LLM4AD: Large Language Model for Algorithm Design

Fei Liu
Department of Computer Science
City University of Hong Kong

Dec, 2024

Outlines

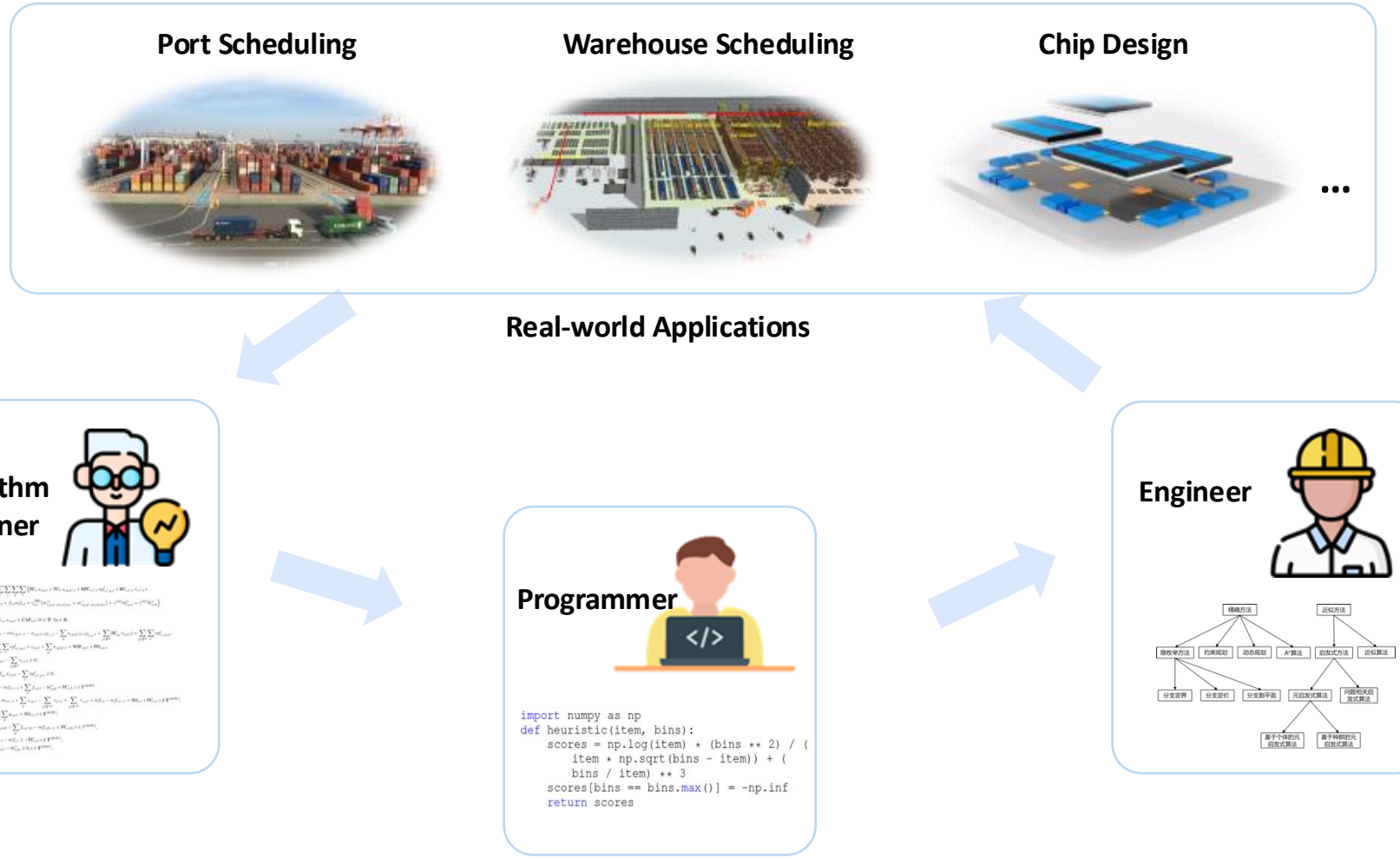
- **Background**
- **LLM4AD review**
- **Evolution of Heuristics (EoH)**
- **MEoH**
- **LLM4AD Platform**
- **Future Works**
- **Conclusion**

Outlines

- **Background**
- **LLM4AD review**
- **Evolution of Heuristics (EoH)**
- **MEoH**
- **LLM4AD Platform**
- **Future Works**
- **Conclusion**

Challenges of Algorithm Design

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



- Automatic Algorithm Design has been a longstanding pursuit

Existing Efforts for Automatic Algorithm Design (AAD)

- Hyperparameter Tuning[1]

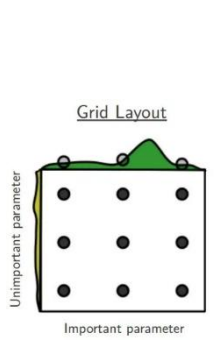
Algorithm 1 AdamW Optimizer

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

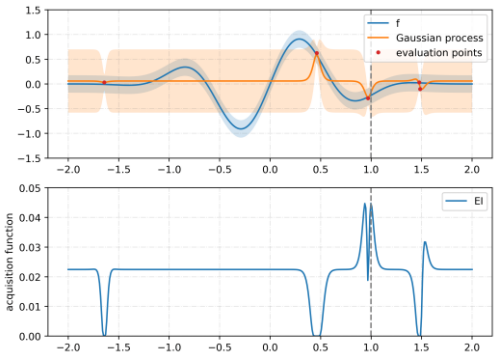
1. Baseline algorithm

- 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;

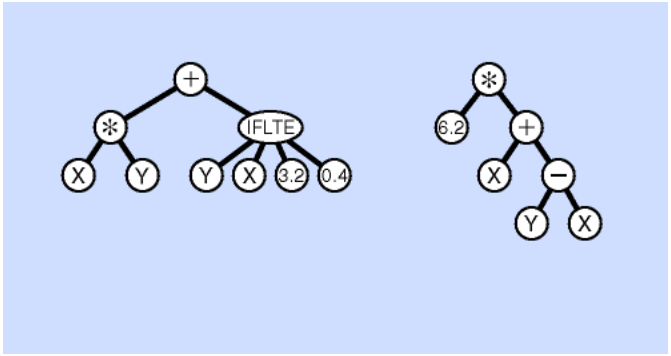
× Search a small vicinity around the baseline;

❑ 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.

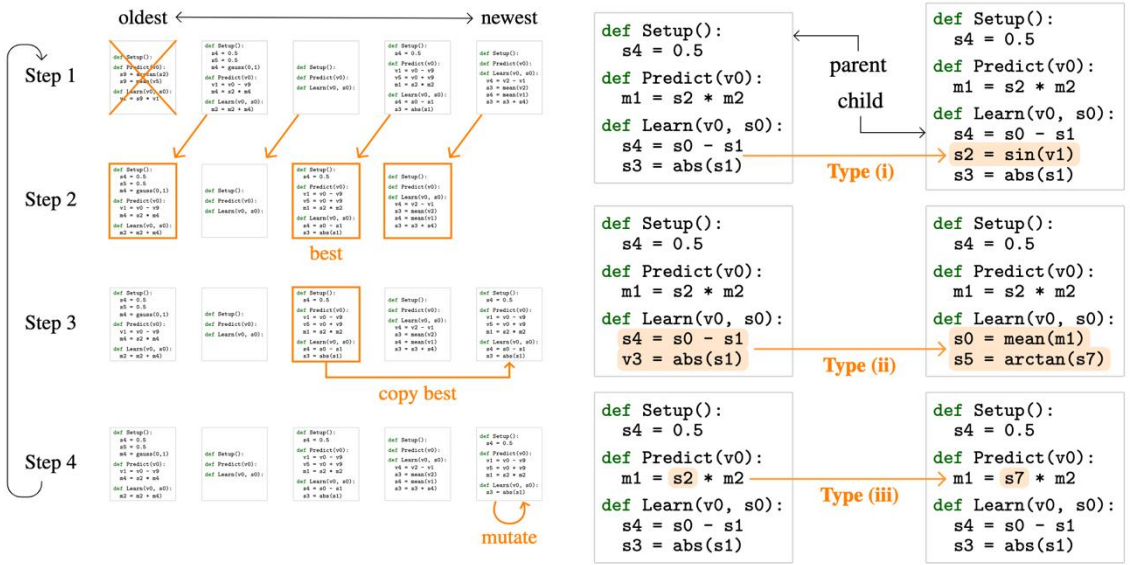
Existing Efforts on Automatic Algorithm Design (AAD)

- Genetic Programming[1]



Better algorithms are created by exchanging subparts between trees

An Application on AutoML[2]



✓ A flexible yet intuitive approach towards AAD;

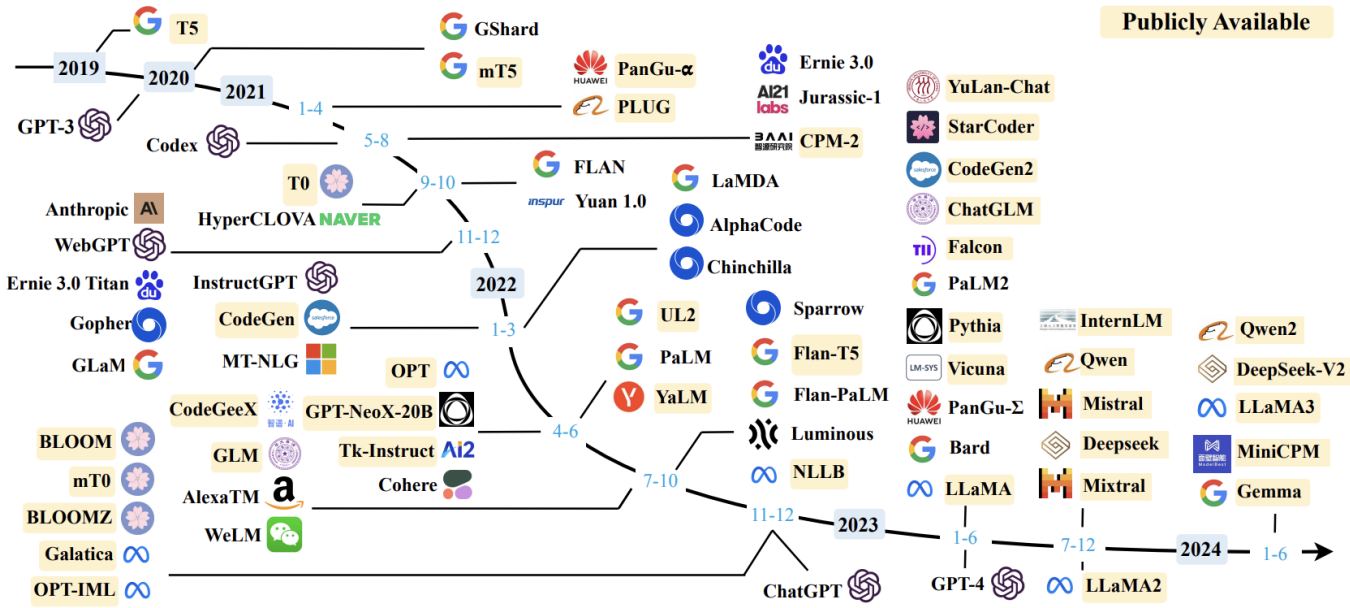
× Pre-define a set of primitives, and low search efficiency;

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

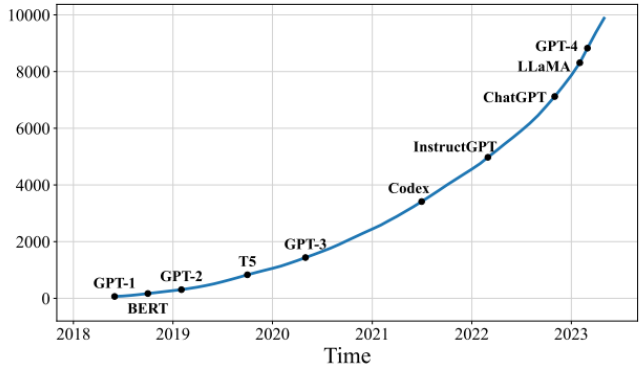
[1] Koza, John R. "Genetic programming as a means for programming computers by natural selection." Statistics and computing 4 (1994): 87-112.
[2] Real, Esteban, et al. "Automl-zero: Evolving machine learning algorithms from scratch." International conference on machine learning. PMLR, 2020.

Large Language Models (LLMs)

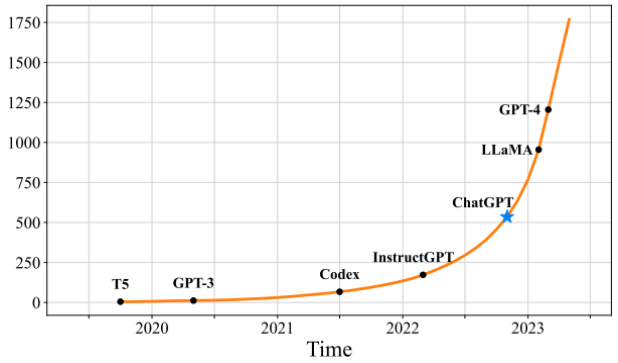
- Rapid Increasing of LLM and its research papers in the last three years



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



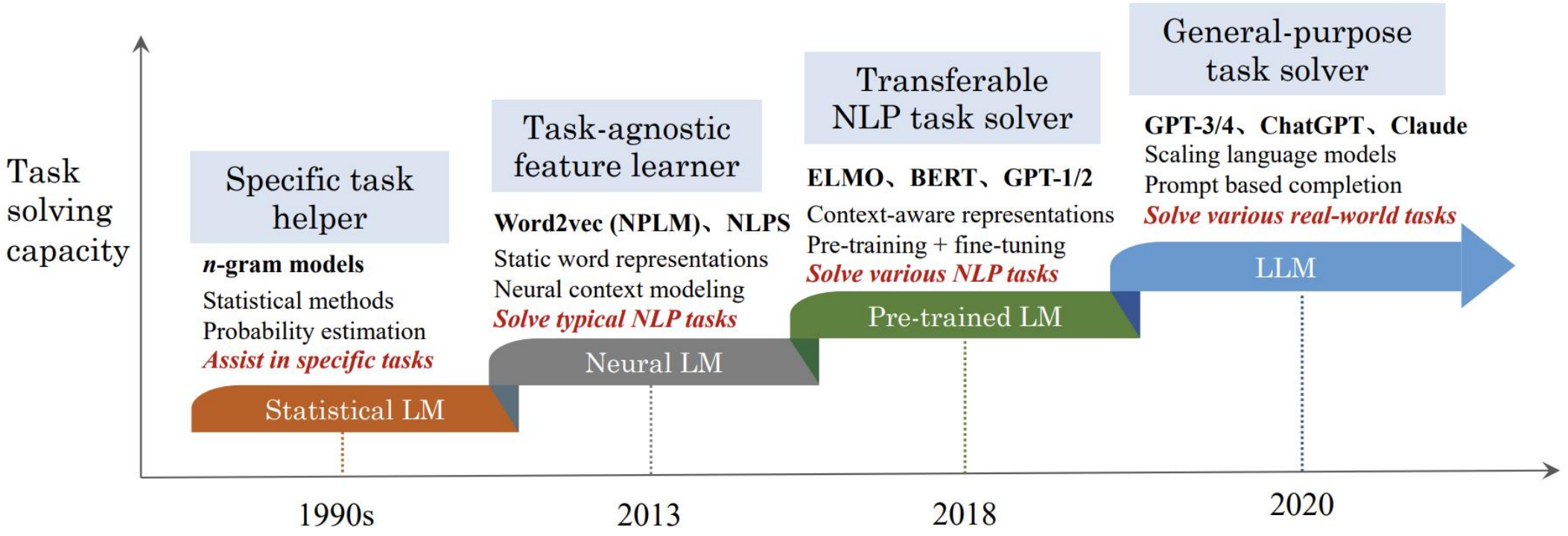
Query = "Language Model" (arXiv)



Query = "Large Language Model" (arXiv)

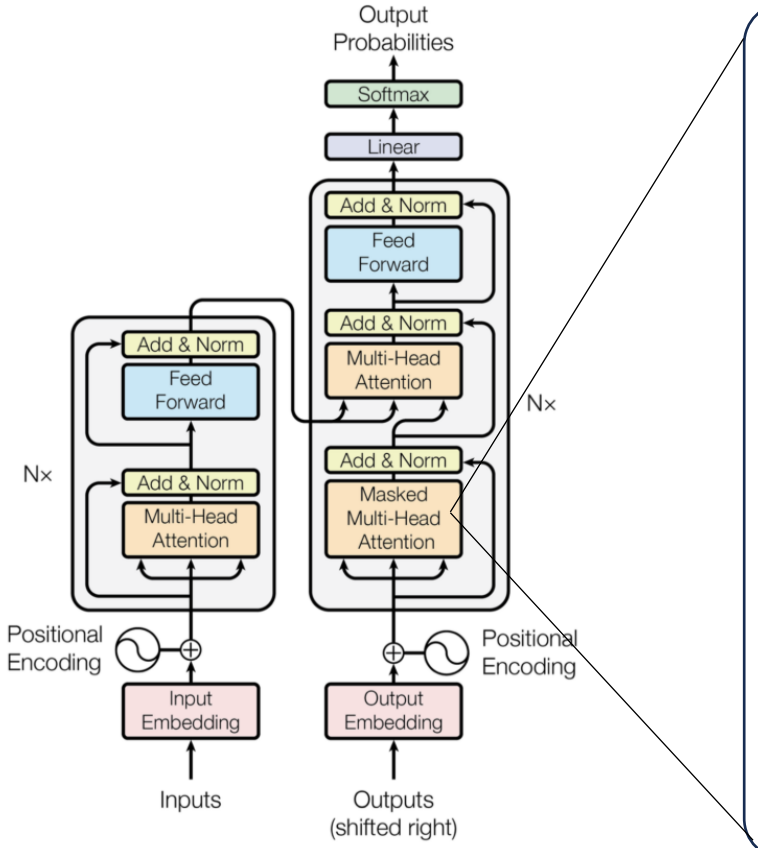
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

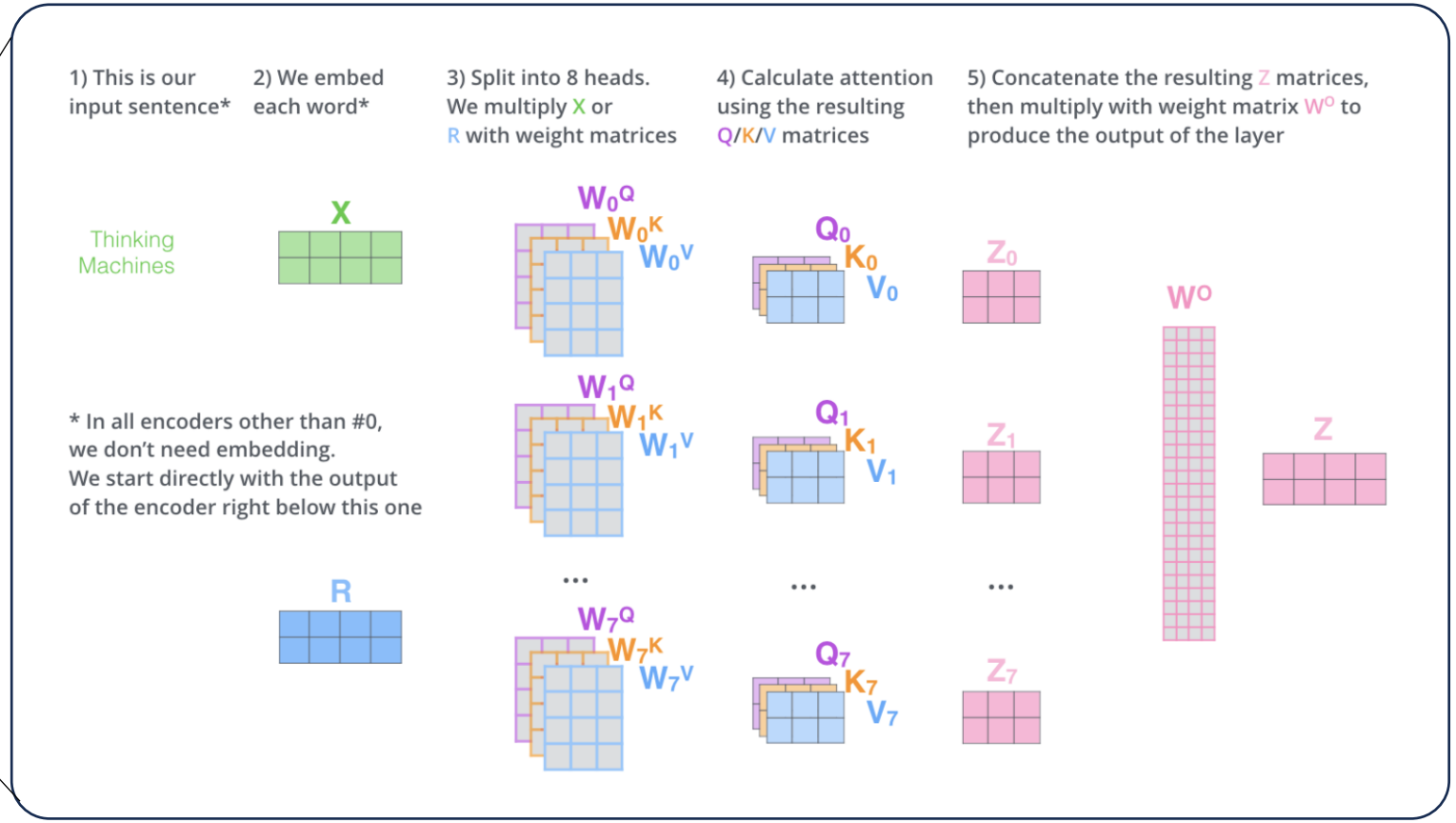
Large Language Models (LLMs)



- 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).

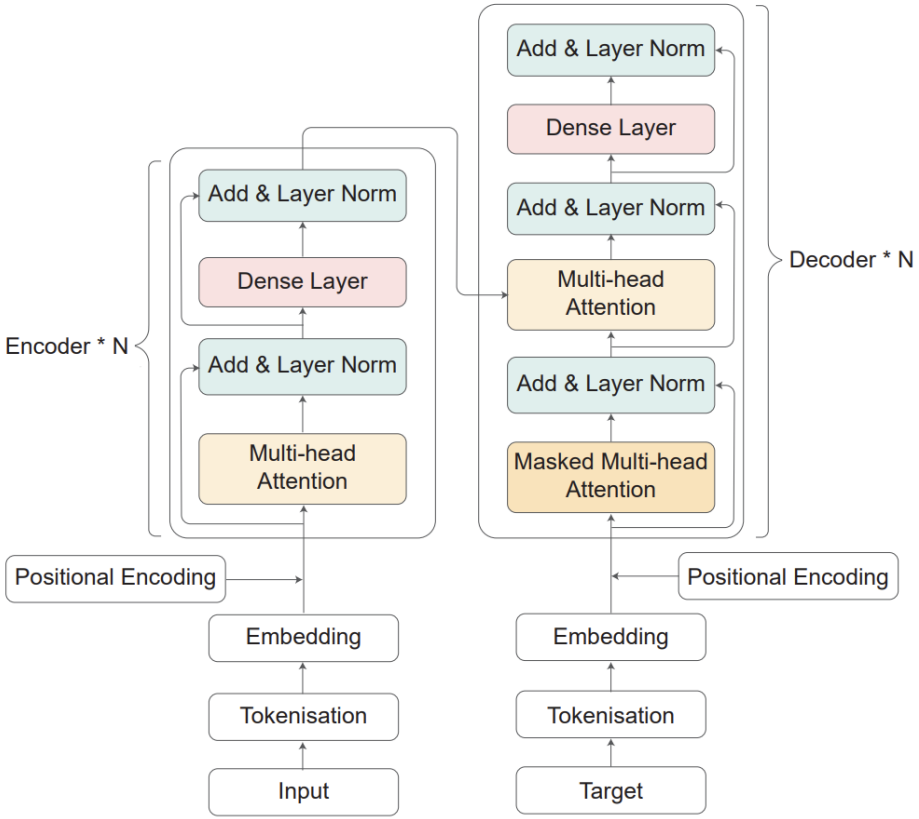
LLM Model Structures

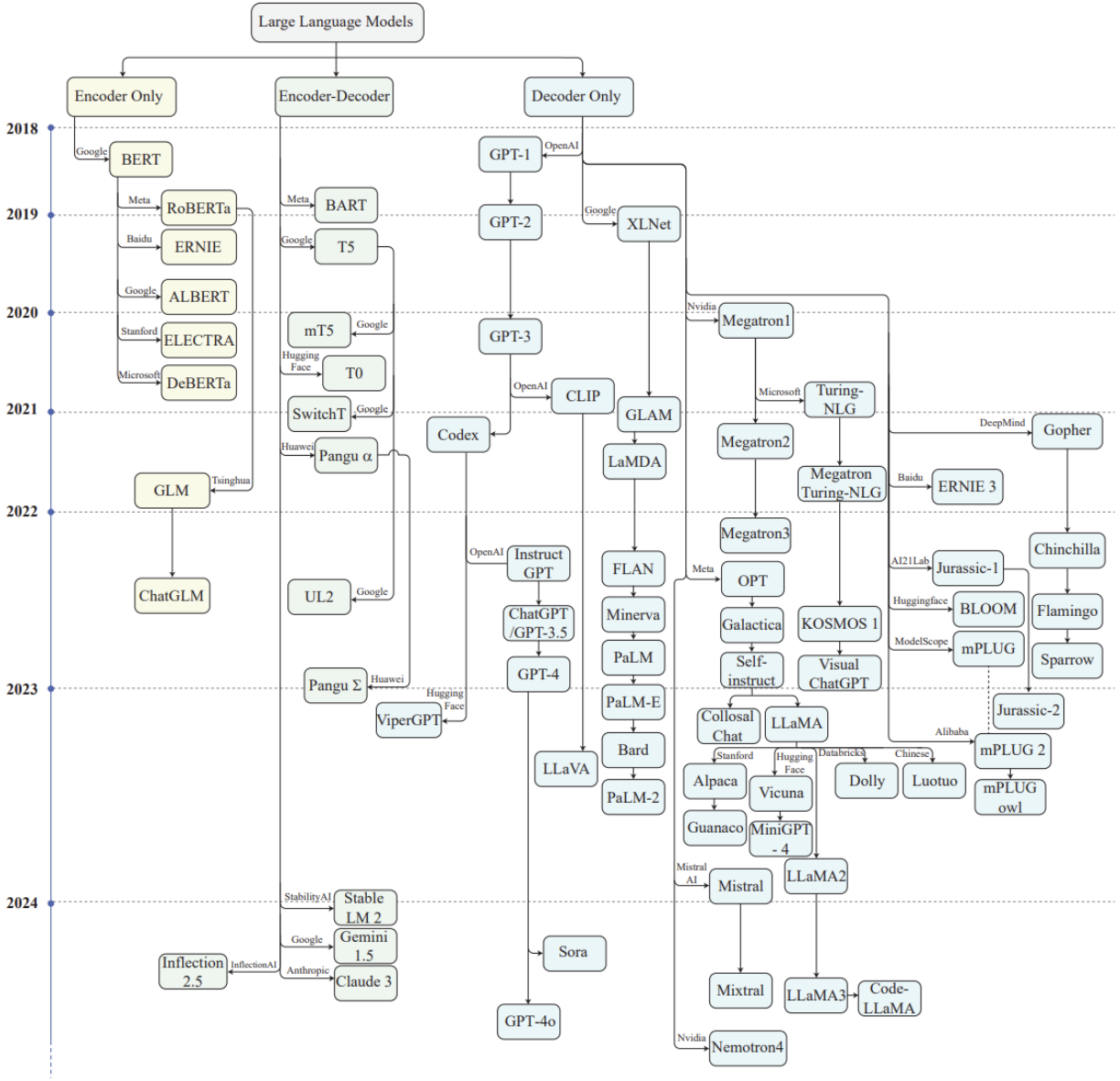




- Vaswani, A. "**Attention is all you need**." NeurIPS 2017.
- **Stanford CS25: Transformers United** https://web.stanford.edu/class/cs25/
- **The Illustrated Transformer** https://jalammar.github.io/illustrated-transformer/

