Monte Carlo Tree Search for Comprehensive Exploration in LLM-Based Automatic Heuristic Design

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, Bryan Hooi

Homepage: https://zz1358m.github.io/zhizheng.github.io/

National University of Singapore

Paper Code: https://github.com/zz1358m/MCTS-AHD-master/tree/main

Jun. 2th 2025

Outline

- **Background: Combinatorial Optimization**
- LLM for Combinatorial Optimization
- Our Method: MCTS-AHD
- Experiment & Discussion

Combinatorial Optimization

$$\begin{aligned} \min \quad & f(\boldsymbol{x}), \\ s.t. \quad & g(\boldsymbol{x}) \geq 0, \\ & \boldsymbol{x} \in \mathcal{Z}. \end{aligned}$$

- non-differentiable
- non-enumerable
- often NP-hard

- It is a subfield of mathematical optimization;
- The variables are **discrete**, and the decision space is finite;
- The number of feasible solutions increases **exponentially**;
- The optimal solution always exists but is hard to obtain in **polynomial running time**;
- It has **important applications** in many practical scenarios, such as logistics, supply chain management, production planning, facility location and layout, portfolio optimization, drug discovery telecommunications network design, and chip design.

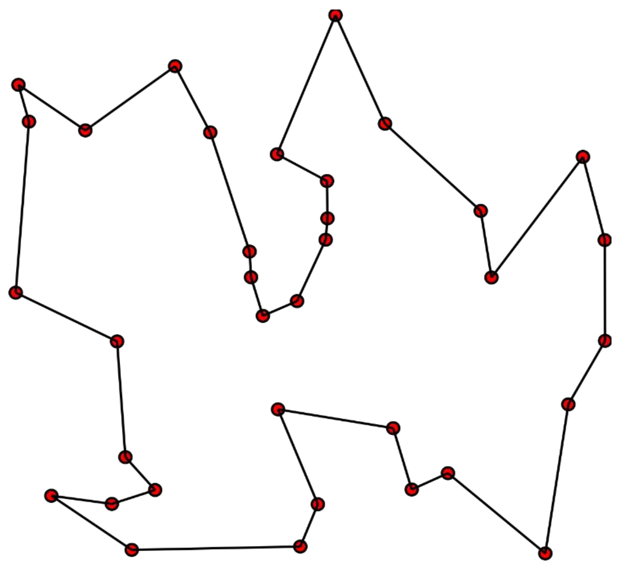
[1] Korte, Bernhard H., et al. Combinatorial optimization. Vol. 1. Heidelberg: Springer, 2011.

[2] Bengio, Yoshua, Andrea Lodi, and Antoine Prouvost. "Machine learning for combinatorial optimization: a methodological tour d'horizon." European Journal of Operational Research 290.2 (2021): 405-421.

Combinatorial Optimization

Travelling Salesman Problem (TSP):

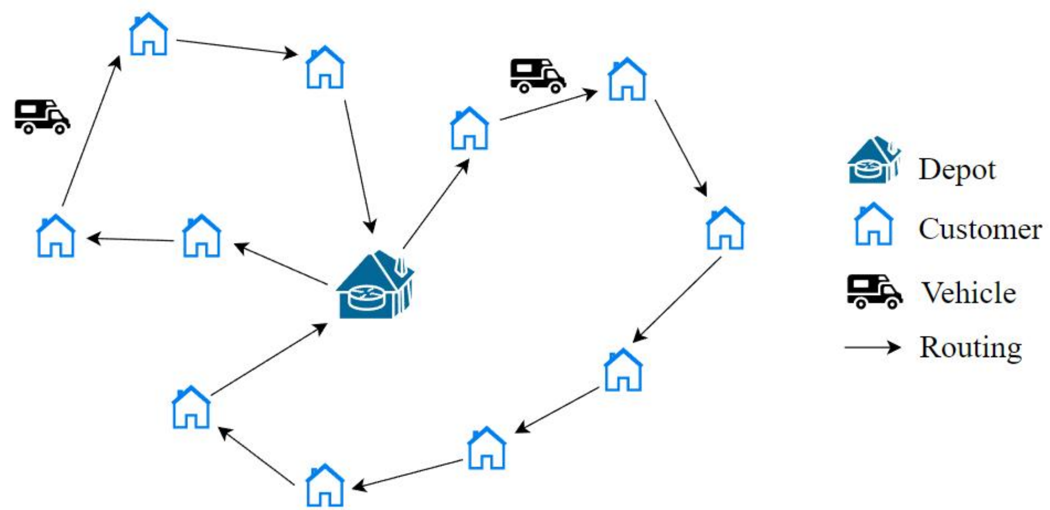
It aims to find the shortest route that visits all the given n nodes exactly once and returns to the origin node. (NP-hard)



A TSP example (also called **instance**) with the optimal solution

Capacitated Vehicle Routing Problem (CVRP):

Vehicles have a limited carrying capacity for the goods that must be delivered. It aims to find the optimal set of routes for a fleet of vehicles in order to deliver to a given set of customers with the lowest cost (e.g., length). (NP-hard)



A CVRP example (also called **instance**) with the optimal solution

[1] Korte, Bernhard H., et al. Combinatorial optimization. Vol. 1. Heidelberg: Springer, 2011.

