HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Tìm kiếm trên cây Monte Carlo để khám phá toàn diện trong thiết kế heuristic tự động dựa trên LLM (MCTS-AHD)

Nhóm 7 – Tính toán tiến hoá

Giảng viên hướng dẫn PGS.TS Huỳnh Thị Thanh Bình

Sinh viên thực hiện Hoàng Anh Quyền - 20224893
Đặng Hiếu Nguyên – 20224998
Nguyễn Tiến Mạnh – 20210570
Phan Thanh Thái - 20224991

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Phần I: Giới thiệu bài báo

ONE LOVE. ONE FUTURE.

Mục đích nghiên cứu

- **Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, Bryan Hooi, "Monte Carlo Tree Search for Comprehensive Exploration in LLM-Based Automatic Heuristic Design,". arXiv:2501.08603v3 [cs.AI], 31 Jan 2025.**
- Heuristic trong tối ưu hóa
  - Giải NP-hard hiệu quả trong thực tế
  - Ví dụ: TSP, Scheduling, Routing
- Vấn đề
  - Thiết kế heuristic thủ công:
    + Phụ thuộc chuyên gia
    + Khó tổng quát
    + Tốn thời gian

➔ Cần tự động hóa thiết kế heuristic

Các nghiên cứu trước (Related Work)

- Population-based LLM-AHD

- Funsearch, EoH, ReEvo, HSEvo
- Duy trì population top-k heuristic
- Dễ rơi vào local optimum
- Giới hạn khả năng khám phá toàn diện

- Neural Combinatorial Optimization (NCO)

- POMO, DeepACO
- Cần task-specific training
- Không tổng quát cho mọi framework

- Manual Heuristics

- TSP: Nearest-greedy
- Knapsack: Largest value-weight ratio
- Tốn công, khó mở rộng

→ Trước MCTS-AHD, các phương pháp này có hạn chế về **khám phá, tính tổng quát, hoặc phụ thuộc dữ liệu**.

Cải tiến của paper này

- MCTS-AHD cải tiến

- Tree structure: giữ toàn bộ lịch sử heuristic → tránh bỏ heuristic kém tạm thời
- LLM-based actions đặc trưng MCTS: i1, m1, m2, e1, e2, s1
- Progressive Widening: mở rộng node theo số lần visit → cân bằng exploration/exploitation
- Thought-alignment: sinh description chính xác → hỗ trợ LLM reasoning
- Exploration-decay: tập trung vào node tốt khi gần hội tụ

- Kết quả nổi bật

- Hiệu năng heuristic vượt population-based và NCO
- Ổn định với NP-hard CO + Bayesian Optimization
- Khám phá heuristic space toàn diện hơn

➢ Tóm lại, paper không chỉ áp dụng MCTS mà còn tối ưu **cấu trúc tree + actions + kỹ thuật tiến hóa**, giúp heuristic tốt hơn và tổng quát hơn.

Automatic Heuristic Design (AHD)

Định nghĩa chính thức

Với bài toán P:

$$h^* = \arg\max_{h \in H} g(h)$$

Trong đó:

- H: không gian heuristic hợp lệ
- $g(h)$: hiệu năng heuristic

Ước lượng hiệu năng

$$g(h) = \mathbb{E}_{ins \in D}[-f(h(ins))]$$

→ AHD không tìm nghiệm trực tiếp, mà tìm **thuật toán giải nghiệm**.

LLM-based AHD

Ý tưởng

- LLM sinh heuristic dưới dạng code
- Đánh giá heuristic trên dataset
- Dùng tiến hóa để cải thiện

Cách làm phổ biến

- Population-based Evolutionary Computation

Population-based LLM-AHD

Quy trình

- Giữ population $\{h_1, \ldots, h_M\}$
- Sinh heuristic mới bằng LLM
- Chỉ giữ heuristic tốt nhất