LLM Model Structures

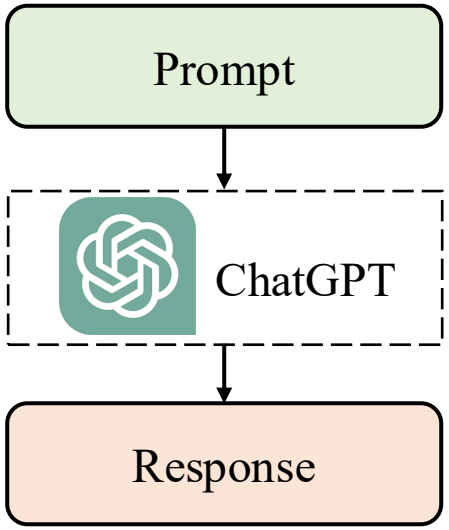


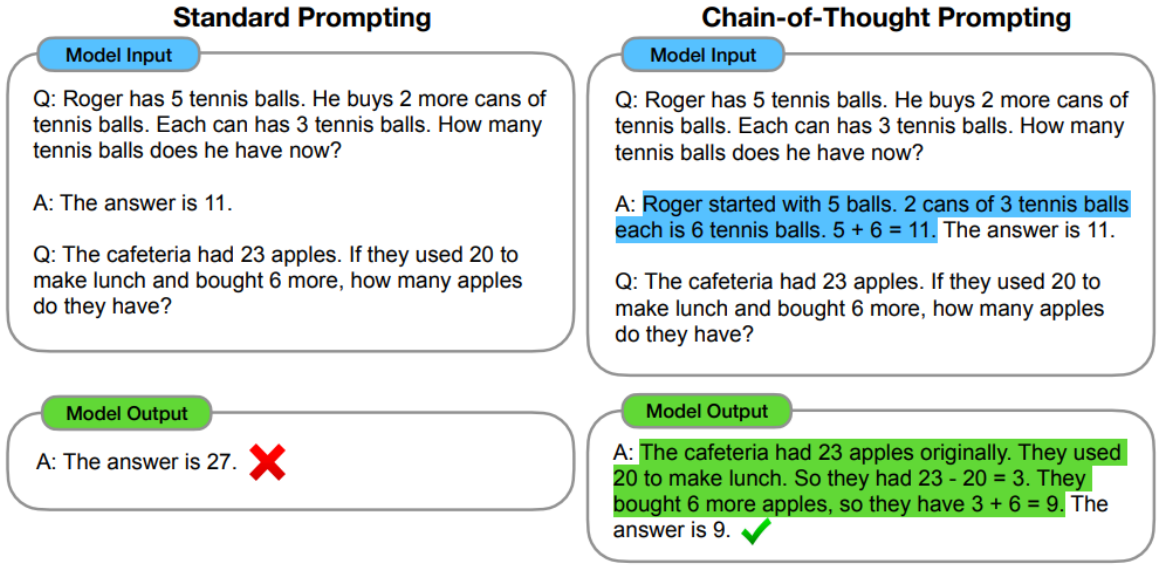


- Minghao Shao, Abdul Basit, Ramesh Karri, and Muhammad Shafique. "Survey of different Large Language Model Architectures: Trends, Benchmarks, and Challenges." IEEE Access (2024).

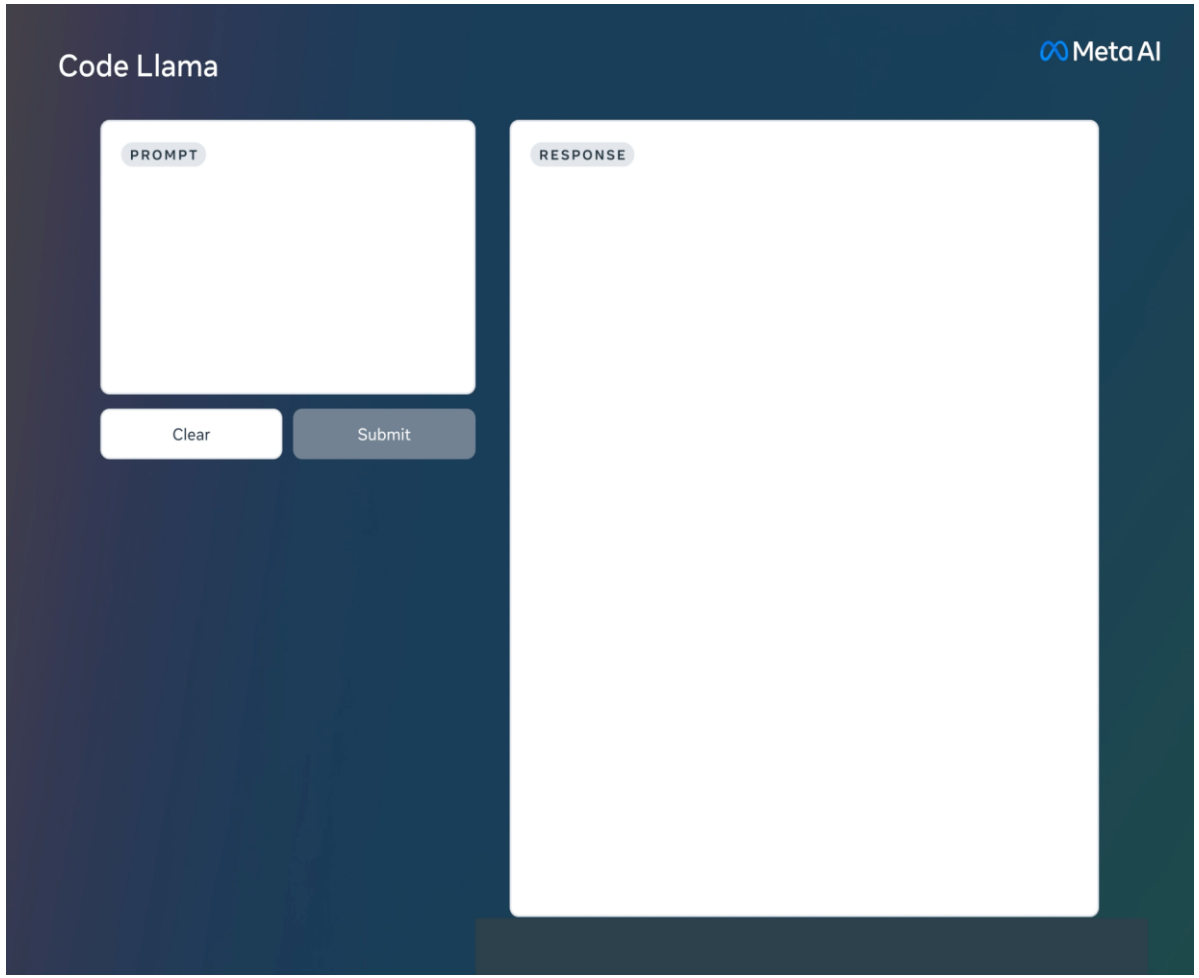
Prompt Engineering

- Pre-trained LLMs are powerful / Train&Finetune LLMs are **time and resource-consuming**
- The results are heavily rely on **input instruction** (i.e., prompt)
- How to do **prompt engineering** is a significant and popular research topic

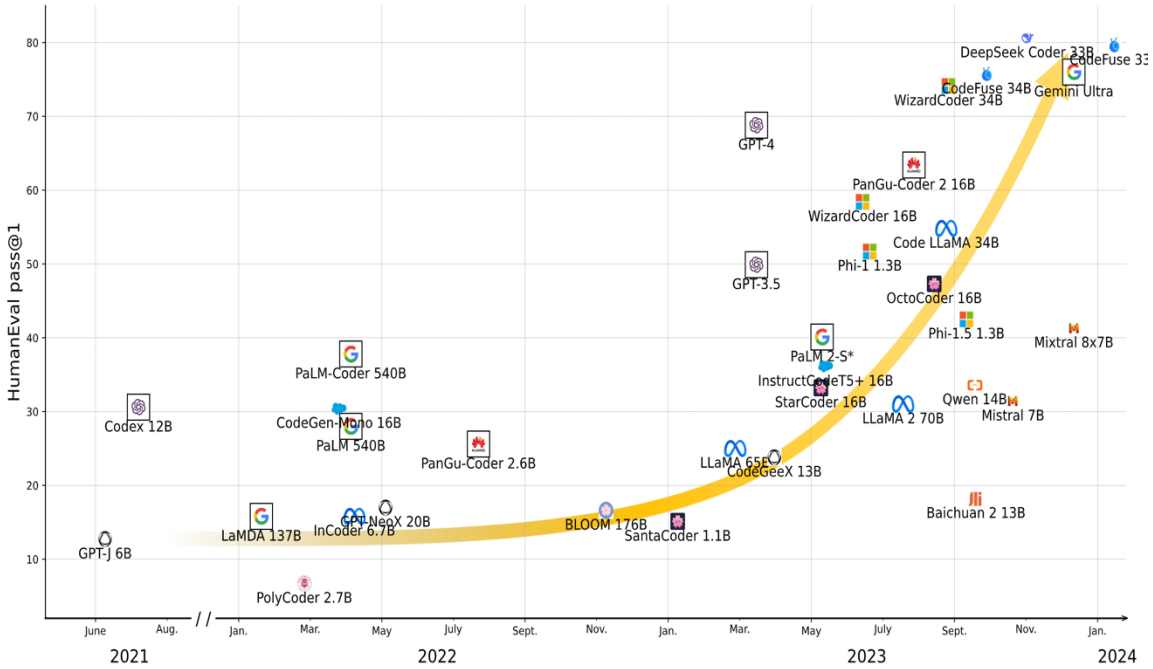




Prompt Engineering

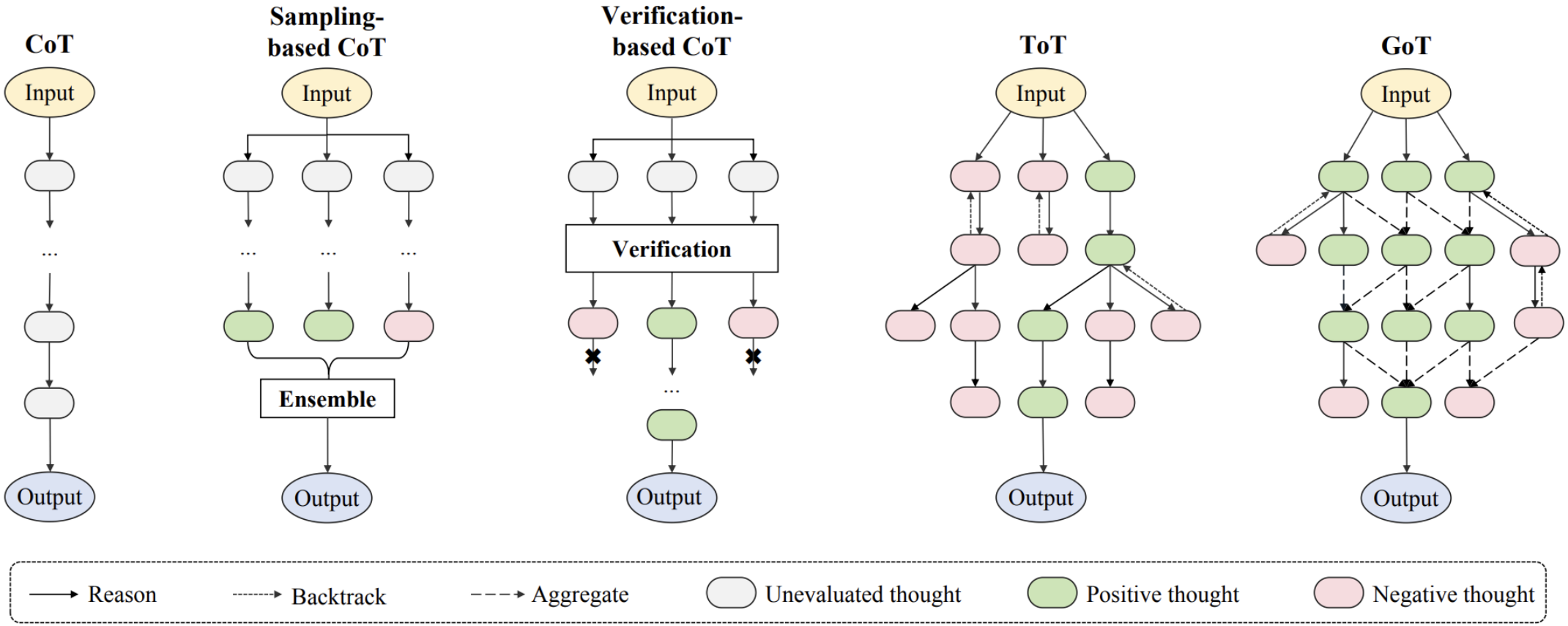


(Source: https://ai.meta.com/blog/code-llama-large-language-model-coding/)



(Source: arXiv 2311.07989)

Prompt Engineering



- 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).

Outlines

- ❑ **Background**
- ❑ **LLM4AD review**
- ❑ **Evolution of Heuristics (EoH)**
- ❑ **MEoH**
- ❑ **LLM4AD Platform**
- ❑ **Future Works**
- ❑ **Conclusion**

LLM4AD Scope

The scope of our LLM4AD survey as follows:

- The term ***Large Language Models*** refers to language models of sufficient scale to enable robust zero-shot performance across various tasks
  - Studies employing smaller models for algorithm design, such as those prevalent in conventional model-based algorithms and machine learning-assisted algorithms, are excluded.
  - Research utilizing other large models that lack language processing capabilities, such as purely vision-based models, are not considered. However, multi-modal LLMs that include language processing are within our scope.

- The term ***Algorithm*** in this context refers to a set of mathematical instructions or rules designed to solve a problem, particularly when executed by a computer. Traditional mathematical algorithms, most heuristic approaches, and certain agents or policies that can be interpreted as algorithms

LLM4AD 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

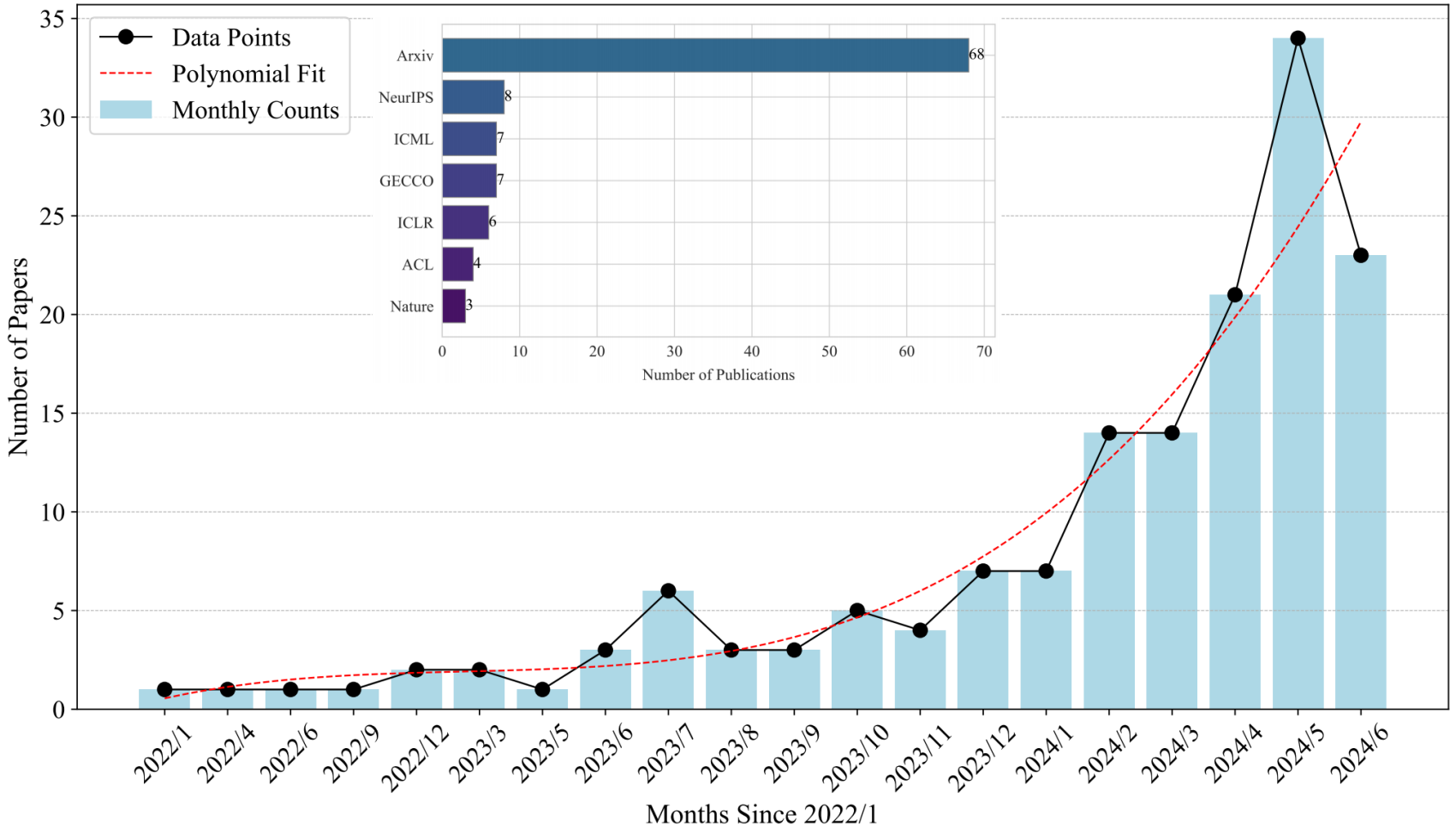
Content: Full paper
Exclusion Criteria: Research relevant to the topic
Remaining Results: 160 papers

Stage IV: Supplementation

Additional pertinent papers gathered from experience
Final Results: 180 papers

Number of publications

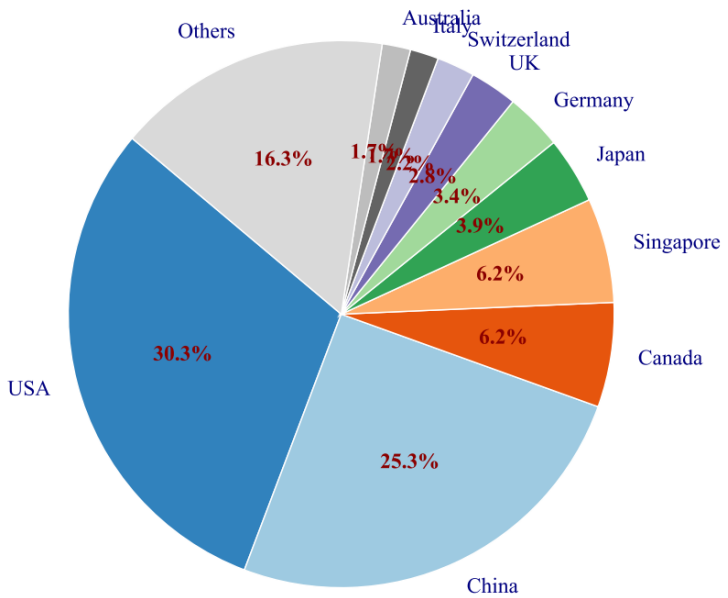
- Sapid increasing in recent months
- Most on Arxiv but accepted by top AI conferences, e.g., NeurIPS, ICML, ICLR, GECCO



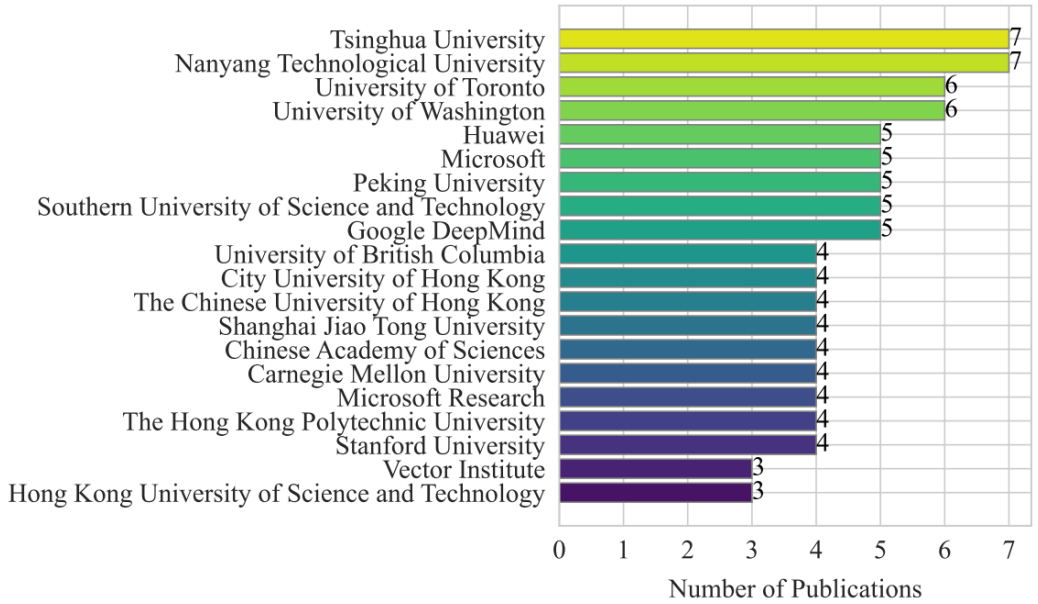
- Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**A Systematic Survey on Large Language Models for Algorithm Design**." *arXiv preprint arXiv:2410.14716* (2024).

Country and Institution

- The United States leads, closely followed by China, with these two countries alone accounting for 50% of the publications.
- Top universities: Tsinghua, NTU, and the University of Toronto, alongside major corporations like Huawei, Microsoft, and Google



(b) Country distribution

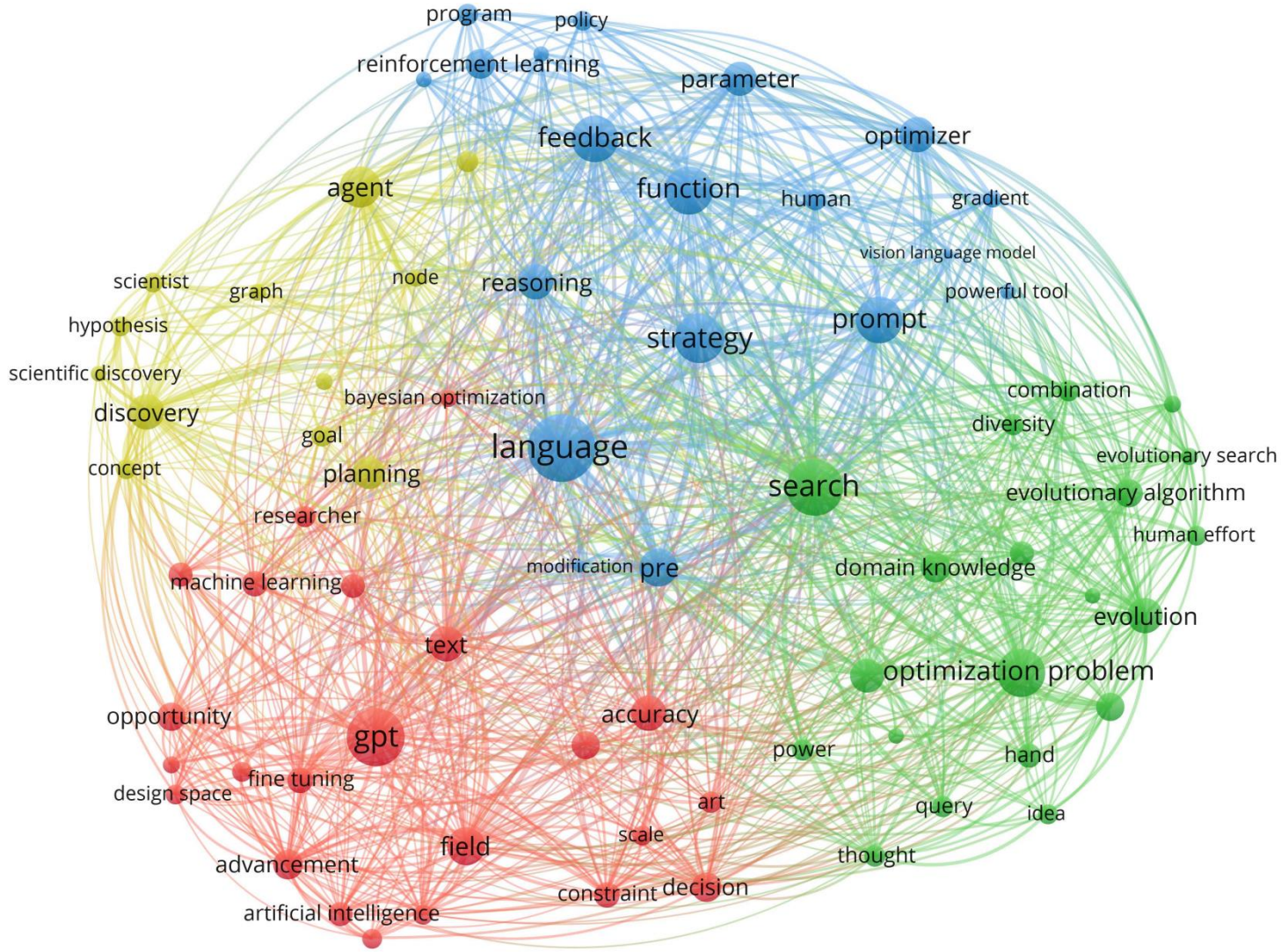


(c) Institutions

- 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).

