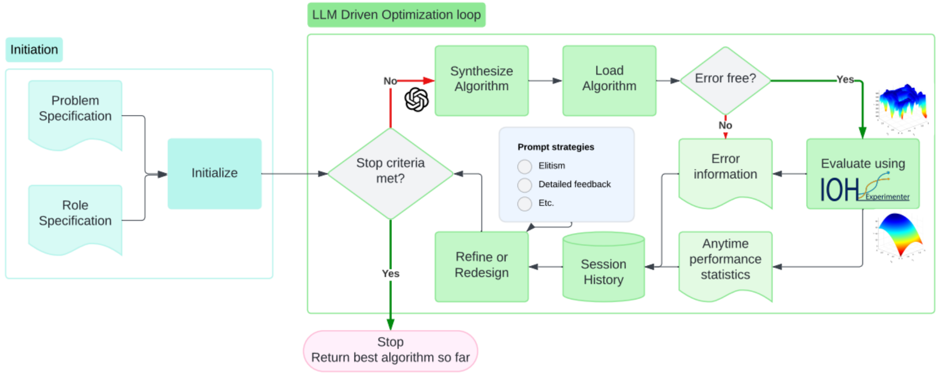
Large Language Models as Designers (LLMaD)



EoH: Heuristic Design [1]



FunSearch: Function Design [2]



LLaMEA: Heuristic Design[3]

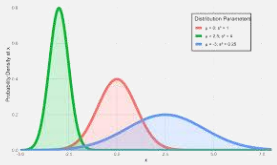
1. Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. **Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model.** ICML 2024 **Oral Top 1.5%**
2. Bernardino Romera-Paredes, et al. **Mathematical discoveries from program search with large language models**. Nature 2024
3. Niki van Stein, and Thomas Bäck. **LLaMEA: A Large Language Model Evolutionary Algorithm for Automatically Generating Metaheuristics**. TEVC 2024.

Part II
Evolution of Heuristics (EoH)

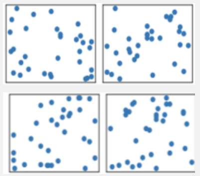
Automatic Algorithm Design (AAD)

- **Given:**

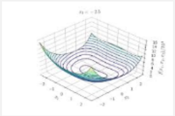
$\boldsymbol{P^I}$: A probability distribution of target problem instances (e.g., Traveling Salesman Problem, TSP)

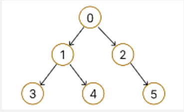


$\boldsymbol{InstS}$: a set of problem instances, some samples from P^I



$\boldsymbol{A^S}$: An algorithm space





$\boldsymbol{f}(a, i)$: A performance metric of $a \in A^S$ for problem instance i

- **Goal:**

Find an algorithm $a \in A^S$ to optimize its expected performance over the instance:

$$F(a) \coloneqq E_{i \sim P^I} f(a, i) \Rightarrow F(a) \coloneqq E_{InstS} f(a, i).$$

Existing Efforts for AAD

Hyperparameter Tuning [1]

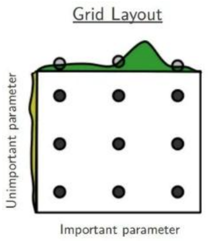
Algorithm 1 AdamW Optimizer

given $\boxed{\beta_1, \beta_2}, \epsilon, \lambda, \boxed{\eta}, f$
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

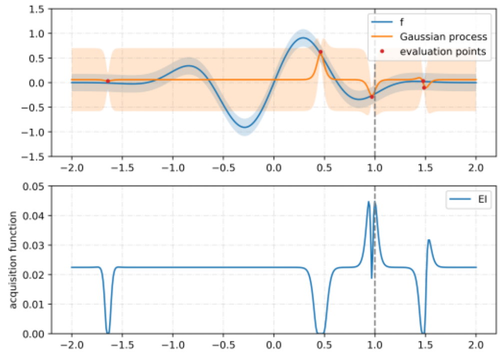
1. Baseline algorithm

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

2. Identify key hyperparameters



Random search [2]



Bayesian Optimization [3]

3. Search

✓ Existing optimization techniques can be readily applied;

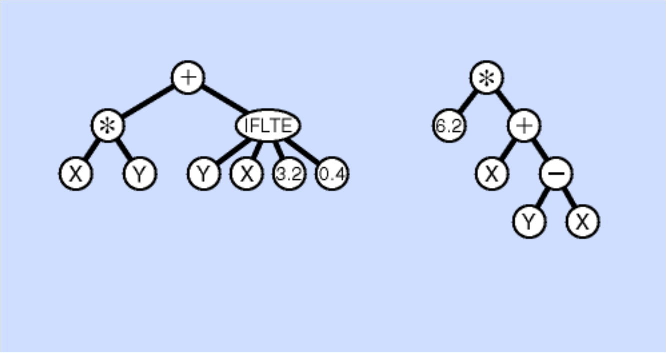
× Search a small vicinity around the baseline;

❑ 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.

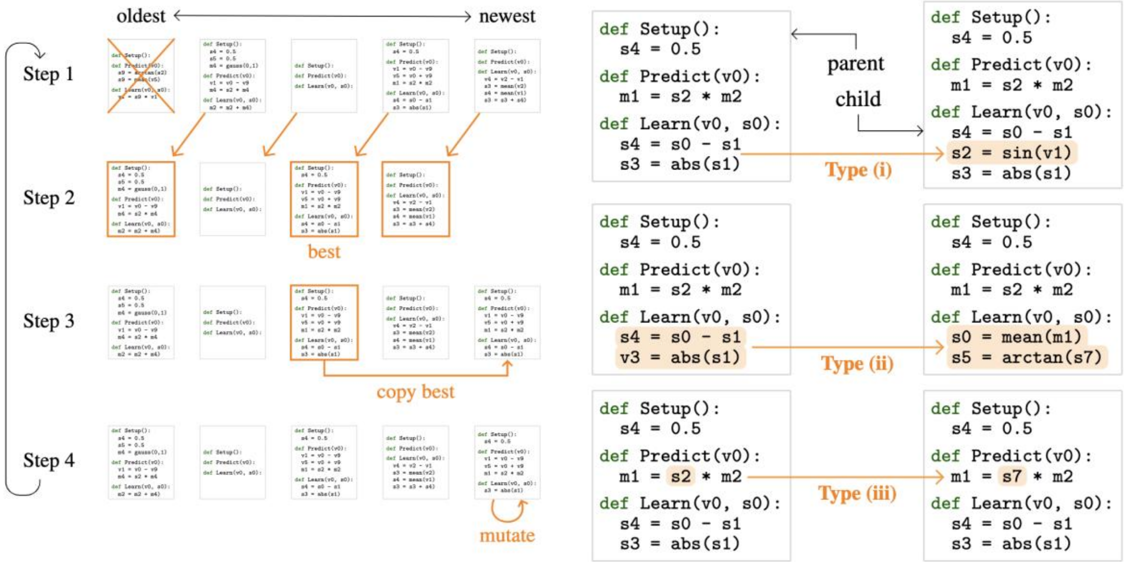
Existing Efforts on AAD

- Genetic Programming [1]



Better algorithms are created by
exchanging subparts between trees

An Application on AutoML [2]



✓ A flexible yet intuitive approach towards AAD;

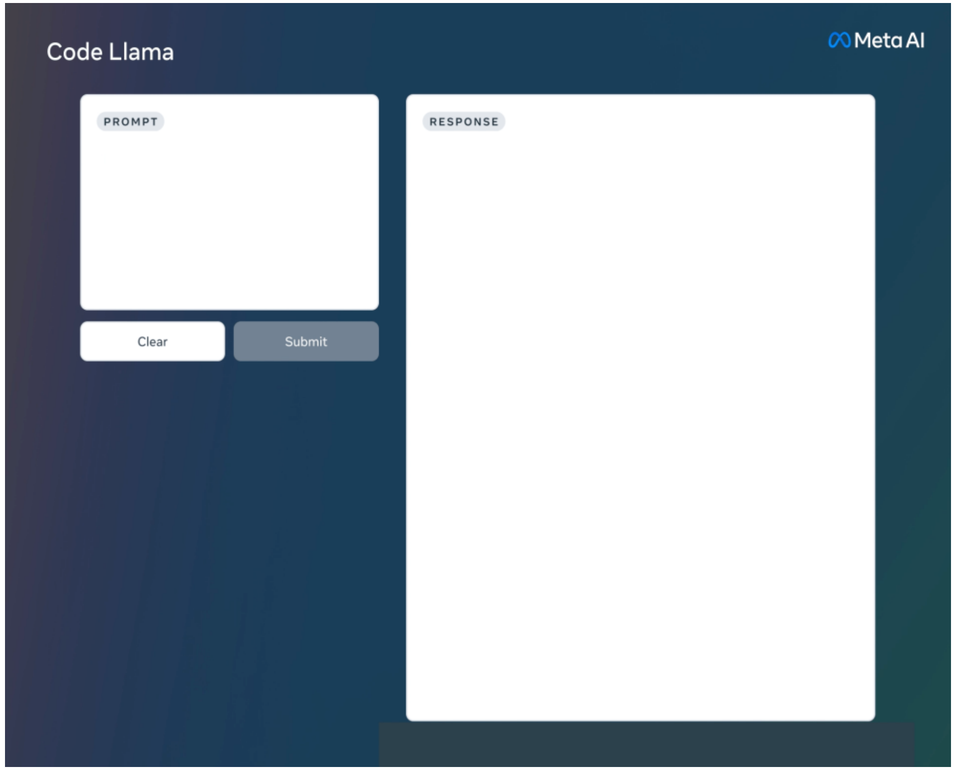
× Pre-define a set of primitives, and low search efficiency;

- The pre-defined primitives and the search rules still require much expert knowledge and manual crafting

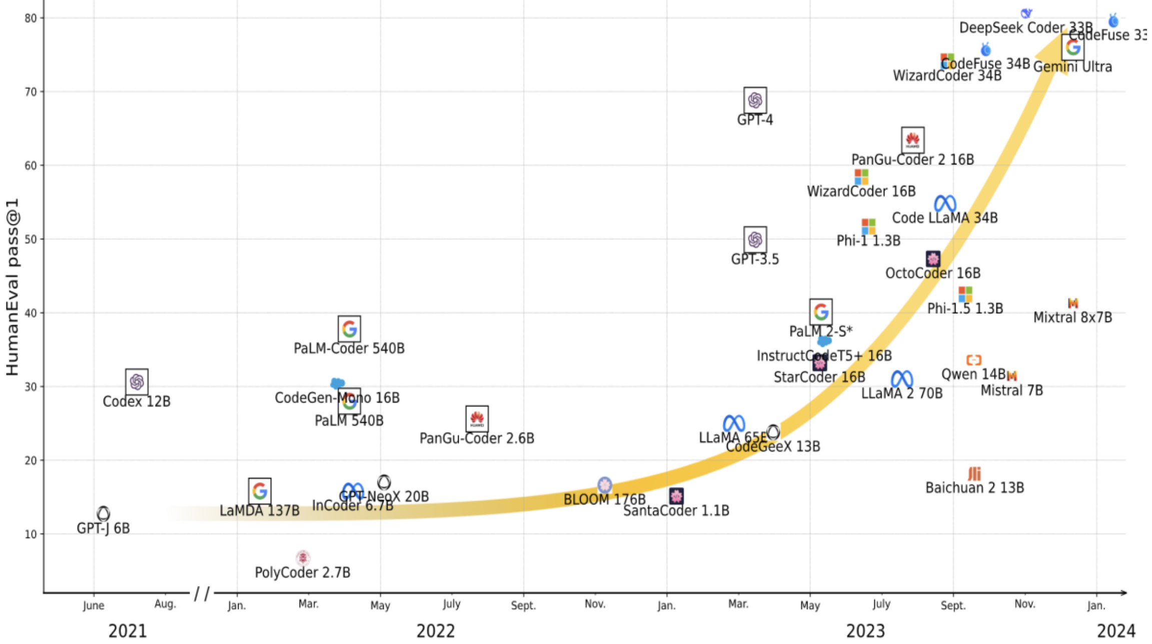
1. Koza, John R. "Genetic programming as a means for programming computers by natural selection." Statistics and computing 4 (1994): 87-112.
2. Real, Esteban, et al. "Automl-zero: Evolving machine learning algorithms from scratch." International conference on machine learning. PMLR, 2020.

LLMs as Designers for Algorithm Design?

- LLMs can operate in both language and code spaces (discrete, complex, and difficult to formulate)
- Directly prompt LLMs to generate algorithms similar to a given sample within the algorithm search space
- An effective search framework is required



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



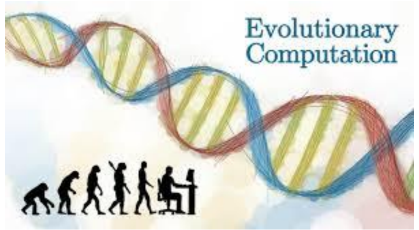
(Source: arXiv 2311.07989)

Evolution of Heuristics (EoH)

<u>Evolution of Heuristics (EoH)</u>:

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

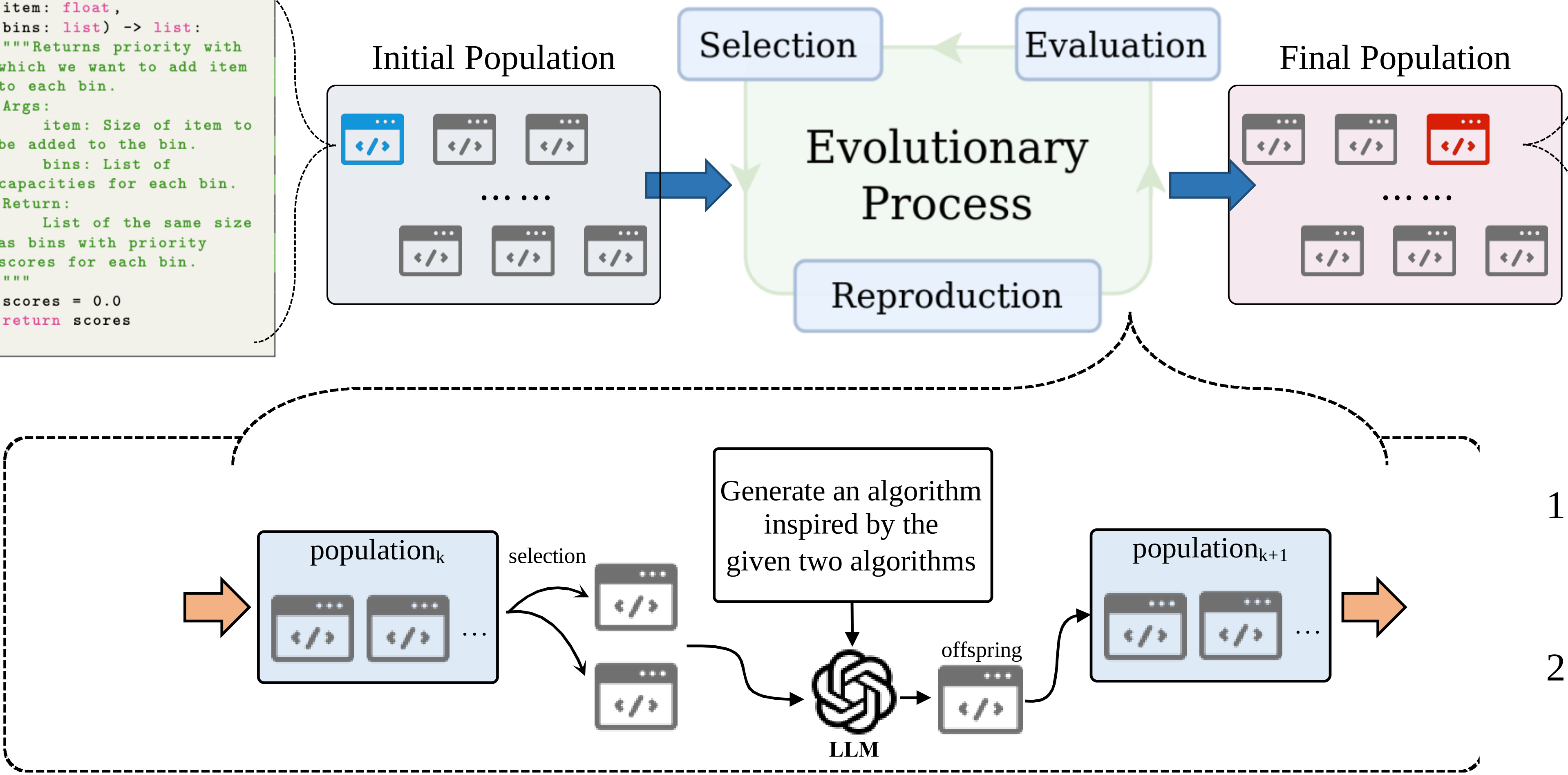
Algorithm Designer generates novel and diverse algorithms to drive exploration and creativity