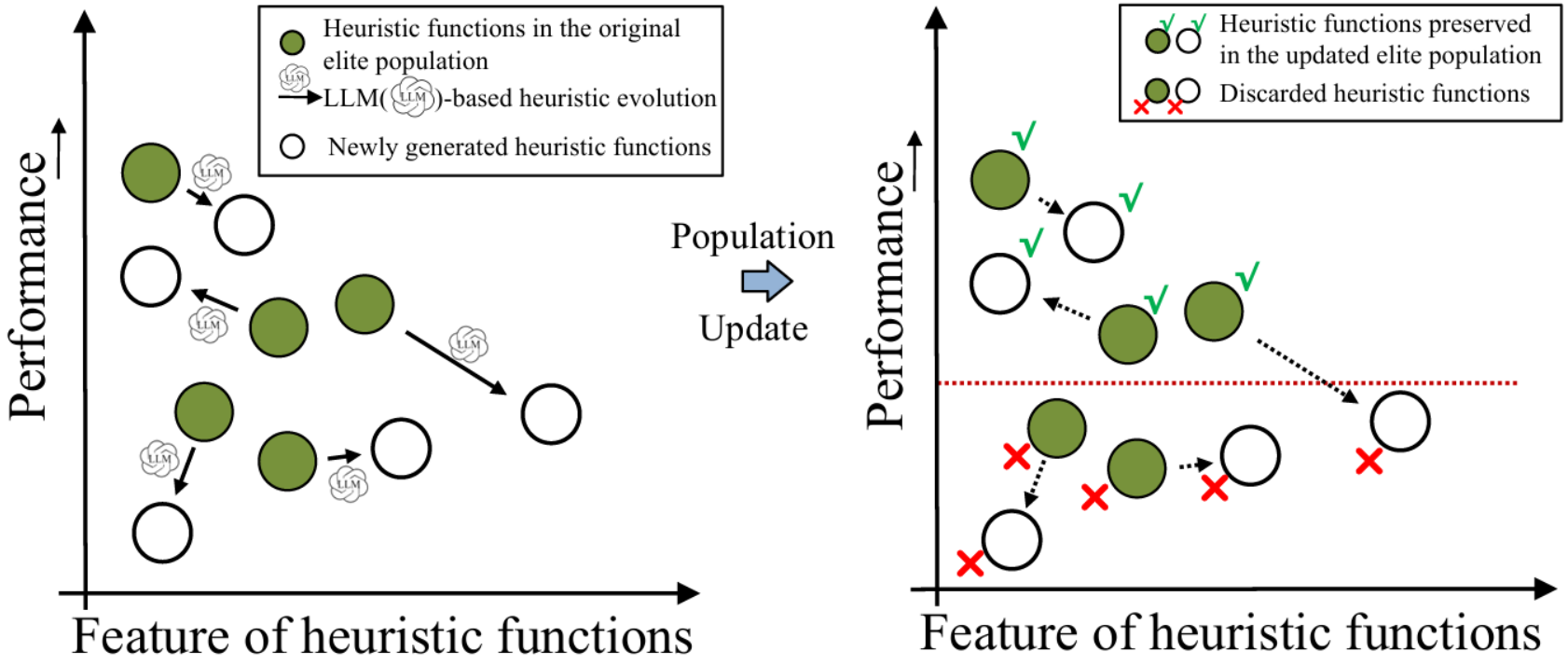
Điều kiện giữ lại

$$g(h_{new}) > \min_i g(h_i)$$

Vấn đề

- Loại bỏ heuristic kém *quá sớm*
- Không cho "worse-before-better"