Word Cloud

- **Blue Cluster** features the keyword "**Language**", includes other frequently used terms such as "strategy", "reasoning", "prompt", and "function", which are crucial for task resolution.
- **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".

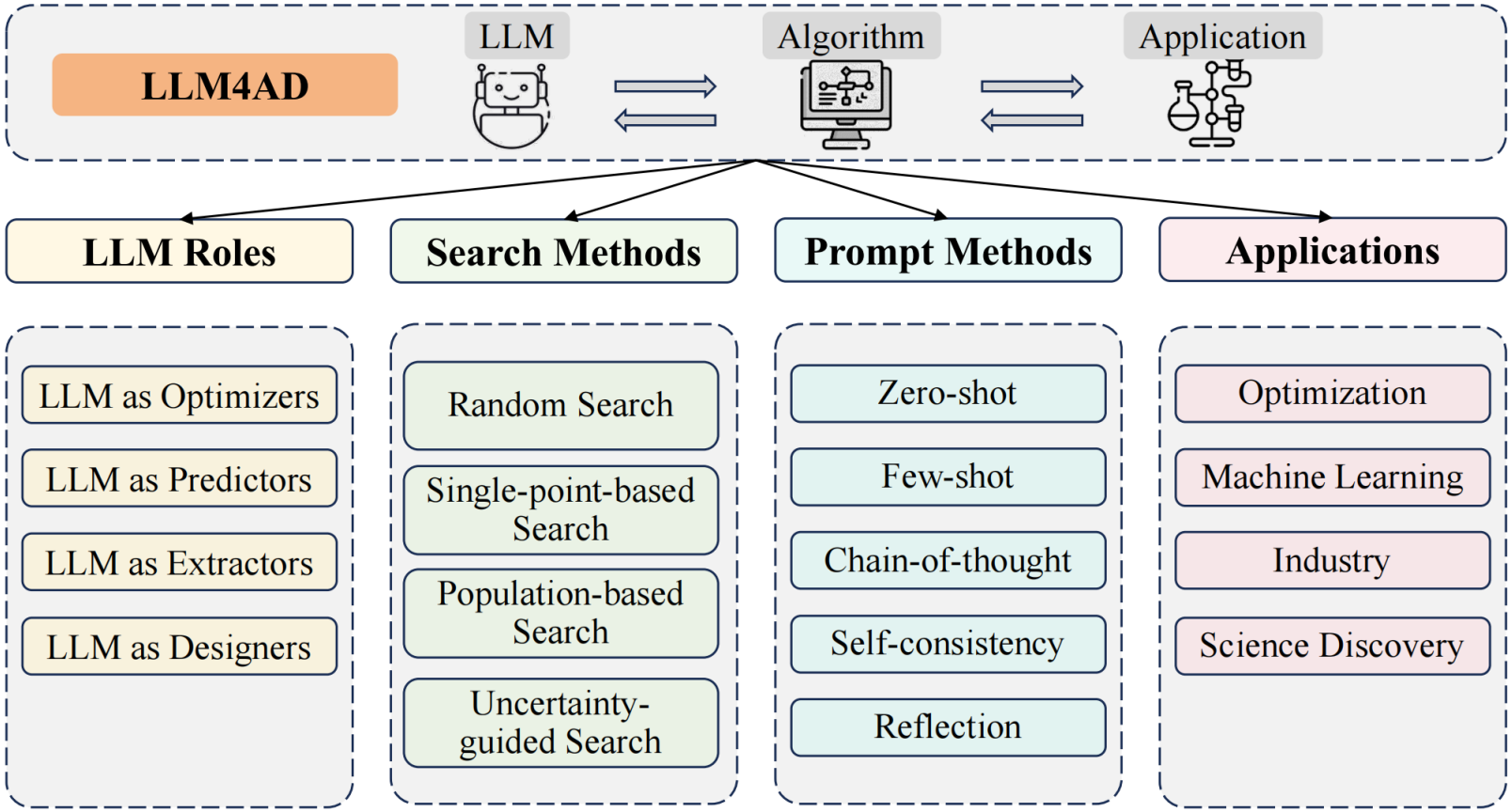


- Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**A Systematic Survey on Large Language Models for Algorithm Design**." *arXiv preprint arXiv:2410.14716* (2024).

Taxonomy

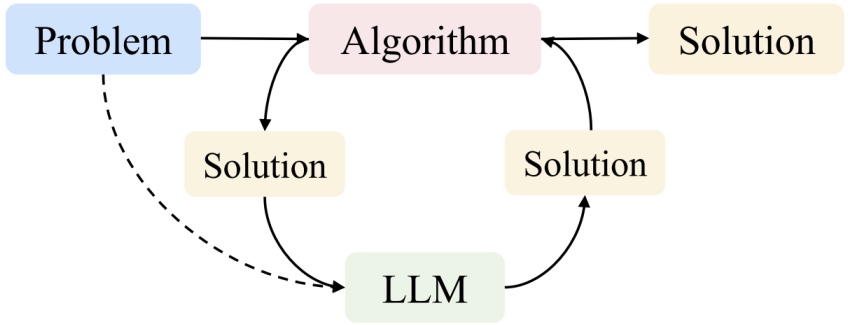
- Three key components: LLM, algorithm, application

1. **LLM roles**
2. **Search methods**
3. **Prompt methods**
4. **Applications**

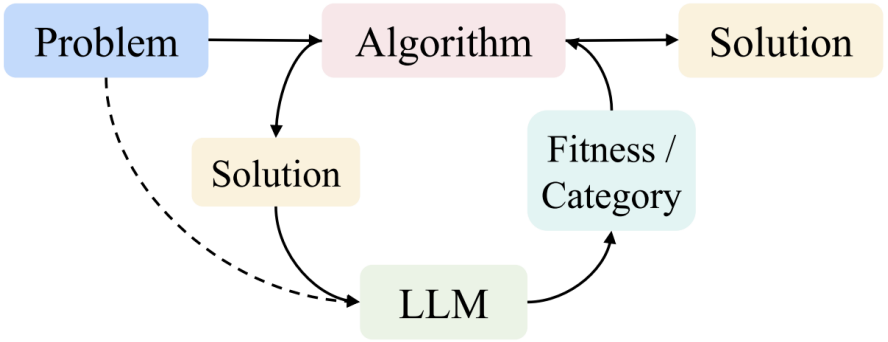


- Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**A Systematic Survey on Large Language Models for Algorithm Design**." *arXiv preprint arXiv:2410.14716* (2024).

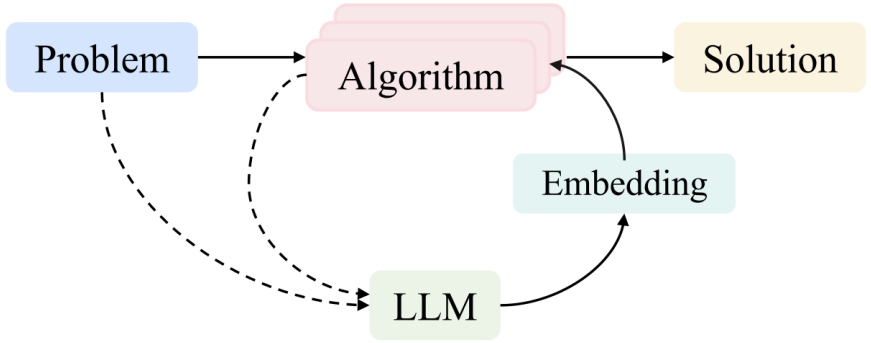
LLM Roles



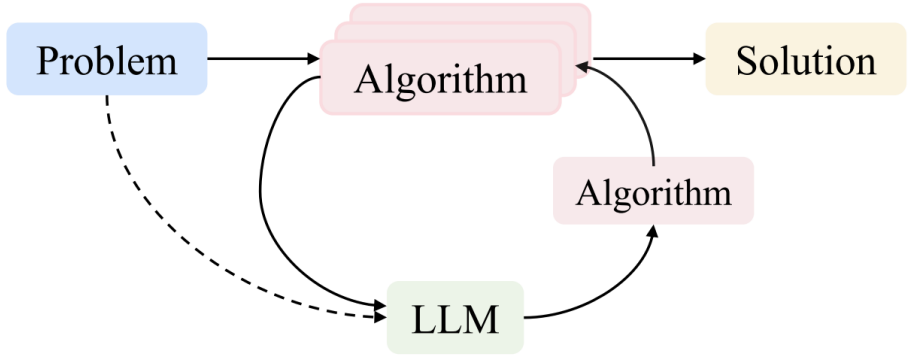
Large Language Models as Optimizers (LLMaO)



Large Language Models as Predictors (LLMaP)



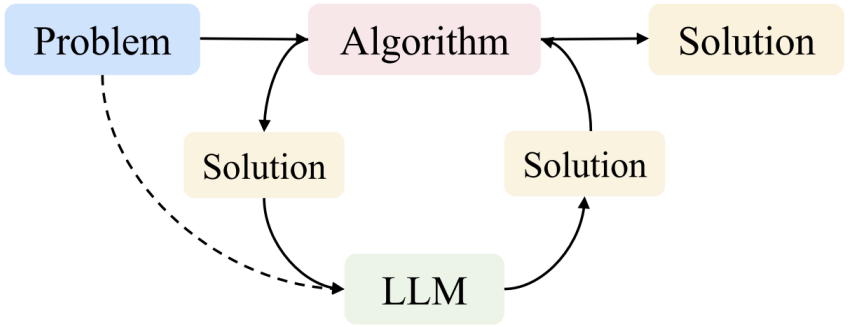
Large Language Models as Extractors (LLMaE)



Large Language Models as Designers (LLMaD)

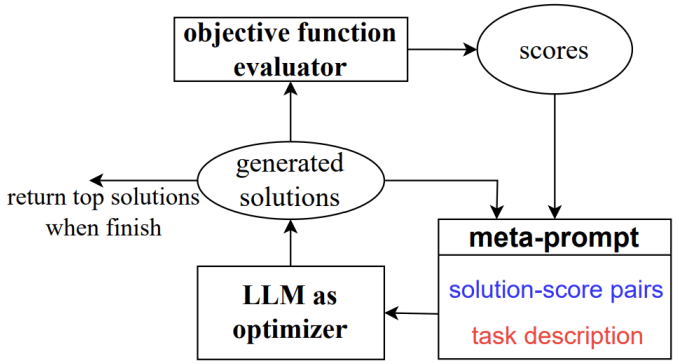
- Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. "**A Systematic Survey on Large Language Models for Algorithm Design**." *arXiv preprint arXiv:2410.14716* (2024).

LLM Roles: LLMaO

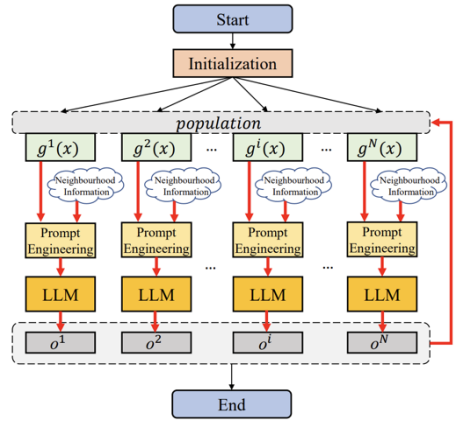


Large Language Models as Optimizers (LLMaO)

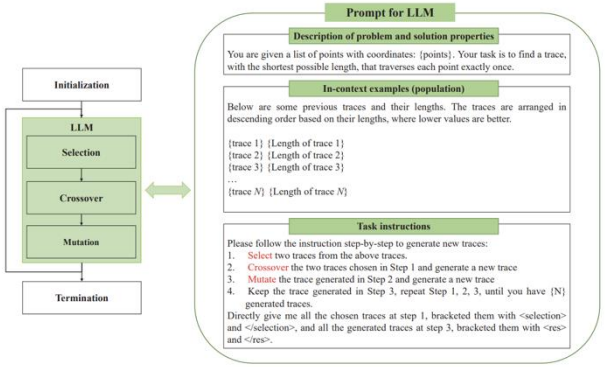
- LLMs are used as optimizer to suggest new solutions, human-designed algorithm
- LLM excels at learning and handling complicated patterns, integrating preference
- Lack generalization and interpretability



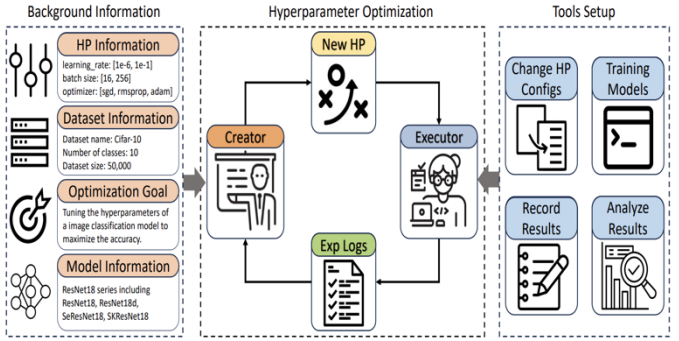
LLM as optimizers



MOEA



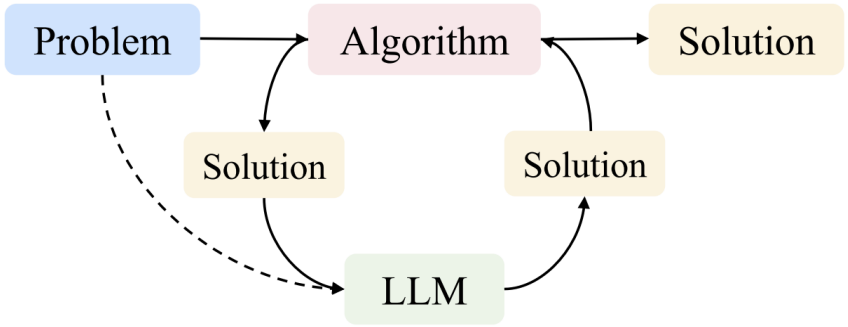
SOEA



Hyperparameter

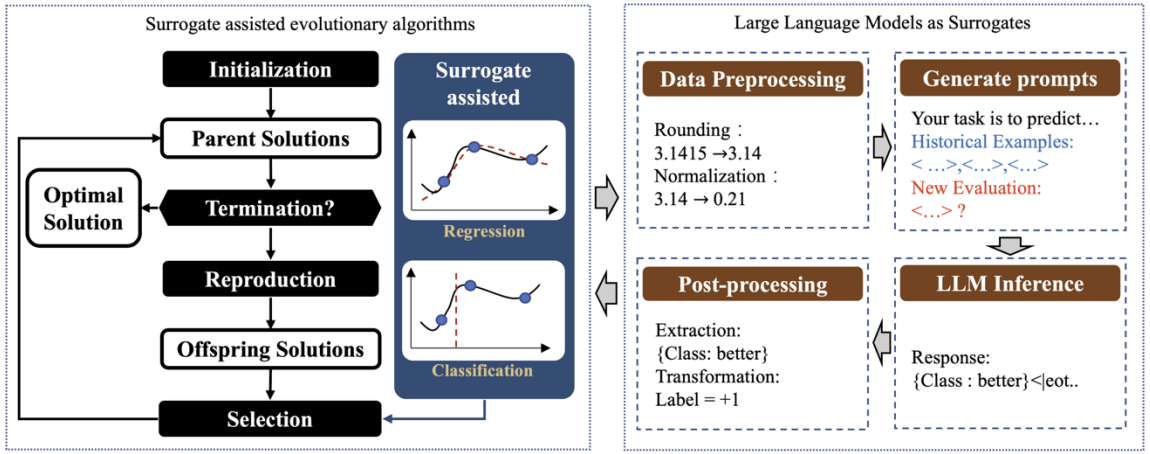
1. Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large Language Models as Optimizers. In *The Twelfth International Conference on Learning Representations*.
2. Fei Liu, Xi Lin, Zhenkun Wang, Shunyu Yao, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. 2023. Large Language Model for Multi-objective Evolutionary Optimization. arXiv preprint arXiv:2310.12541 (2023).
3. Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. 2024. Large language models as evolutionary optimizers. In 2024 IEEE Congress on Evolutionary Computation (CEC). IEEE, 1–8.
4. Siyi Liu, Chen Gao, and Yong Li. "Large Language Model Agent for Hyper-Parameter Optimization." arXiv preprint arXiv:2402.01881 (2024).

LLM Roles: LLMaP

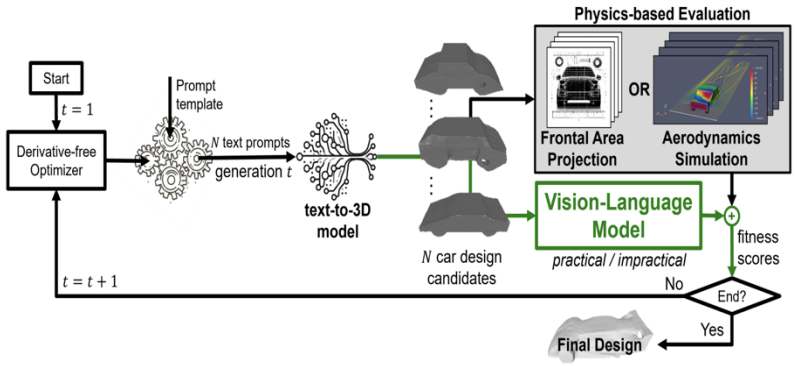


Large Language Models as Predictors (LLMaP)

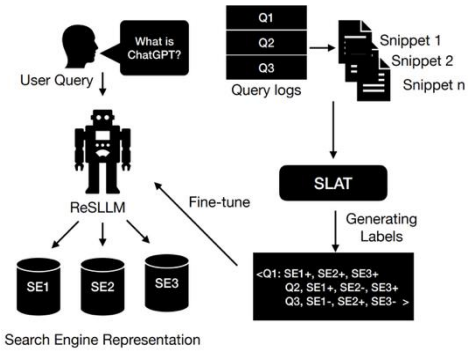
- LLMs are used as predictors predict a solution's outcomes or responses
- LLMs excel at processing and generating human-like response, reduce the computational load and time required in model training



LLMs as predictors (regression and classification) for EA



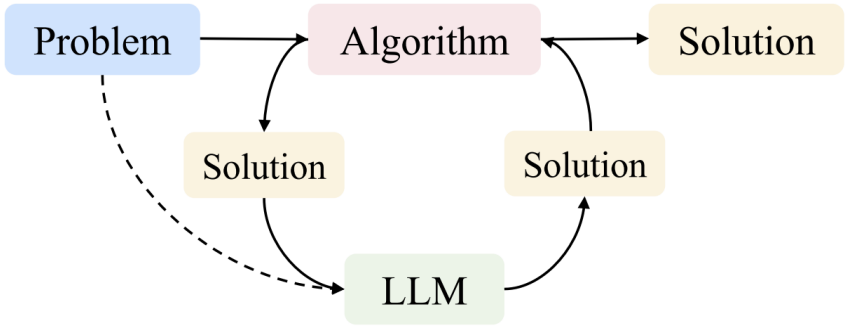
Car shape score



Resource selector

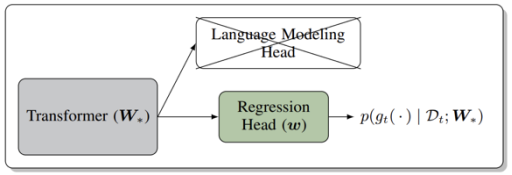
1. Hao Hao, Xiaoqun Zhang, and Aimin Zhou. 2024. Large Language Models as Surrogate Models in Evolutionary Algorithms: A Preliminary Study. arXiv preprint arXiv:2406.10675 (2024).
2. Melvin Wong, Thiago Rios, Stefan Menzel, and Yew Soon Ong. 2024. Generative AI-based Prompt Evolution Engineering Design Optimization With Vision-Language Model. arXiv preprint arXiv:2406.09143 (2024).
3. Shuai Wang, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. 2024. ReSLLM: Large Language Models are Strong Resource Selectors for Federated Search. arXiv preprint arXiv:2401.17645 (2024).

LLM Roles: LLMaE

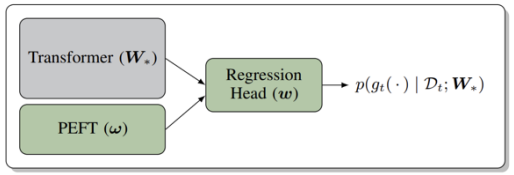


Large Language Models as Extractors (LLMaE)

- LLMs are employed to extract features or specific knowledge from target problem and/or algorithms to enhance problem-solving
- LLMs excel at extract high-level features and comprehends text and code

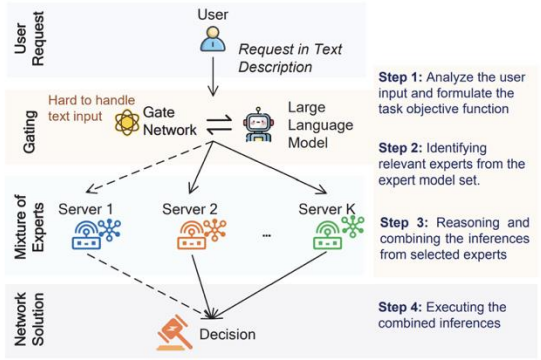


(a) Fixed-feature LLM surrogate

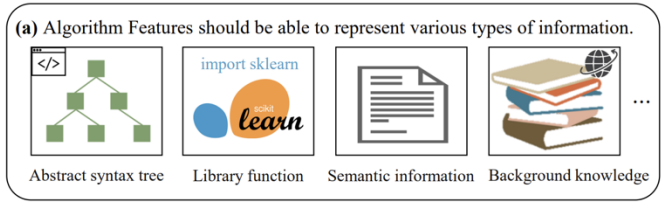


(b) Adaptive-feature LLM surrogate

Extractor for Bayesian Opt.



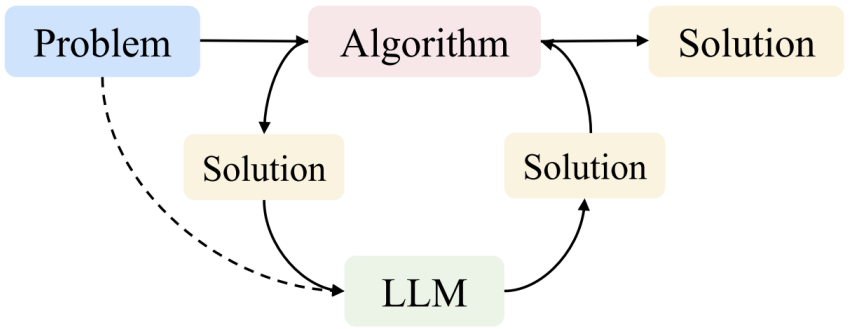
Intelligent Networks



Extractor for Algorithm Selection

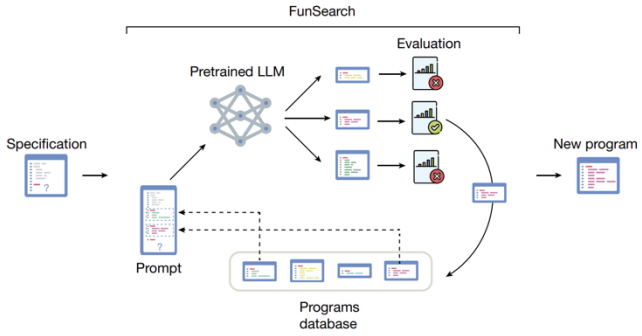
1. Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and Geoff Pleiss. 2024. A Sober Look at LLMs for Material Discovery: Are They Actually Good for Bayesian Optimization Over Molecules? ICML (2024).
2. Xingyu Wu, Yan Zhong, Jibin Wu, Bingbing Jiang, Kay Chen Tan, et al. 2024. Large language model-enhanced algorithm selection: towards comprehensive algorithm representation. IJCAI 2024.
3. Hongyang Du, Guangyuan Liu, Yijing Lin, Dusit Niyato, Jiawen Kang, Zehui Xiong, and Dong In Kim. 2024. Mixture of Experts for Network Optimization: A Large Language Model-enabled Approach. arXiv preprint arXiv:2402.09756 (2024).