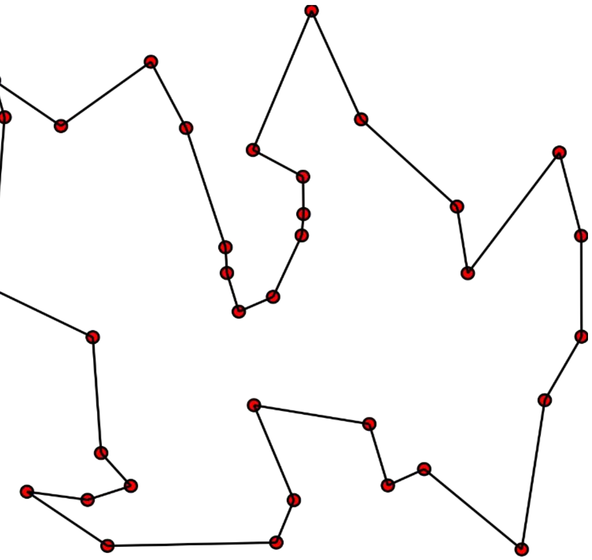
[2] Travelling salesman problem - Wikipedia, https://en.wikipedia.org/wiki/Travelling_salesman_problem

Solving Combinatorial Optimization

Exact Algorithms:

- Brute-force search: $\boldsymbol{O}(n!)$
- Dynamic programming: $\boldsymbol{O}(n^2 2^n)$
- Branch and Bound: $\boldsymbol{O}(2^n)$

Optimal but
Super time-sonsuming



A TSP example (also called **instance**) with the optimal solution

Approximation Algorithms (Heuristics):

- Christofides Algorithm:
- Greedy Algorithm:
- Evolutionary Algorithm:
- Ant Colony Optimization (ACO):

Efficient but
Need manpower to design parameters and workflows

Neural Combinatorial Optimization (NCO):

- Learning to Construct (**L2C**):
  Using NNs to construct solutions from scratch
- Learning to Improve (**L2I**):
  Using NNs to iteratively improve feasible solutions

Super-Efficient but
Performs bad especially on Out-of-domain problems and instances

[1] Bengio, Yoshua, Andrea Lodi, and Antoine Prouvost. "Machine learning for combinatorial optimization: a methodological tour d'horizon." European Journal of Operational Research 290.2 (2021): 405-421.

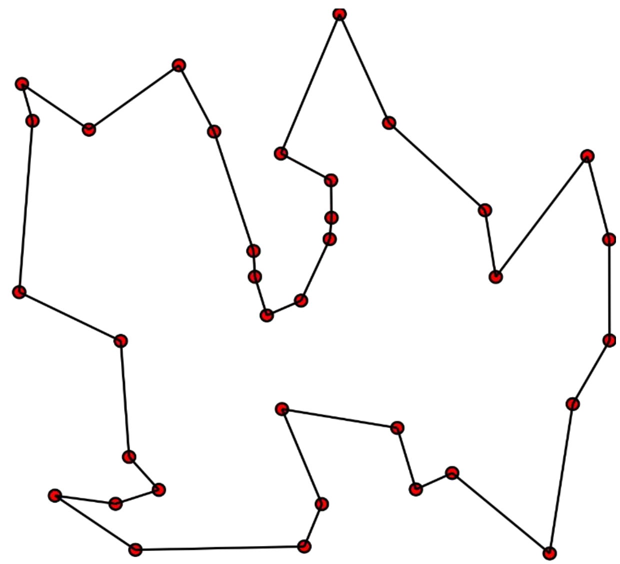
Outline

- ➢Background: Combinatorial Optimization
- ➢**LLM for Combinatorial Optimization**
- ➢Our Method: MCTS-AHD
- ➢Experiment & Discussion

LLM for Combinatorial Optimization

Directly using LLM to plan [1]:

- Input node coordinates to LLMs
- Let the LLM to plan solutions and gradually update the solutions.



A TSP example (also called **instance**) with the optimal solution

LLM-based Automatic Heuristic Design [2]:

Approximation Algorithms (Heuristics):

- Christofides Algorithm:
- Greedy Algorithm:
- Evolutionary Algorithm:
- Ant Colony Optimization (ACO):

Efficient but Need manpower to design parameters and workflows

- Select a heuristic shown in the page before and using LLMs to initallize gradually update the heuristics, taking the original ones as the starting nodes.

Methods	LEMA*	OPRO*	MCTS-AHD(step-by-step construction)
TSP20	3.94%	4.40%	7.71%
TSP50	-	133.00%	11.82%

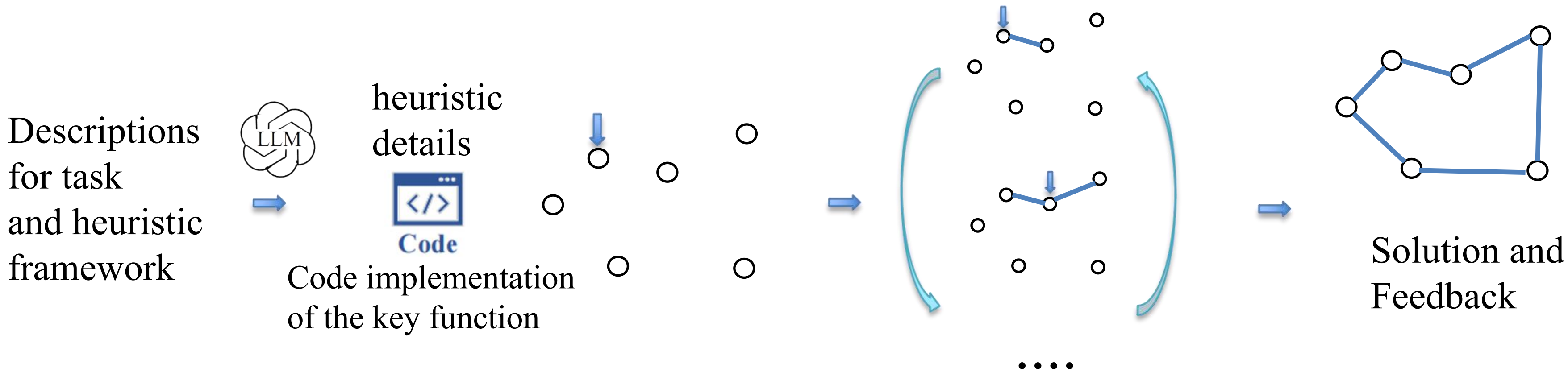
[1] Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., and Chen, X. Large language models as optimizers, 2024. URL https://arxiv.org/abs/2309.03409.
[2] Liu, F., Tong, X., Yuan, M., and Zhang, Q. Algorithm evolution using large language model. arXiv preprint arXiv:2311.15249, 2023, ICML2024.