Population-based LLM-AHD



(a) Populations in LLM-based AHD

Hạn chế cốt lõi của Population

Trực giác

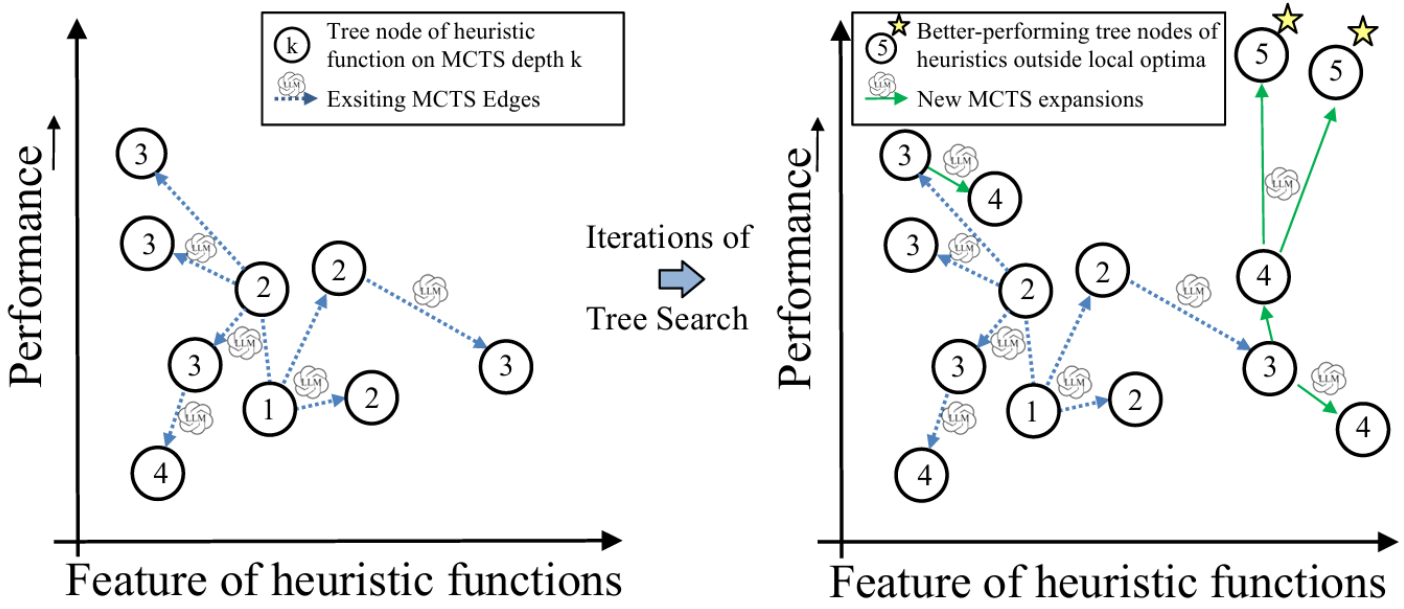
- Không gian heuristic rất lớn
- Heuristic yếu hiện tại có thể trở thành mạnh sau refinement
- Population → **khó duy trì đa dạng**

→ **Dễ kẹt local optimum**

Ý tưởng MCTS-AHD

Thay Population → Tree

- Mỗi node = 1 heuristic
- Không loại bỏ heuristic
- Lưu toàn bộ lịch sử tiến hóa



(b) MCTS for higher-quality LLM-based AHD (Ours)

→ Thay vì hỏi "heuristic nào tốt nhất", ta hỏi "heuristic nào đáng phát triển tiếp".