LLM Roles: LLMaD

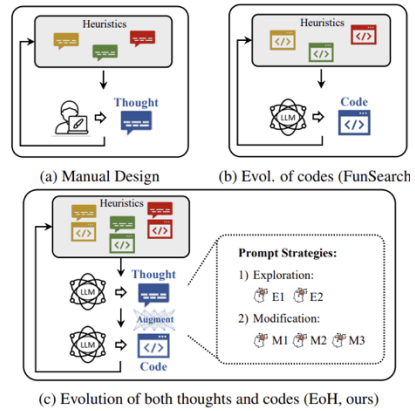


Large Language Models as Designers (LLMaD)

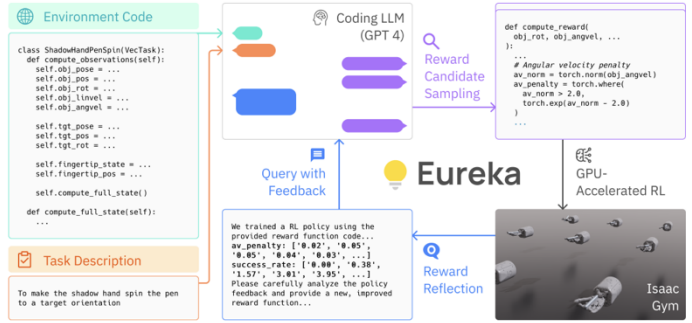
- LLMs are employed to directly create algorithms or specific components
- LLMs excel at code generation, text comprehension, and reasoning



FunSearch



EoH

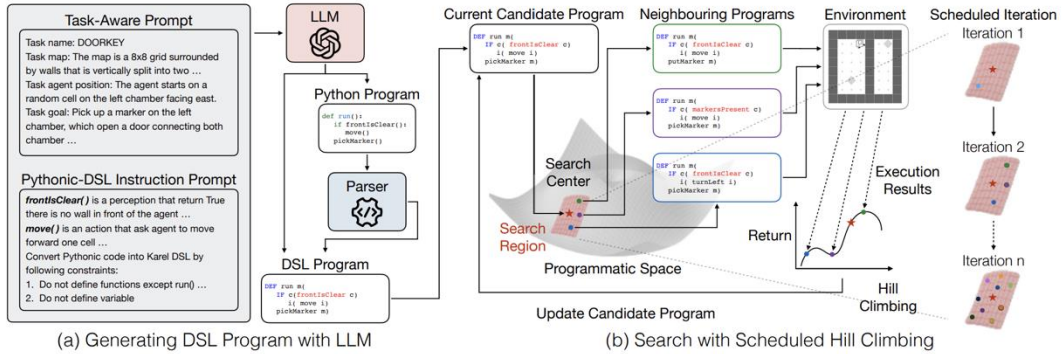


EUREKA

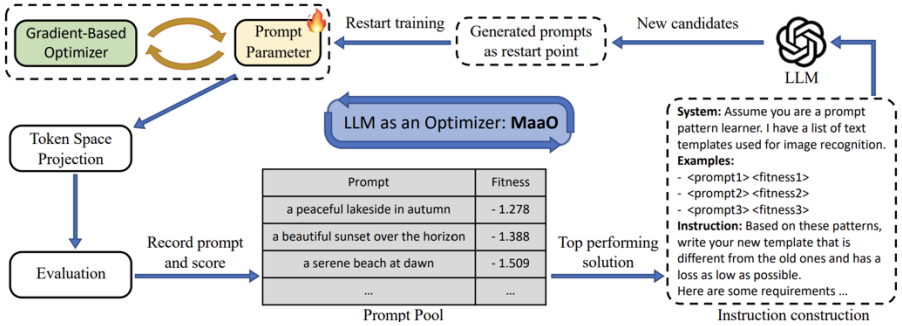
1. Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-Level Reward Design via Coding Large Language Models. ICLR 2024
2. Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. Nature2024
3. 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

Search Methods

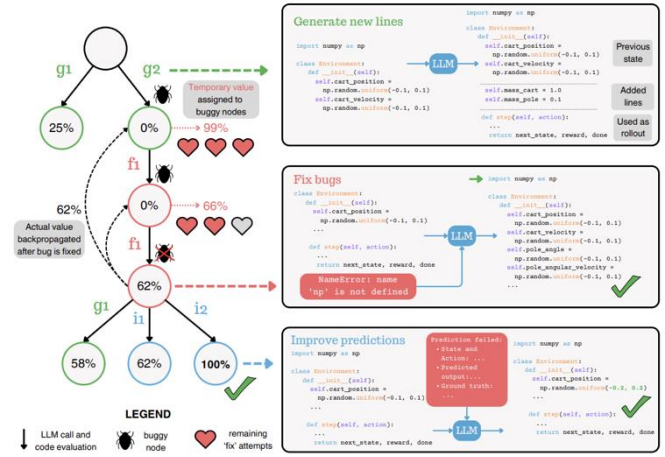
- **Sampling**
  - Sampling
  - Beam Search
  - MCTS
- **Single-point-based**
  - Hillclimb
  - Neighborhood Search
  - Gradient-based Search
  - Reinforcement Learning
- **Multi-point-based**
  - Single-objective
  - Multi-objective
- **Uncertainty-guided**
  - Bayesian Optimization



Hillclimb, Neighborhood Search



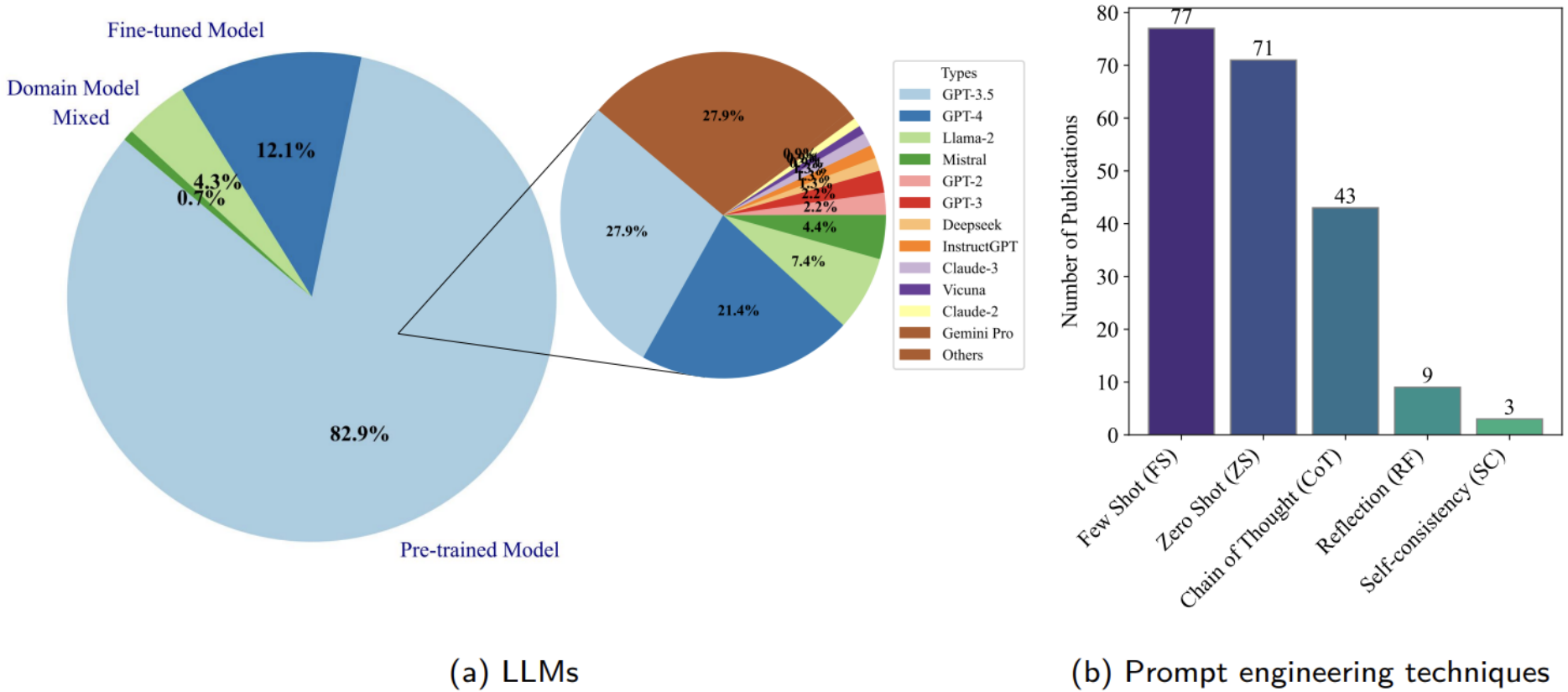
Gradient-based Search



MCTS

1. Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. 2024. Generating Code World Models with Large Language Models Guided by Monte Carlo Tree Search. *arXiv preprint arXiv:2405.15383* (2024)
2. Max Liu, Chan-Hung Yu, Wei-Hsu Lee, Cheng-Wei Hung, Yen-Chun Chen, and Shao-Hua Sun. 2024. Synthesizing Programmatic Reinforcement Learning Policies with Large Language Model Guided Search. arXiv preprint arXiv:2405.16450 (2024).
3. Zixian Guo, Ming Liu, Zhilong Ji, Jinfeng Bai, Yiwen Guo, and Wangmeng Zuo. 2024. Two Optimizers Are Better Than One: LLM Catalyst Empowers Gradient-Based Optimization for Prompt Tuning. arXiv:2405.19732 [cs.CV] https://arxiv.org/abs/2405.19732
4. Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. GPTSwarm: Language Agents as Optimizable Graphs. In Forty-first International Conference on Machine Learning.

Prompt Strategies



(a) LLMs

(b) Prompt engineering techniques

- 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).

Applications: 1) Optimization

Application	Method	Role of LLM	Prompt Strategy	Specific Problems or Tasks
Combinatorial Optimization	AEL [97]	LLMaD	FS, CoT	TSP
	ReEvo [192]	LLMaD	CoT, RF	TSP
	OPRO [183]	LLMaO	FS	TSP
	LMEA [101]	Mixed	FS	TSP
	MLLM [76]	Mixed	ZS, FS	CVRP
	FunSearch [139]	LLMaD	FS	Cap Set Problem, Online BPP
	EoH [98]	LLMaD	FS, OS, CoT	TSP, Online BPP, FSSP
	MH-LLM* [142]	LLMaD	ZS, FS	Social Networks Problem
Continuous Optimization	LLaMEA [161]	LLMaD	CoT	BBOB
	EvoLLM [84]	LLMaO	FS	BBOB, Neuroevolution
	OPRO [183]	LLMaO	FS	Linear Regression
	LEO [18]	LLMaO	FS	Numerical Benchmarks, Industrial Engineering Problems
	LAEA [64]	FS, CoT	LLMaP	Ellipsoid, Rosenbrock, Ackley, Griewank
	MOEA/D-LMO [96]	LLMaO	FS	ZDT, UF
	LLM-MOEA* [147]	LLMaE	ZS	Multi-objective Sustainable Infrastructure Planning Problem
	CMOEA-LLM [170]	LLMaO	FS	DAS-CMOP

Applications: 1) Optimization

Bayesian Optimization	HPO-LLM* [197]	LLMaO	FS	HPOBench
	LLAMBO [102]	Mixed	FS, ZS, CoT	Bayesmark, HPOBench
	BO-LIFT* [137]	LLMaD	FS	Catalyst Optimization
	EvolCAF [187]	LLMaD	FS, OS, CoT	Synthetic Functions, JAHS-Bench-201
	FunBO [2]	LLMaD	FS	Synthetic Functions, HPO
Prompt Optimization	APE [208]	LLMaO	FS	Instruction Induction, BBH
	OPRO [183]	LLMaO	FS	GSM8K, BBH, MultiArith, AQuA
	APO [134]	LLMaO	ZS, FS	NLP Benchmark Classification Tasks
	GPO [155]	LLMaO	FS	BBH, GSM8K, MMLU, WSC, WebNLG
	MaaO [60]	LLMaO	FS	NLU Tasks, Image Classification Tasks
	EvoPrompt [58]	LLMaO	CoT, FS	Language Understanding and Generation Tasks, BBH
Optimization Modeling	DOCP [171]	LLMaO	FS, CoT	Facility Location Problems
	OptiMUS [3]	Mixed	ZS, FS	NL4LP
	ORLMs [157]	LLMaD	ZS	NL4Opt, MAMO, IndustryOR
	LM4OPT [4]	LLMaD	FS, ZS	NL4Opt
Other Applications	AS-LLM [178]	LLMaE	ZS	Algorithm Selection
	LLaMoCo [109]	LLMaD	FS	Optimization Code Generation
	StrategyLLM [52]	Mixed	ZS, FS, CoT	Math & Commonsense & Algorithmic & Symbolic Reasoning

Applications: 2) Machine Learning

Application	Method	Role of LLM	Prompt Strategy	Specific Problems or Tasks
Task Planning	Zero-Planner [73]	LLMaP	ZS, FS	VirtualHome Tasks
	SayCan [6]	LLMaP	FS	Real-world Kitchen Tasks with Robots
	LLM-GM [74]	LLMaP	FS	Real-world Kitchen Tasks with Robots
	Text2Motion [94]	LLMaP	FS	Long Sequence Tasks for Robots
	ProgPrompt [148]	LLMaD	FS, COT	VirtualHome Tasks
	SayCanPay [65]	LLMaP	FS	VirtualHome Tasks, Ravens Tasks, BabyAI Tasks
Reinforcement Learning	LFG [144]	LLMaP	FS, CoT	ObjectNav Tasks, Real-world Tasks
	SLINVIT [201]	LLMaP	ZS, FS	ALFWorld Tasks, InterCode Tasks, BlocksWorld Tasks
	MEDIC [14]	LLMaP	ZS	BabyAI Tasks
	Eureka [108]	LLMaD	FS, CoT	IsaacGym Tasks, Bidexterous Manipulation Tasks
	EROM [123]	LLMaD	ZS, CoT	IsaacGym Tasks
	LLM-MOE [39]	LLMaP	FS, CoT	Intelligent Networks
Neural Architecture Search	EvoPrompting [25]	LLMaD	FS, ZS, CoT	MNIST Dataset, CLRS Algorithmic Reasoning
	HS-NAS [79]	LLMaP	FS, ZS, CoT	Machine Translation Tasks
	LLMatic [124] LAPT [207]	LLMaD LLMaD	FS, ZS, CoT ZS, FS	CIFAR-10 Dataset, NAS-bench-201 Benchmarks NAS201, Trans101, DARTs
	LLM-GE [121]	LLMaD	FS, ZS, CoT	CIFAR-10 Dataset

Applications: 2) Machine Learning

Graph Learning	LLM-Critical [112]	LLMaD	FS, ZS, CoT	Critical Node Identification
	LLM-GNN [29]	LLMaP	FS, CoT	Label-free Node Classification
	ReStruct [27]	LLMaE	FS, ZS	Meta-structure Discovery
	AutoAlign [200]	LLMaE	FS, ZS	Entity Type Inference
	KSL [49]	LLMaP	FS	Knowledge Search
Dataset Labeling	Inter-Classier [30]	LLMaO	FS, ZS	iNaturalist Datasets, KikiBouba datasets
	DataSculpt [56]	LLMaP	FS	Label Function Design
Other Applications	LLM2FEA [174]	LLMaP	FS, CoT	Objective-oriented Generation
	tnGPS [195]	LLMaD	FS, OS, CoT	Tensor Network Structure Search
	L-AutoDA [57]	LLMaD	FS, OS, CoT	Adversarial Attack

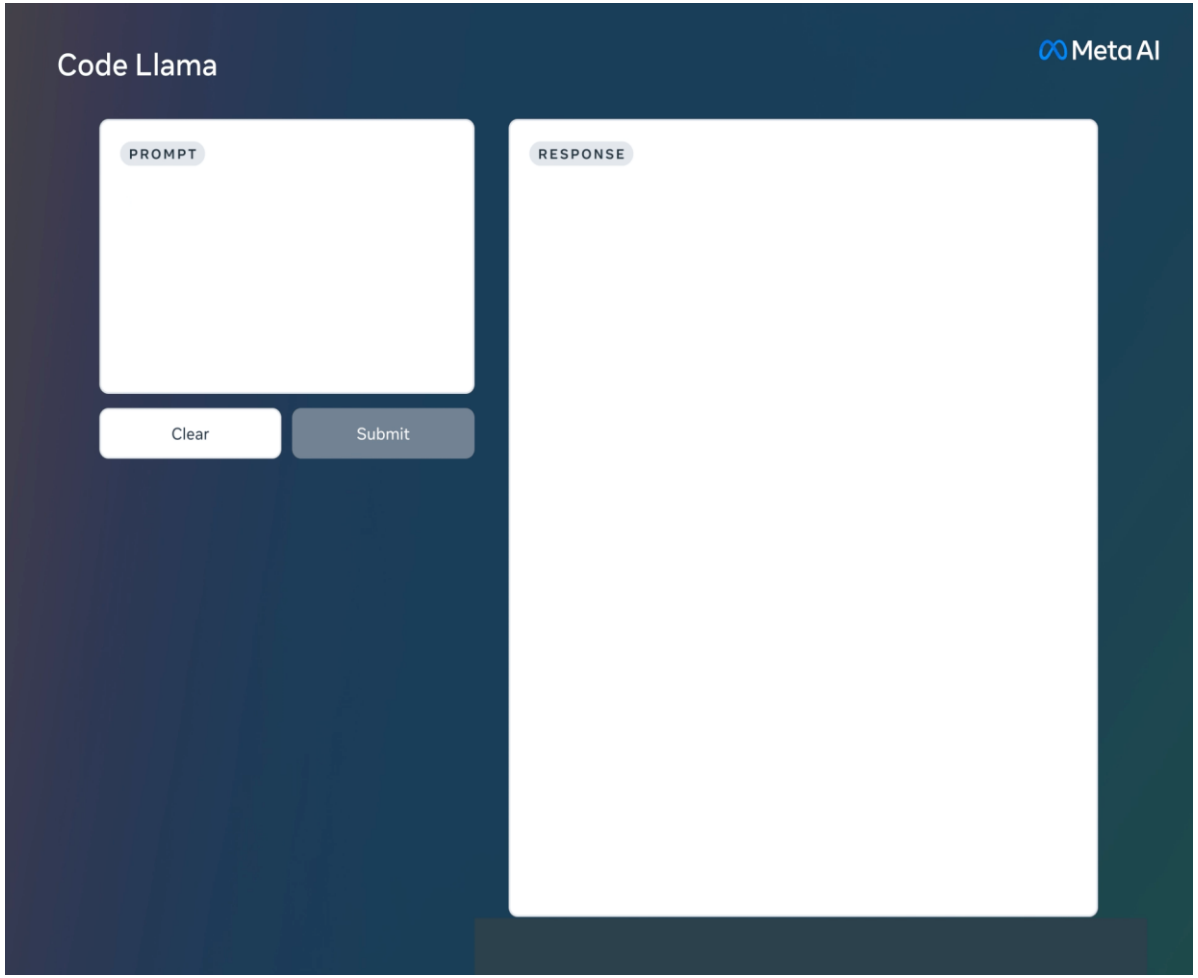
Applications: 3) Science Discovery

Application	Method	Role of LLM	Specific Problems or Tasks
General Scientific Equation Discovery	Bilevel [106]	LLMaD	Physical Scientific Discovery
	LLM-SR [146]	LLMaD	Scientific Equation Discovery
	LLM4ED [41]	LLMaD	Equation Discovery
Chemical	ChatChemTS [77]	LLMaD	Molecule Design
	Debjyoti Bhattacharya, et al. [15]	LLMaO	Molecule Design
	Agustinus Kristiadi, et al. [83]	LLMaE	Molecule Design
	BoChemian [138]	LLMaE	Chemical Reaction
	Multi-modal MoLFormer [149]	LLMaP	Battery Electrolytes Formulation Design
	CataLM [164]	LLMaP	Catalyst Design
	Gavin Ye [189]	LLMaD	Drug Design
	DrugAssist [190]	LLMaO	Drug Design
Biology	MLDE [158]	LLMaP	Protein Design
	X-LoRA [21]	LLMaP	Protein Design
	CodonBERT [92]	LLMaP	mRNA Design and Optimization
	Revisiting-PLMs [70]	LLMaP	Protein Function Prediction
Mechanics	FLUID-LLM [210]	LLMaP	Computational Fluid Dynamics
	MechAgents [125]	LLMaD	Mechanics Design
	MeLM [22]	LLMaP, LLMaD	Carbon Nanotubes and Proteins Design
	Mengge Du, et al. [40]	LLMaD	Nonlinear Dynamics Equation Discovery

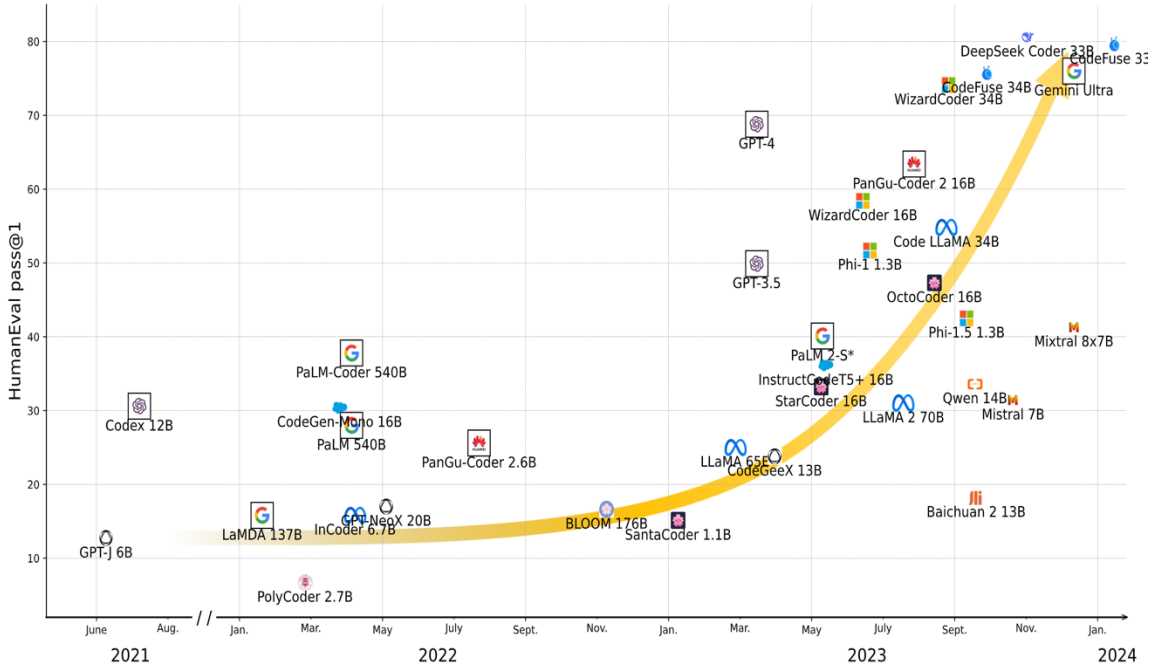
Outlines

- ❑ **Background**
- ❑ **LLM4AD review**
- ❑ **Evolution of Heuristics (EoH)**
- ❑ **MEoH**
- ❑ **LLM4AD Platform**
- ❑ **Future Works**
- ❑ **Conclusion**

LLM for Algorithm Design?



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



(Source: arXiv 2311.07989)

Standalon LLM is Insufficient for AAD

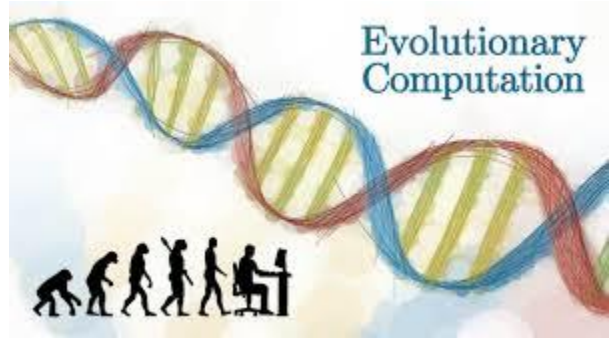
- BBH (Big Bench Hard): multistep arithmetic, algorithmic reasoning
- MuSR (Multistep Soft Reasoning)

T	Model	Average	IFEval	BBH	MATH Lvl 5	GPQA	MUSR	MMLU-I
◆	dnhkng/RYS-XLarge	44.75	79.96	58.77	38.97	17.9	23.72	49.2
💬	MaziyarPanahi/calme-2.1-qwen2-72b	43.61	81.63	57.33	36.03	17.45	20.15	49.05
💬	Qwen/Qwen2-72B-Instruct	42.49	79.89	57.48	35.12	16.33	17.17	48.92
💬	alpindale/magnum-72b-v1	42.17	76.06	57.65	35.27	18.79	15.62	49.64
💬	meta-llama/Meta-Llama-3.1-70B-Instruct	41.74	86.69	55.93	28.02	14.21	17.69	47.88
💬	abacusai/Smaug-Qwen2-72B-Instruct	41.08	78.25	56.27	35.35	14.88	15.18	46.56
◆	pankajmathur/orca_mini_v7_72b	39.06	59.3	55.06	26.44	18.01	24.21	51.35
💬	MaziyarPanahi/calme-2.2-llama3-70b	37.98	82.08	48.57	22.96	12.19	15.3	46.74
◆	VAGOsolutions/Llama-3-SauerkrautLM-70b-Instruct	37.82	80.45	52.03	21.68	10.4	13.54	48.8
💬	NousResearch/Hermes-3-Llama-3.1-70B	37.31	76.61	53.77	13.75	14.88	23.43	41.41
◆	ValiantLabs/Llama3-70B-Fireplace	36.82	77.74	49.56	19.64	13.98	16.77	43.25