LLM for Combinatorial Optimization

LLM-based Automatic Heuristic Design [2]:

- Select a heuristic shown in the page before and using LLMs to initallize greadually update the heuristics, taking the original ones as the starting nodes.

Design greedy heuristics for solving a TSP instance



LLM-based AHD can be applied to any problem and heursitic.
It can also generate powerful heuristics using the pre-trained knolwedge in LLMs.

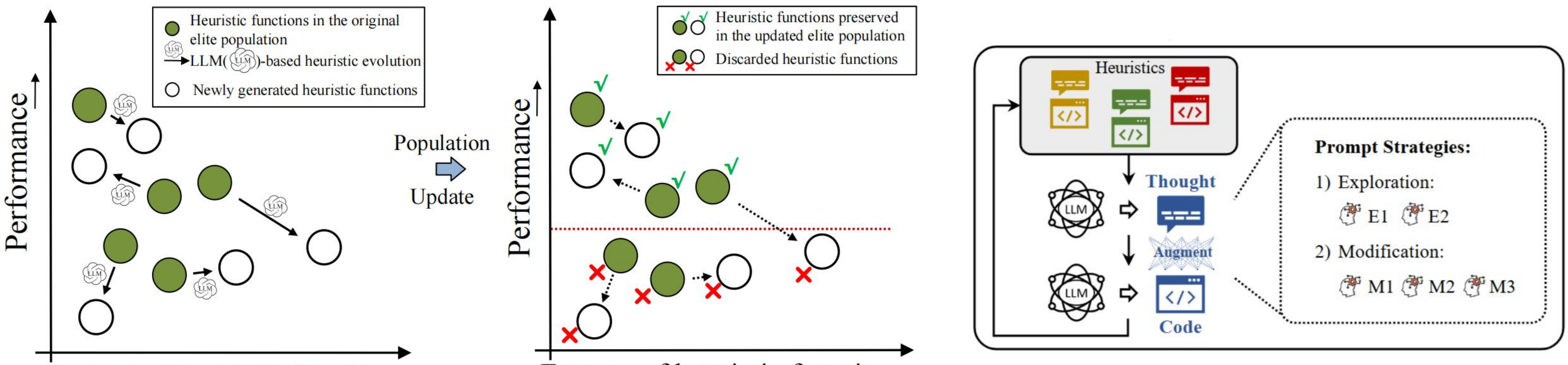
[1] Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., and Chen, X. Large language models as optimizers, 2024. URL https://arxiv.org/abs/2309.03409.
[2] Liu, F., Tong, X., Yuan, M., and Zhang, Q. Algorithm evolution using large language model. arXiv preprint arXiv:2311.15249, 2023, ICML2024.

LLM for Combinatorial Optimization

LLM-based Automatic Heuristic Design [2]:

- Select a heuristic shown in the page before and using LLMs to initallize greadually update the heuristics, taking the original ones as the starting nodes.

EoH [1] Designs to maintain a population of code and its correspoing designing idea for heurisitc evolution.



(c) Evolution of both thoughts and codes (EoH, ours)

[1] Liu, F., Tong, X., Yuan, M., and Zhang, Q. Algorithm evolution using large language model. arXiv preprint arXiv:2311.15249, 2023, ICML2024.

Outline

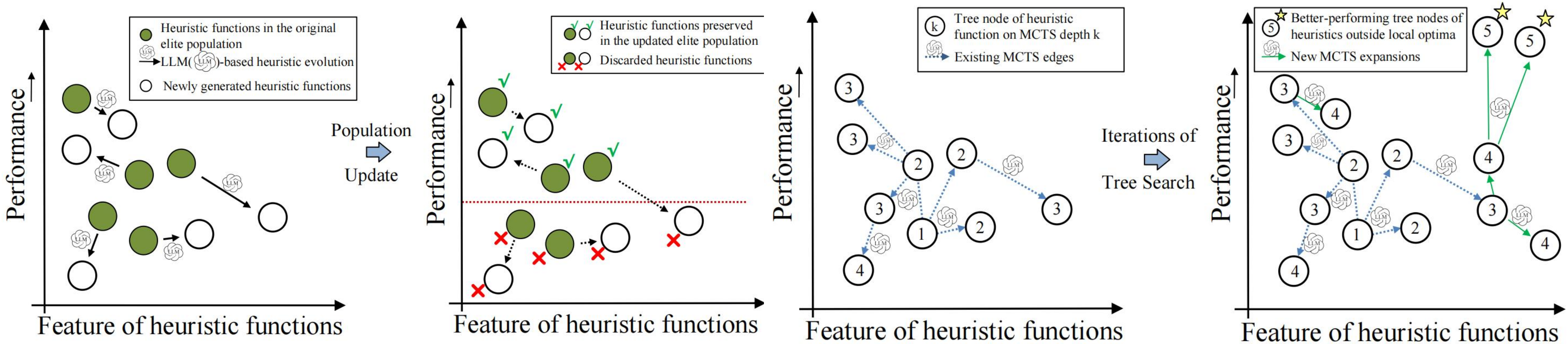
- Background: Combinatorial Optimization
- LLM for Combinatorial Optimization
- **Our Method: MCTS-AHD**
- Experiment & Discussion