Cấu trúc cây trong MCTS-AHD

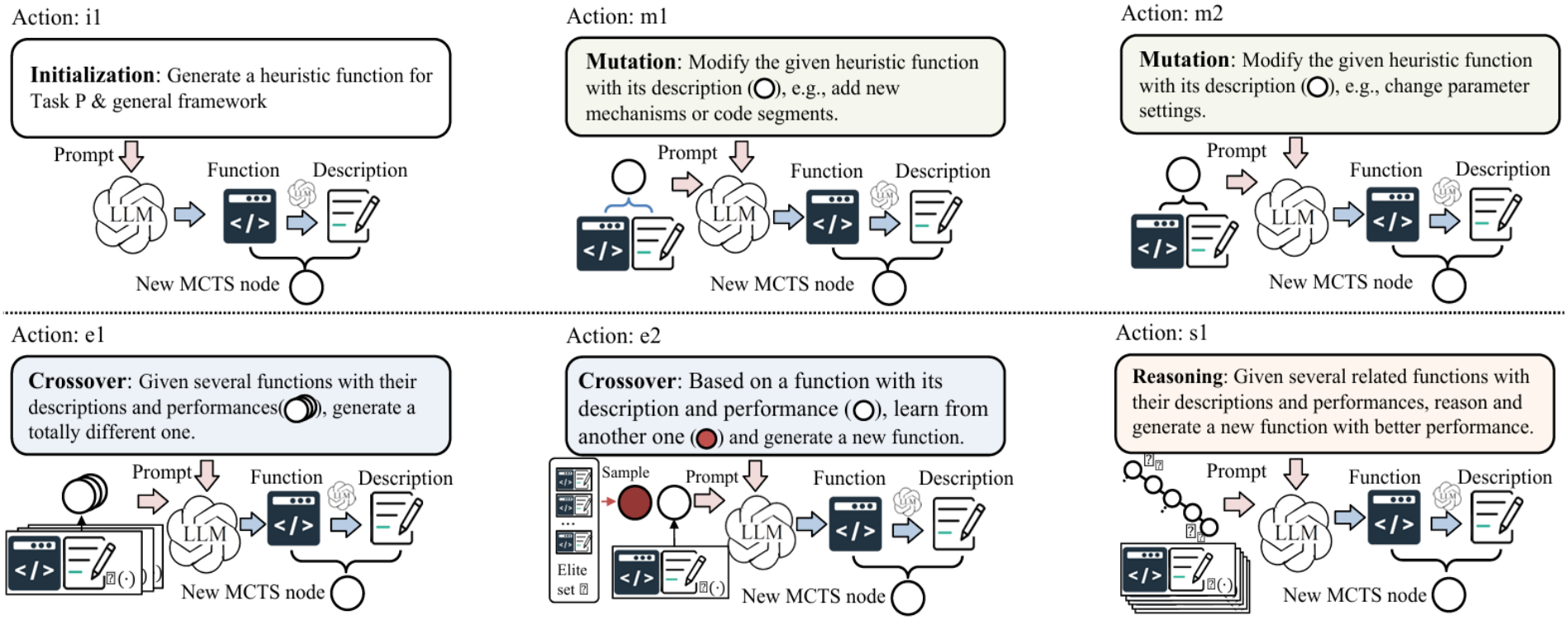
- Root là một **nút ảo**, không chứa heuristic.
- Mỗi node con chứa 1 heuristic:
  - code Python
  - mô tả heuristic
  - $Q(n)$: chất lượng theo hàm đánh giá
  - $N(n)$: số lượt thăm

Cây càng sâu → heuristic càng được chỉnh sửa/hoàn thiện.

8. LLM-based Actions

Các hành động

- **i1**: Khởi tạo heuristic
- **m1**: Thêm cơ chế
- **m2**: Chỉnh tham số
- **e1**: Crossover nhiều heuristic
- **e2**: Học từ elite
- **s1**: Tree-path reasoning



Tree-path Reasoning (Action s1)

Ý tưởng

- Lấy các heuristic trên đường:

$$n_r \rightarrow \cdots \rightarrow n_l$$

- Phân tích ưu điểm chung
- Sinh heuristic mới tốt hơn

→ Đây là lợi thế **chỉ có tree mới làm được**, population không có "ngữ cảnh tiến hóa".

Monte Carlo Tree Search (MCTS)

MCTS gồm 4 bước

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

Monte Carlo Tree Search (MCTS)

Bước 1 — Selection

Từ root, thuật toán chọn node con có **UCT cao nhất** theo công thức:

$$UCT(c) = \frac{Q(c) - q_{min}}{q_{max} - q_{min}} + \lambda \cdot \sqrt{\frac{\ln(N(parent) + 1)}{N(c)}}$$

Trong đó:

- Phần thứ nhất: **Normalized Quality**

  → chuẩn hoá giúp Q của nhiều bài toán khác nhau so sánh được.
- Phần thứ hai: **Exploration Term**

  → ưu tiên các node ít được thăm.
- λ **giảm dần theo thời gian**

  → đầu kỳ khám phá rộng, cuối kỳ tập trung khai thác.