Combination of LLM and EC

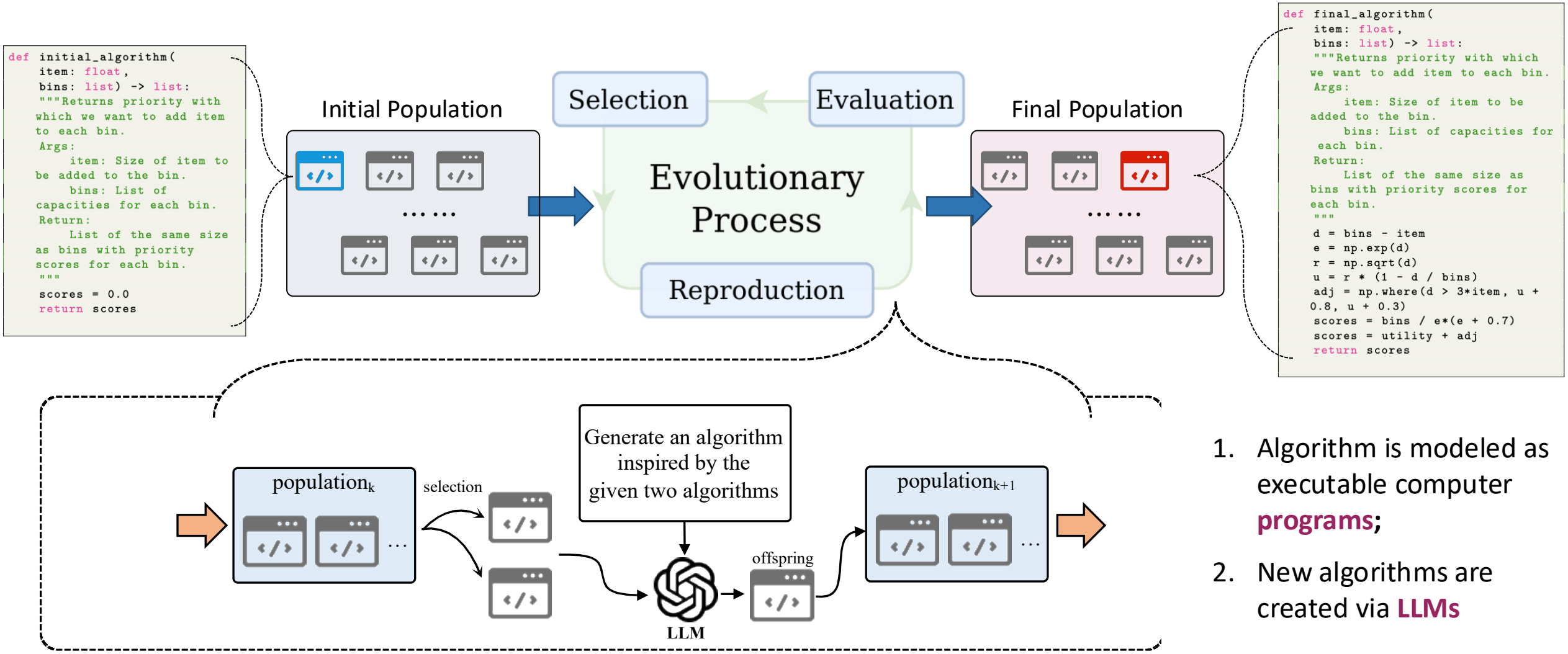
Automatic Algorithm Design

Evolution of Heuristics (EoH): Large language models + Evolutionary computation



LLM+EC Pipeline

```
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
```

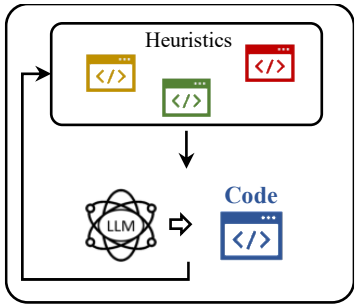



```
def final_algorithm(
    item: float,
    bins: list) -> list:
    """Returns priority with which
    we want to add item to each bin.
    Args:
        item: Size of item to be
    added to the bin.
        bins: List of capacities for
     each bin.
    Return:
        List of the same size as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

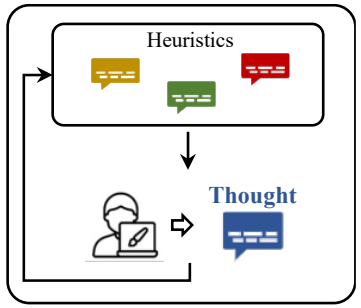
1. Algorithm is modeled as executable computer **programs**;
2. New algorithms are created via **LLMs**

[1] Fei Liu, 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.

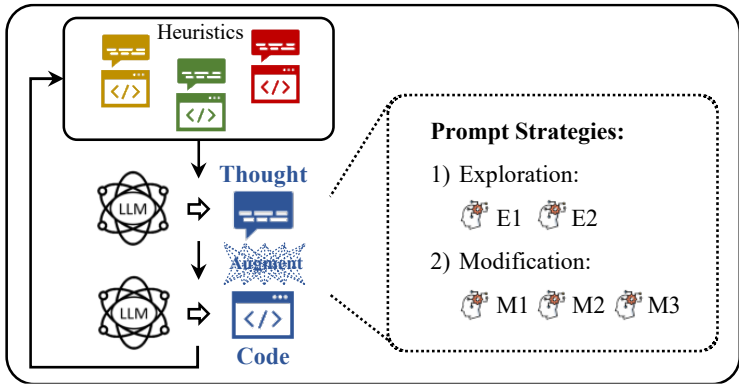
Evolution of Heuristics (Thoughts and Codes)



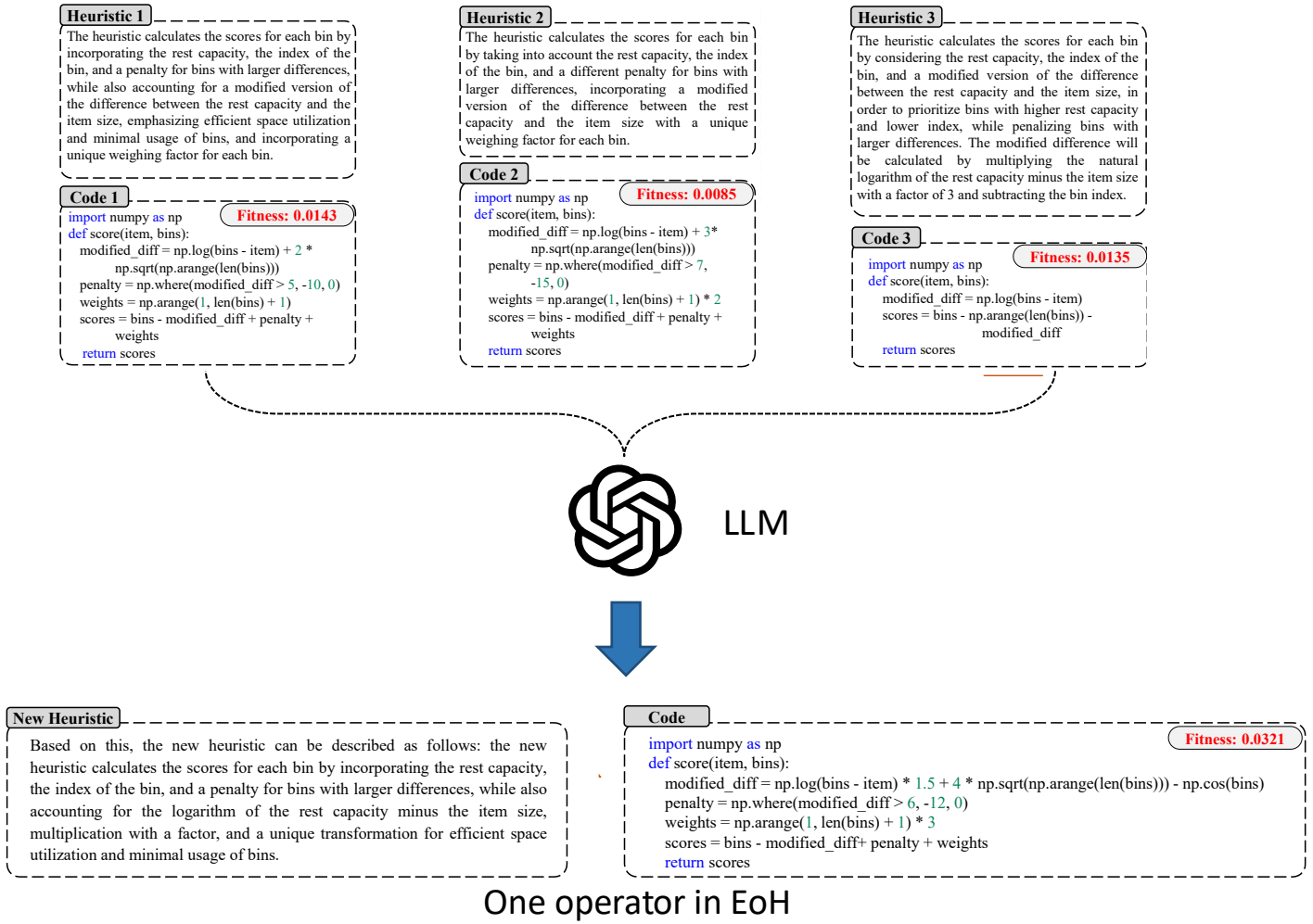
Evolution of Codes



Evolution of Thoughts



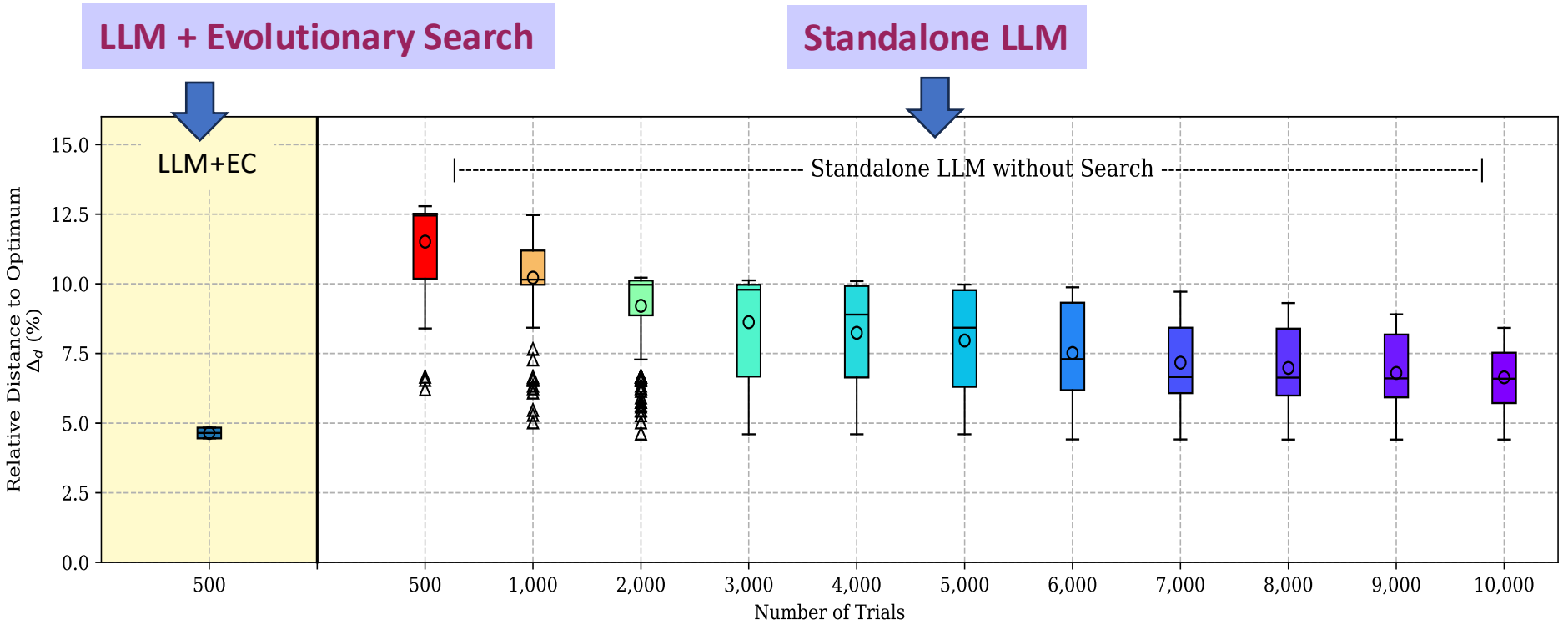
Evolution of Heuristics (EoH)



One operator in EoH

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang.
Towards Efficient Automatic Algorithm Design Using Large Language Model. **ICML 2024**. (**Oral Top 1.52%**)

Evolutionary Search is Important



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

OpenAI GPT-3.5 Turbo

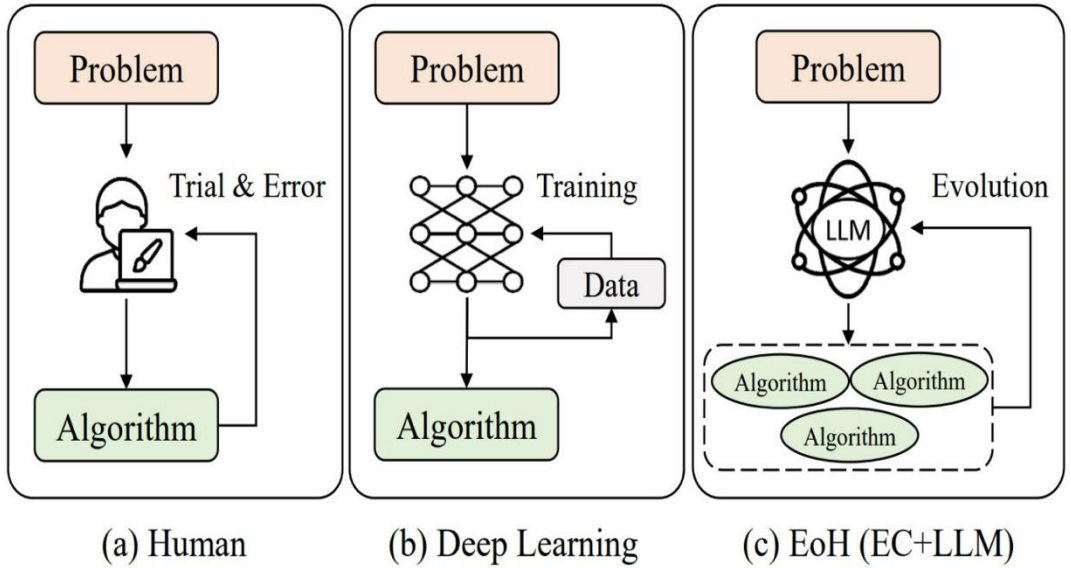
- Merely allowing a LLM to attempt multiple times is not enough.
- Combining LLM with search significantly enhances overall performance.
- Evolutionary algorithm is a highly effective option.

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**

A New Paradigm for Algorithm Design

- Evolution of Heuristics (EoH) is a new paradigm of algorithm design

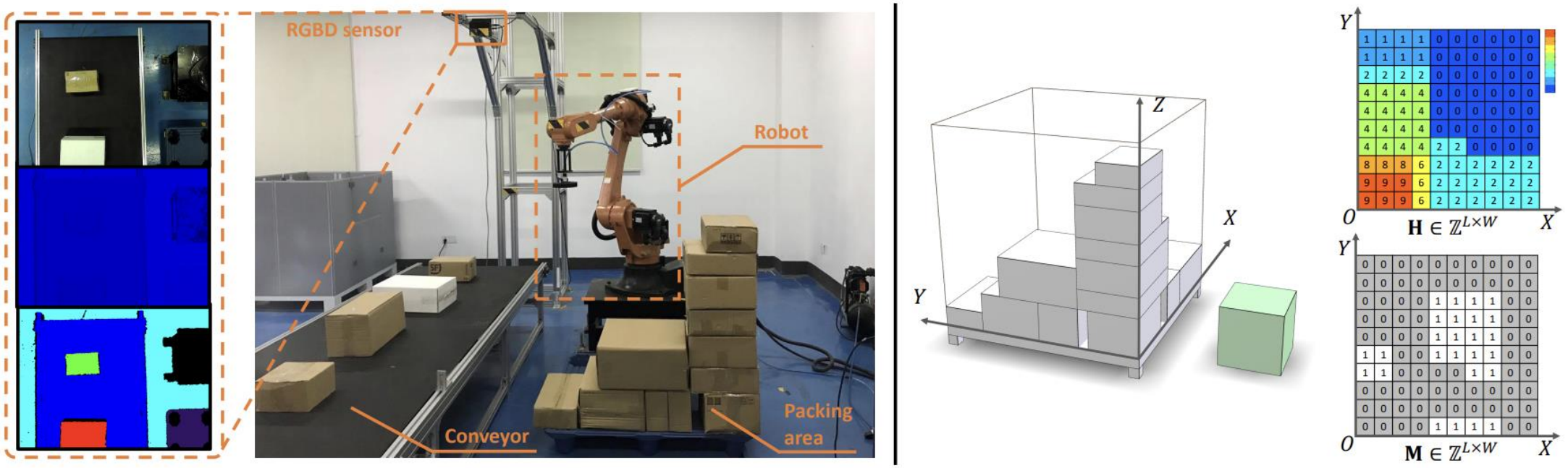
1. No model training and minimized manual crafting
2. Open-ended heuristic development
3. Automation and enhancement of problem-solving



- Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Towards Efficient Automatic Algorithm Design Using Large Language Model. **ICML 2024**. (**Oral Top 1.52%**)

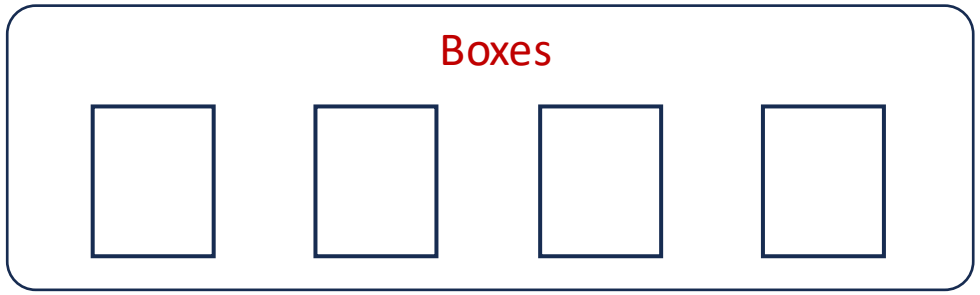
Online Bin Packing Problem

- **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

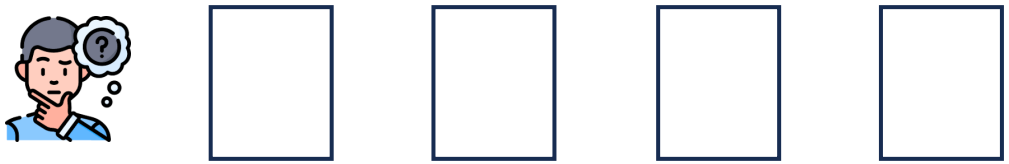


- Hang Zhao, Chenyang Zhu, Xin Xu, Hui Huang, and Kai Xu. "Learning practically feasible policies for online 3D bin packing." *Science China Information Sciences* 65, no. 1 (2022): 112105.

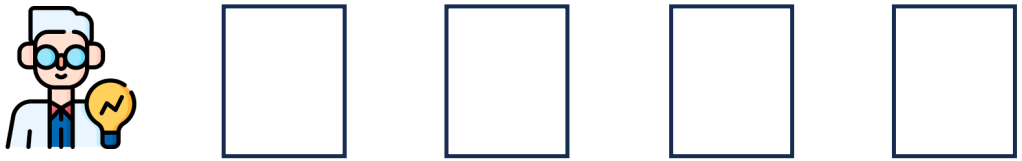
Online Bin Packing Problem



First fit Heuristic



Expert Heuristic



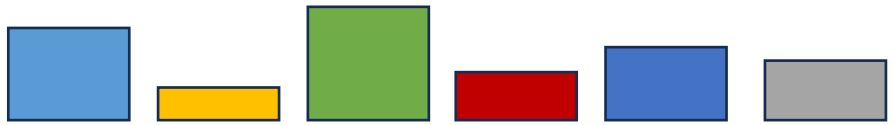
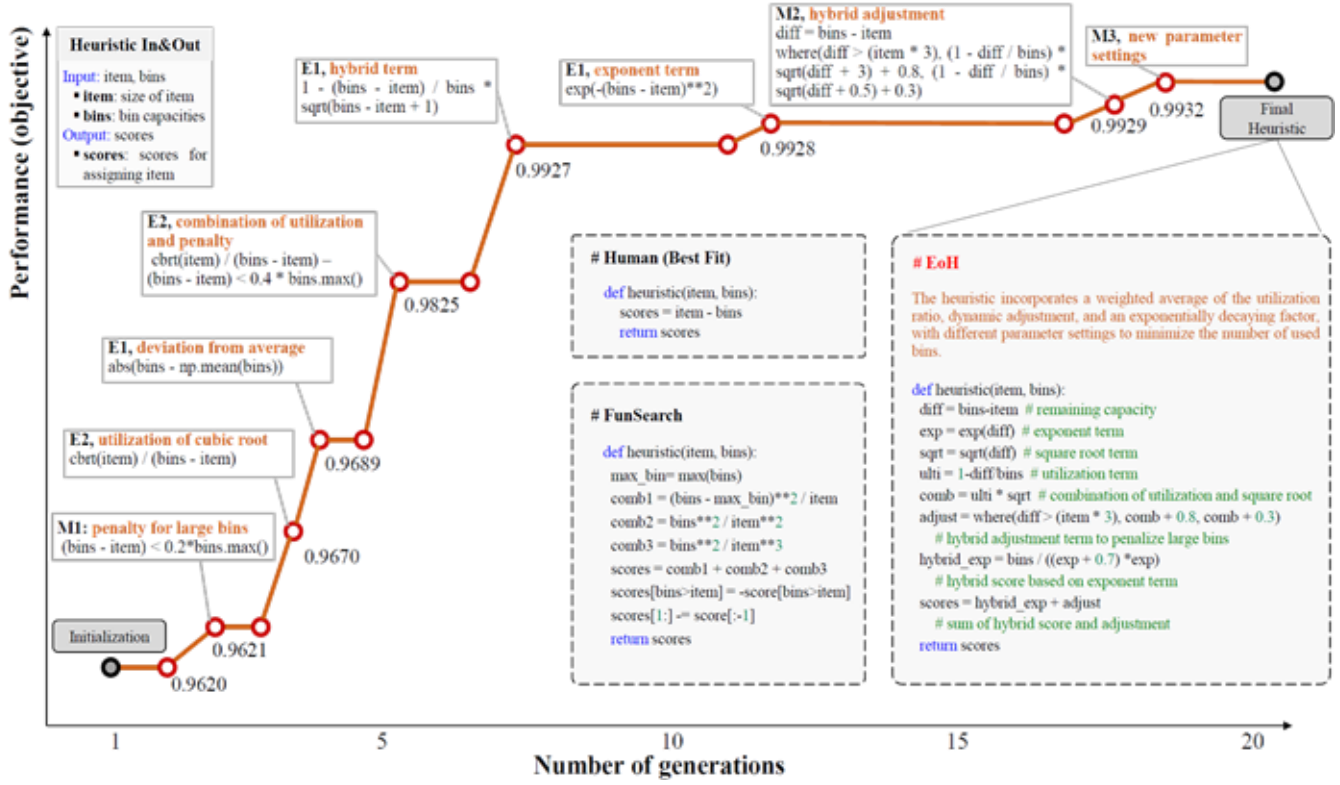


Illustration on Online Bin Packing

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 on Online Bin Packing

- Online Bin Packing (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 used bins
  - **Online:** one item each step, unknown future items, packed items not moved

Compared Algorithms

a) Human design: **First Fit**, **Best Fit**

b) Automatic algorithm design: **FunSearch**

(FunSearch is published on Nature 2024, recognized as a milestone)

a) Ours: **EoH**

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%**

Results on 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)

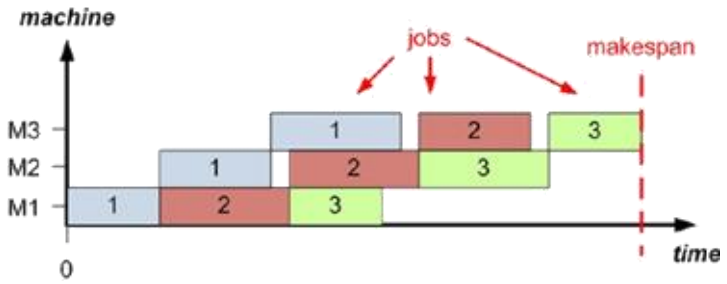
c) Ours: **EoH**

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**

Results on 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)

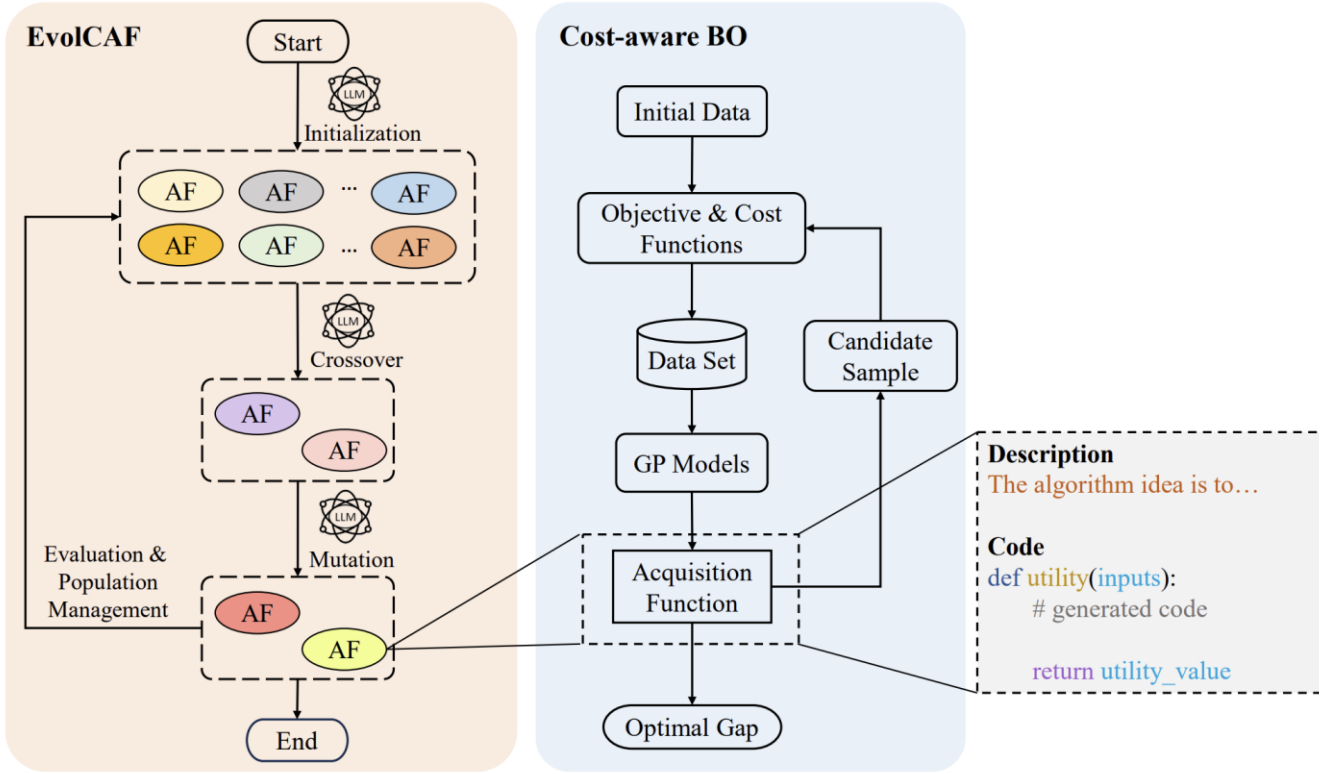
c) Ours: **EoH**

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**

Results on Bayesian Optimization

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

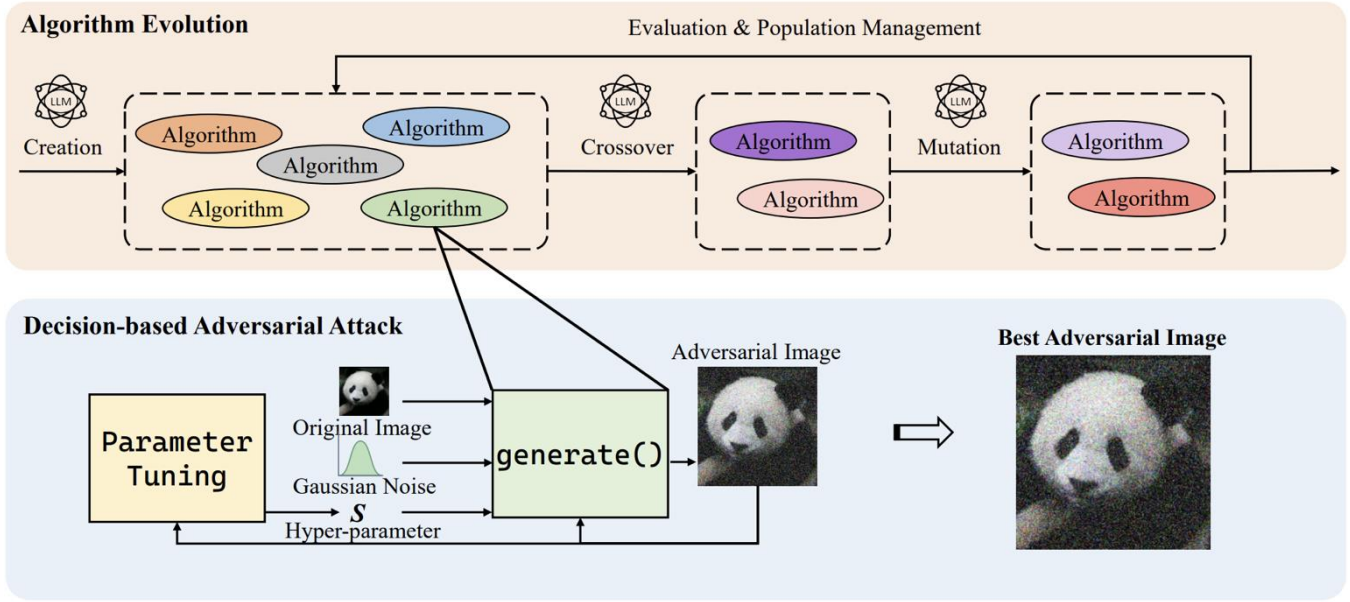


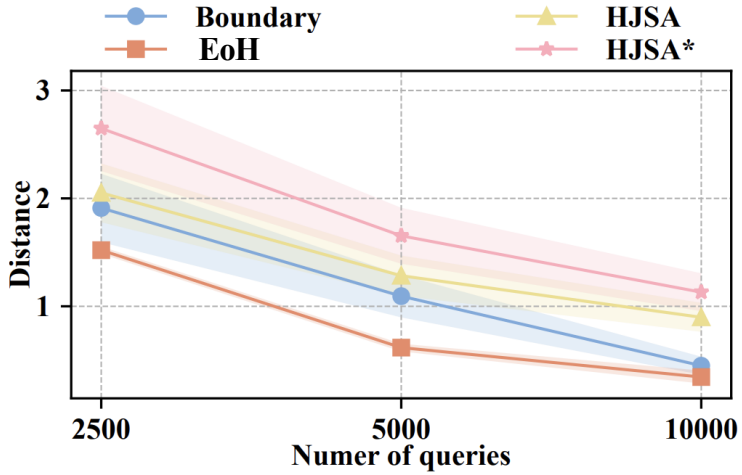
Test Instances	Budget	EI	EIpu	EI-cool	EoH
Ackley 2D	30	2.6600(40)	2.3302(40)	2.7369(40)	**0.4277(34)**
	300	1.2295(395)	0.8582(399)	0.8317(399)	**0.0505(306)**
Rastrigin 2D	30	4.7425(41)	5.6155(41)	5.7754(40)	**0.0511(34)**
	300	1.6656(410)	1.6678(408)	1.8518(408)	**0.0046(306)**
Griewank 2D	30	0.4875(35)	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**)

Results on 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**

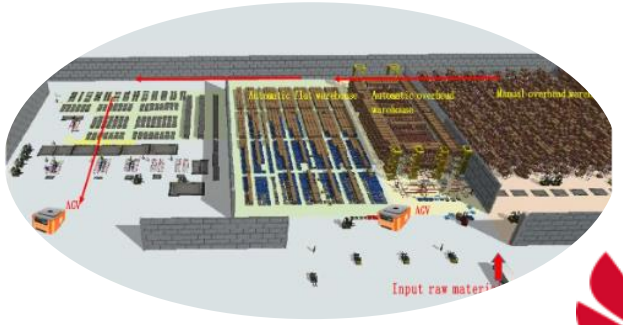
Ping Guo, Fei Liu, Xi Lin, Qingchuan Zhao, and Qingfu Zhang. L-AutoDA: Leveraging large language models for automated decision-based adversarial attacks. **GECCO *2024***

Industry Applications

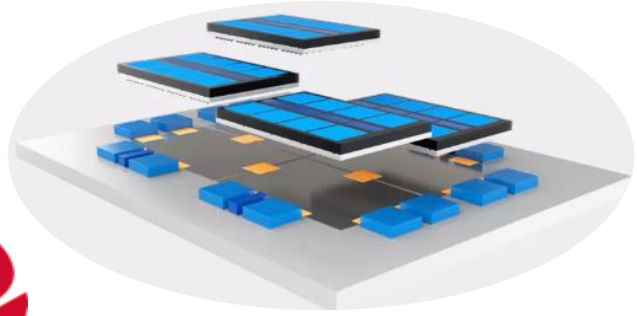