Our Method: MCTS-AHD

Motivation

All the existing methods like EoH [1], Funsearch [2] and ReEvo [3] adopt a population-based method.

Population-based methods will make fast convergence but these methods are difficult in comprehensive exploration in the space of all possible heuristics.

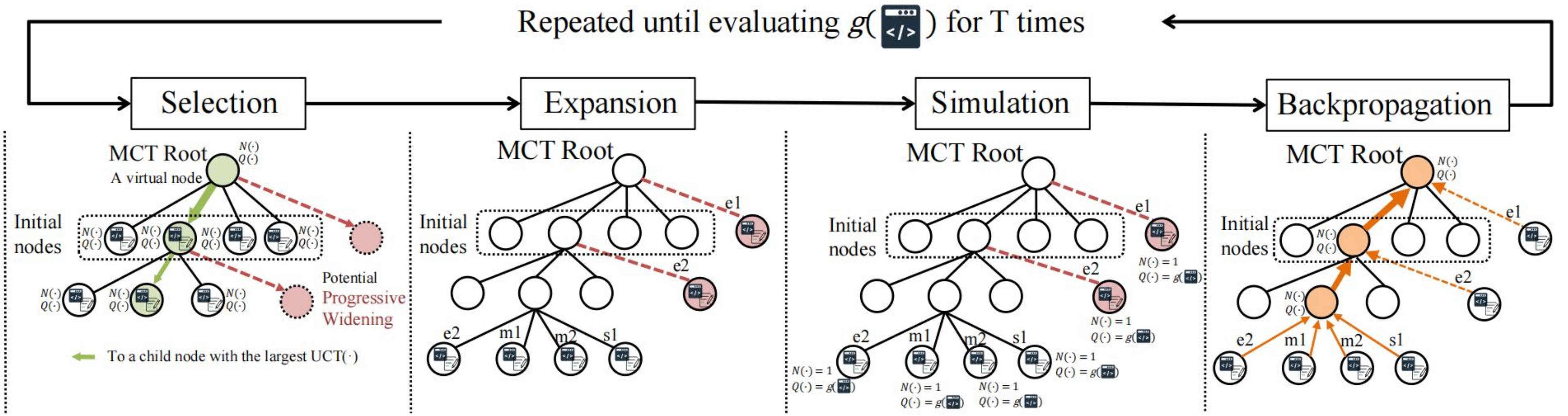


This paper propose to use MCTS as a better structure for heuristic evolution.
MCTS enable multi-step heuristic evolution for comprehensive exploration.

[1] Liu, F., Tong, X., Yuan, M., and Zhang, Q. Algorithm evolution using large language model. arXiv preprint arXiv:2311.15249, 2023, ICML2024.
[2] Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J., Ellenberg, J. S., Wang, P., Fawzi, O., et al. Mathematical discoveries from program search with large language models. Nature, 625 (7995):468–475, 2024.
[3] Ye, H., Wang, J., Cao, Z., Berto, F., Hua, C., Kim, H., Park, J., and Song, G. Reevo: Large language models as hyper-heuristics with reflective evolution. arXiv preprint arXiv:2402.01145, 2024a.

Our Method: MCTS-AHD

How is the MCTS process of MCTS-AHD?



Selection, Expanion, Simulation, and Backpropagation are general steps for MCTS.

Selection is based on the UCT function.

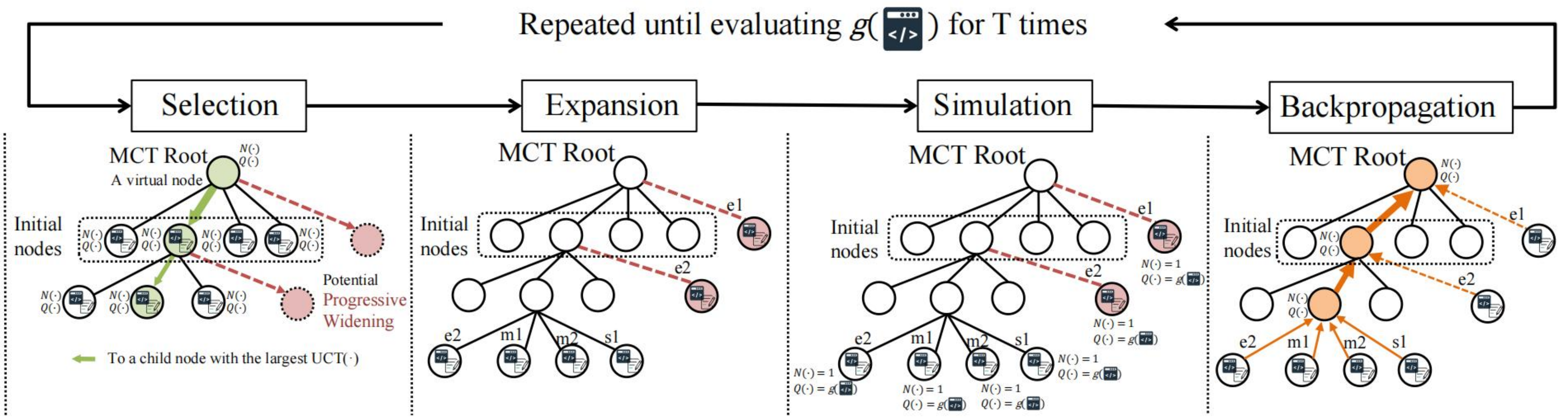
$$\mathrm{UCT}(c) = \left(\frac{Q(c) - q_{min}}{q_{max} - q_{min}} + \lambda \cdot \sqrt{\frac{\ln(N(n_c)+1)}{N(c)}} \right), \tag{5}$$

Expansion in MCTS-AHD is done by LLMs. It involves several prompt strategies:

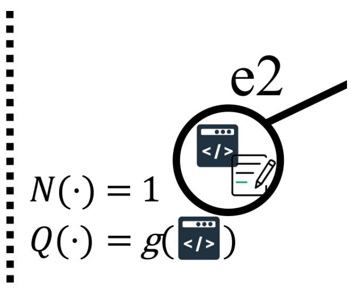
i1, e1, e2, m1, m2, s1.

Our Method: MCTS-AHD

How is the MCTS process of MCTS-AHD?



Simulation assess the performance of leaves.



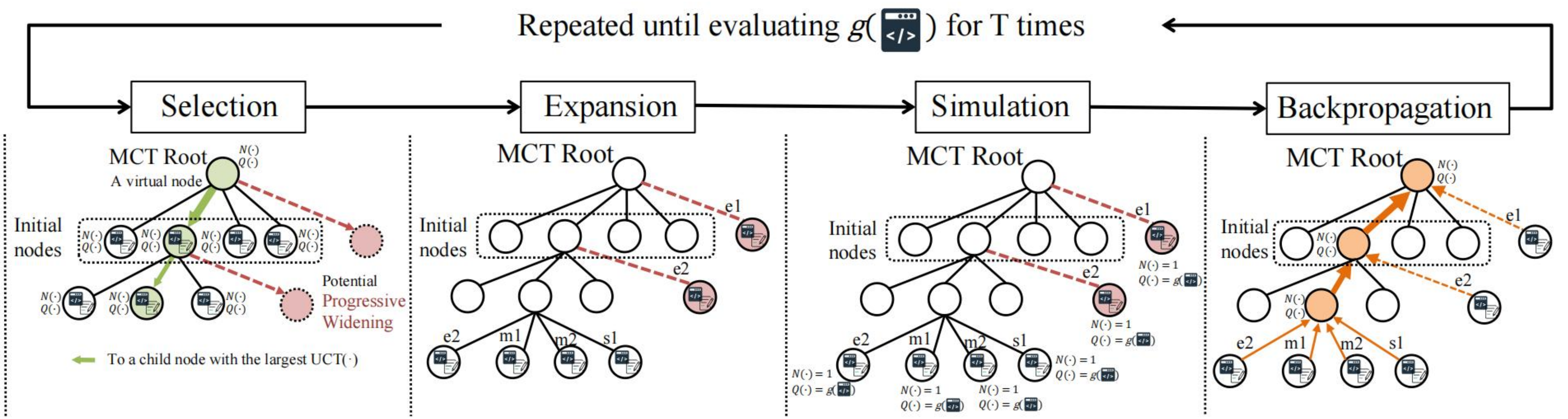
Backpropagation updates the Q and N value of nodes by:

$$Q(n_c) \leftarrow \max_{c \in \text{Children}(n_c)} Q(c),$$

$$N(n_c) \leftarrow \sum_{c \in \text{Children}(n_c)} N(c).$$

Our Method: MCTS-AHD

How is the MCTS process of MCTS-AHD?



We also involve ***Progressive Widening***, to cope with the dynamic development of the heuristic space.

Codition:

$$\lfloor N(n)^{\alpha} \rfloor \geq |\text{children}(n_c)|,$$

We design ***Exploration-Decay*** to improve the convergence of heuristic evaluation in the final steps.