⏩ **Đi đến khi gặp leaf node.**

Exploration Decay

Công thức

$$\lambda_t = \lambda_0 \cdot \frac{T - t}{T}$$

- Đầu: khám phá mạnh
- Cuối: hội tụ

→ Tránh lãng phí tài nguyên về sau

Monte Carlo Tree Search (MCTS)

Bước 2 — Expansion

Tại leaf node, MCTS-AHD dùng LLM để sinh **2k+2 heuristic mới** thông qua 6 action:

3 loại mutation:

- **i1** – tạo heuristic hoàn toàn mới
- **m1** – thêm logic
- **m2** – điều chỉnh tham số

2 loại crossover:

- **e1** – trộn nhiều heuristic từ nhiều nhánh → tạo heuristic mới khác biệt
- **e2** – cải thiện heuristic hiện tại dựa trên elite set

1 loại reasoning:

- **s1** – học từ đường phát triển của heuristic (path từ root đến node hiện tại)

Monte Carlo Tree Search (MCTS)

Bước 3 — Simulation

- Mỗi heuristic mới sinh ra được chạy **thực tế** trên tập dữ liệu bài toán.
- Trả về giá trị:

$$Q(h) = g(h)$$

- N(node) = 1
- Update:
  - elite set (top 10 heuristic)
  - qmin/qmax

⏩ Đây là bước tốn thời gian nhất.

Monte Carlo Tree Search (MCTS)

Bước 4 — Backpropagation

Cập nhật toàn bộ node tổ tiên:

- **Q(node) = max(children Q)**
  → vì mục tiêu là tìm heuristic tốt nhất trong nhánh.
- **N(node) = sum(children N)**
  → phản ánh độ sôi động của nhánh.

⏩ Đây là điểm khiến MCTS-AHD không loại bỏ heuristic nào, mà thay đổi đánh giá dựa trên "tốt nhất trong nhánh".

Thiết lập thực nghiệm và kết quả chính

- Bài toán: TSP, KP, CVRP, BPP,BO
- Framework: Greedy, ACO,GLS
- Baseline:
  - Funsearch
  - EoHRe
  - EvoHS
  - Evo

Task	TSP						KP					
N=	N=50		N=100		N=200		N=50, W=12.5		N=100, W=25		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%**

Ablation Study

Loại bỏ

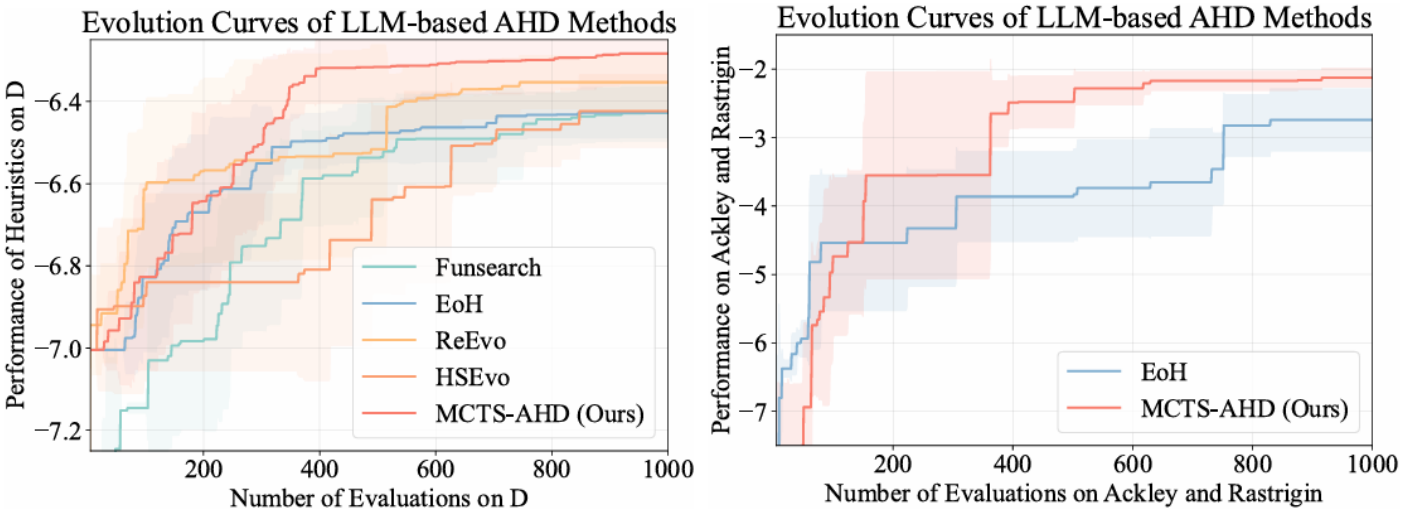
- Progressive Widening
- Tree-path reasoning
- Exploration decay