MIPLIB Performance: Among 20 heterogeneous problems, the new algorithm successfully solves 18, with 8 reaching optimal beating human designed algorithms

Heuristics	Average Primal Gap	Wins
coefficient	1510	0/18
distributional	4826	1/15
farkas	1180	3/11
fractional	1268	1/14
linesearch	1589	1/16
pseudocost	1242	4/15
vectorlength	4803	2/17
EoH	**844**	**8/18**

Warehouse Management

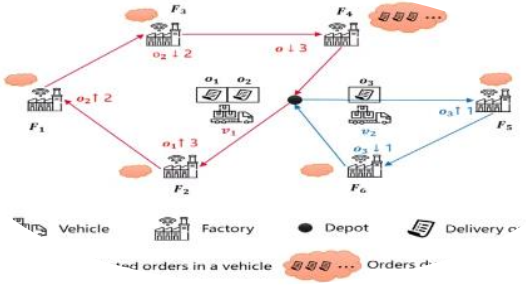


EDA



HUAWEI

Logistic



Port Scheduling

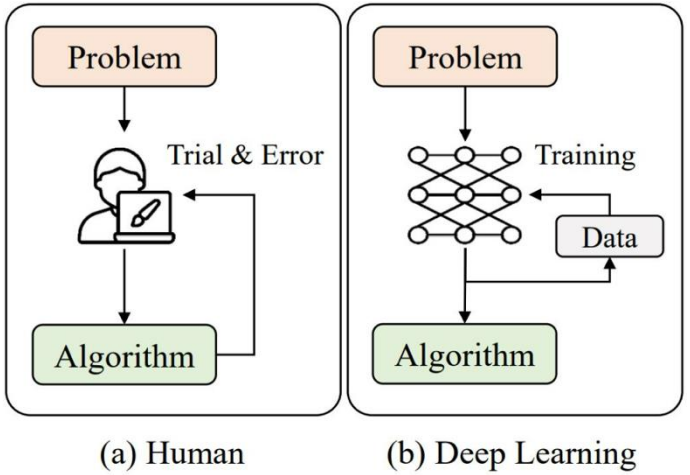
Conclusion

- EoH: A New Paradigm for Algorithm Design

  1. No model training required and reduced manual crafting
  2. Open-ended heuristic development
  3. Automation and enhancement of problem-solving

- Demonstration on Widely-studied CO Problems, Bayesian Opt. and ML Tasks:
  - Beating human design and existing AAD methods

- Industry Applications and Open-source



Outlines

- ❑ **Background**
- ❑ **LLM4AD review**
- ❑ **Evolution of Heuristics (EoH)**
- ❑ **MEoH**
- ❑ **LLM4AD Platform**
- ❑ **Future Works**
- ❑ **Conclusion**

Related works

	Heuristic design	Considered objectives
Configuration, selection, and composition	Expert knowledge, hand crafted rules	Single objective
LLM-based automated heuristic design	LLM	Single objective
Multi-objective heuristic design	Expert knowledge, hand crafted rules	Multiple objectives
MEoH	LLM	Multiple objectives

Method

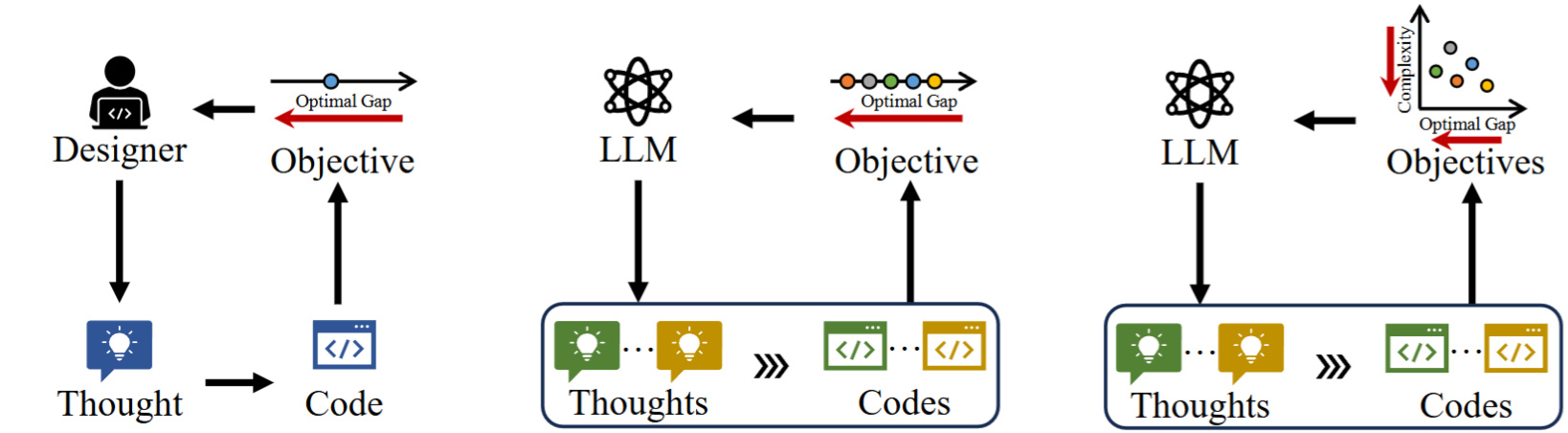


Figure 1: Comparison to human design and existing LLM-based heuristic design (a) manual heuristic design by human experts, (b) single-objective LLM-based heuristic design (e.g., FunSearch and EoH), and (c) our proposed multi-objective heuristic design (MEoH).

Method

- Dominance-dissimilarity mechanism
  - MOEAs
    - The dominance relationships in the objective space
  - LLM-based automated heuristic design
    - The dissimilarity of heuristics in the search space

Method

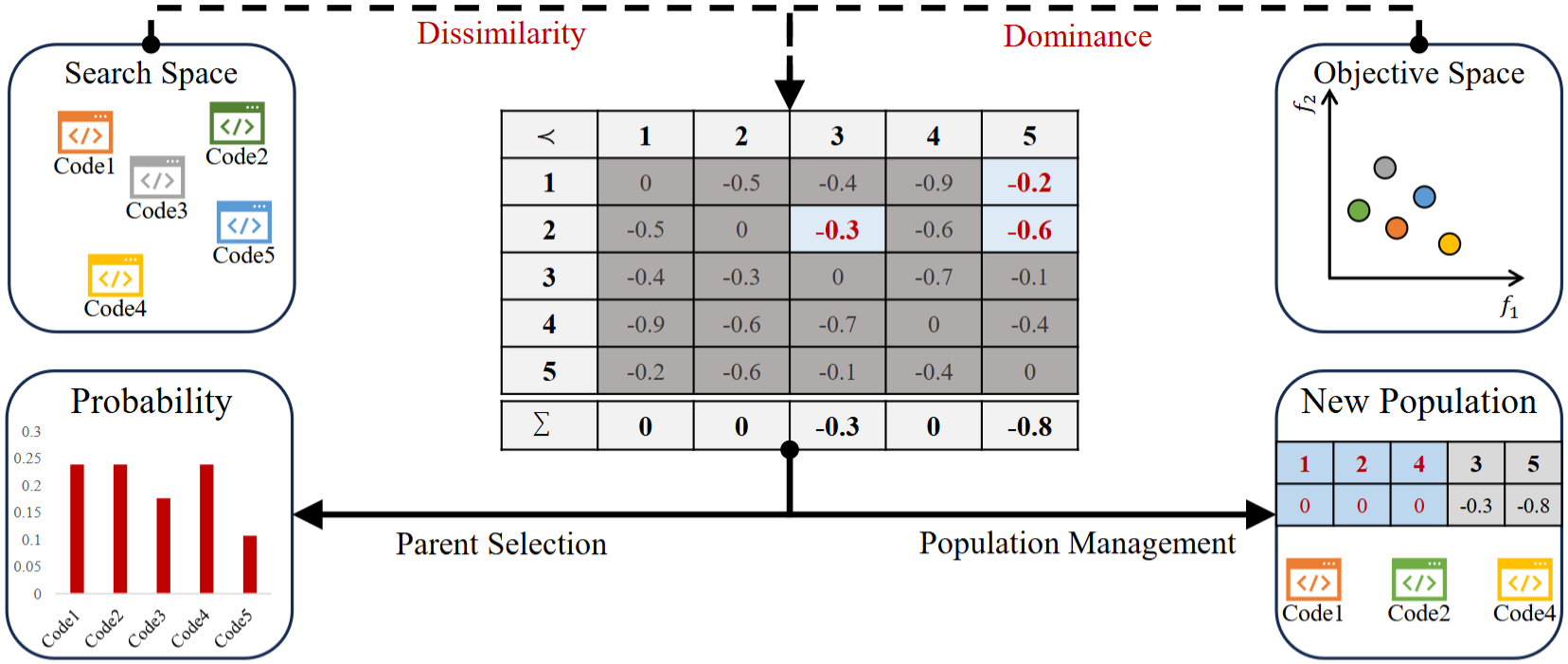
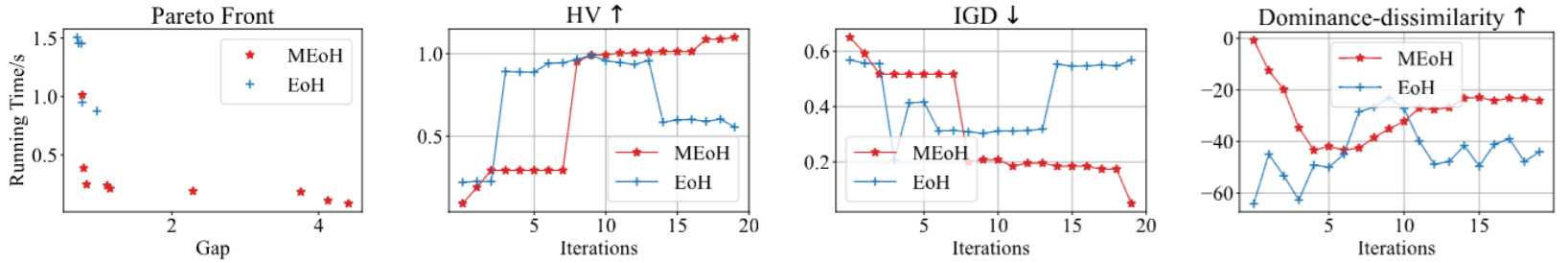


Figure 2: An illustration of parent selection and population management with dominance-dissimilarity mechanism. By incorporating code dissimilarity in the search space and dominance relationships in the objective space, the parent selection and population management are improved to foster diversity and search efficiency.

Experiments

- Online BPP

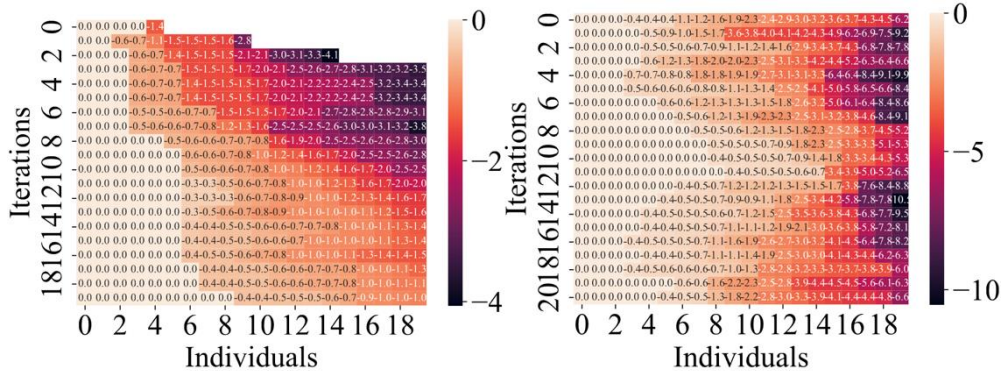


(a) Pareto Front (b) HV (c) IGD (d) Score

Figure 3: Comparations of EoH and MEoH on BPP5k.

Table 1: Results of In- and Out-of-distribution BPP.

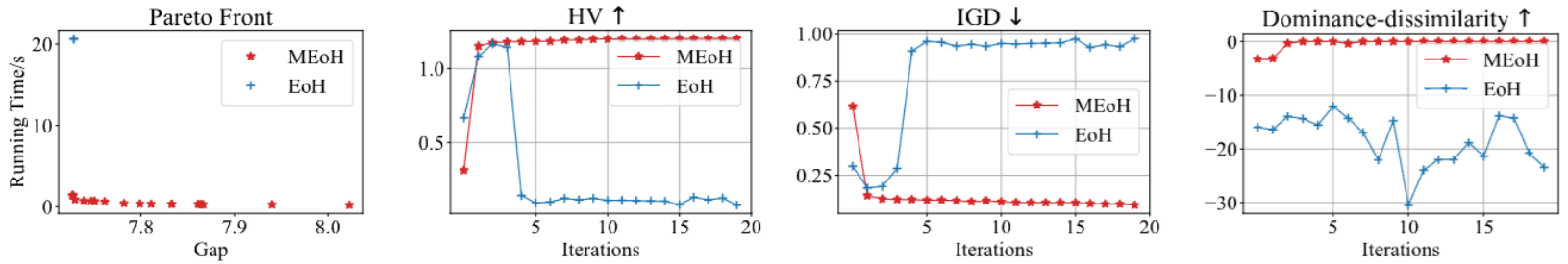
Weibull	FunSearch		EoH		MEoH	
	Gap	Time/s	Gap	Time/s	Gap	Time/s
5k C100	0.802%	0.728	0.753%	1.362	1.387%	0.191
10k C100	2.595%	2.128	0.537%	5.128	0.651%	0.650
100k C100	3.319%	195.734	0.391%	502.938	0.080%	59.078
Avg.	2.239%	66.197	0.560%	169.809	0.706%	19.973
5k C500	29.494%	0.750	0.100%	1.672	0.351%	0.100
10k C500	47.734%	2.459	0.125%	6.337	0.473%	0.306
100k C500	53.640%	259.094	0.099%	646.828	0.410%	22.078
Avg.	43.623%	87.434	0.108%	218.279	0.411%	7.495



(a) BPP, MEoH (b) BPP, EoH

Experiments

- TSP

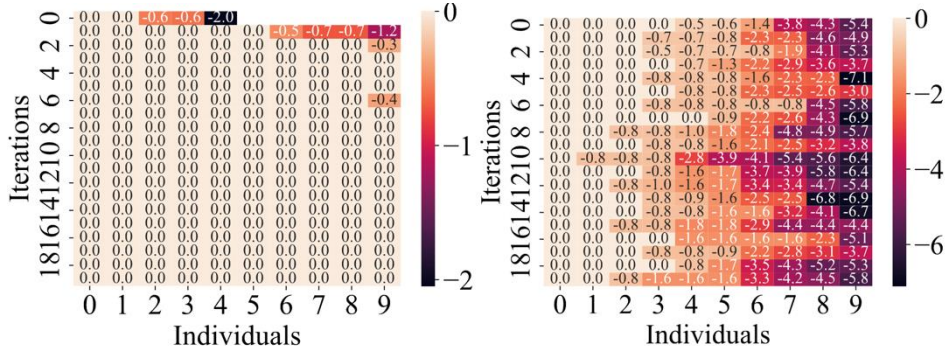


(a) Pareto Front (b) HV (c) IGD (d) Score

Figure 4: Comparations of EoH and MEoH on TSP100.

Table 2: Results of in- and out-of-distribution randomly generated TSP.

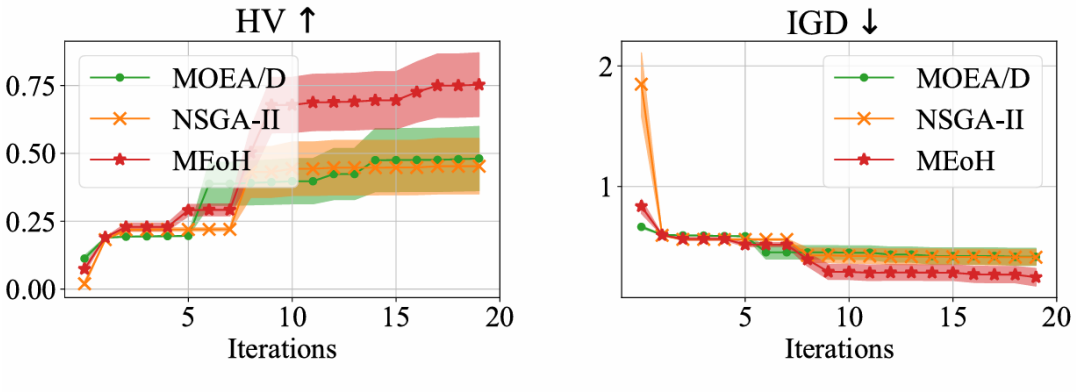
	TSP100		TSP500		TSP1000	
	Gap	Time/s	Gap	Time/s	Gap	Time/s
FunSearch	0.100%	1.452	1.525%	27.598	2.344%	161.124
EoH	0.113%	22.434	1.750%	43.541	2.524%	262.603
MEoH(Best)	0.109%	1.373	1.733%	30.945	4.208%	26.844
MEoH(Fast)	3.690%	0.175	4.402%	3.306	4.536%	21.900



(c) TSP, MEoH (d) TSP, EoH

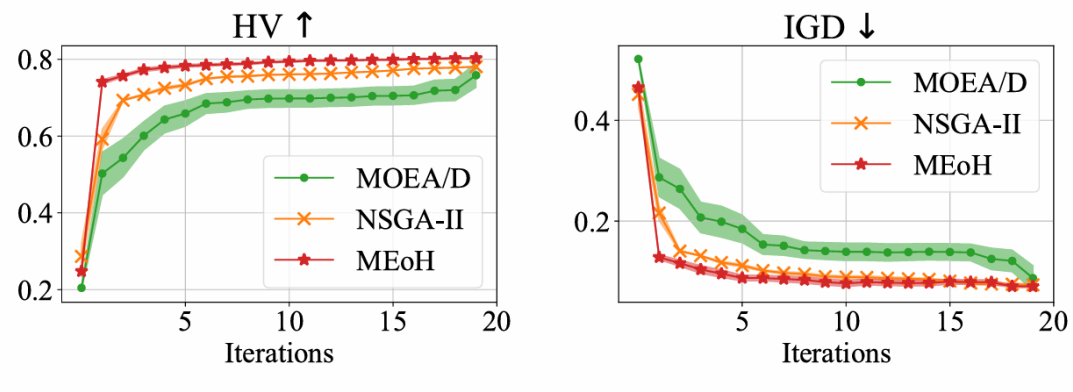
Experiments

- Conventional MOEAs



(a) HV (b) IGD

Figure 3: Comparison to conventional MOEAs on BPP.



(a) HV (b) IGD

Figure 4: Comparison to conventional MOEAs on TSP.

Conclusion

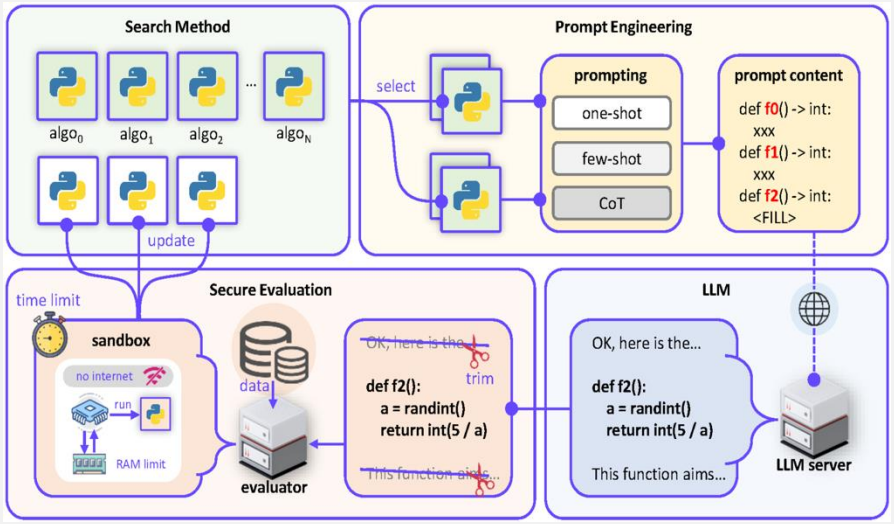
- The first LLM-based automated heuristic design framework
- A dominance-dissimilarity mechanism considering both the objective and search space

Outlines

- ❑ **Background**
- ❑ **LLM4AD review**
- ❑ **Evolution of Heuristics (EoH)**
- ❑ **MEoH**
- ❑ **LLM4AD Platform**
- ❑ **Future Works**
- ❑ **Conclusion**

Large Language Model for Algorithm Design (LLM4AD)

LLM4AD 基于大模型的算法设计开源平台

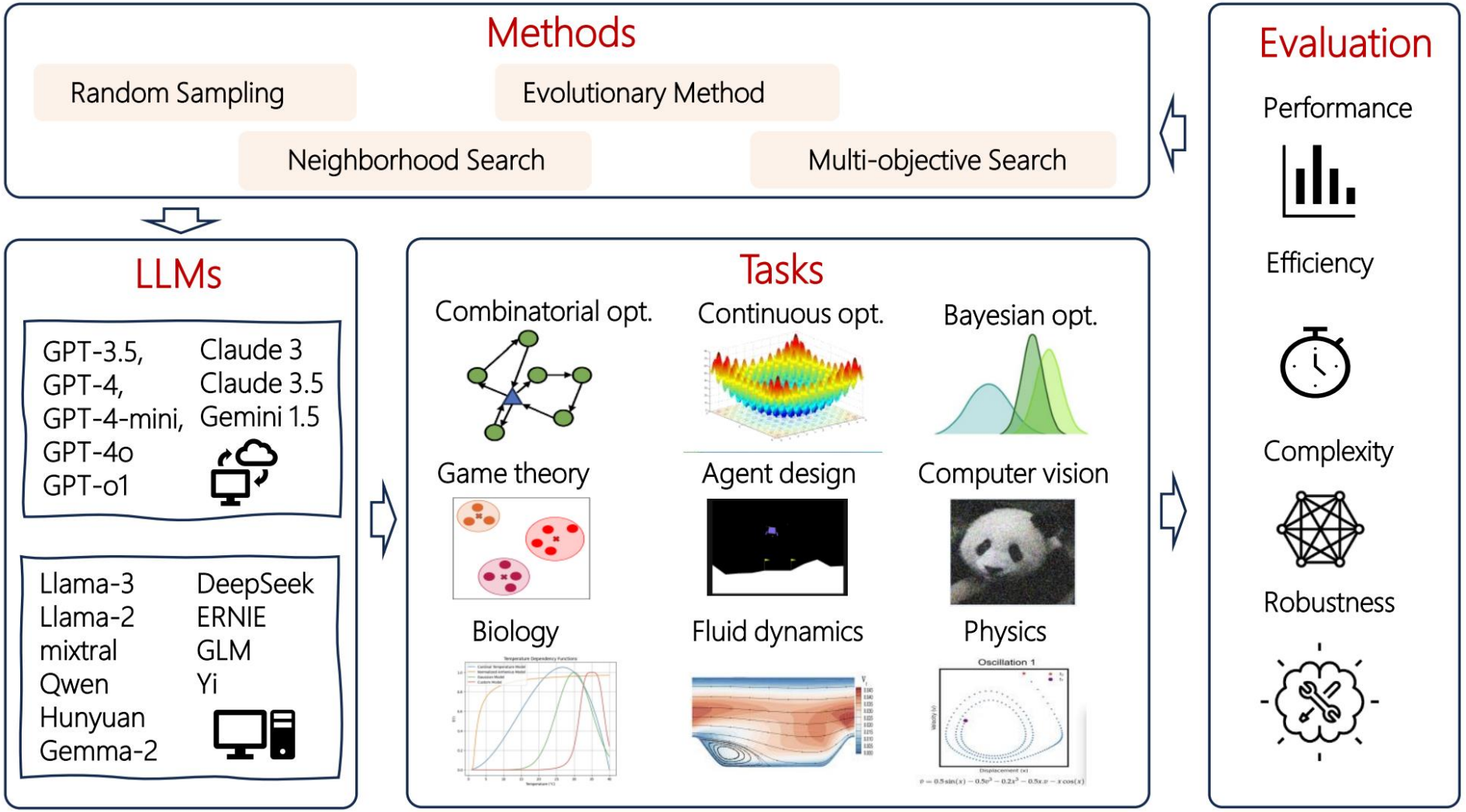


- Github 开源仓库地址：
https://github.com/Optima-CityU/llm4ad

- LLM4AD 手册：
https://llm4ad-doc.readthedocs.io/en/latest/

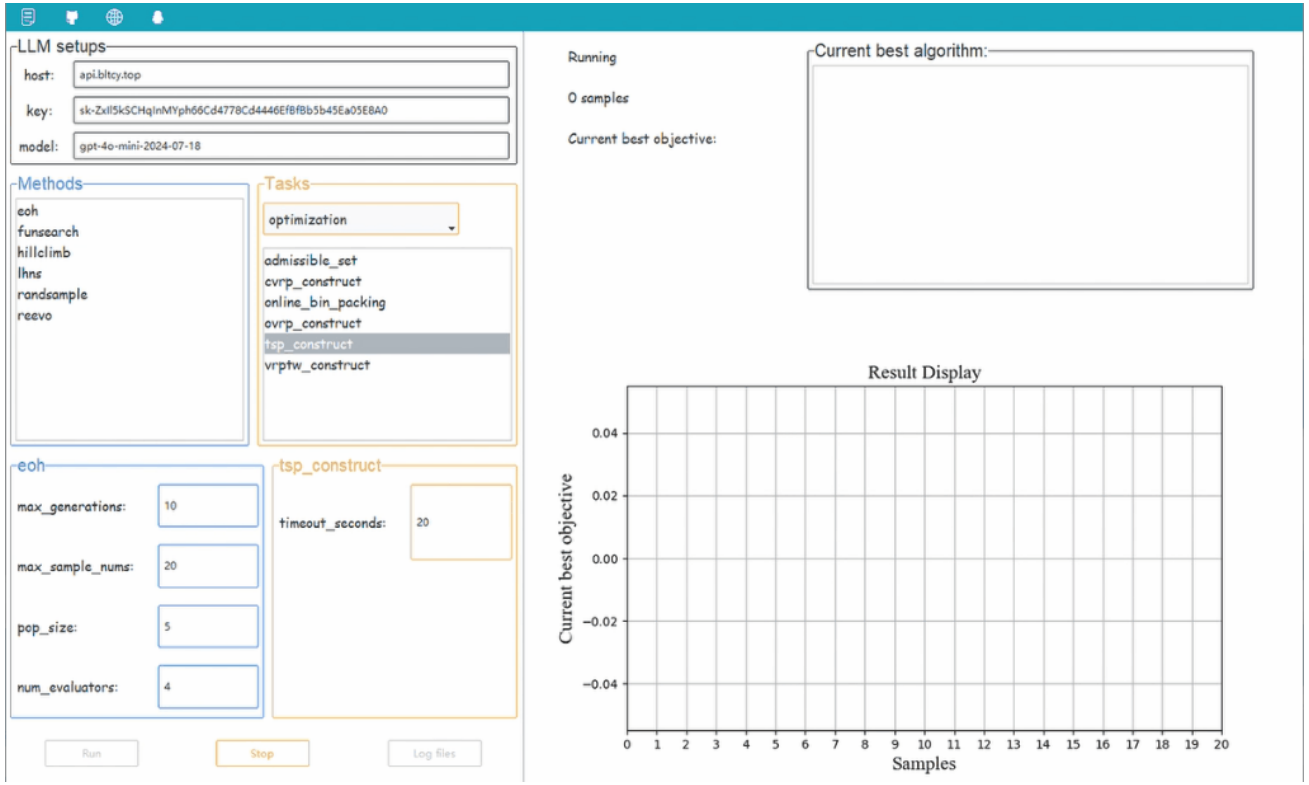
- LLM4AD 网站：
www.llm4ad.com

LLM4AD Overview

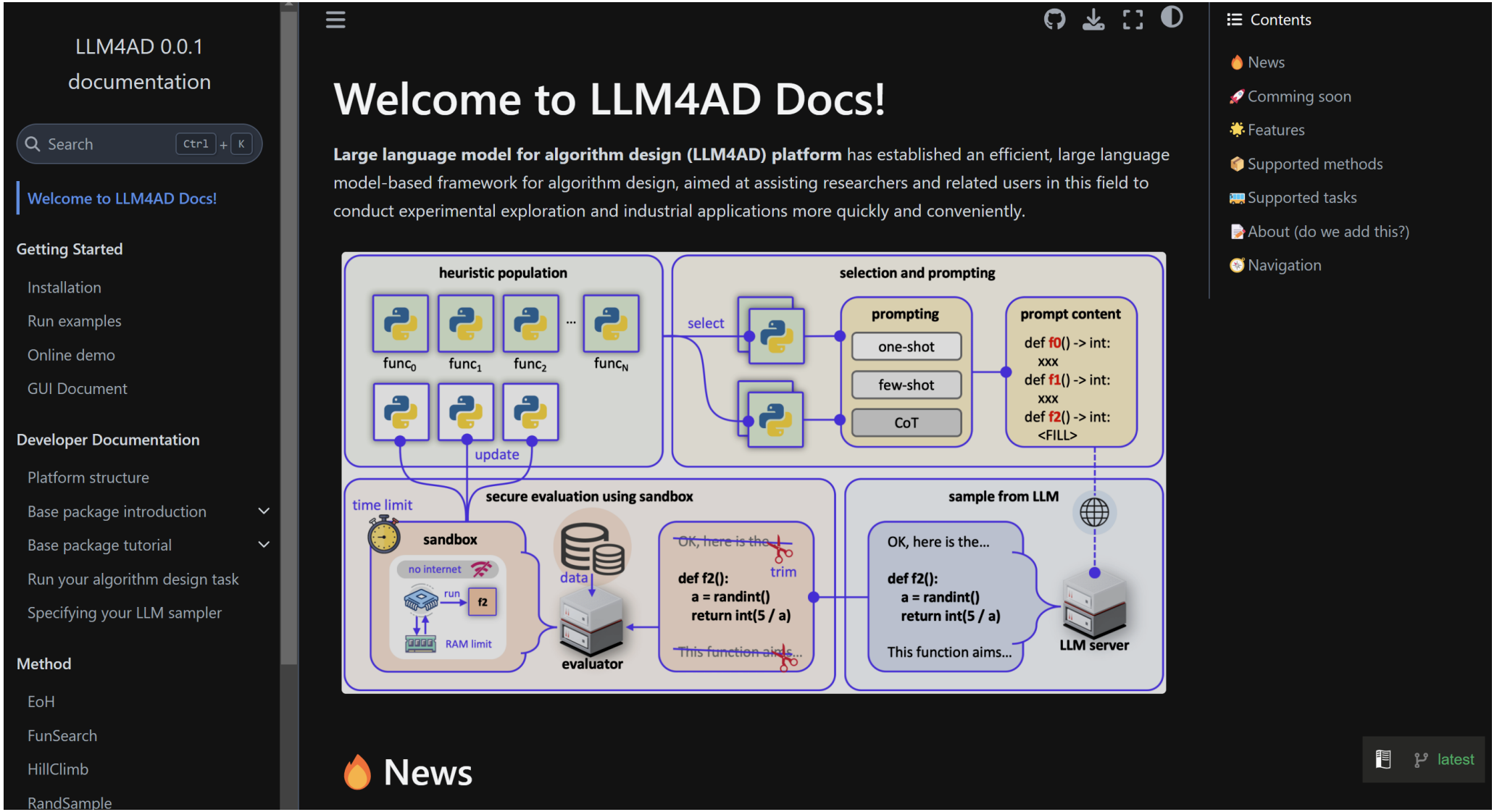


Terminal & GUI Usage

```python
from llm4ad.task.optimization.online_bin_packing import OBPEvaluation
from llm4ad.tools.llm.llm_api_https import HttpsApi
from llm4ad.method.eoh import EoH, EoHProfiler


def main():

    llm = HttpsApi(host="xxx",  # your host endpoint, e.g., api.openai.com/v1/completions, api.deepseek
                   key="sk-xxx",  # your key, e.g., sk-abcdefghijklmn
                   model="xxx",  # your llm, e.g., gpt-3.5-turbo, deepseek-chat
                   timeout=20)

    task = OBPEvaluation()

    method = EoH(llm=llm,
                 profiler=EoHProfiler(log_dir='logs/eoh', log_style='simple'),
                 evaluation=task,
                 max_sample_nums=20,
                 max_generations=10,
                 pop_size=4,
                 num_samplers=1,
                 num_evaluators=1,
                 debug_mode=False)

    method.run()


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



Support Docs



Supports

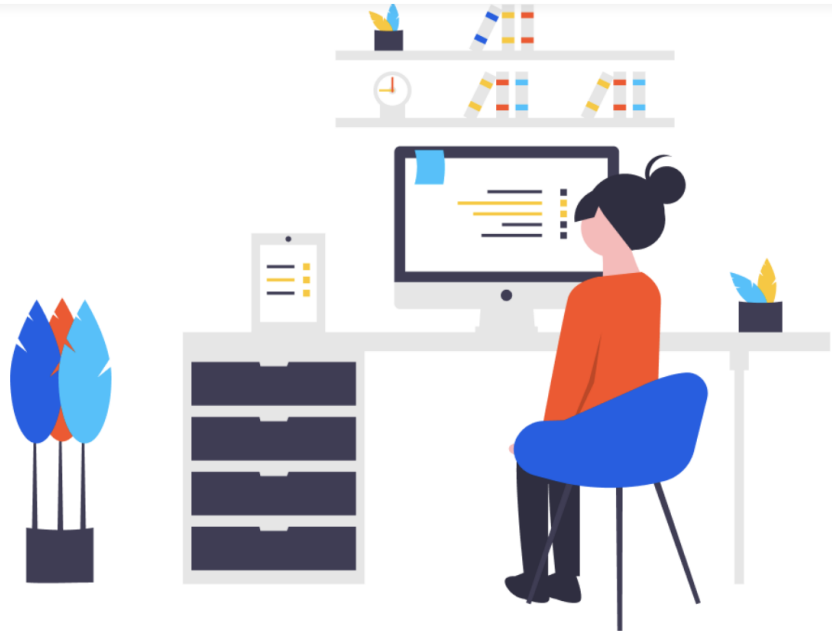
LLM4AD

Home News Examples About Us

Large Language Model for Algorithm Design

LLM4AD harnesses the power of large language models to streamline and enhance the process of algorithm design. Our platform enables users to quickly generate, test, and refine algorithms using natural language inputs, making complex algorithm tasks more accessible and efficient.

Document GitHub



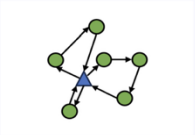
Get started with LLM4AD

```
# Import necessary libraries
import numpy as np

def heuristic(item, bins):
```

Run quickstart

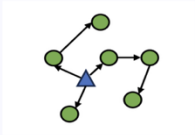
Algorithm Design Task Examples in LLM4AD

Optimization



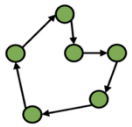
Capacitated Vehicle Routing Problem

Constructive Heuristics for Capacitated Vehicle Routing Problem (CVRP).



Open Vehicle Routing Problem

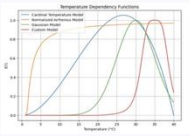
The Open Vehicle Routing Problem (OVRP) is a variant of VRP that has open routes.



Traveling Salesman Problem

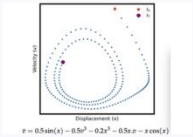
Constructive Heuristics for Traveling Salesman Problem (TSP).

Science Discovery



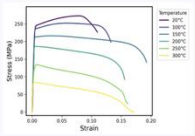
Bacteria Growth

A biology-focused task aiming to discover growth patterns.



Oscillator1

A mathematical task aimed at uncovering oscillator behaviors.



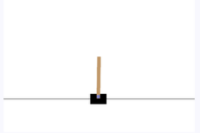
Stress & Strain

A physics task focused on discovering relationships.

Machine Learning

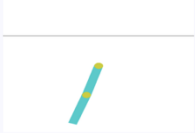
Mountain Car

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



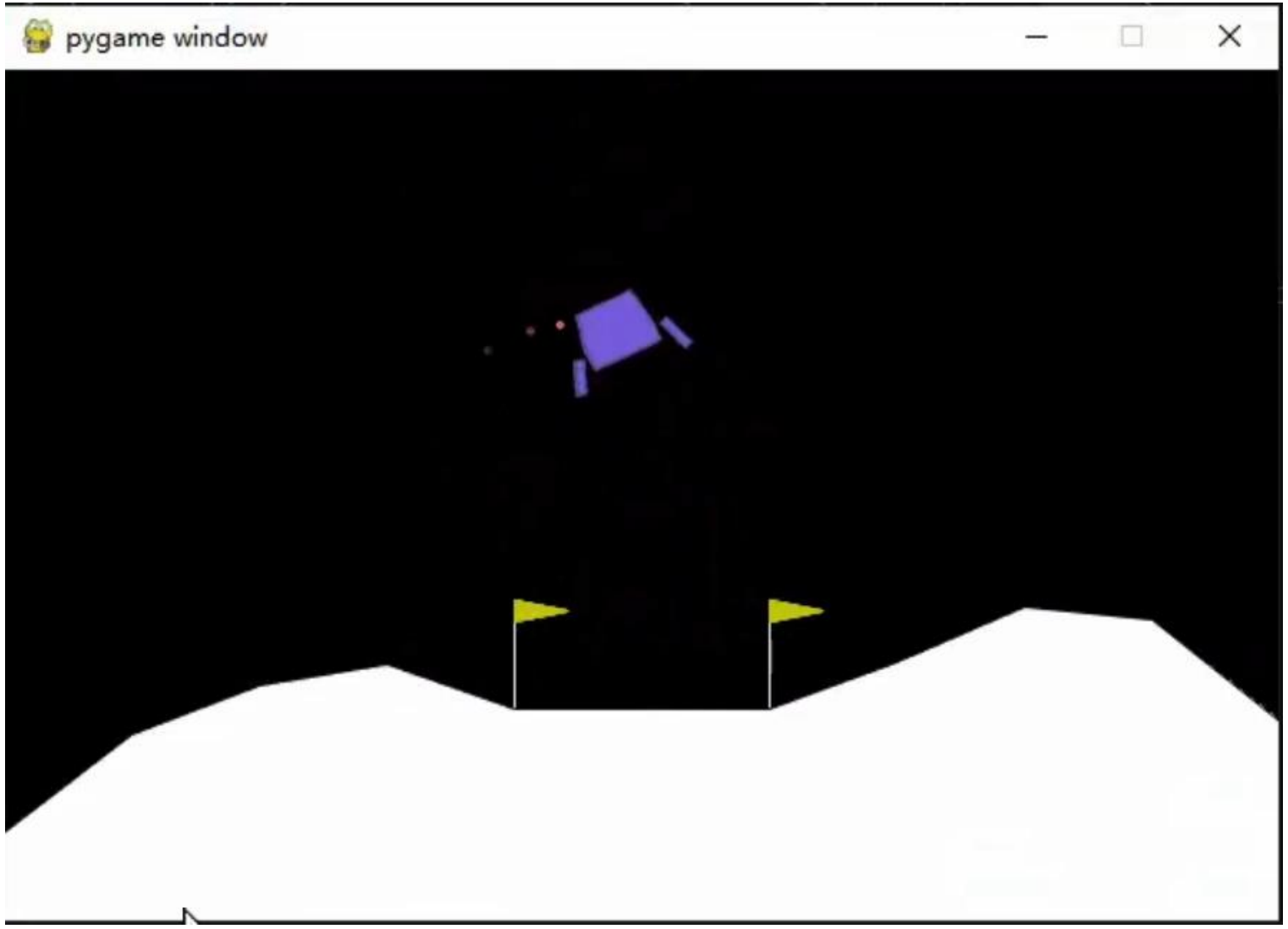
Cart Pole

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

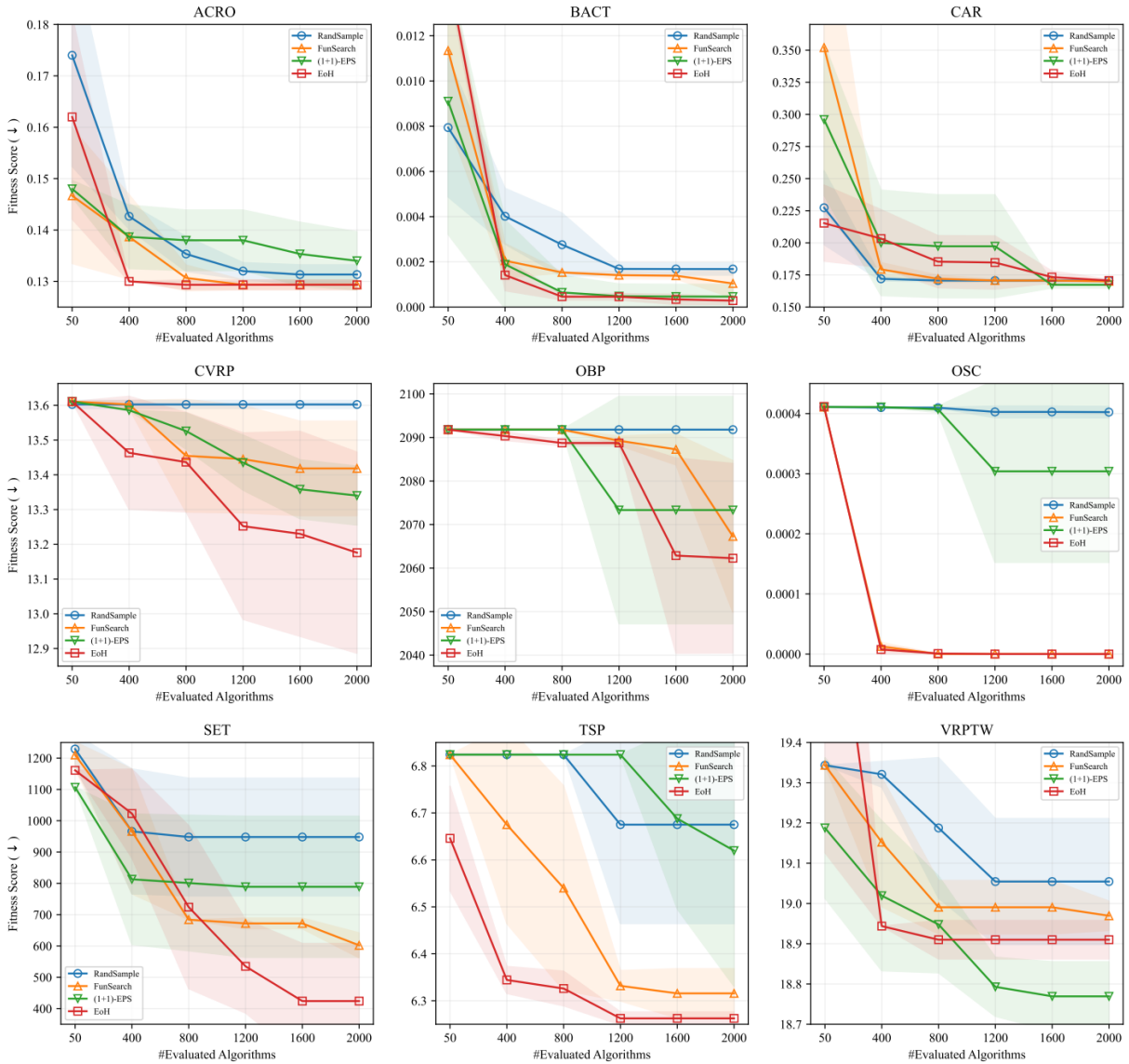
- ✓ Admissible Set (Set)
- ✓ Online Bin Packing (OBP)
- ✓ Traveling Salesman Problem (TSP)
- ✓ Capacitated Vehicle Routing Problem (CVRP)
- ✓ Vehicle Routing Problem with Time Windows (VRPTW)

Science Discovery

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

Machine Learning

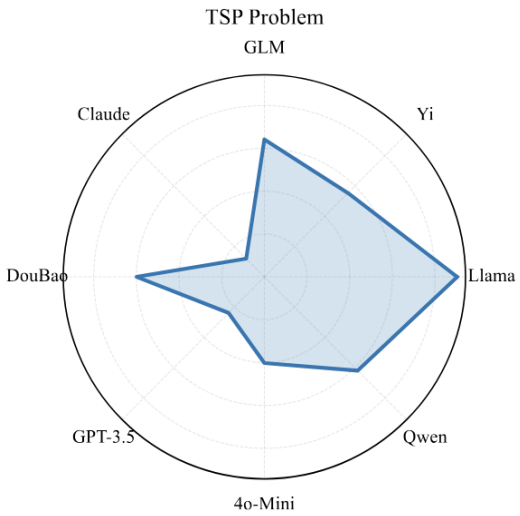
- ✓ Acrobot (ACRP)
- ✓ Car Mountain (CAR)

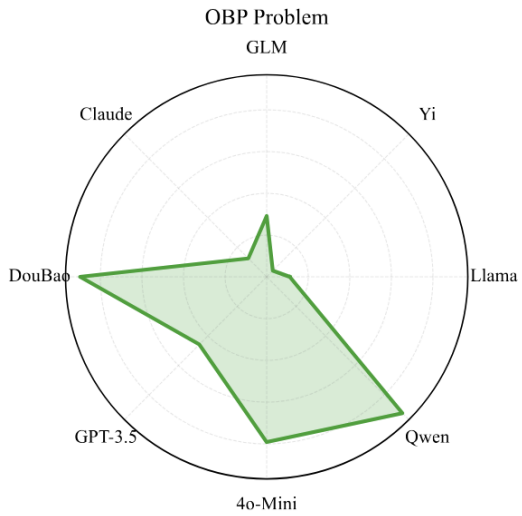


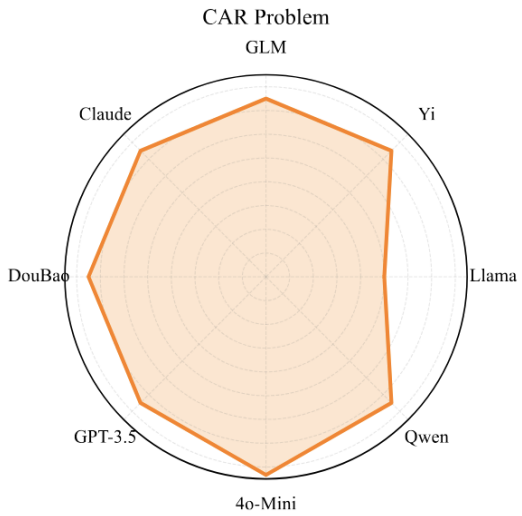
Benchmarking Results

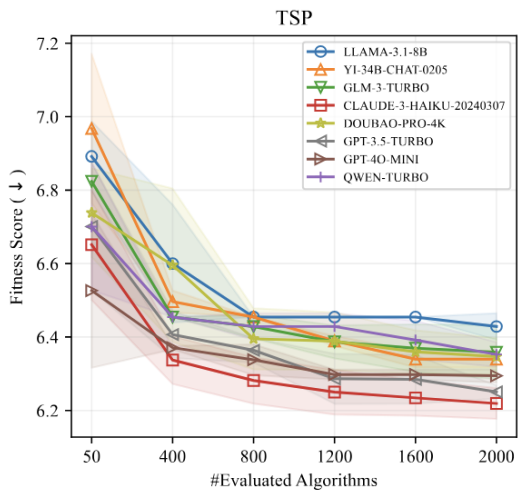
Eight LLMs

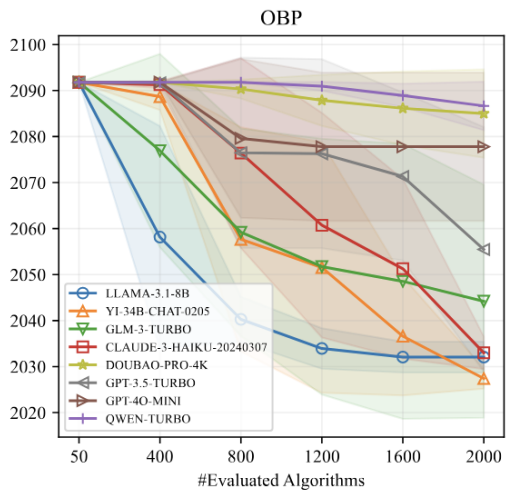
- ✓ **Open AI**
  - GPT-3.5
  - GPT4o-mini
- ✓ **Claude**
- ✓ **Llama**
- ✓ **Qwen**
- ✓ **GLM**
- ✓ **YI**
- ✓ **DouBao**

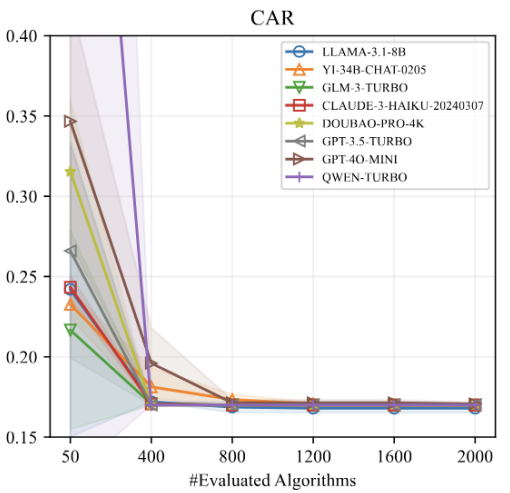












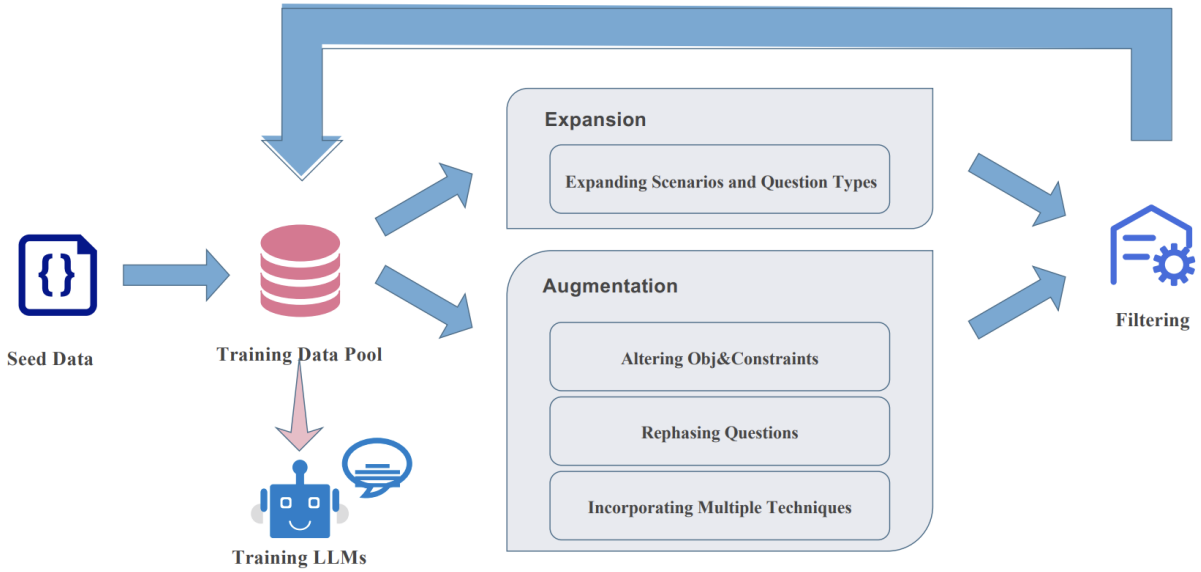
Outlines

- ❑ **Background**
- ❑ **LLM4AD review**
- ❑ **Evolution of Heuristics (EoH)**
- ❑ **MEoH**
- ❑ **LLM4AD Platform**
- ❑ **Future Works**
- ❑ **Conclusion**

Future Directions

1) Domain LLM for Algorithm Design

- Cost
- Date generation
- Reasoning



An Expected Training Example

Input - Natural Language Problem:
A company has three transportation options to choose from to transport 25 tons of cargo, namely trucks, airplanes, and ships with costs \$100, \$120, \$80 per ton and capacities of 10, 20, 30 tons respectively. The company can't choose trucks and ships together. How should the company optimize the selection and allocation of these methods to minimize overall costs?

Target - Mathematical Model and Program:
Mathematical Model:
Variables:
- x_1, x_2, x_3: 0-1 variables indicating whether trucks, airplanes, and ships are selected, respectively.
- y_1, y_2, y_3: Non-negative continuous variables indicating the volume of cargo transported by trucks, airplanes, and ships.

Objectives and Constraints:

$$\text{Minimize}_{x_i \in \{0,1\}, y_i \geq 0} \quad 100y_1 + 120y_2 + 80y_3$$

subject to

$$\begin{aligned} & x_1 + x_2 + x_3 \geq 1 && \text{(At least one transportation)} \\ & y_1 \leq 10x_1, \quad y_2 \leq 20x_2, \quad y_3 \leq 30x_3 && \text{(Capacity limits)} \\ & x_1 + x_3 \leq 1 && \text{(Mutual exclusion constraint)} \\ & y_1 + y_2 + y_3 \geq 25 && \text{(Transportation requirements)} \end{aligned}$$

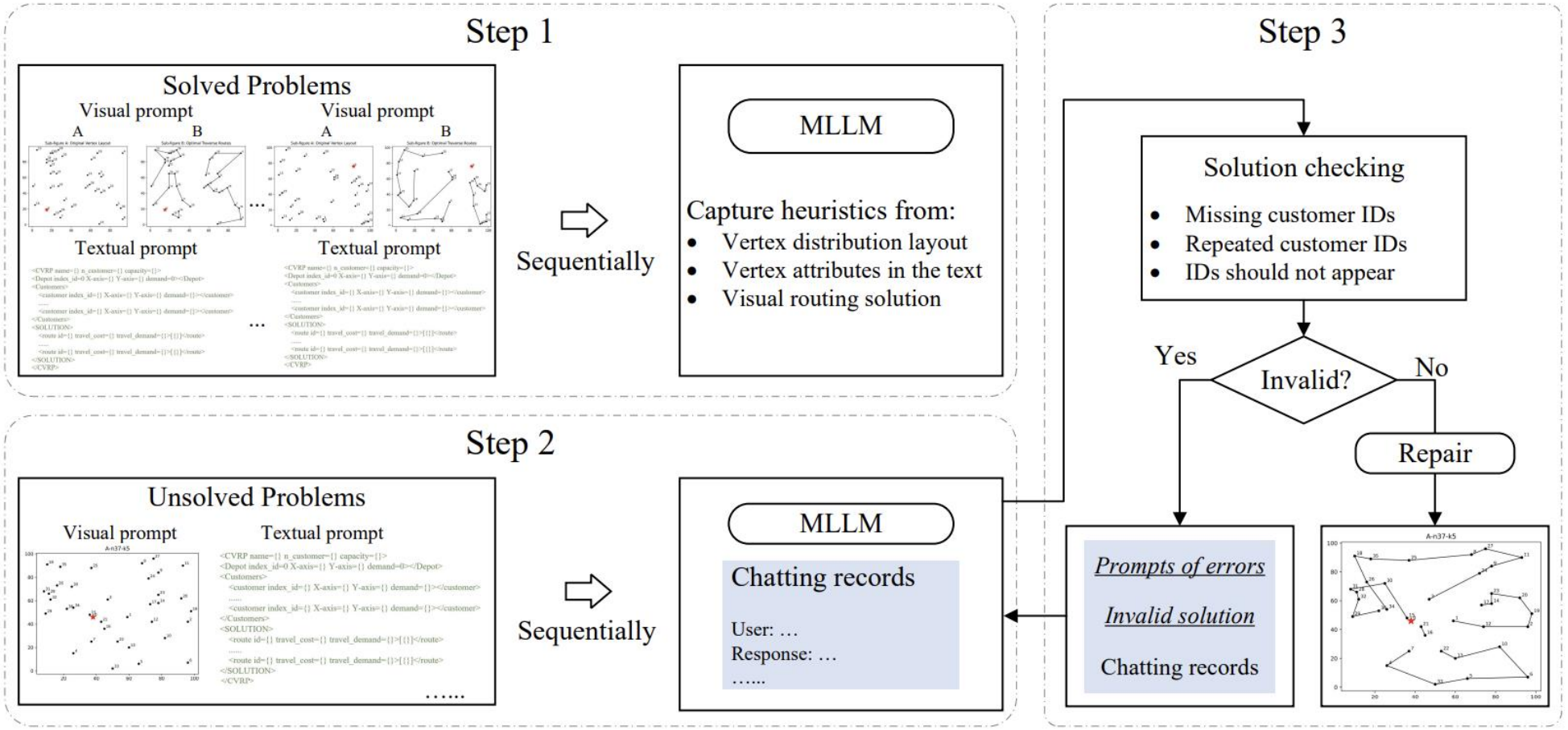
Program using COPT solver:

```
import coptpy as cp
env = cp.Envr()
model = env.createModel("TransportationOptimization")
...
model.setObjective(cp.quicksum(costs[mode] * y[mode] for mode in costs),
    ↪ sense=cp.COPT.MINIMIZE)
model.addConstr(x['trucks'] + x['ships'] <= 1, name="ModeExclusivity")
model.addConstr(cp.quicksum(x[mode] for mode in costs) >= 1, name="
    ↪ AtLeastOneMode")
for mode in costs:
    model.addConstr(y[mode] <= capacities[mode] * x[mode], name=f"Capacity_{
        ↪ mode}")
model.solve()
...
```

Zhengyang Tang, Chenyu Huang, Xin Zheng, Shixi Hu, Zizhuo Wang, Dongdong Ge, and Benyou Wang. 2024. ORLM: Training Large Language Models for Optimization Modeling. *arXiv preprint arXiv:2405.17743* (2024).

Future Directions

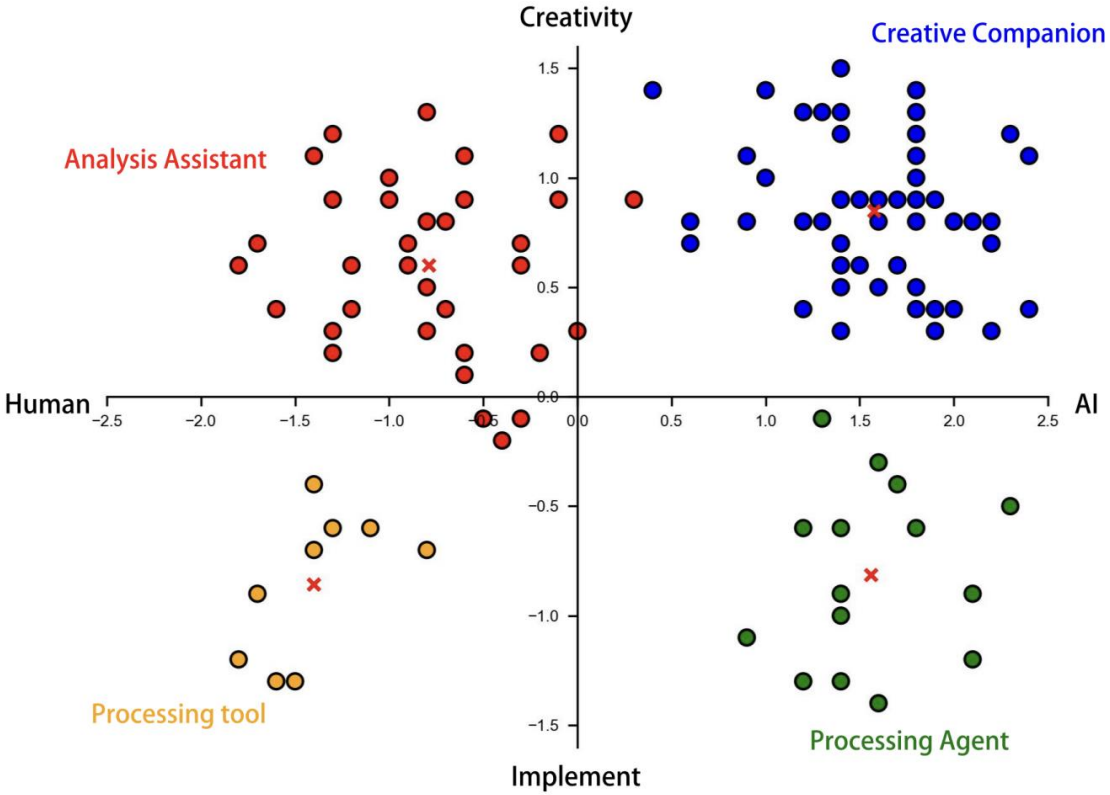
2) Multi-modal LLM for Algorithm Design



1. Yuxiao Huang, Wenjie Zhang, Liang Feng, Xingyu Wu, and Kay Chen Tan. 2024. How multimodal integration boost the performance of llm for optimization: Case study on capacitated vehicle routing problems. arXiv preprint arXiv:2403.01757 (2024).

Future Directions

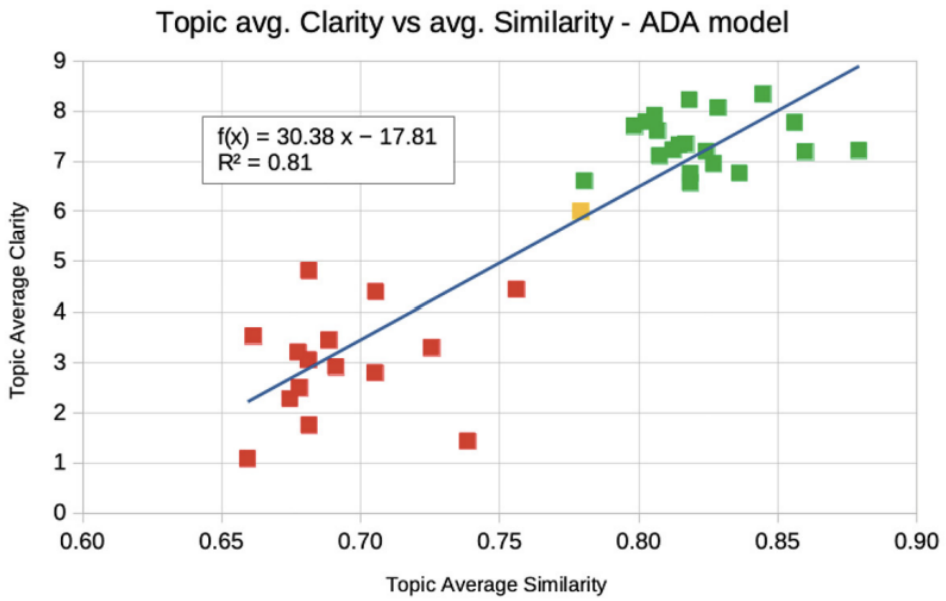
3) Interaction with Human Experts

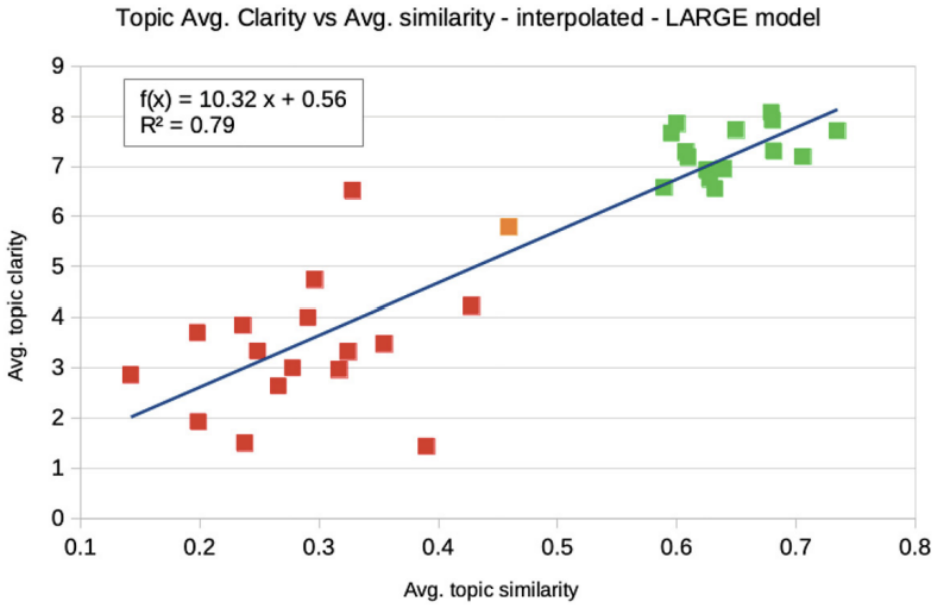


1. Jiayang Li, Jiale Li, and Yunsheng Su. 2024. A Map of Exploring Human Interaction Patterns with LLM: Insights into Collaboration and Creativity. In International Conference on Human-Computer Interaction. Springer, 60–85.

Future Directions

4) LLM-based Algorithm Assessment

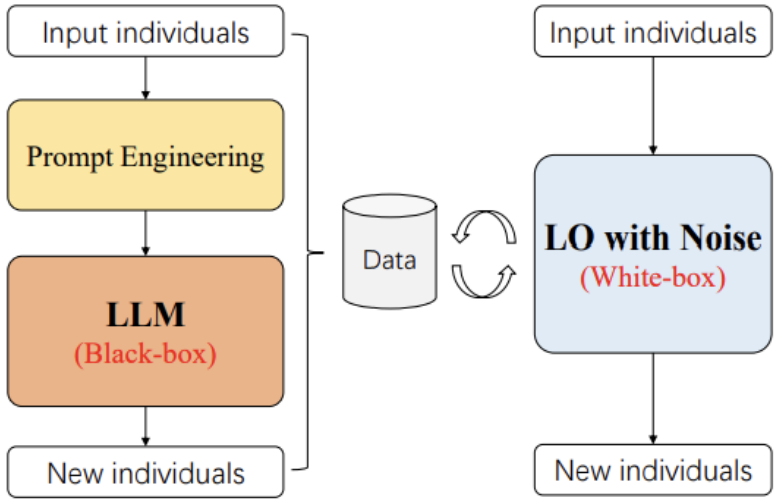




1. Andrea Sterbini and Marco Temperini. 2024. Automated Analysis of Algorithm Descriptions Quality, Through Large Language Models. In International Conference on Intel ligent Tutoring Systems. Springer, 258–271.

Future Directions

5) Understand the Behavior of LLM



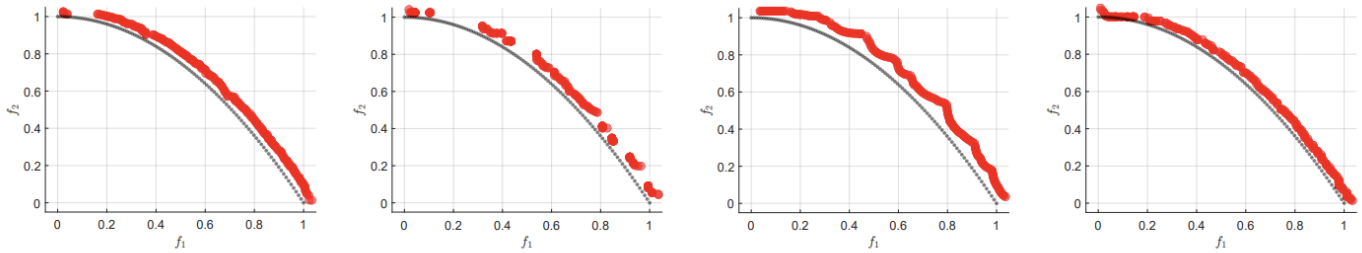


Fig. 9. Approximated PFs on UF4 instance: (a) NSGA-II, (b) MOEA/D, (c) MOEA/D-DE, and (d) MOEA/D-LO.

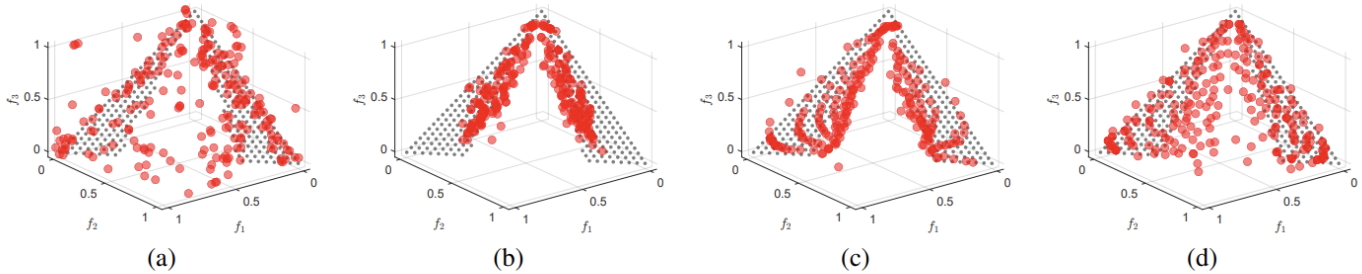
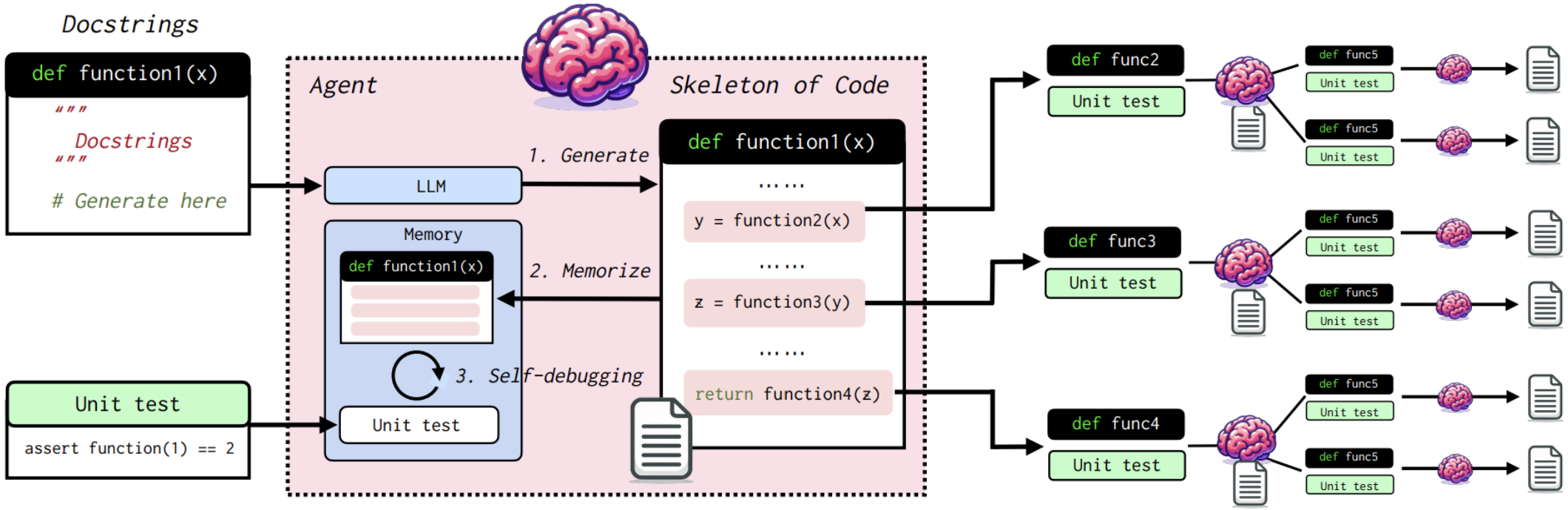


Fig. 10. Approximated PFs on UF9 instance: (a) NSGA-II, (b) MOEA/D, (c) MOEA/D-DE, and (d) MOEA/D-LO.

1. Fei Liu, Xi Lin, Zhenkun Wang, Shunyu Yao, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. 2023. Large Language Model for Multi-objective Evolutionary Optimization. arXiv preprint arXiv:2310.12541 (2023).

Future Directions

5) Fully Automatic Algorithm Design



1. Ishibashi, Yoichi, and Yoshimasa Nishimura. "Self-organized agents: A llm multi-agent framework toward ultra large-scale code generation and optimization." arXiv preprint arXiv:2404.02183 (2024).

Future Directions

6) LLM4AD Benchmarks

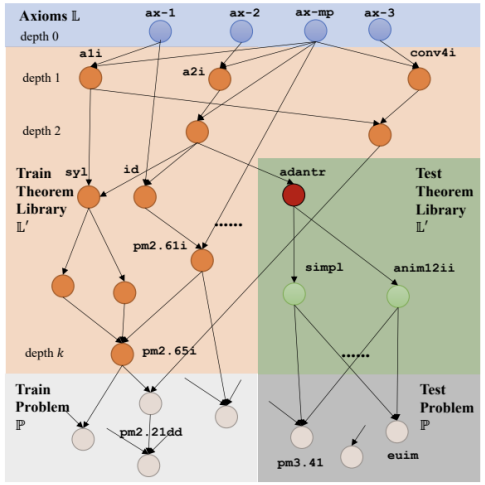


Figure 4: The construction process of the ATG benchmark. Each node is a theorem in "set.mm" and edges represent if a node refers to another in its proof. We assign each node a depth and use it to split the theorem library $\mathbb{L}'$ and $\mathbb{P}$. Lastly, we select the red node and use all its successor nodes for testing.

Dataset	Axioms & Hypotheses	Theo-rems	Set Type	Split Depth	Theorem Library	Problem Set
wb	83	272	train	10	82	32
			test	20	54	21
wif	247	1284	train	33	518	220
			test	39	211	88
minimp	373	2048	train	36	754	298
			test	40	441	182

Table 1: Statistics of proposed ATG datasets.

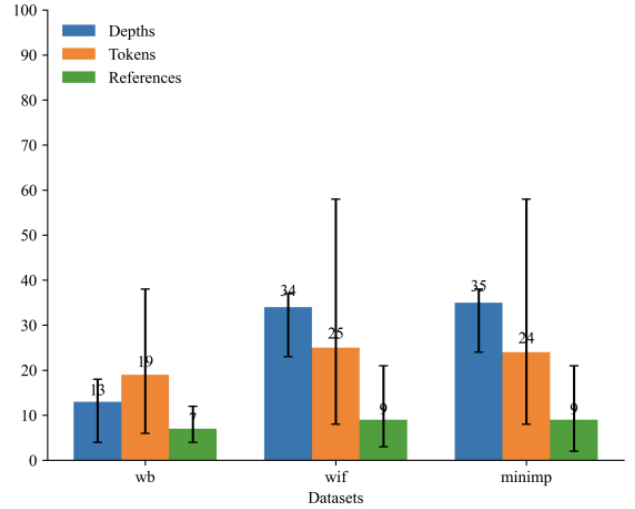


Figure 5: Statistics of proof depth, tokens, and referred theorems of test theorem library.

1. Xiaohan Lin, Qingxing Cao, Yinya Huang, Zhicheng Yang, Zhengying Liu, Zhenguo Li, and Xiaodan Liang. 2024. ATG: Benchmarking Automated Theorem Generation for Generative Language Models. arXiv preprint arXiv:2405.06677 (2024).

Conclusion

- We have reviewed large language model for algorithm design **(LLM4AD)**
- We have proposed a new algorithm design paradigm combining evolutionary computation and large language model, named **Evolution of Heuristics (EoH)**
- We have open-sourced a **Python-based LLM4AD platform**
- We have highlighted some future research directions