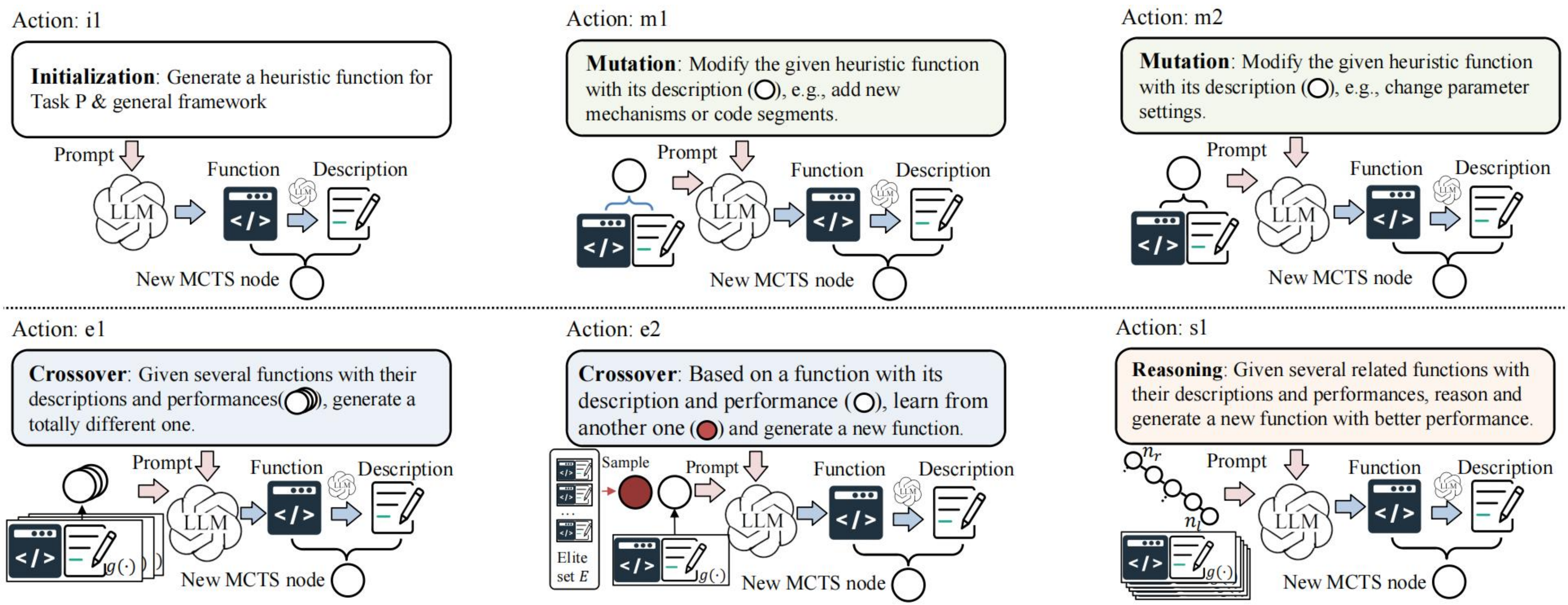
$$\mathrm{UCT}(c) = \left(\frac{Q(c) - q_{min}}{q_{max} - q_{min}} + \lambda \cdot \sqrt{\frac{\ln(N(n_c) + 1)}{N(c)}} \right), \quad (5)$$

$$\lambda = \lambda_0 * \frac{T - t}{T}.$$

Our Method: MCTS-AHD

How does MCTS-AHD make node expansion?

Adopted from [1] where LLM are used to simulate crossover and mutation with several prompt strategies.
We design six prompt strategies for LLM-based heuristic modification.



[1] Liu, F., Tong, X., Yuan, M., and Zhang, Q. Algorithm evolution using large language model. arXiv preprint arXiv:2311.15249, 2023, ICML2024.

Outline

- ➢Background: Combinatorial Optimization
- ➢LLM for Combinatorial Optimization
- ➢Our Method: MCTS-AHD
- ➢**Experiment & Discussion**

Experiment & Discussion

Experiment

To show the effectiveness and the wide application of MCTS-AHD, we condsider 16 heuristic evolution senarios.

NP-hard CO Problems as Tasks

- Step-by-step construction framework:
  - Travelling Salesman Problem (TSP)
  - TSP-copy for a reference on simultaneous heuristic function evaluations
  - 0-1 Knapsack (KP)
  - Online Bin Packing Problem (Online BPP) **(Please set max_fe = 2000 for re-implementing the report results for Online BPP)**
  - Admissible Set Problem (ASP)
- Ant Colony Optimization (ACO) **(Please set init_pop_size = 10 in re-implementing the report results for Black-box settings)**:
  - TSP and Black-box settings
  - Capacitated Vehicle Routing Problem (CVRP) and Black-box settings
  - Multiple Knapsack Problem (MKP) and Black-box settings
  - Offline Bin Packing Problem (Offline BPP) and Black-box settings
- Guided Local Search:
  - (Large-scale) TSP

Experiment & Discussion

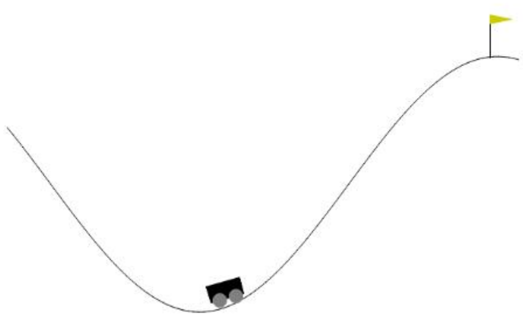
Experiment

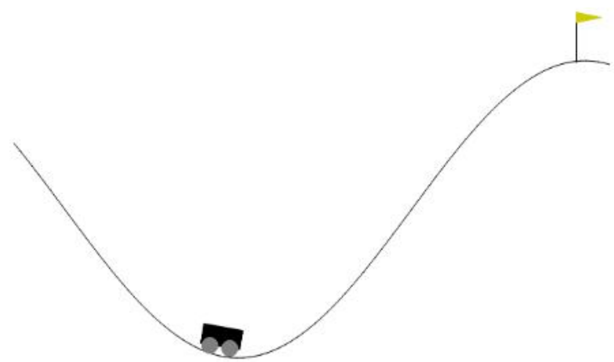
To show the effectiveness and the wide application of MCTS-AHD, we condsider 16 heuristic evolution senarios.

Other Complex Tasks

- Bayesian Optimization (BO):
  - Cost-aware Function Design in Active Learning **(Please set botorch according to the requirements.txt for the report results)**

And a ***mountain_car*** optimization problem.





Experiment & Discussion

Experiment Results

MCTS-AHD can get significantly better results on nearly all of these senarios.

Table 1. Designing heuristics with the step-by-step construction framework for TSP and KP. We evaluate methods on 6 test sets with 1,000 instances each. Test sets with in-domain scales (i.i.d. to the evaluation dataset D) are underlined. Since AHD methods have no guarantees for generalization ability, the effect on in-domain datasets is more important. Optimal for TSP is obtained by LKH (Lin & Kernighan, 1973), and Optimal for KP is the result of OR-Tools. Each LLM-based AHD method is run three times and we report the average performances. The best-performing method with each LLM is shaded, and each test set's overall best result is in bold.

Task	TSP						KP					
N=	<u>N=50</u>		N=100		N=200		N=50, W=12.5		<u>N=100, W=25</u>		N=200, W=25	
Methods	Obj.↓	Gap	Obj.↓	Gap	Obj.↓	Gap	Obj.↑	Gap	Obj.↑	Gap	Obj.↑	Gap
Optimal	5.675	-	7.768	-	10.659	-	20.037	-	40.271	-	57.448	-
Greedy Construct	6.959	22.62%	9.706	24.94%	13.461	26.29%	19.985	0.26%	40.225	0.12%	57.395	0.09%
POMO	**5.697**	**0.39%**	**8.001**	**3.01%**	12.897	20.45%	19.612	2.12%	39.676	1.48%	57.271	0.09%
LLM-based AHD: *GPT-3.5-turbo*												
Funsearch	6.683	17.75%	9.240	18.95%	12.808	19.61%	19.985	0.26%	40.225	0.12%	57.395	0.09%
EoH	6.390	12.59%	8.930	14.96%	12.538	17.63%	19.994	0.21%	40.231	0.10%	57.400	0.08%
MCTS-AHD(Ours)	6.346	11.82%	8.861	14.08%	12.418	16.51%	19.997	0.20%	40.233	0.09%	57.393	0.10%
LLM-based AHD: *GPT-4o-mini*												
Funsearch	6.357	12.00%	8.850	13.93%	12.372	15.54%	19.988	0.24%	40.227	0.11%	57.398	0.09%
EoH	6.394	12.67%	8.894	14.49%	12.437	16.68%	19.993	0.22%	40.231	0.10%	57.399	0.09%
MCTS-AHD(Ours)	6.225	9.69%	8.684	11.79%	**12.120**	**13.71%**	**20.015**	**0.11%**	**40.252**	**0.05%**	**57.423**	**0.04%**

Experiment & Discussion

Experiment Results

MCTS-AHD can get significantly better results on nearly all of these senarios.

Table 4. Designing CAFs for BO. The table shows the gaps to optimal when running BO on instances with manually designed CAFs and CAFs designed by LLM-based AHD methods. LLM-based AHD methods are run three times for the average gaps. In testing, the evaluation budgets for BO are 30 and we run 10 trials for average gaps. The results of EI, EIpu, and EI-cool are from Yao et al. (2024c).

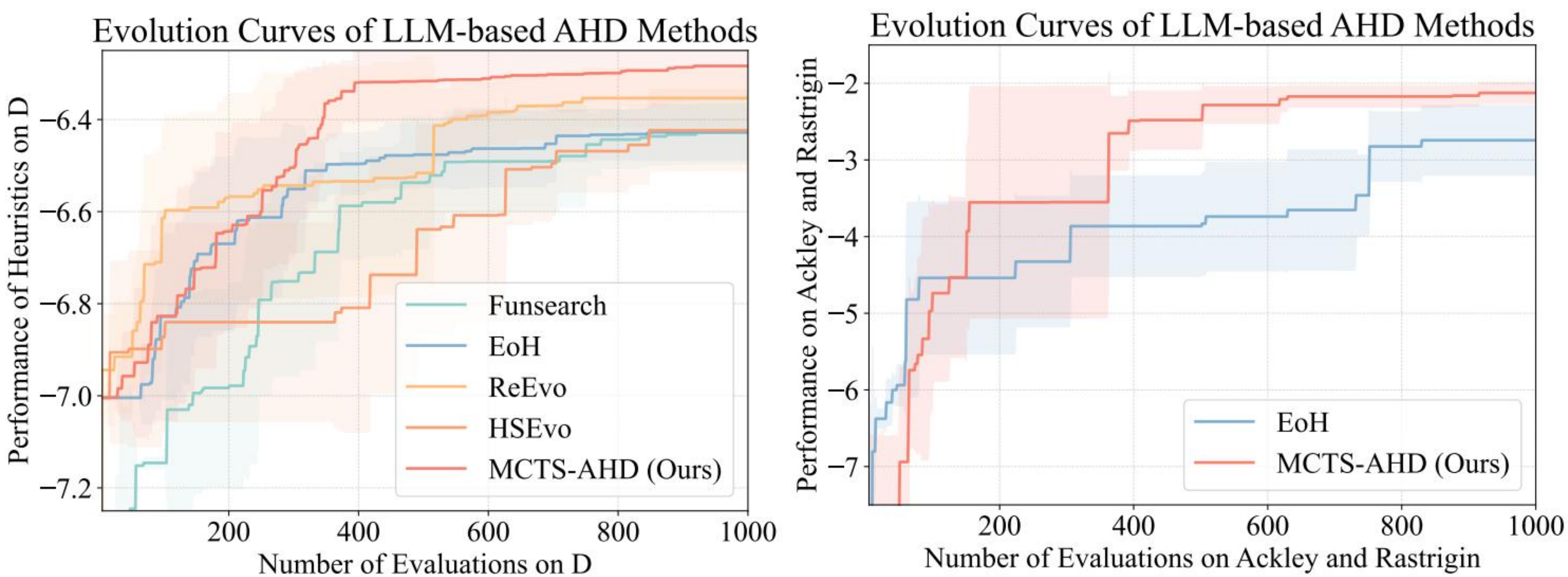
Instances	Ackley	Rastrigin	Griewank	Rosenbrock	Levy	ThreeHumpCamel	StyblinskiTang	Hartmann	Powell	Shekel	Hartmann	Cosine8
EI	2.66%	4.74%	0.49%	**1.26%**	**0.01%**	0.05%	0.03%	0.00%	18.89%	7.91%	0.03%	0.47%
EIpu	2.33%	5.62%	0.34%	2.36%	0.01%	0.12%	**0.02%**	0.00%	19.83%	7.92%	0.03%	0.47%
EI-cool	2.74%	5.78%	**0.34%**	2.29%	0.01%	0.07%	0.03%	**0.00%**	14.95%	8.21%	**0.03%**	0.54%
LLM-based AHD: *GPT-4o-mini*												
EoH	2.45%	0.90%	0.54%	56.78%	0.20%	0.26%	0.79%	0.04%	70.89%	4.56%	0.33%	0.36%
MCTS-AHD	**2.40%**	**0.77%**	0.36%	1.68%	0.01%	**0.02%**	0.20%	0.01%	**1.27%**	**3.94%**	0.38%	**0.34%**

CAF: Cost-Aware Acquisition Function

Experiment & Discussion

Experiment Results

Evaluation curves show that MCTS-AHD can promote comprehensive exploration without losing convergence speed.



(a) Design Step-by-step Construction heuristics for TSP

(b) Design CAFs for BO

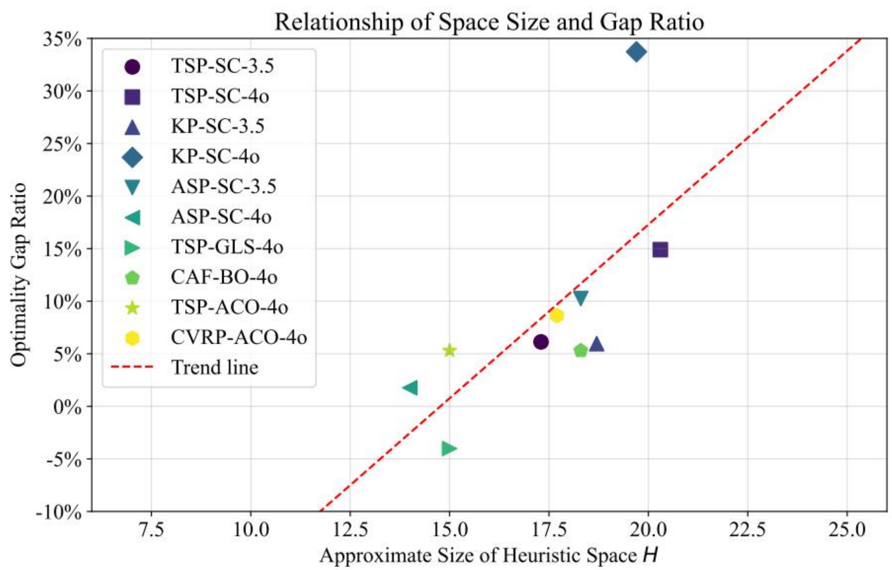
Experiment & Discussion

What is the advantage scope of MCTS-AHD, compared to existing population-based methods

- The relation between performance and the complexity of heuristic space.

Any heuristic function can be expressed in the weighted-sum form of sub-functions as follows:

$$a_1 f_1(x) + a_2 f_2(x) + a_3 f_3(x) + \ldots + a_n f_n(x)$$



MCTS-AHD performs better in ***application scenarios with more complex heuristic spaces.***

Experiment & Discussion

What is the advantage scope of MCTS-AHD, compared to existing population-based methods

- The relation between performance and the amount of descriptions.

ReEvo propose to consider ***black-box optimization problems*** in LLM-based AHD.

Table 16. Implementing MCTS-AHD on Black-box CO tasks with ACO general frameworks. We follow the settings of Ye et al. (2024a) in heuristic evolution and run each LLM-based method three times for average performance. The white-box results are the same as Table 2.

	TSP	CVRP	MKP	Offline BPP
N=	N=50	N=50, C=50	N=100,m=5	N=500,C=150
Methods	Obj.↓	Obj.↓	Obj.↑	Obj.↓
ACO	5.992	11.355	22.738	208.828
DeepACO	5.842	8.888	23.093	203.125
White-box Setting: *GPT-4o-mini*				
EoH	5.828	9.359	23.139	204.646
ReEvo	5.856	9.327	23.245	206.693
MCTS-AHD(Ours)	5.801	9.286	23.269	204.094
Black-box Setting: *GPT-4o-mini*				
EoH	5.831	9.401	23.240	204.615
ReEvo	5.860	9.404	23.196	206.021
MCTS-AHD(Ours)	5.830	9.444	23.191	205.375

MCTS-AHD performs better in ***application scenarios with more descriptions.***

Thanks so much for Listening

Zhi Zheng 郑执

zhi.zheng@u.nus.edu