→ Hiệu năng giảm rõ rệt

Methods	*TSP50*	*KP100*
MCTS-AHD (Original, 10 runs)	10.661%	0.059%
w/o Progressive Widening	12.132%	0.064%
w/o Thought-alignment	11.640%	0.061%
w/o Exploration-decay	11.606%	0.064%
w/o Action s1	11.919%	0.062%
w/o Action m1	10.921%	0.083%
w/o Action m2	11.679%	0.061%
MCTS-AHD variant $\lambda_0 = 0.05$	11.080%	0.056%
MCTS-AHD variant $\lambda_0 = 0.2$	12.124%	0.034%

So sánh hội tụ

- Population-based: hội tụ sớm
- MCTS-AHD: cải thiện liên tục



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

(b) Design CAFs for BO

Figure 4. Evolution curves on two diverse application scenarios.

Kết luận

Đóng góp

- Khám phá toàn diện heuristic space
- Tránh local optimum

Hướng tương lai

- Hybrid MCTS + population

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Phần II: Cài đặt thực tế

ONE LOVE. ONE FUTURE.

2. Chuyển đổi mô hình

Thay đổi từ mô hình của OpenAI sang dùng mô hình của Vertex AI (Google Cloud)

- **Cài đặt client mới:** Thêm module GeminiClient trong gemini.py để giao tiếp với API của Vertex AI Gemini.
  - Sử dụng xác thực OAuth2 với file service account JSON.
  - Tự động lấy access token và gửi request tới endpoint của Vertex AI.
  - Chuẩn hóa giao diện để tương thích với các phần còn lại của hệ thống (giống OpenAIClient).

- **Cấu hình:** Thêm file cấu hình gemini.yaml để định nghĩa các tham số như tên mô hình, đường dẫn credentials, location, v.v.

- **Tích hợp vào pipeline:** Trong file cấu hình tổng (config.yaml), chuyển mục llm_client từ openai sang gemini. Hệ thống sẽ tự động khởi tạo đúng client dựa trên config.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Phần III: Hướng tiếp cận mới

ONE LOVE. ONE FUTURE.

MultiObjective

Trong **MCTS-AHD gốc**, mỗi heuristic h chỉ có **1 giá trị**:

$$Q(h) = \text{solution quality}$$

Nhưng trong hệ thống thật:

- Heuristic A:
  - Quality: **rất tốt**
  - Runtime: **rất chậm**
- Heuristic B:
  - Quality: **kém hơn chút**
  - Runtime: **rất nhanh**

MultiObjective

Với mỗi heuristic h, thay vì 1 giá trị Q(h), ta có:

$$\mathbf{Q}(h) = (Q_{sol}(h), Q_{time}(h))$$

Cụ thể:

$$Q_{\text{scalar}}(h) = w(t) \cdot \hat{Q}_{sol}(h) - (1 - w(t)) \cdot \hat{Q}_{time}(h)$$

- $\hat{Q}$: giá trị đã chuẩn hóa [0,1]
- $w(t)$: trọng số **giảm dần theo t**

$$\frac{Q(c) - q_{min}}{q_{max} - q_{min}}$$

MultiObjective

$$w(t) = 1 - \frac{t}{T}$$

Diễn giải:

Giai đoạn	w	Hành vi
Đầu	≈ 1	Tìm heuristic **tốt nhất có thể**
Giữa	≈ 0.5	Cân bằng tốt – nhanh
Cuối	≈ 0	Ưu tiên heuristic **nhanh / deploy được**

- Ban đầu, MCTS-AHD tập trung tìm heuristic có chất lượng cao nhất;
- Về cuối, nó ưu tiên heuristic có chi phí tính toán thấp hơn.