Search Engine maintains, evaluates, and explores elite individuals

EoH Pipeline

```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
```

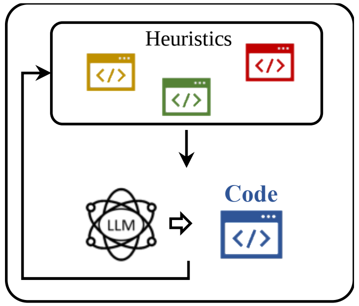



```python
def final_algorithm(
    item: float,
    bins: list) -> list:
    """Returns priority with which
    we want to add item to each bin.
    Args:
        item: Size of item to be
    added to the bin.
        bins: List of capacities for
     each bin.
    Return:
        List of the same size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

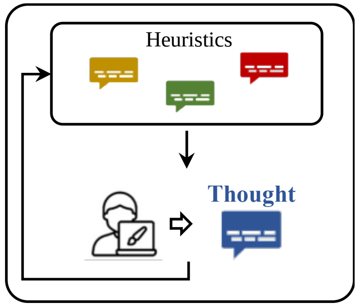
1. Algorithm is modeled as executable computer **programs**;
2. New algorithms are created via **LLMs**

- Fei Liu, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. "Algorithm evolution using large language model." arXiv preprint arXiv:2311.15249 (2023).
- Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. **Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model.** ICML 2024 (**Oral Top 1.5%**)

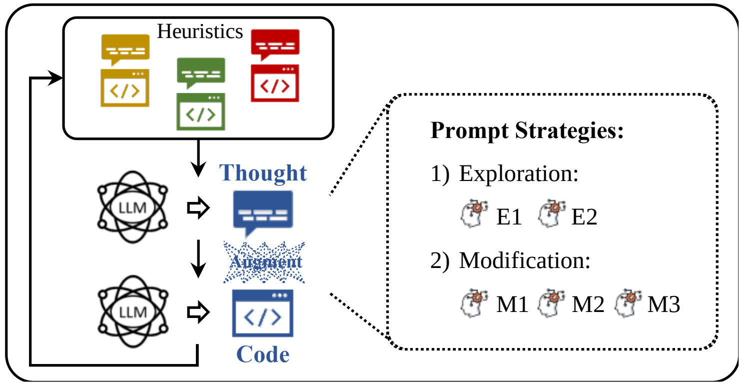
EoH Evolves both Thoughts and Codes



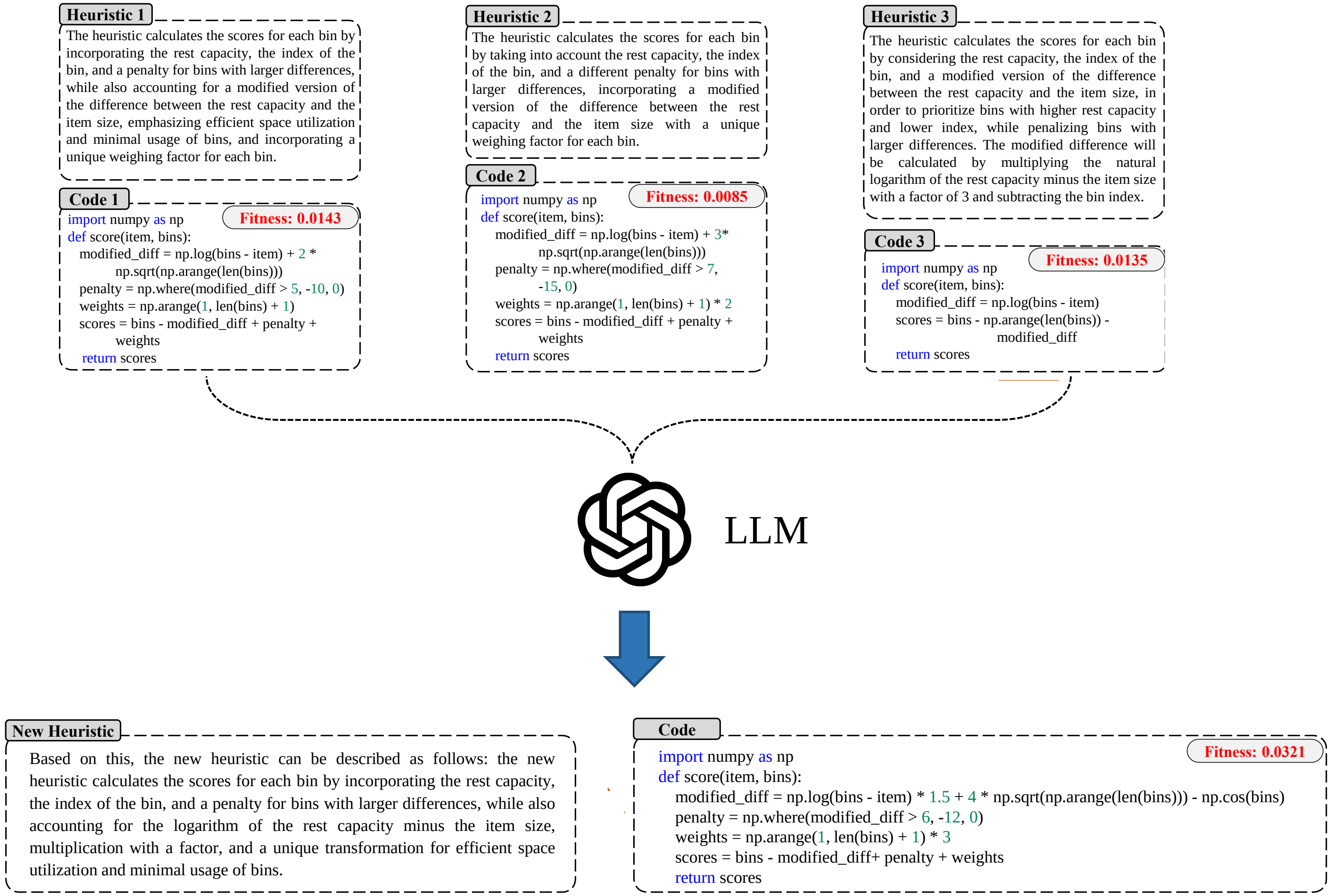
Evolution of Codes



Evolution of Thoughts



Evolution of Heuristics (EoH)

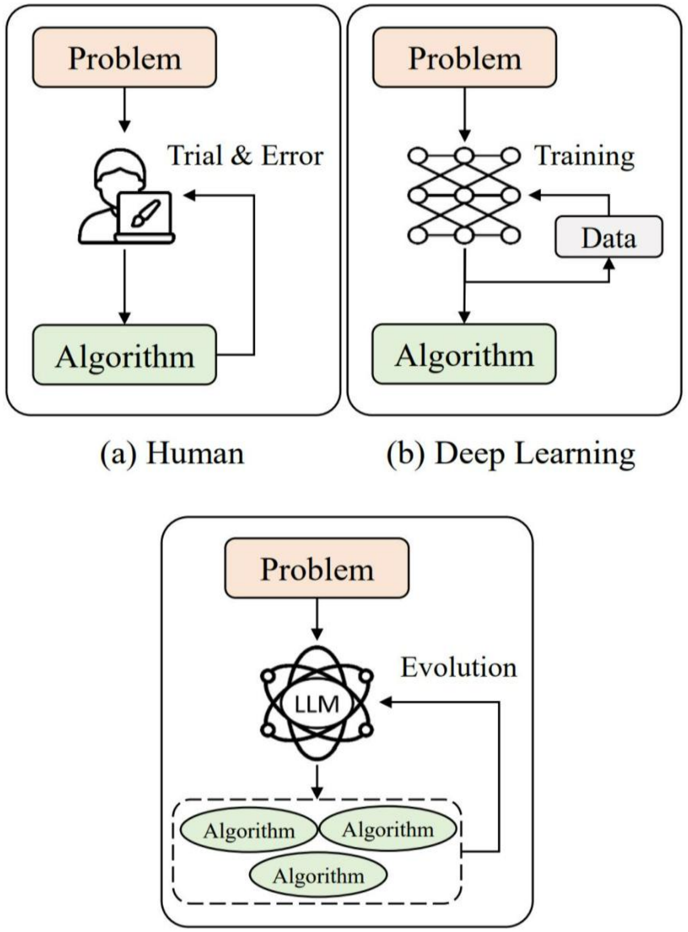


One operator in EoH

- Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. **Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model.** ICML 2024 (**Oral Top 1.5%**)

EoH: A New Paradigm for Algorithm Design

1. No model training and minimized manual crafting
2. Open-ended heuristic development
3. Enhancement of problem-solving and new insights



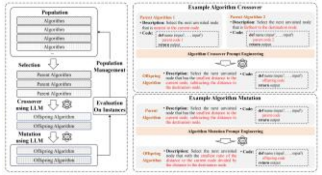
(a) Human (b) Deep Learning

(c) EoH (EC+LLM)

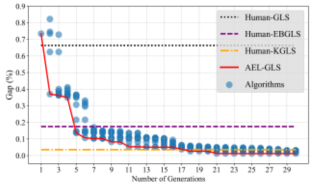
EC+LLM for AAD

Ours (EoH)

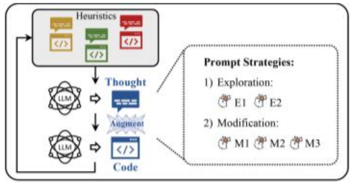
2023-11-26 AEL: Algorithm Evolution using LLM



2024-01-04 Beating Human: Design of GLS

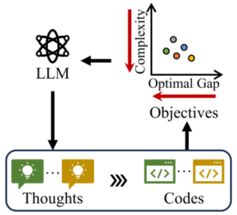


2024-05-01 EoH: Evolution of Heuristics … Using LLM. (ICML 2024 Oral)



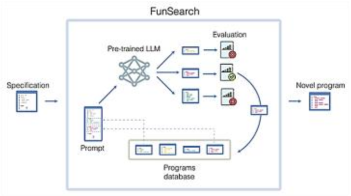
(c) Coevolution of thoughts and codes (ours)

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

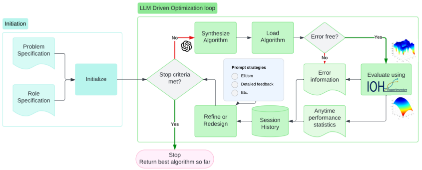


Others

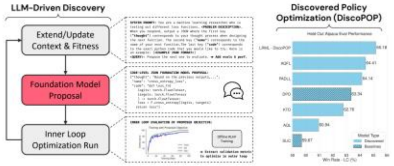
2023-12-14 FunSearch (Nature, Google)



2024-05-30 LLaMEA (TEVC, Leiden University)



2024-06 Discovering Preference Optimization Algorithms with and for LLM (Cambridge)

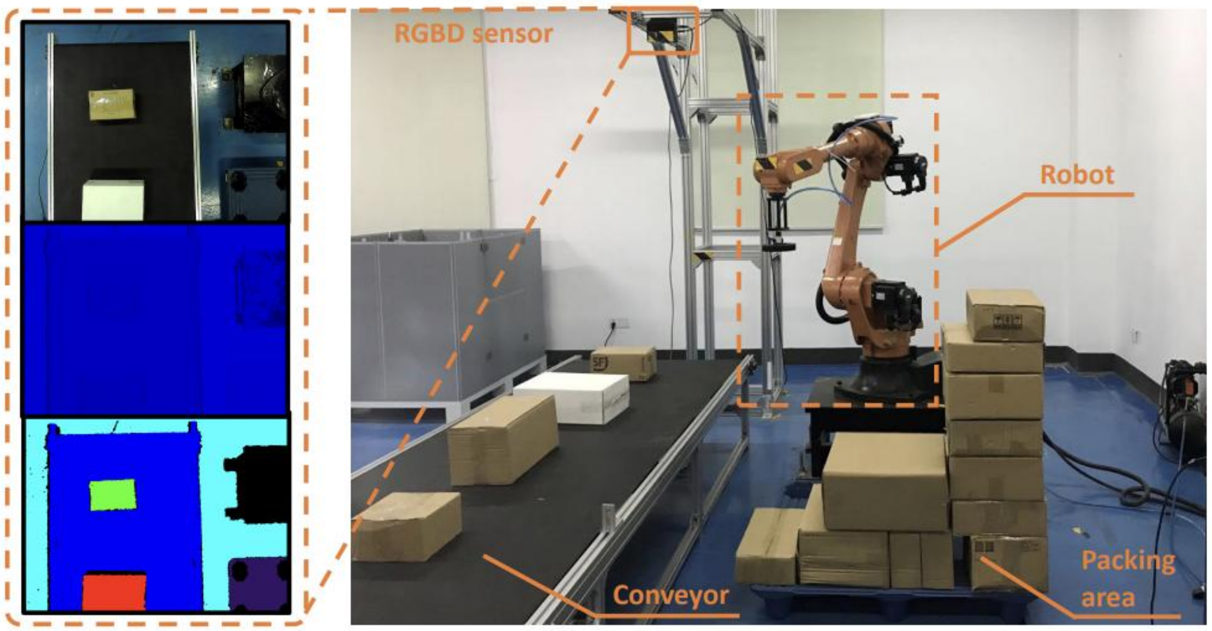


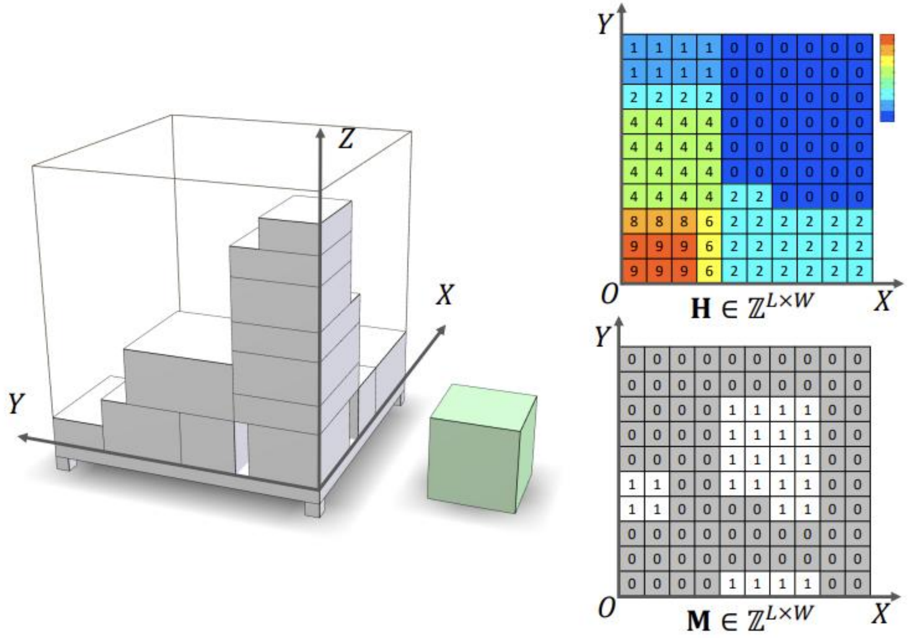
- Fei Liu, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. "**Algorithm Evolution using Large Language Model**." *arXiv preprint* 2023.
- Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. "**An example of evolutionary computation+ large language model beating human: Design of efficient guided local search**." arXiv 2024.
- Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. **Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model**. ICML 2024. (**Oral Top 1.5%**)
- Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**Multi-objective Evolution of Heuristic Using Large Language Model.**" AAAI 2025 (**Oral**)
- Romera-Paredes, et al. "**Mathematical discoveries from program search with large language models**." Nature 2024.
- Niki van Stein, and Thomas Bäck. "**LLaMEA: A Large Language Model Evolutionary Algorithm for Automatically Generating Metaheuristics**." TEVC 2024.
- Chris Lu, Samuel Holt, Claudio Fanconi, Alex Chan, Jakob Foerster, Mihaela van der Schaar, and Robert Lange. "**Discovering preference optimization algorithms with and for large language models**." NeurIPS 2025

Part III
Case Study and LLM4AD Platform

Online Bin Packing Problem (OBP)

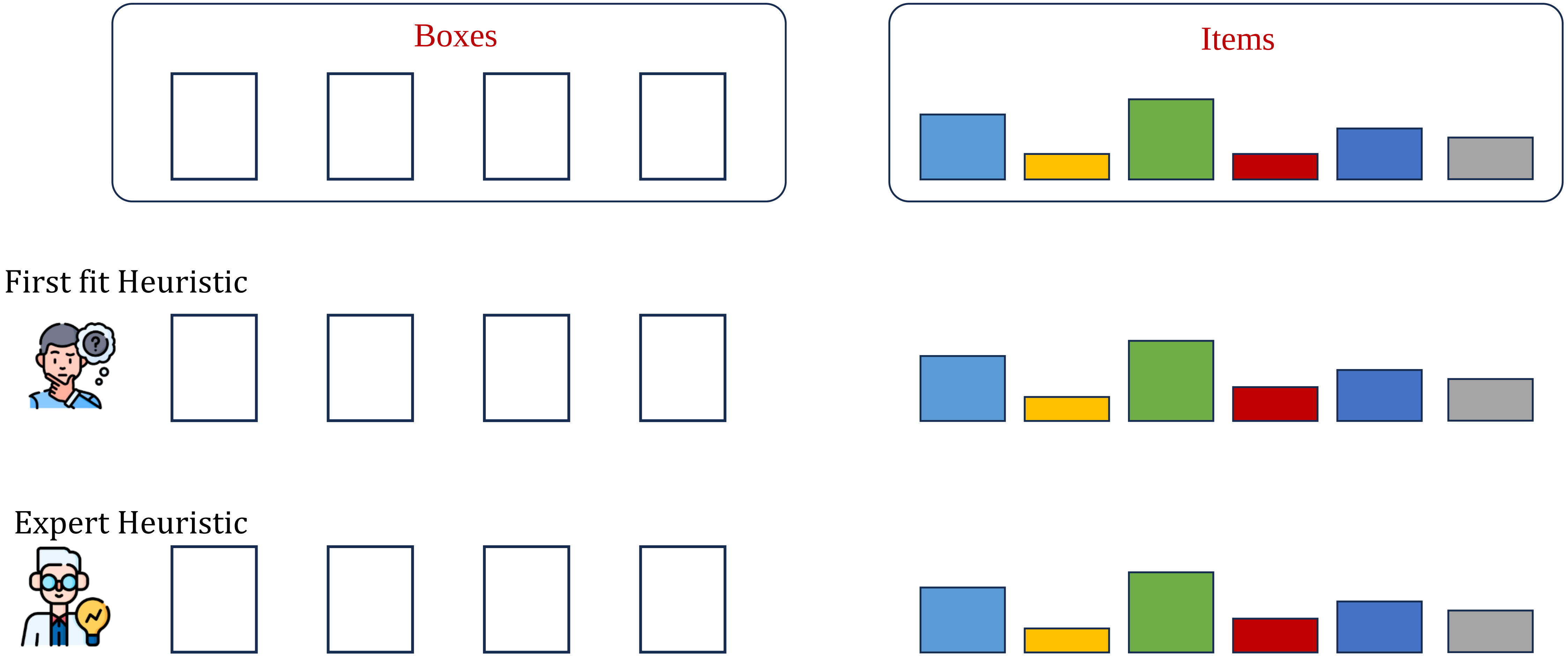
- **Given:** a set of items of various sizes and a set of fixed-sized bins
- **Goal:** pack the items into bins to minimize the number of used bins
- **Online:** one item each step, unknown future items, packed items not moved





- Hang Zhao, Chenyang Zhu, Xin Xu, Hui Huang, and Kai Xu. "Learning practically feasible policies for online 3D bin packing." *Science China Information Sciences* 65, no. 1 (2022): 112105.

Heuristics for OBP



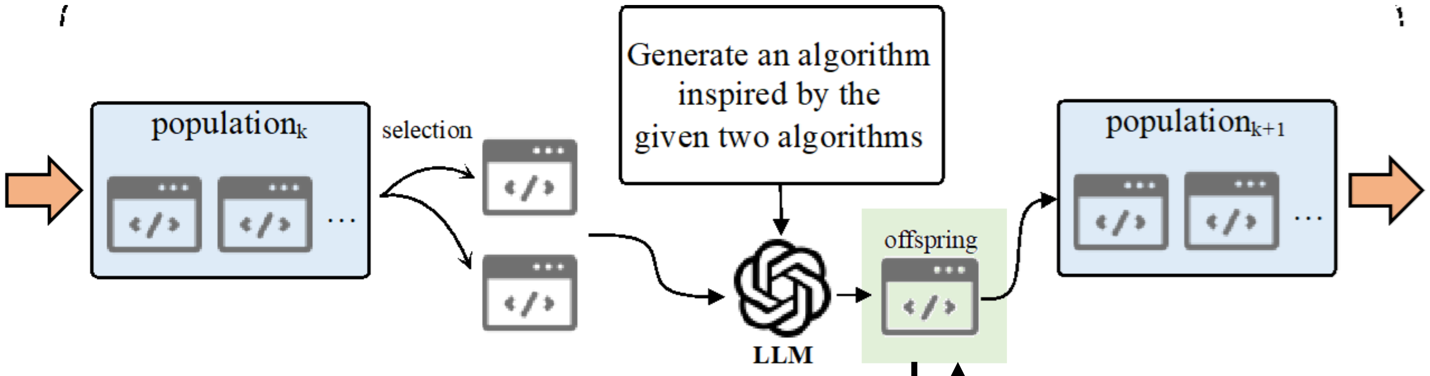
Template

```python
def online_binpack(self,
                   items: tuple[float, ...], bins: np.ndarray, priority: callable
                   ) -> tuple[list[list[float, ...], ...], np.ndarray]:
    """Performs online binpacking of `items` into `bins`."""
    # Track which items are added to each bin.
    packing = [[] for _ in bins]
    # Add items to bins.
    for item in items:
        # Extract bins that have sufficient space to fit item.
        valid_bin_indices = self.get_valid_bin_indices(item, bins)
        # Score each bin based on heuristic.
        priorities = priority(item, bins[valid_bin_indices])
        # Add item to bin with highest priority.
        best_bin = valid_bin_indices[np.argmax(priorities)]
        bins[best_bin] -= item
        packing[best_bin].append(item)
    # Remove unused bins from packing.
    packing = [bin_items for bin_items in packing if bin_items]
    return packing, bins
```

Template used in EoH

```python
def priority(item: float, bins: np.ndarray) -> np.ndarray:
    """Returns priority with which we want to add item to each bin.
    Args:
        item: Size of item to be added to the bin.
        bins: Array of capacities for each bin.
    Return:
        Array of same size as bins with priority score of each bin.
    """
    return bins
```

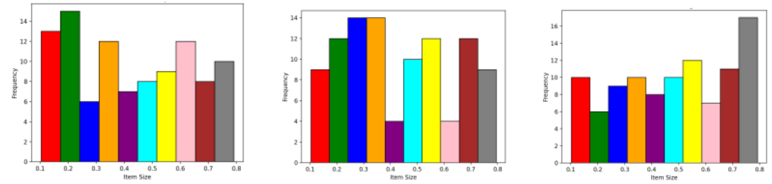
Evaluation



Designed heuristic

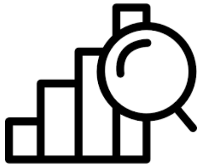
```python
def priority(item: float, bins: np.ndarray) -> np.ndarray:
    """Returns priority with which we want to add item to each bin.
    Args:
        item: Size of item to be added to the bin.
        bins: Array of capacities for each bin.
    Return:
        Array of same size as bins with priority score of each bin.
    """
    return bins
```

Instance



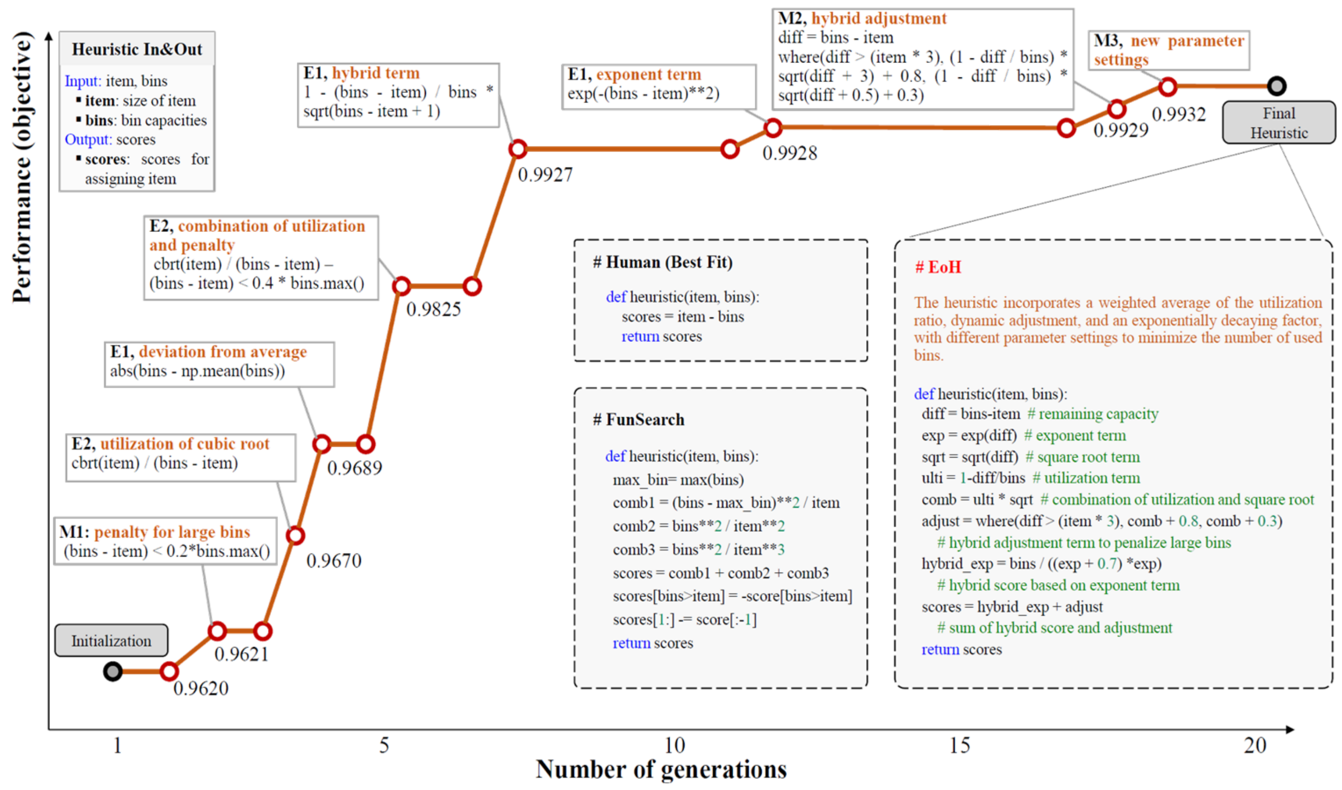
Evaluation block

```python
def online_binpack(self,
                   items: tuple[float, ...], bins: np.ndarray, priority: callable
                   ) -> tuple[list[list[float, ...], ...], np.ndarray]:
    """Performs online binpacking of `items` into `bins`."""
    # Track which items are added to each bin.
    packing = [[] for _ in bins]
    # Add items to bins.
    for item in items:
        # Extract bins that have sufficient space to fit item.
        valid_bin_indices = self.get_valid_bin_indices(item, bins)
        # Score each bin based on heuristic.
        priorities = priority(item, bins[valid_bin_indices])
        # Add item to bin with highest priority.
        best_bin = valid_bin_indices[np.argmax(priorities)]
        bins[best_bin] -= item
        packing[best_bin].append(item)
    # Remove unused bins from packing.
    packing = [bin_items for bin_items in packing if bin_items]
    return packing, bins
```



Evolution Process and Optimal Heuristic

Heuristic Evolution Process



A Comparison of the Heuristics

Heuristic Designed by Human (First Fit)

```
import numpy as np
def heuristic(item, bins):
    scores = -np.arange(len(bins))
    return scores
```

Heuristic Designed by Human (Best Fit)

```
def heuristic(item, bins):
    scores = item - bins
    return scores
```

Heuristic Designed by EoC

```
import numpy as np
def heuristic(item, bins):
    scores = np.log(item) * (bins ** 2) / (
        item * np.sqrt(bins - item)) + (
        bins / item) ** 3
    scores[bins == bins.max()] = -np.inf
    return scores
```

Heuristic Designed by FunSearch

```
def heuristic(item, bins):
  max_bin_cap = max(bins)
  score = (bins - max_bin_cap)**2 / item +
      bins**2 / (item**2)
  score += bins**2 / item**3
  score[bins > item] = -score[bins > item]
  score[1:] -= score[:-1]
  return score
```

Heuristic Designed by EoH

<Heuristic Description>
The heuristic incorporates a weighted average of the utilization ratio, dynamic adjustment, and an exponentially decaying factor, with different parameter settings to minimize the number of used bins.

<Code>

```
import numpy as np
def heuristic(item, bins):
  diff = bins-item  # remaining capacity
  exp = np.exp(diff)  # exponent term
  sqrt = np.sqrt(diff)  # square root term
  ulti = 1-diff/bins  # utilization term
  comb = ulti * sqrt  # combination of utilization and square root
  adjust = np.where(diff > (item * 3), comb + 0.8, comb + 0.3)
      # hybrid adjustment term to penalize large bins
  hybrid_exp = bins / ((exp + 0.7) *exp)
      # hybrid score based on exponent term
  scores = hybrid_exp + adjust
      # sum of hybrid score and adjustment
  return scores
```

Results

a) Human design: **First Fit**, **Best Fit**
b) FunSearch
c) EoH (Ours)

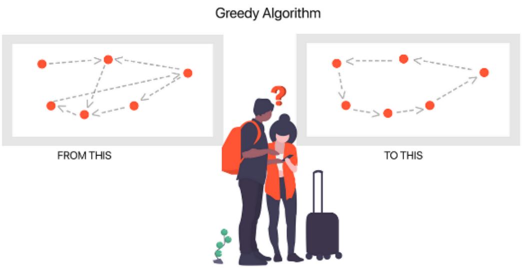
Comparison of the fraction of excess bins to lower bound (lower is better)
for various bin packing heuristics on Weibull instances.

	1k_C100	5k_C100	10k_C100	1k_C500	5k_C500	10k_C500
First Fit	5.32%	4.40%	4.44%	4.97%	4.27%	4.28%
Best Fit	4.87%	4.08%	4.09%	4.50%	3.91%	3.95%
FunSearch	3.78%	**0.80%**	**0.33%**	6.75%	1.47%	0.74%
EoH (Ours)	**2.24%**	**0.80%**	0.61%	**2.31%**	**0.78%**	**0.61%**

More Results: Traveling Salesmen Problem

□ Traveling Salesman Problem (TSP)

- **Given:** a set of positions (cities)
- **Goal:** find the shortest possible route that visits each city exactly once and returns to the origin city



Compared Algorithms

a) Human design heuristics: Nearest Insert (**NI**), Farthest Insert (**FI**), **OR-Tools** (recognized as one of the most powerful solvers)

b) Automatic algorithm design: Attention model (**AM**), **POMO**, **LEHD** (state-of-the-art neural solver)

c) Ours: **EoH**

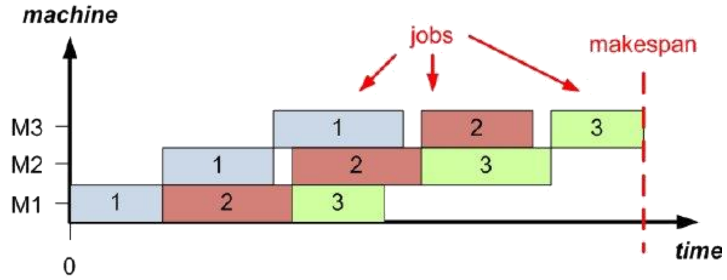
Comparison of the relative distance (%) to the best-known solutions (lower is better) for various routing heuristics on a subset of TSPLib instances.

	rd100	pr124	bier127	kroA150	u159	kroB200
NI	19.91	15.50	23.21	18.17	23.59	24.10
FI	9.38	4.43	8.04	8.54	11.15	7.54
OR-Tools	0.01	0.55	0.66	0.02	1.75	2.57
AM	3.41	3.68	5.91	3.78	7.55	7.11
POMO	0.01	0.60	13.72	0.70	0.95	1.58
LEHD	0.01	1.11	4.76	1.40	1.13	0.64
EoH (Ours)	**0.01**	**0.00**	**0.42**	**0.00**	**0.00**	**0.20**

More Results: Flow Shop Scheduling Problem

Flow Shop Scheduling Problem (FSSP)

- **Given: n jobs** with varying processing times on **m machines**
- **Goal:** minimize the total schedule length (makespan)
- Each job consists of m operations that must be executed in a specific order on the corresponding machine. Same processing order for all jobs. No machine can perform multiple operations simultaneously



Compared Algorithms

a) Human design heuristics: **GUPTA**, **CDS**, **NEH**, **NEHFF**

(NEH and its variants are most commonly used and powerful heuristics)

b) Automatic algorithm design: **PFSPNet** and **PFSPNet_NEH**

(state-of-the-art neural solver)

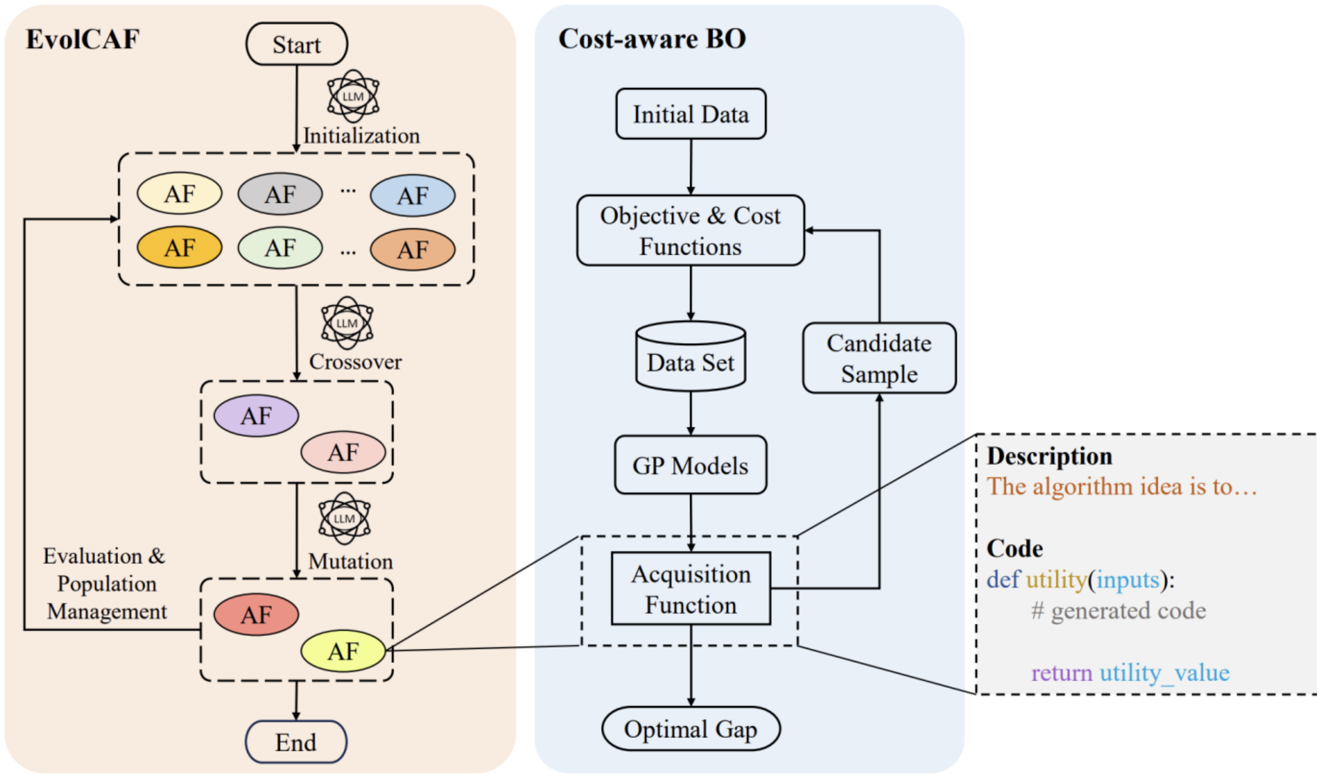
c) Ours: **EoH**

Average relative makespace (%) to the lower bound on FSSP Taillard instances. Lower is better.

	n20m10	n20m20	n50m10	n50m20	n100m10	n100m20
GUPTA	23.42	21.79	20.11	22.78	15.03	21.00
CDS	12.87	10.35	12.72	15.03	9.36	13.55
NEH	4.05	3.06	3.47	5.48	2.07	3.58
NEHFF	4.15	2.72	3.62	5.10	1.88	3.73
PFSPNet	14.78	14.69	11.95	16.95	8.21	16.47
PFSPNet_NEH	4.04	2.96	3.48	5.05	1.72	3.56
EoH (Ours)	**0.30**	**0.10**	**0.19**	**0.60**	**0.14**	**0.41**

More Results: Bayesian Optimization

- Design cost-aware acquisition function in Gaussian process
- EoH outperforms SOTA human-designed methods

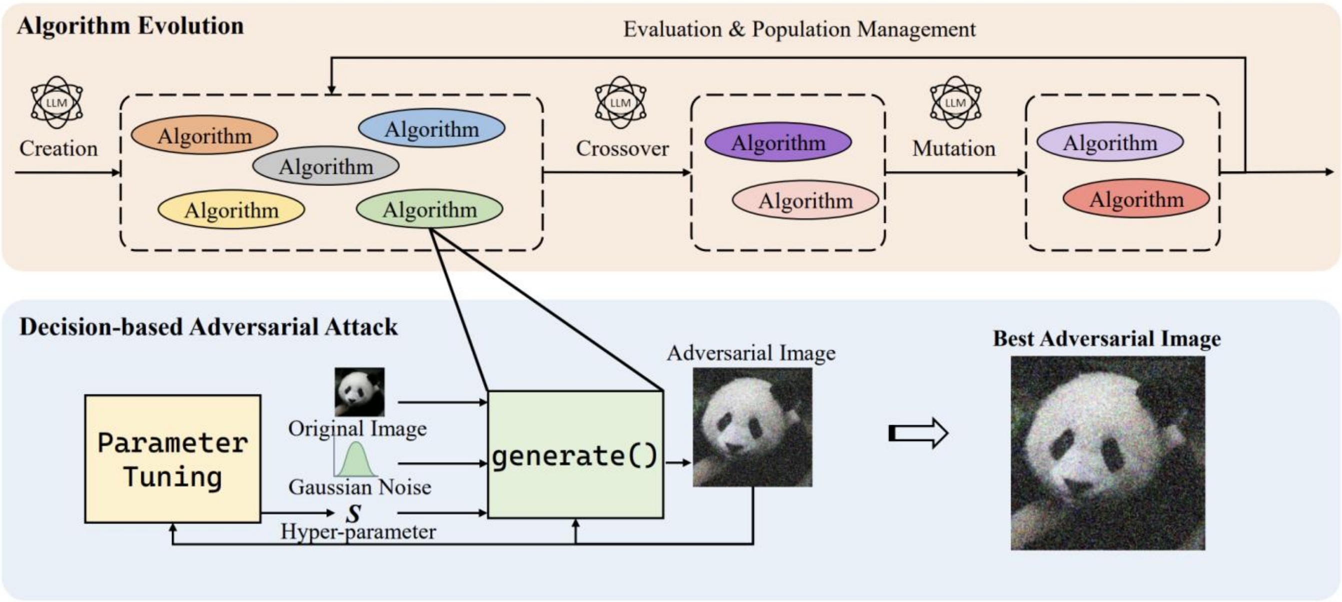


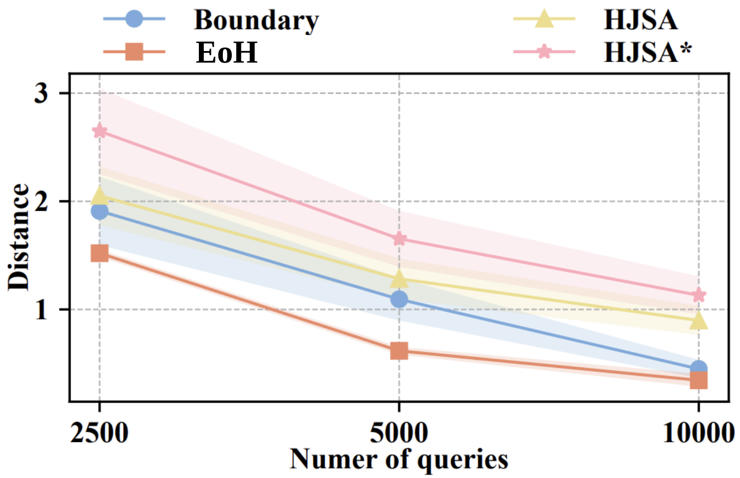
Test Instances	Budget	EI	EIpu	EI-cool	EoH
Ackley 2D	30	2.6600(40)	2.3302(40)	2.7369(40)	**0.4277(34)**
	300	1.2295(395)	0.8582(399)	0.8317(399)	**0.0505(306)**
Rastrigin 2D	30	4.7425(41)	5.6155(41)	5.7754(40)	**0.0511(34)**
	300	1.6656(410)	1.6678(408)	1.8518(408)	**0.0046(306)**
Griewank 2D	30	0.4875(35)	0.3384(36)	0.3374(36)	**0.1762(33)**
	300	0.1305(323)	0.1195(323)	0.1360(323)	**0.0361(307)**
Rosenbrock 2D	30	1.2609(41)	2.3601(44)	2.2909(42)	**0.0304(33)**
	300	0.0332(369)	0.0406(394)	**0.0317(372)**	0.0402(307)
Levy 2D	30	0.0056(38)	0.0098(38)	0.0116(38)	**0.0013(33)**
	300	1.1517e-4(314)	**5.9321e-5(316)**	8.1046e-5(317)	3.7248e-4(307)
ThreeHumpCamel 2D	30	0.0483(39)	0.1182(40)	0.0710(39)	**0.0007(33)**
	300	5.0446e-4(322)	7.4557e-4(326)	**2.6392e-4(325**	7.5310e-4(306)
StyblinskiTang 2D	30	0.0286(41)	0.0233(42)	0.0266(41)	**0.0071(33)**
	300	1.4420e-4(332)	1.8616e-4(339)	**6.1798e-5(343**	2.0142e-3(306)
Hartmann 3D	30	5.6696e-5(40)	1.0364e-4(41)	**4.6158e-5(40)**	4.8127e-4(36)
	300	1.8263e-5(420)	1.3089e-5(429)	**9.0599e-6(432**	2.3656e-4(311)
Powell 4D	30	18.8892(48)	19.8281(51)	14.9481(49)	**0.1285(38)**
	300	2.9839(376)	1.1173(395)	1.6806(391)	**0.0136(316)**
Shekel 4D	30	7.9123(48)	7.9210(49)	8.2132(48)	**2.6367(39)**
	300	6.5193(545)	6.9044(545)	7.0135(551)	**0.1993(315)**
Hartmann 6D	30	0.0326(52)	0.0296(52)	**0.0278(52)**	0.0384(44)
	300	0.0122(710)	0.0054(705)	0.0154(695)	**0.0042(327)**
Cosine8 8D	30	0.4723(48)	0.4738(48)	0.5351(48)	**0.4357(53)**
	300	0.1707(532)	0.2364(533)	0.2779(527)	**0.0148(342)**

Yiming Yao, Fei Liu, Ji Cheng, and Qingfu Zhang. Evolve Cost-aware Acquisition Functions Using Large Language Models. **PPSN 2024**. (**Best Paper Nomination**)

More Results: Image Adversarial Attack

- ❑ Design decision-based adversarial attack method
- ❑ EoH outperforms many human-designed decision-based adversarial attack methods



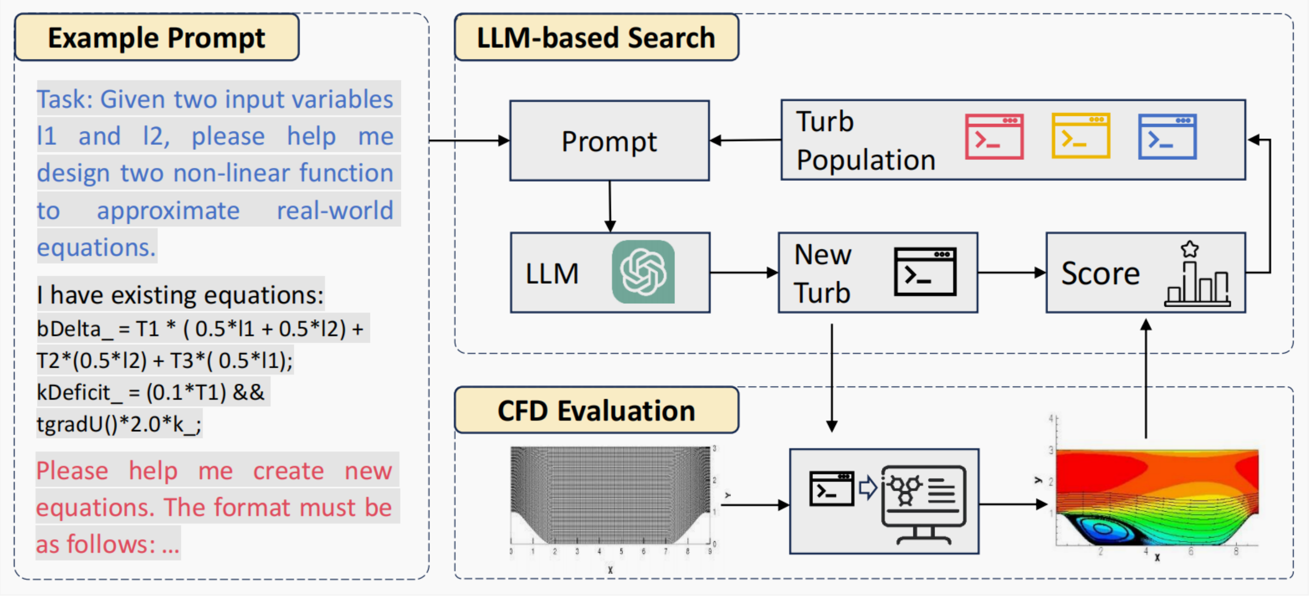


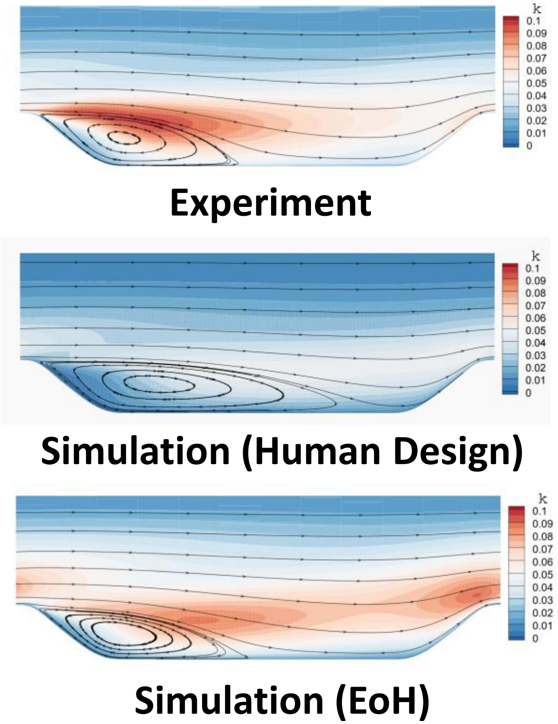
Attack Name	Distance (ℓ_2-norm)			Attack Success Rate		
# of Queries	2500	5000	10000	2500	5000	10000
Boundary	$1.9107_{1.2665}$	$1.0938_{0.7861}$	$0.4495_{0.3340}$	**14.7**	**26.2**	65.5
HSJA	$2.0512_{1.0876}$	$1.2833_{0.7442}$	$0.8978_{0.5360}$	9.2	16.1	24.6
HSJA*	$2.6482_{1.5790}$	$1.6532_{1.0347}$	$1.1306_{0.6987}$	7.9	13.9	19.6
EoH	$\mathbf{1.5202_{0.1337}}$	$\mathbf{0.6171_{0.1430}}$	$\mathbf{0.3445_{0.2386}}$	0.0	0.5	**80.3**

- Ping Guo, Fei Liu, Xi Lin, Qingchuan Zhao, and Qingfu Zhang. L-AutoDA: Leveraging large language models for automated decision-based adversarial attacks. **GECCO** ***2024***

More Results: Computational Fluid Dynamics

- In numerical simulation using Navier-Stocks equations, existing turbulence models often fail to represent the complex turbulent flows due to **liner assumption**.
- **Using EoH to design algebraic expressions for correcting the turbulence model in the Reynolds-averaged Navier-Stokes equations.**

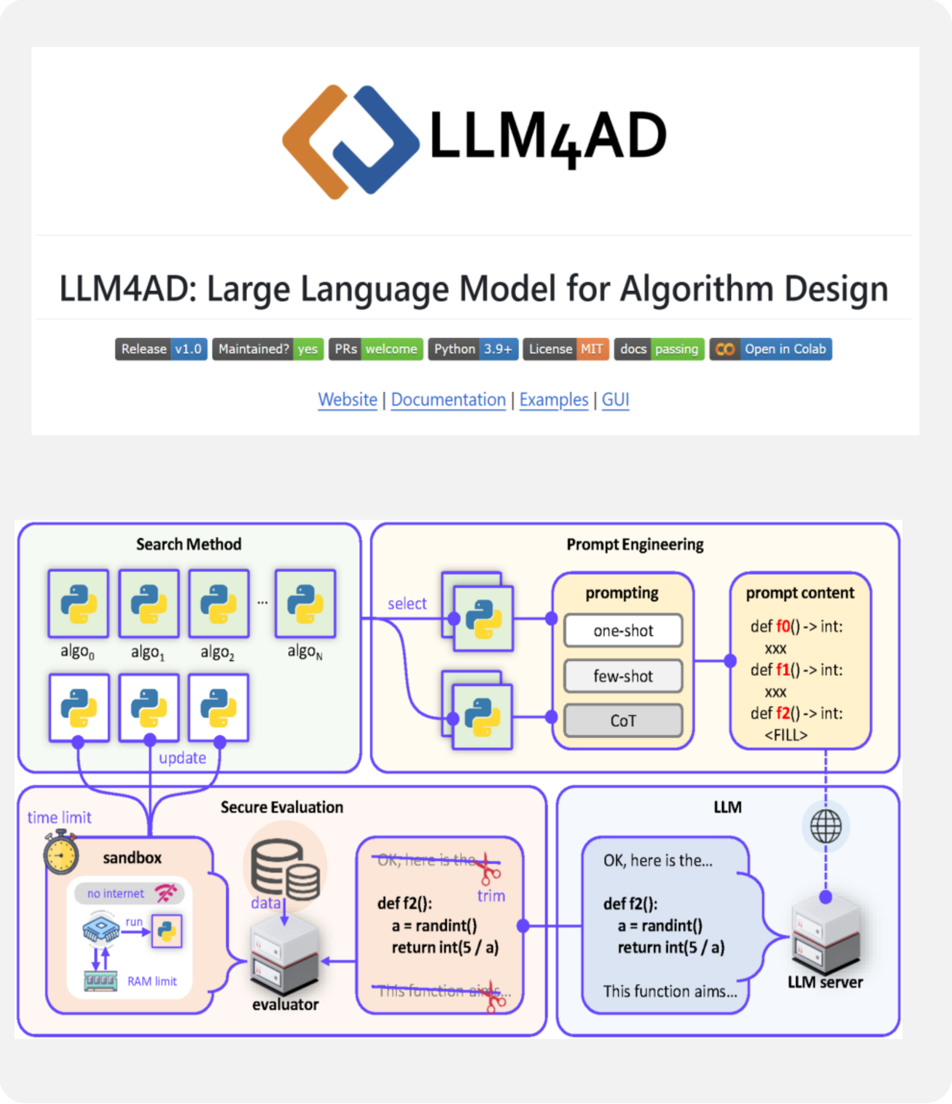




- Zhang, Yu, Kefeng Zheng, Fei Liu, Qingfu Zhang, and Zhenkun Wang. "**AutoTurb: Using Large Language Models for Automatic Algebraic Model Discovery of Turbulence Closure**." Physics of Fluid, 2025.

Large Language Model for Algorithm Design (LLM4AD)

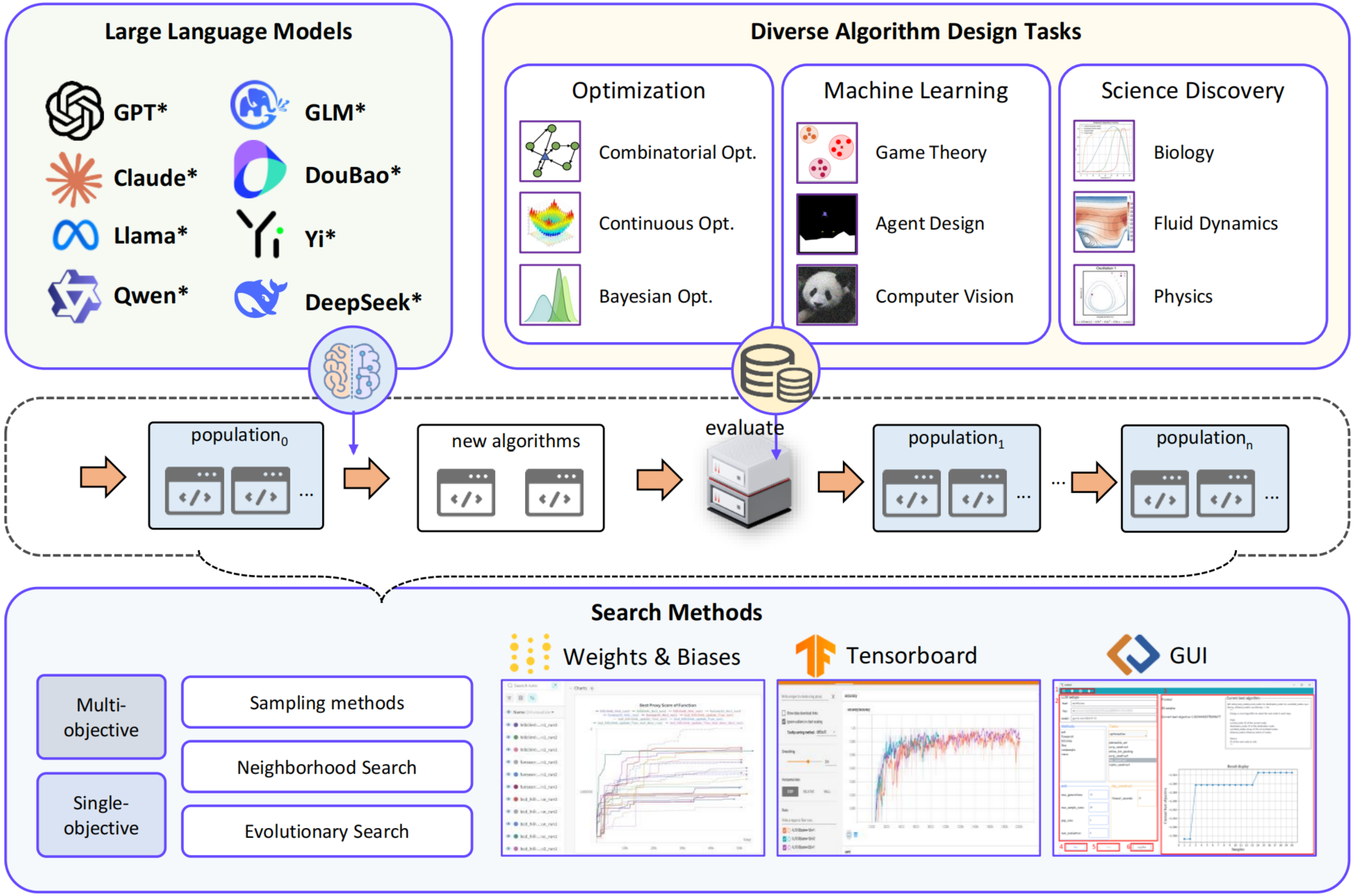
LLM4AD Platform



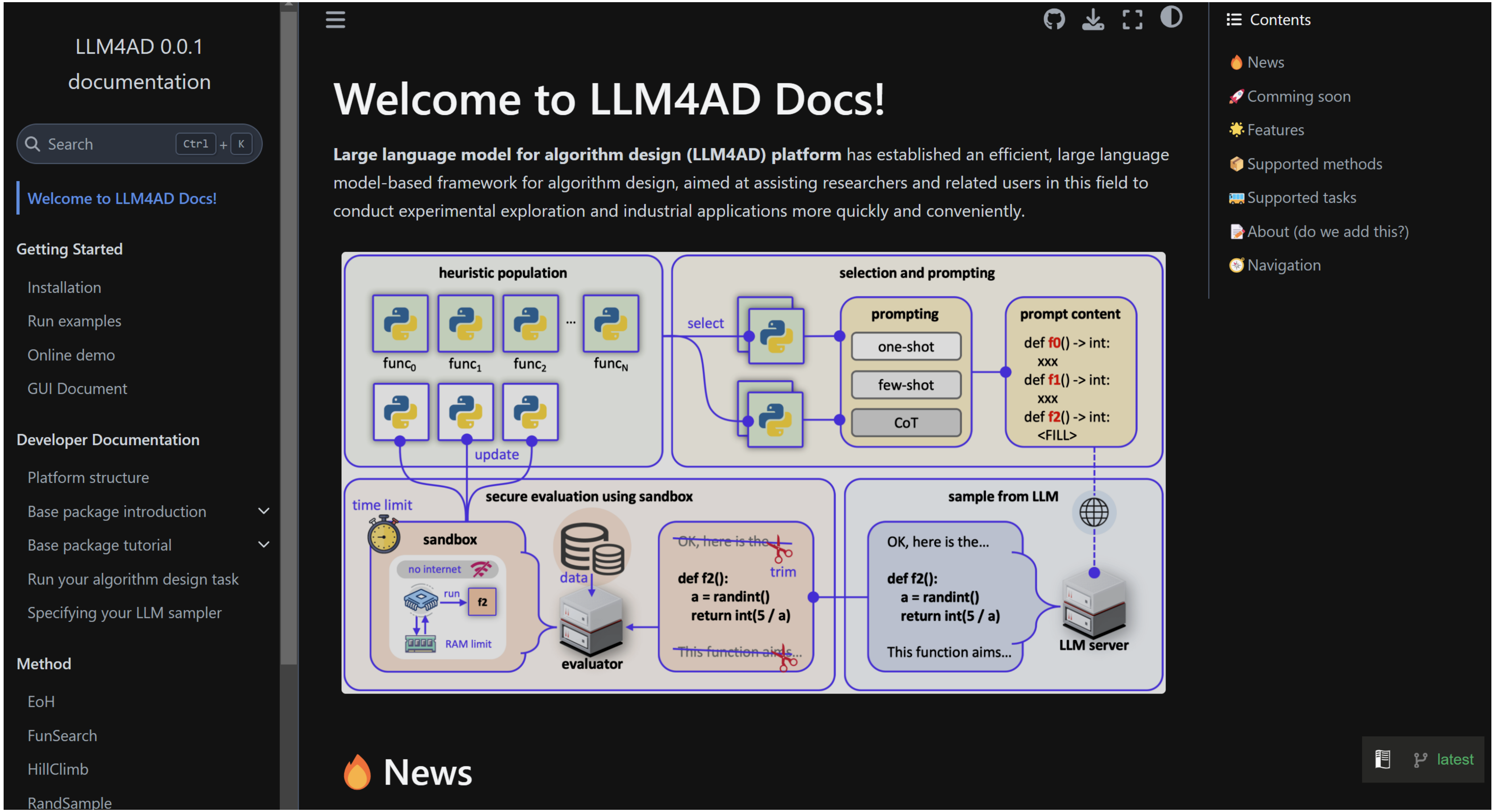
- Github Rep：
  https://github.com/Optima-CityU/llm4ad
- LLM4AD Documents：
  https://llm4ad-doc.readthedocs.io/en/latest/
- LLM4AD Web：
  www.llm4ad.com

LLM4AD Overview

- **Diverse search methods**
  - EoH
  - MEoH
  - Hill-climb
  - Local Search
  - Sampling
- **10+ LLMs**
  - GPT, Deepseek, Claude, Llama, GLM,Qwen, DouBao, Yi….
- **100+Tasks**
  - Optimization
  - Machine Learning
  - Science Discovery
- **Supports**
  - Handbook
  - Tutorial
  - GUI
  - …

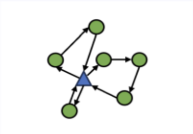


Support Docs



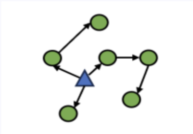
Algorithm Design Task Examples

Optimization



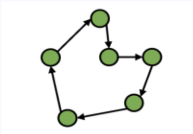
Capacitated Vehicle Routing Problem

Constructive Heuristics for Capacitated Vehicle Routing Problem (CVRP).



Open Vehicle Routing Problem

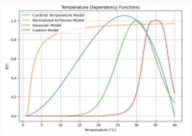
The Open Vehicle Routing Problem (OVRP) is a variant of VRP that has open routes.



Traveling Salesman Problem

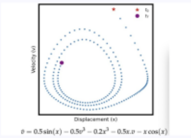
Constructive Heuristics for Traveling Salesman Problem (TSP).

Science Discovery



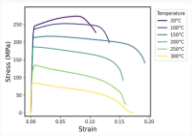
Bacteria Growth

A biology-focused task aiming to discover growth patterns.



Oscillator1

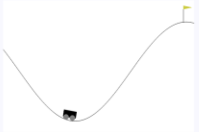
A mathematical task aimed at uncovering oscillator behaviors.



Stress & Strain

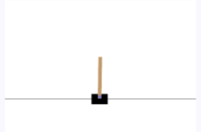
A physics task focused on discovering relationships.

Machine Learning



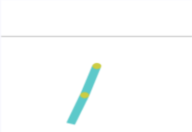
Mountain Car

To optimize the car's actions to reach a target with minimal iterations under specific position and velocity constraints.



Cart Pole

Aiming to maximize the duration that a pole remains balanced on a moving cart within specific position and angle constraints.



Acrobot

A target height by applying torque to the actuated joint within specified angular constraints.

Benchmarking Results

4 Evolutionary Methods

- ✓ EoH
- ✓ 1+1 EPS
- ✓ FunSearch
- ✓ Random Sample

9 Algorithm Design Tasks

Optimization

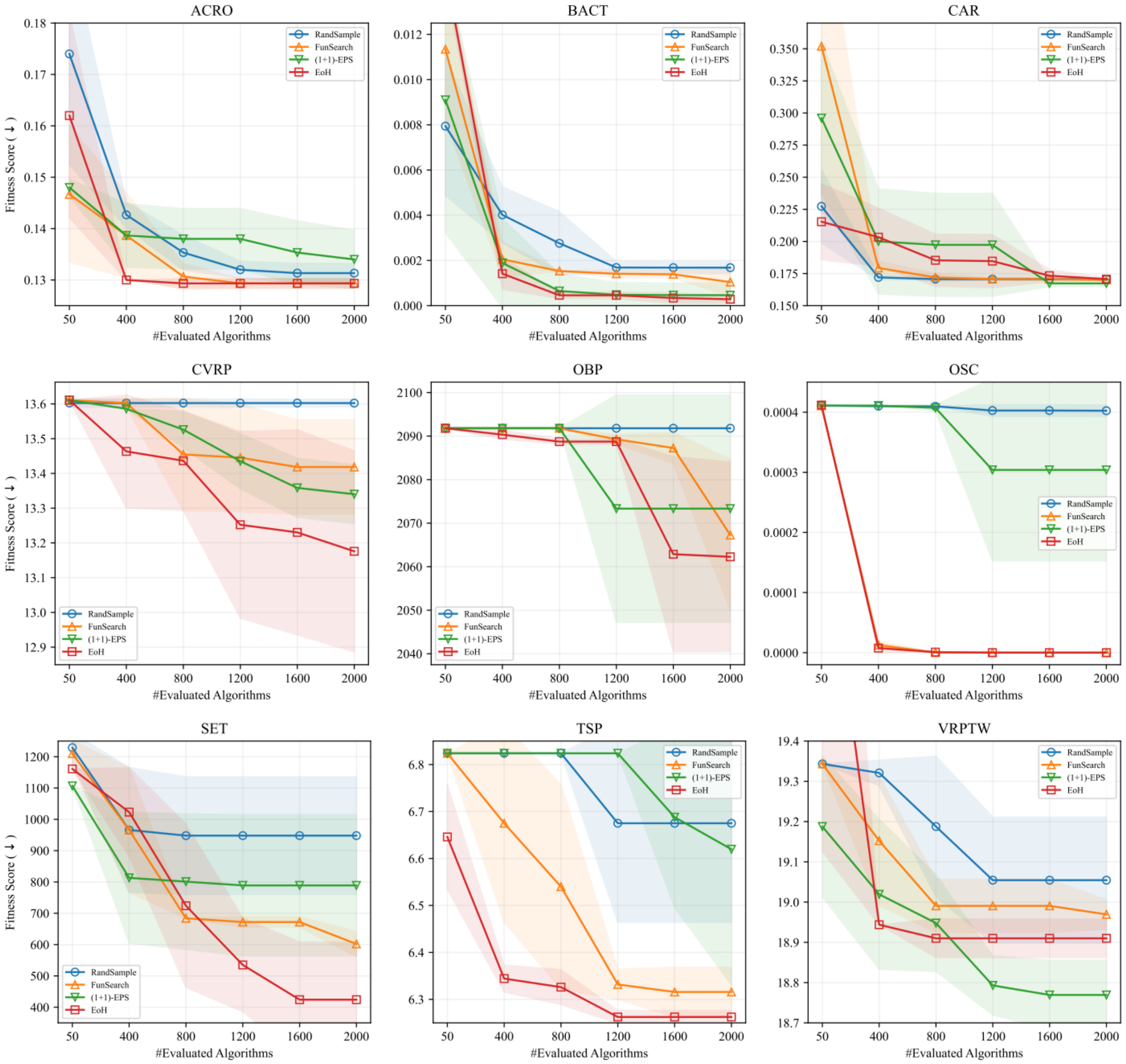
- ✓ Admissible Set (Set)
- ✓ Online Bin Packing (OBP)
- ✓ Traveling Salesman Problem (TSP)
- ✓ Capacitated Vehicle Routing Problem (CVRP)
- ✓ Vehicle Routing Problem with Time Windows (VRPTW)

Science Discovery

- ✓ Oscillator (OSC)
- ✓ Bacterial Growth (BACT)

Machine Learning

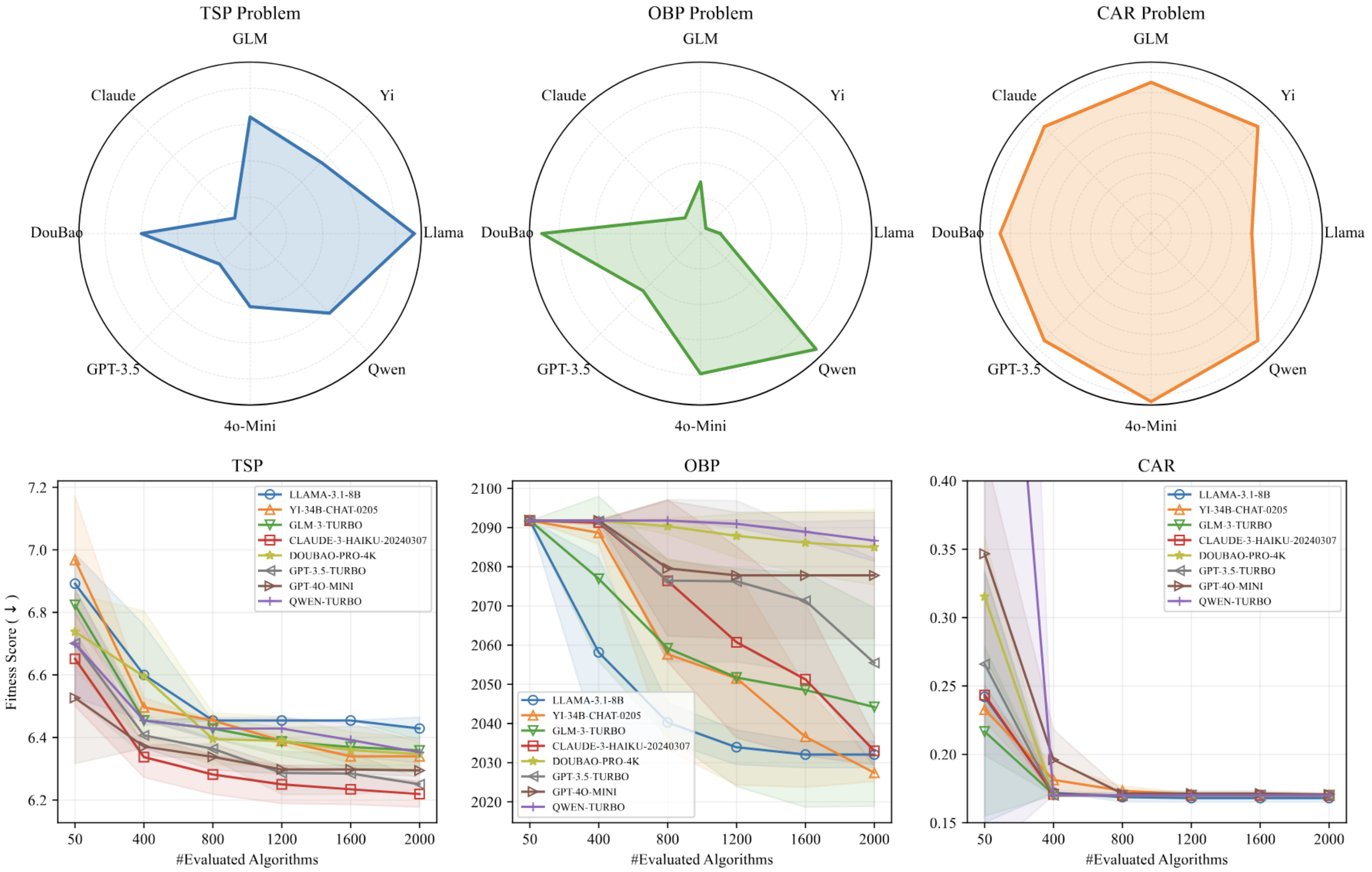
- ✓ Acrobot (ACRP)
- ✓ Car Mountain (CAR)



Benchmarking Results

Eight LLMs

- ✓ **Open AI**
  - GPT-3.5
  - GPT4o-mini
- ✓ **Claude**
- ✓ **Llama**
- ✓ **Qwen**
- ✓ **GLM**
- ✓ **YI**
- ✓ **DouBao**



Usage

```python
from llm4ad.task.optimization.tsp_construct import TSPEvaluation
from llm4ad.tools.llm.llm_api_https import HttpsApi
from llm4ad.tools.profiler import ProfilerBase
from llm4ad.method.eoh import EoH


def main():
    llm = HttpsApi(host='xxx',  # your host endpoint, e.g., api.openai.com, api.deepseek.com
                   key='sk-xxx',  # your key, e.g., sk-abcdefghijklmn
                   model='xxx',  # your llm, e.g., gpt-3.5-turbo, deepseek-chat
                   timeout=20)

    task = TSPEvaluation()

    method = EoH(llm=llm,
                 profiler=ProfilerBase(log_dir='logs', log_style='complex'),
                 evaluation=task,
                 max_sample_nums=20,
                 max_generations=5,
                 pop_size=2,
                 num_samplers=1,
                 num_evaluators=1)

    method.run()


if __name__ == '__main__':
    main()
```

Import task and method

Set up LLM interface

Set up Task, e.g., TSP

Set up Method, e.g., EoH

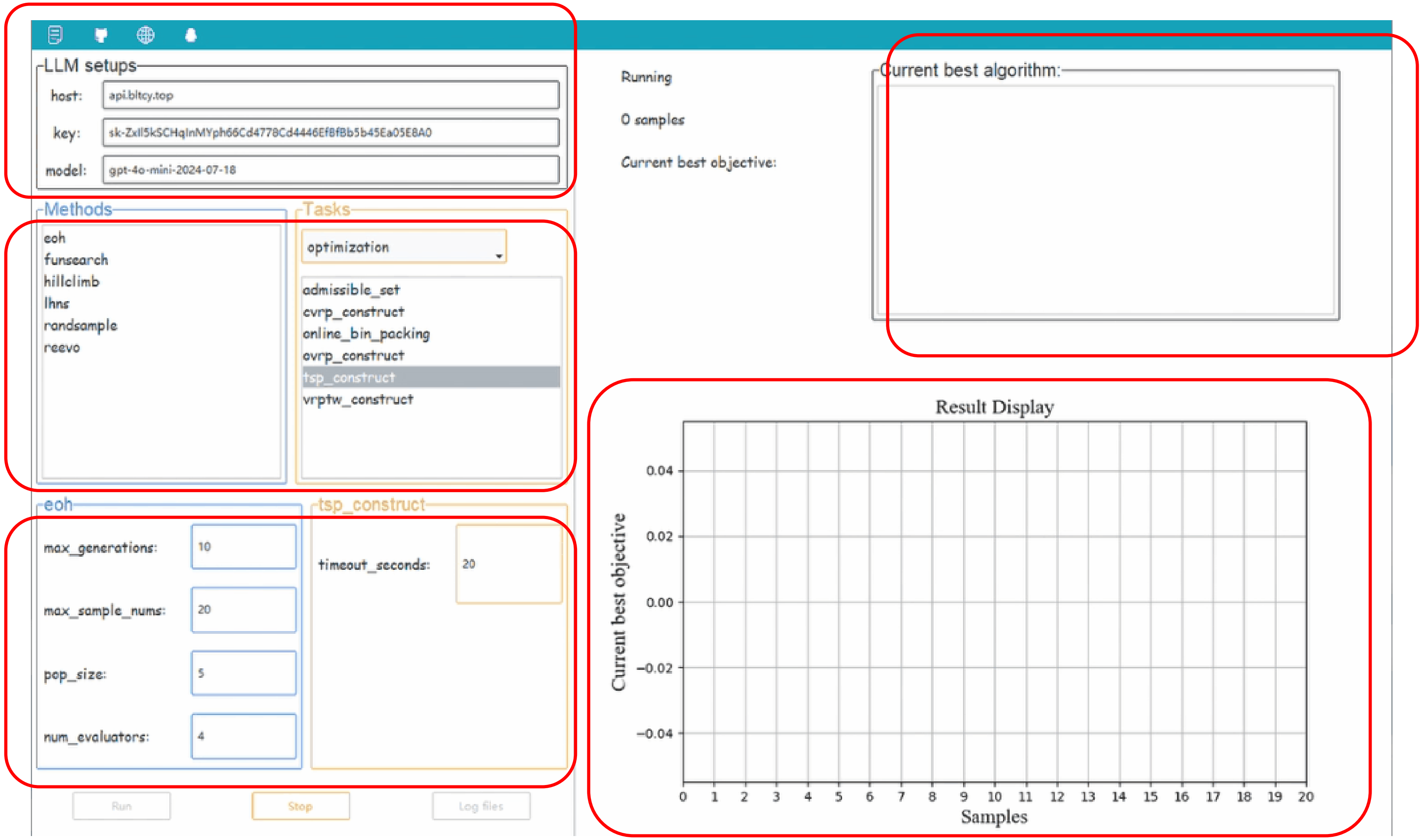
Run

GUI Usage

LLM Interface

Methods & Tasks

Settings of Methods & Tasks



Current Best Algorithm

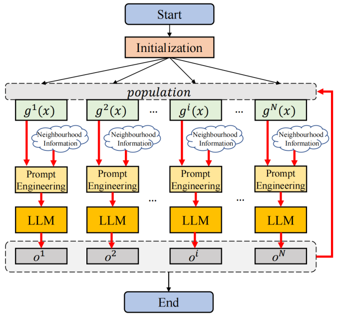
Convergence

Part IV
LLM with/for Multiobjective Optimization

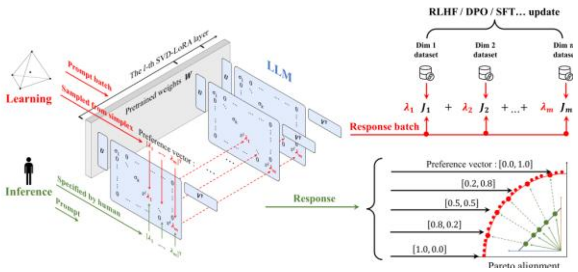
EC+LLM with/for Multiobjective Optimization

Ours

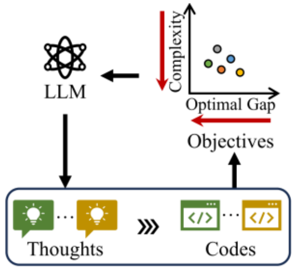
2023-10-19 LLM for MOEA



2024-02-03 Panacea
Pareto alignment for LLMs

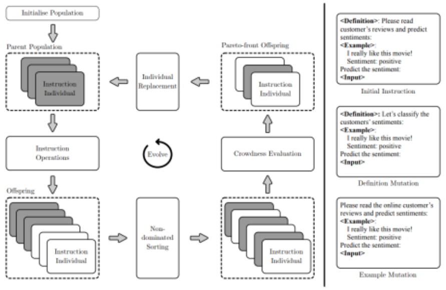


2024-09 MEoH: Multi-objective Evolution of Heuristic using LLM (AAAI 2025 Oral)

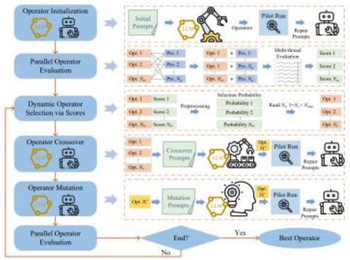


Others

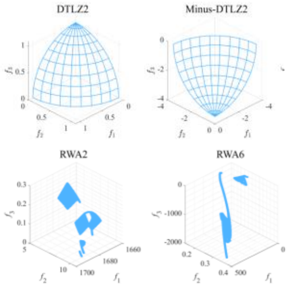
2023-10-26 Instoptima



2024-6-13 Autonomous MOO using LLM

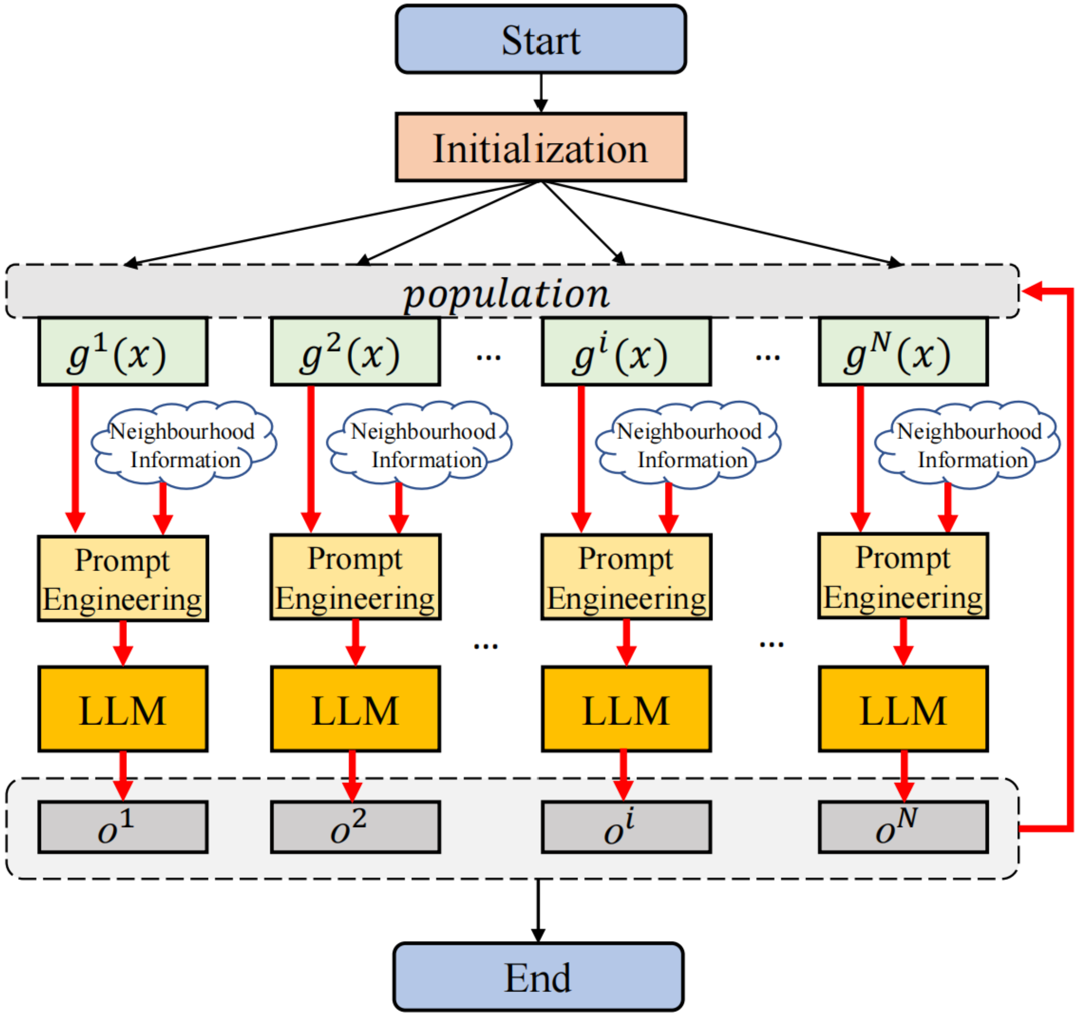


2025-02-28
Benchmarking Experiment Settings using LLMs



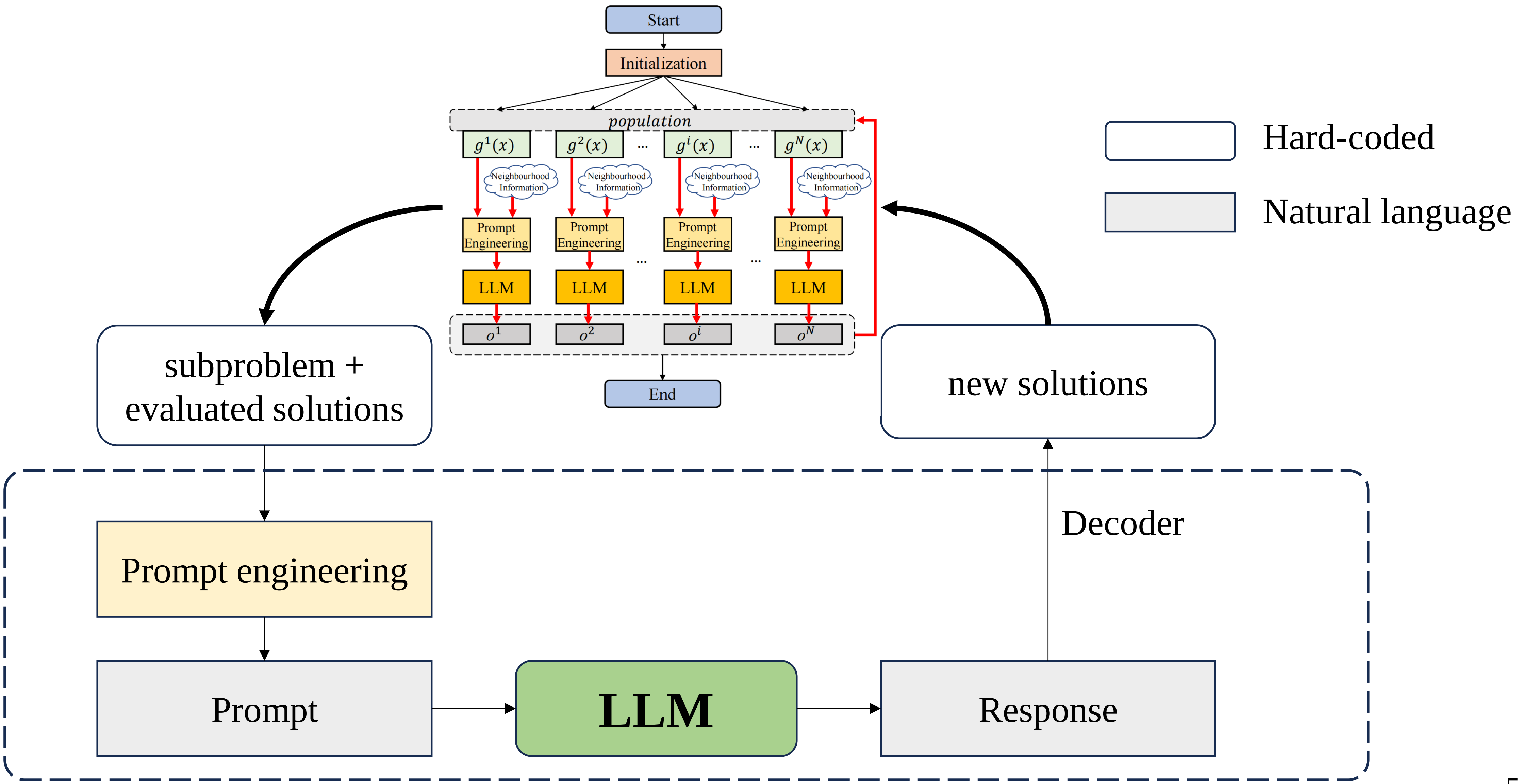
- Fei Liu, Xi Lin, Zhenkun Wang, Shunyu Yao, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. "**Large language model for multi-objective evolutionary optimization**." arXiv 2023, EMO2025.
- Heng Yang, and Ke Li. "**Instoptima: Evolutionary multi-objective instruction optimization via large language model-based instruction operators**." EMNLP 2023
- Yifan Zhong, Chengdong Ma, Xiaoyuan Zhang, Ziran Yang, Haojun Chen, Qingfu Zhang, Siyuan Qi, and Yaodong Yang. "**Panacea: Pareto alignment via preference adaptation for llms**." NeurIPS 2024
- Yuxiao Huang, Shenghao Wu, Wenjie Zhang, Jibin Wu, Liang Feng, and Kay Chen Tan. "**Autonomous multi-objective optimization using large language model**." arXiv 2024.
- Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**Multi-objective Evolution of Heuristic Using Large Language Model.**" AAAI 2025 (**Oral**)
- Lie Meng Pang, Hisao Ishibuchi, "**Large Language Model-Based Benchmarking Experiment Settings for Evolutionary Multi-Objective Optimization**" arXiv 2025.

LLMs as Optimizers for MOEAs



- LLMs are used as **pre-trained black-box operators** to suggest new candidate solutions for each subproblem
- **Prompt engineering** is used for LLM **in-context learning**

LLMs as Optimizers for MOEAs



LLMs as Optimizers for MOEAs

☐ Example Prompt

1. **Description of task**
2. **In-context samples**
3. **Expected outputs**

Example Prompt:

Now you will help me minimize a function with 4 variables. I have some points and the function values of them. The points start with <start> and end with <end>. The points are arranged in descending order based on their function values, where lower values are better.

point: <start>0.344,0.940,0.582,0.878<end>
value: 4.582
point: <start>0.376,0.973,0.604,0.828<end>
value: 4.530
...
point: <start>0.787,0.610,0.053,0.420<end>
value: 2.399
point: <start>0.012,0.532,0.001,0.196<end>
value: 1.474

Give me 2 new points that are different from all points above, and have a function value lower than any of the above. **Do not write code. Do not give any explanation. Each output new point must start with <start> and end with <end>.**

LLMs as Optimizers for MOEAs

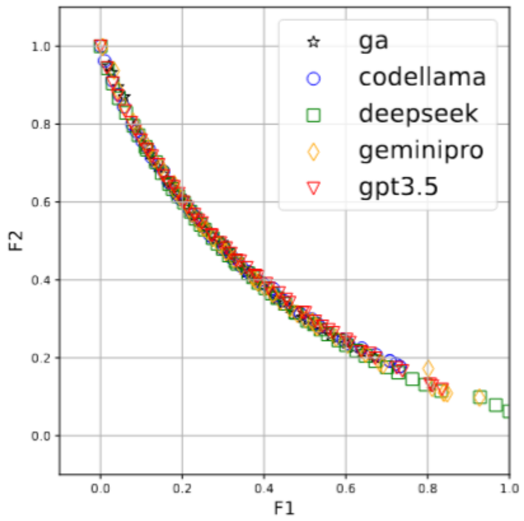
☐ Settings

- **Instances:** RE instances [1]
- **LLMs:** GPT-3.5-turbo, Deepseek, GeminiPro, Codellama

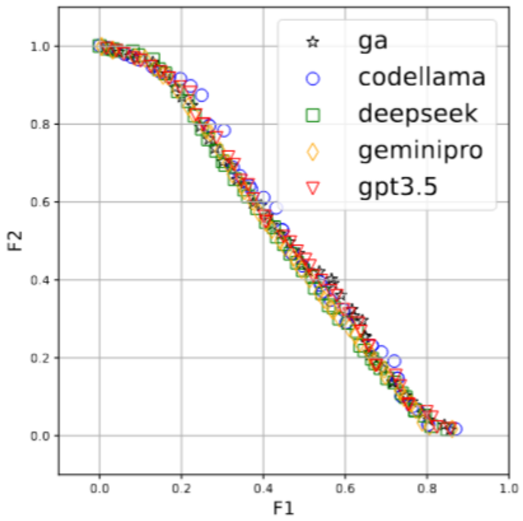
➢ Competitive performance when compared to MOEA with conventional hand-crafted operators (e.g., GA)

➢ Exhibits robust performance across various LLMs.

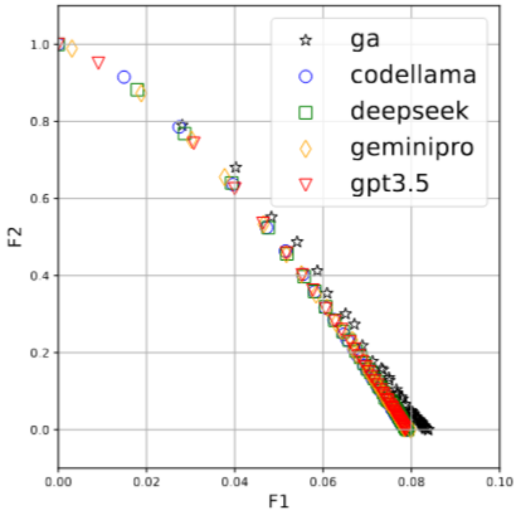
➢ However, it lacks interpretability and requires interaction with LLMs during optimization.



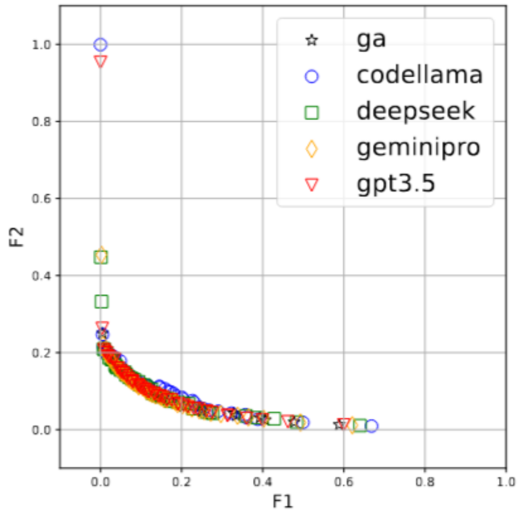
a) RE21



b) RE22



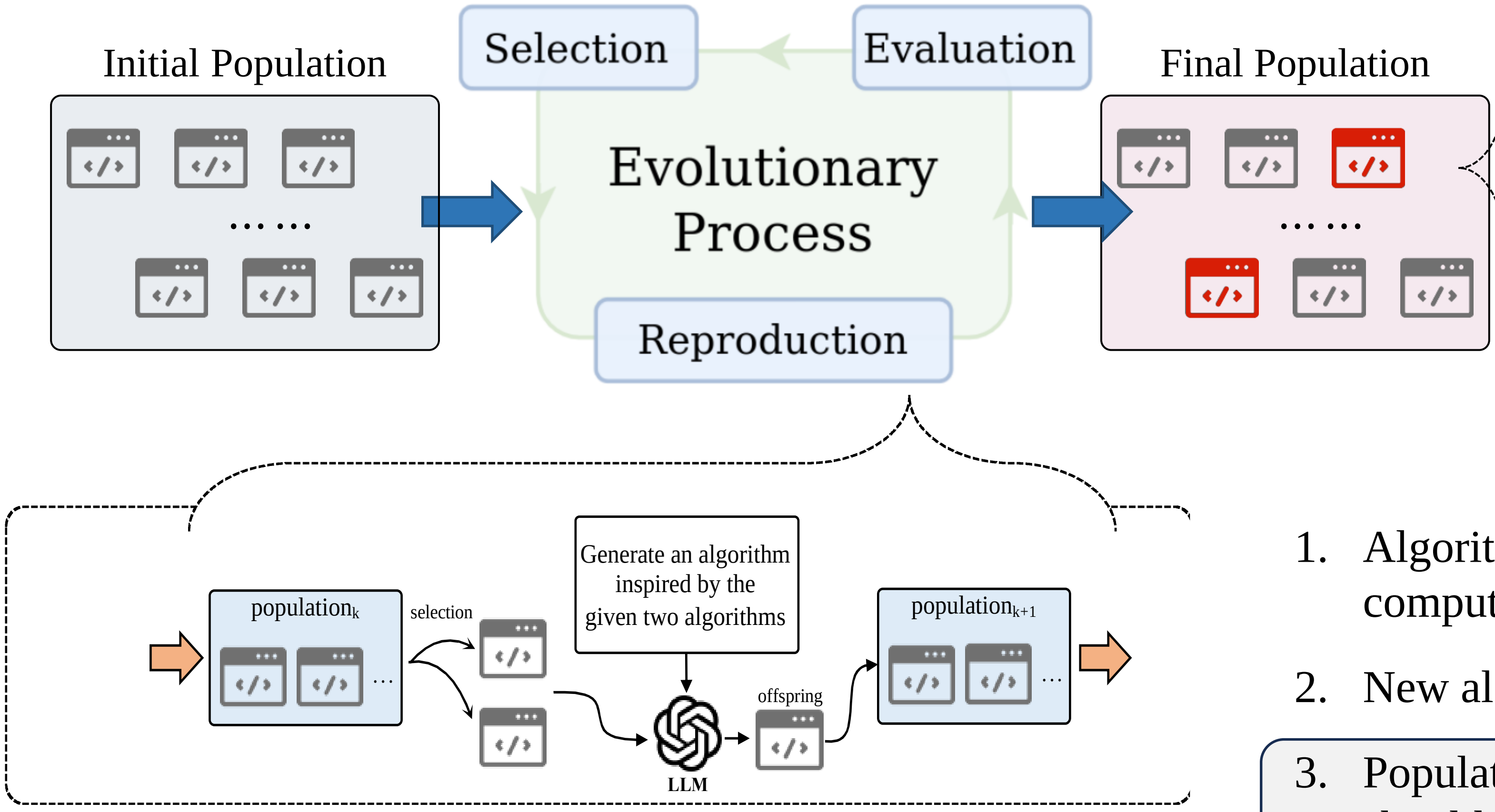
c) RE23

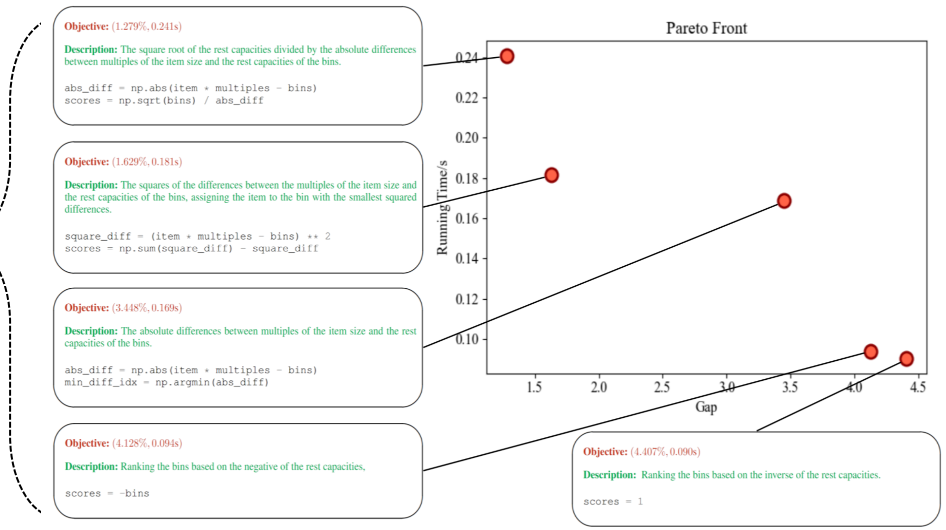


d) RE24

1. Tanabe, Ryoji, and Hisao Ishibuchi. "An easy-to-use real-world multi-objective optimization problem suite." *Applied Soft Computing* 89 (2020): 106078.

Multobjective Evolution of Heuristics (MEoH)



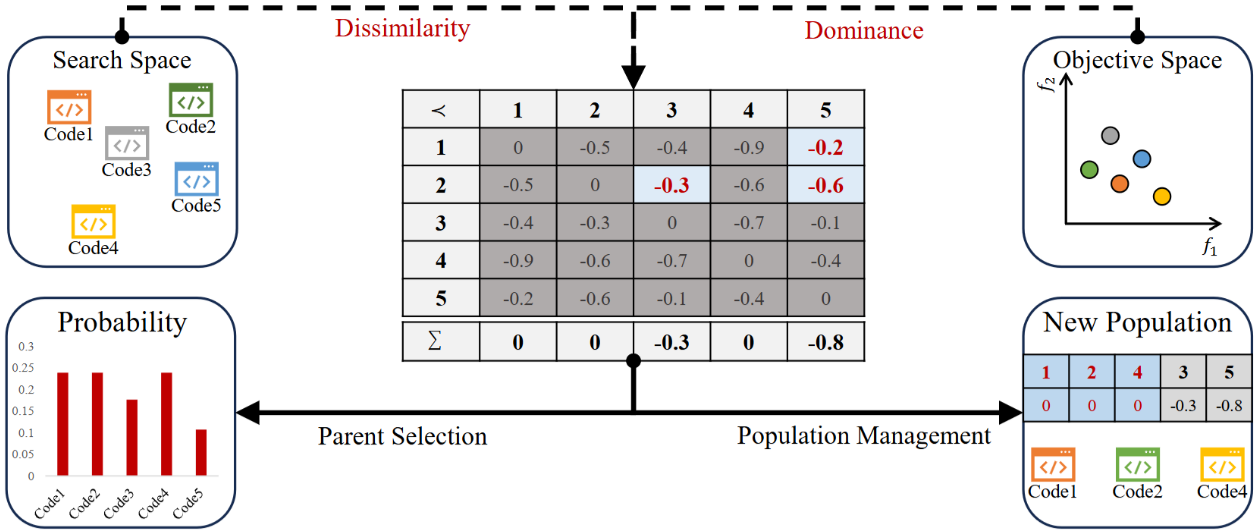


1. Algorithms are represented as executable computer programs.
2. New algorithms are generated using LLMs.
3. Population management and parent selection should account for both **dominance** and **diversity**.

- Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**Multi-objective Evolution of Heuristic Using Large Language Model.**" AAAI 2025 (**Oral**)

Multobjective Evolution of Heuristics (MEoH)

- **Dominance:** Pareto dominance on objective space
- **Dissimilarity:**
  - Algorithms are represented using natural language descriptions and Python code
  - We evaluate the dissimilarity between code segments
  - We choose to utilize Abstract Syntax Tree (AST) for similarity measurement



- Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**Multi-objective Evolution of Heuristic Using Large Language Model.**" AAAI 2025 (**Oral**)

Multobjective Evolution of Heuristics (MEoH)

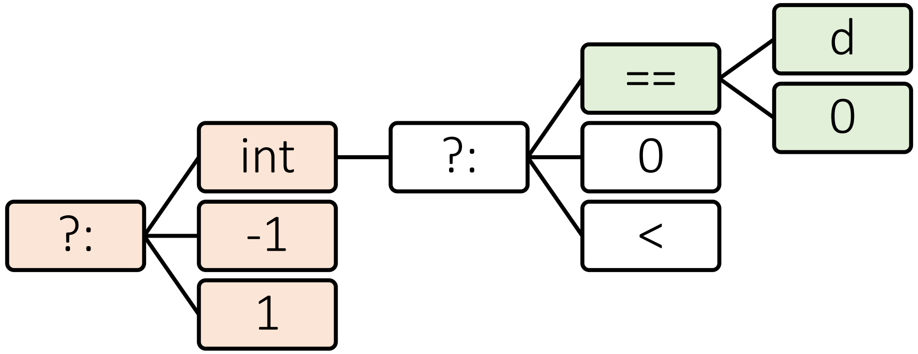
☐ Dissimilarity measurement

$$\mathrm{Sim}_{\mathrm{AST}}(a,b) = \mathrm{Count}_{\mathrm{clip}}(\mathrm{Tree}_a)/\mathrm{Count}(\mathrm{Tree}_b)$$

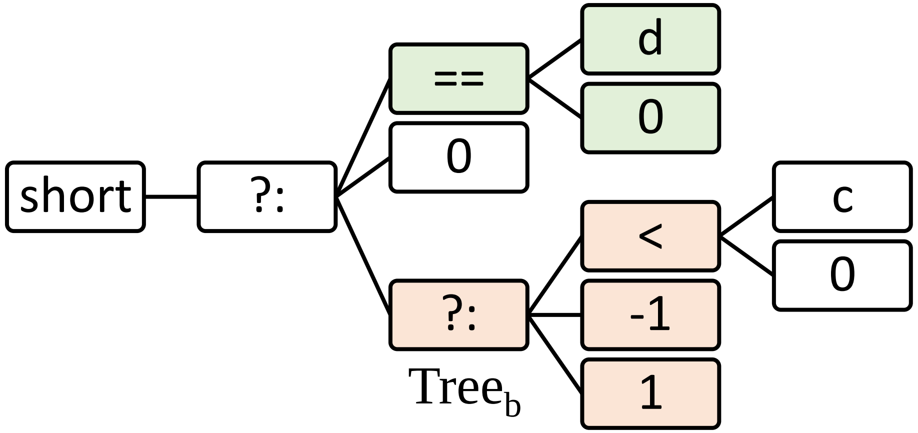
- Count(): The number of subtrees,
- $\mathrm{Count}_{\mathrm{clip}}$(): The number of matched subtrees
- Considering the syntactic structure and ignoring the names

```
public static int Sign(double d){
    return ((int)((d==0)?0:(d<0)))?-1:1;
}
```

```
public static int Sign(double d){
    return (short)((d==0)?0:(c<0)?-1:1);
}
```



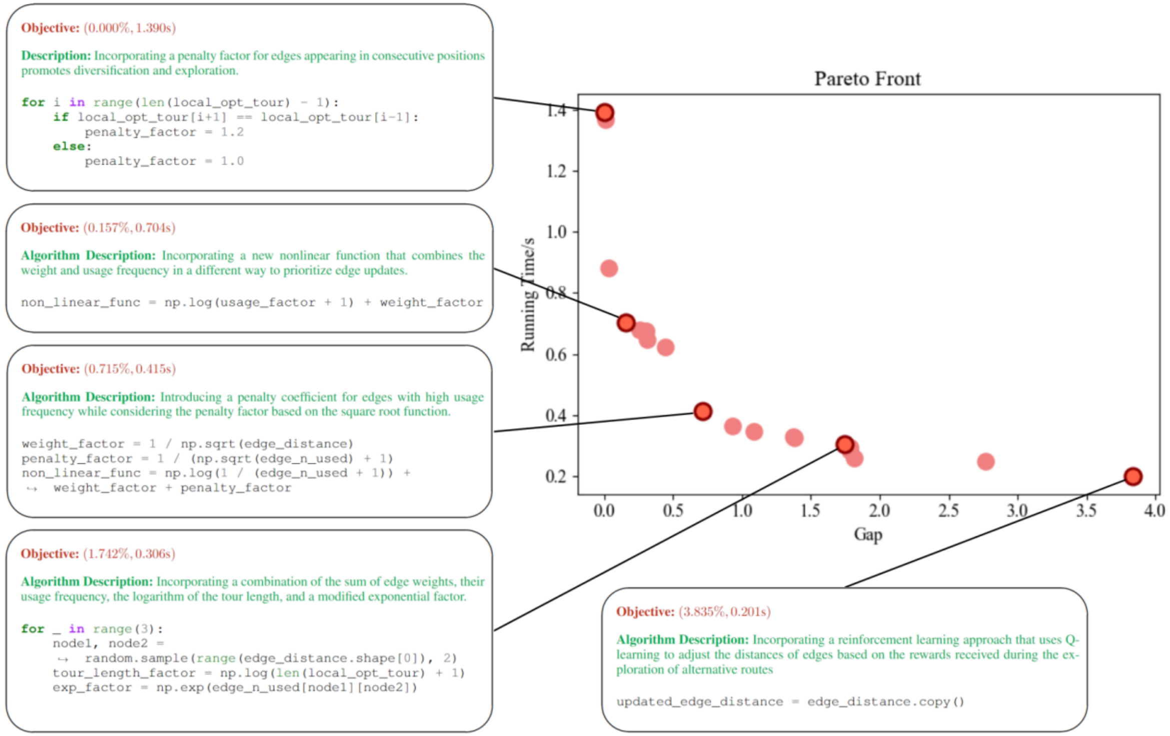
Tree_a

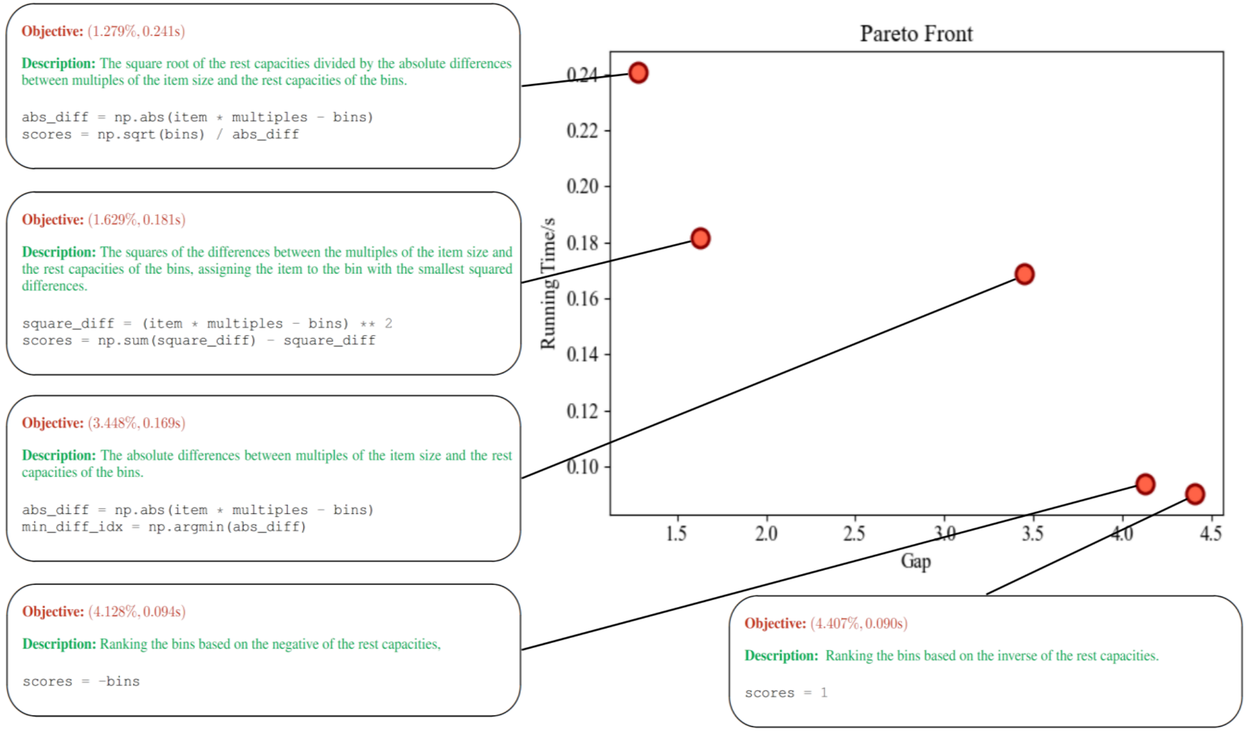


Tree_b

Multobjective Evolution of Heuristics (MEoH)

- Online Bin Packing Non-dominated Heuristics
- TSP Non-dominated Heuristics





- Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**Multi-objective Evolution of Heuristic Using Large Language Model.**" AAAI 2025 (**Oral**)

Summary

Takeaways

- Algorithm design with LLMs is a growing direction.
- **EoH** combines LLMs and Evolutionary Search, it introduces a new paradigm for automated algorithm development.
- **LLM4AD** is an open-source, user-friendly platform for LLM-based algorithm design.
- LLMs can be effectively utilized with/for **MOEAs**.

Ongoing work: a system that reads papers/books, designs algorithms (using EoH) and writes tech reports/papers **automatically**.

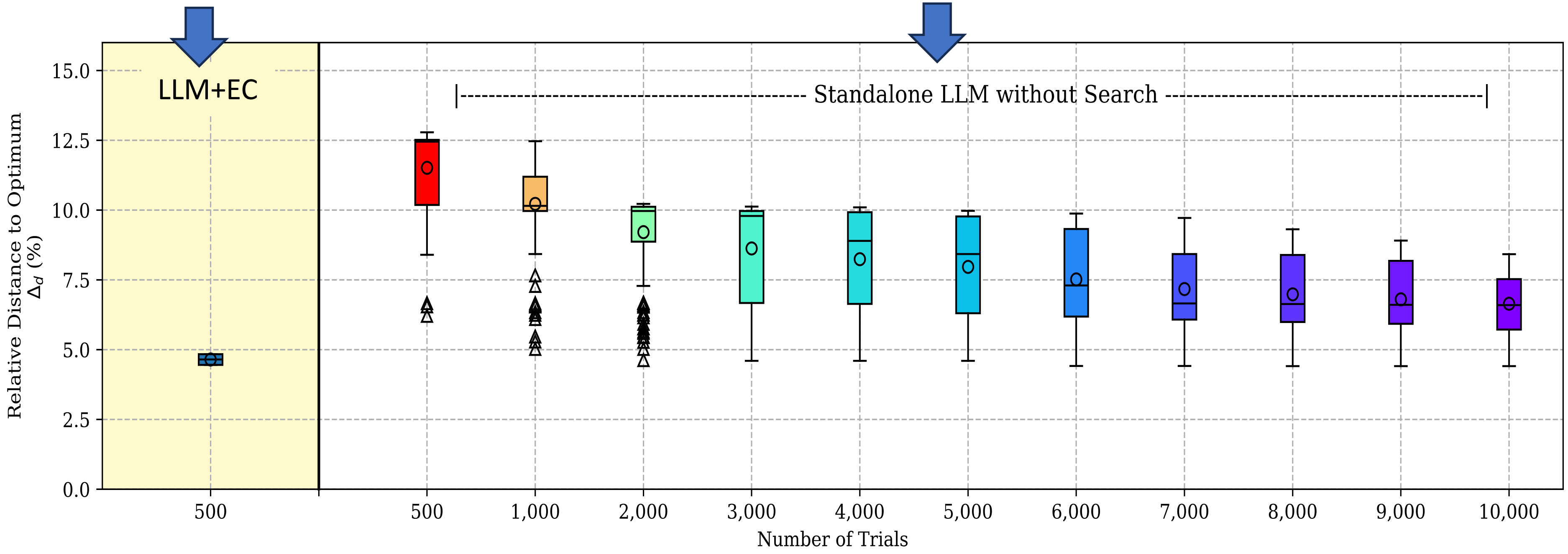
References

- Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. "**Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model**." *ICML 2024.* (**Oral Top 1.5%**)
- Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**Multi-objective Evolution of Heuristic Using Large Language Model."** AAAI 2025 (**Oral**)
- Yiming Yao, Fei Liu, Ji Cheng, and Qingfu Zhang. "**Evolve Cost-aware Acquisition Functions using Large Language Models.**" *PPSN 2024.* (**Best Paper Nomination**)
- Pengkun Wang, Zhe Zhao, HaiBin Wen, Fanfu Wang, Binwu Wang, Qingfu Zhang, and Yang Wang. "**LLM-AutoDA: Large Language Model-Driven Automatic Data Augmentation for Long-tailed Problems**.“ *NeurIPS 2024.*
- Rui Zhang, Fei Liu, Xi Lin, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. "**Understanding the Importance of Evolutionary Search in Automated Heuristic Design with Large Language Models**." *PPSN 2024.*
- Ping Guo, Fei Liu, Xi Lin, Qingchuan Zhao, and Qingfu Zhang. "**L-AutoDA: Large language Models for Automatically Evolving Decision-based Adversarial Attacks**." *GECCO 2024.*
- Fei Liu, Xi Lin, Zhenkun Wang, Shunyu Yao, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. "**Large language model for multi-objective evolutionary optimization**." *EMO 2025*.
- Yu Zhang, Kefeng Zheng, Fei Liu, Qingfu Zhang, and Zhenkun Wang. "**AutoTurb: Using Large Language Models for Automatic Algebraic Model Discovery of Turbulence Closure.**“ *Physics of Fluid 2025.*
- Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**A Systematic Survey on Large Language Models for Algorithm Design**." arXiv preprint 2024

Questions ?

Evolutionary Search is Important

LLM + Evolutionary Search **Standalone LLM**



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

OpenAI GPT-3.5 Turbo

- Merely allowing a LLM to attempt multiple times is not enough.
- Combining LLM with search significantly enhances overall performance.
- Evolutionary algorithm is a highly effective option.

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**