-> Thiết kế này tận dụng trực tiếp ngân sách đánh giá sẵn có trong MCTS-AHD, không phá cấu trúc thuật toán, nhưng giúp hệ thống tìm được heuristic vừa tốt vừa deploy được.

MultiObjective dựa trên Pareto Dominance

Không gian tìm kiếm

- Mỗi **heuristic** h∈H được sinh bởi LLM (qua prompt, mutation, refinement).
- Ta cần tối ưu **mục tiêu**

$$\mathbf{f}(h) = \big(f_1(h), f_2(h), \ldots, f_K(h)\big), \quad f_k : \mathcal{H} \to \mathbb{R}$$

Ví dụ:

- f_1: chất lượng nghiệm (cost, makespan, tour length)
- f_2: thời gian chạy
- f_3: độ ổn định / variance
- f_4: độ tổng quát hóa trên nhiều instance

-> Mục tiêu là tìm **Pareto Set:**

$$\mathcal{P}^* = \{h \in \mathcal{H} \mid \nexists h' : \mathbf{f}(h') \succ \mathbf{f}(h)\}$$

MultiObjective dựa trên Pareto Dominance

Định nghĩa (Pareto dominance):

Cho hai vector mục tiêu $\mathbf{x}, \mathbf{y} \in \mathbb{R}^K$:

$$\mathbf{x} \succ \mathbf{y} \iff \begin{cases} x_k \geq y_k & \forall k \\ \exists k : x_k > y_k \end{cases}$$

Trong phương pháp này:

- Mỗi heuristic tương ứng **một điểm trong không gian mục tiêu**
- Mỗi node trong MCTS **đại diện cho một tập heuristic con**

MultiObjective dựa trên Pareto Dominance

Cấu trúc Node:

Trong MCTS chuẩn, mỗi node lưu:

$$(N(s), Q(s))$$

Trong phương pháp này, mỗi node lưu:

Số lượt thăm:

$$N(s) \in \mathbb{N}$$

Tập vector reward:

$$\mathcal{R}(s) = \{\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \dots\}$$

Ví dụ:
r(h)=(solution quality,−runtime,−variance)

Rút gọn thành ParetoFront cục bộ:

$$\mathcal{P}(s) = \mathrm{ND}(\mathcal{R}(s))$$

ND: Non-Dominated set (tập không bị thống trị)

MultiObjective dựa trên Pareto Dominance

Selection:

UCB cũ:

$$\mathrm{UCB}(a) = Q(a) + c\sqrt{\frac{\ln N}{N(a)}}$$

Vấn đề gặp phải:

- Q(a)∈R
- P(a)⊂R^k

Trong phương pháp mới :

Scalarization ngẫu nhiên:

Chọn

$$\mathbf{w} \sim \mathrm{Dirichlet}(\alpha)$$

Tính:

$$V_{\mathbf{w}}(s') = \max_{\mathbf{r} \in \mathcal{P}(s')} \mathbf{w}^\top \mathbf{r}$$

=> Biến ParetoFront ở trên thành 1 số

Tính UCB:

$$\text{MO-UCB}(s') = V_{\mathbf{w}}(s') + c\sqrt{\frac{\ln N(s)}{N(s')}}$$

MultiObjective dựa trên Pareto Dominance

Backpropagation:

MCTS:

$$Q(s) \leftarrow \frac{N(s)Q(s) + r}{N(s) + 1}$$

- Trong phương pháp mới : thêm heuristic mới r vào tập vector reward,cập nhật lại non-dominated set

$$\mathcal{R}(s) \leftarrow \mathcal{R}(s) \cup \{\mathbf{r}\}$$

$$\mathcal{P}(s) \leftarrow \mathrm{ND}(\mathcal{R}(s))$$

MultiObjective dựa trên Pareto Dominance

Tiêu chí	MCTS	Multi Objective MCTS
Mục tiêu	Tối ưu 1 reward	Tối ưu nhiều mục tiêu
Giá trị node	$Q(s) \in R$	$P(s) \subset R^K$ (Pareto set)
Selection	UCB	Scalarization + UCB
Output	1 nghiệm	Pareto Front
Độ phức tạp	Thấp	Thấp

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !