EoH-S: Evolution of Heuristic Set Using LLMs for Automated Heuristic Design

Fei Liu[1*], Yilu Liu[1*], Qingfu Zhang[1†], Xialiang Tong[2], Mingxuan Yuan[2]

[1]City University of Hong Kong
[2]Huawei Noah's Ark Lab
fliu36-c@my.cityu.edu.hk, yilu.liu@my.cityu.edu.hk, qingfu.zhang@cityu.edu.hk, tongxialiang@huawei.com, yuan.mingxuan@huawei.com

Abstract

Automated Heuristic Design (AHD) using Large Language Models (LLMs) has achieved notable success in the past two years. Despite the effectiveness of existing approaches, they only design a single heuristic to serve all problem instances, often inducing poor generalization across different distributions or sizes. To address this issue, we propose **Automated Heuristic Set Design (AHSD)**, a new methodology for LLM-driven AHD. The aim of AHSD is to automatically design a small-sized complementary heuristic set to serve diverse problem instances, such that each problem instance could be optimized by at least one heuristic in this set. We propose **Evolution of Heuristic Set (EoH-S)**, which realizes AHSD using an evolutionary search framework. It incorporates a complementary population management and a memetic search to design a set of heuristics. Extensive experiments on online bin packing, traveling salesman problem, and capacitated vehicle routing problem show that EoH-S consistently outperforms existing AHD methods. The resulting heuristics exhibit complementary performance across instances of varying sizes and distributions.

Code — https://github.com/FeiLiu36/EoH-S

Introduction

With the code generation and language comprehension capability of LLMs, automation and flexibility of algorithm design can be significantly improved (Liu et al. 2024c; Da Ros et al. 2025; Zhang et al. 2025) . A very successful paradigm in LLM-driven Automated Heuristic Design (AHD) is to integrate LLMs as designers within iterative search frameworks such as evolutionary search (Liu et al. 2024b; Zhang et al. 2024; Ye et al. 2024; van Stein and Bäck 2024; Yao et al. 2025; Hu and Zhang 2025), neighborhood search (Xie et al. 2025), and Monte Carlo Tree Search (MCTS) (Zheng et al. 2025b). These approaches have found applications across diverse domains, including different optimization tasks (Liu et al. 2024b; Ye et al. 2024; van Stein and Bäck 2024; Yao et al. 2024; Ye et al. 2025; Dat, Doan, and Binh 2025; Li et al. 2025b; Çetinkaya et al. 2025), mathematics (Romera-Paredes et al. 2024; Novikov et al. 2025), machine learning (Mo et al. 2025; Zhou et al. 2025; Liu et al. 2024d; Hu, Lu, and Clune 2025; Li et al. 2025a), and engineering (Senkerik et al. 2025; Zheng et al. 2025a; Sartaj and Ali 2025).

A first attempt along this line includes EoH (its early version is called AEL) (Liu et al. 2023, 2024b), which adopts an evolutionary search framework with different prompting strategies to evolve both thoughts and codes for automated heuristic design. Meanwhile, FunSearch (Romera-Paredes et al. 2024) utilizes a multi-island evolutionary framework with a single prompt strategy for automated function search. Moreover, ReEvo (Ye et al. 2024) integrates short- and long-term reflection strategies to enhance the heuristic design process.

Despite these advancements, existing methods mainly focus on identifying a single heuristic with the best average performance across a set of training instances. This approach may suffer from generalization limitations due to the following two reasons: 1) finding a single heuristic with the best performance across all diverse instances is inherently difficult (Sim, Renau, and Hart 2025), and 2) even when a heuristic excels in most training instances, it often fails to generalize to unseen test instances with different distributions or scales (Shi et al. 2025).

A common-used approach to tackle the generalization limitations of heuristic design is to use an algorithm portfolio (Gomes and Selman 2001; Tang et al. 2021) (i.e., a combination of different algorithms). It is natural for us to leverage this approach to deal with the challenges faced by LLM-driven AHD: using different heuristics rather than a single heuristic. However, given the typically enormous number of problem instances, it is impractical, if not impossible, to find one heuristic for each instance. With these concerns, this paper makes the following contributions:

- We introduce an **Automated Heuristic Set Design (AHSD)** formulation for LLM-driven AHD. Unlike prior work that focuses on designing a single heuristic, AHSD aims to design a small-sized complementary heuristic set to serve diverse problem instances, such that each problem instance could be optimized by at least one heuristic in this set. We show that the objective function of AHSD is monotone and supermodular. This formulation is general and, while demonstrated on cross-distribution scenarios, is readily applicable to other domains.

[*]These authors contributed equally.
[†]Corresponding author.

- We propose an **Evolution of Heuristic Set (EoH-S)** approach to apply the AHSD formulation for LLM-driven AHD. EoH-S adopts the same heuristic representation and an evolutionary framework as EoH. Different from EoH, EoH-S incorporates two key components including complementary population management and diversity-aware memetic search to design a small-sized set of heuristics to optimize the complementary performance.
- We conduct experimental studies on three AHD tasks, including online bin packing, traveling salesman problem, and capacitated vehicle routing problem, with instances of different distributions and sizes. Results show that EoH-S outperforms state-of-the-art AHD methods and the designed heuristics demonstrate strong complementary behavior.

Automated Heuristic Set Design (AHSD)

Problem Formulation

Suppose we are given a target heuristic design task $\mathcal{T}$ (e.g., Traveling Salesman Problem (TSP)) with m diverse problem instances $I = \{i_1, \ldots, i_m\}$, the aim of AHSD is to automatically generate a heuristic set $H = \{h_1, \ldots, h_k\} \subseteq D$, where $1 < k \ll m$ and D is the search space, such that each problem instance could be optimized by at least one heuristic in this set, which could be mathematically expressed as

$$\min_{H \subseteq D} \left(f^*_{H,i_1}, f^*_{H,i_2}, \ldots, f^*_{H,i_m} \right), \tag{1}$$

where $f^*_{H,i} = \min_{h \in H} f_i(h)$, and $f_i(h)$ denotes the performance score of heuristic h on instance i (lower is better). Since (1) evaluates H from multiple criteria, it is difficult to directly optimize it. To address this issue, we follow the aggregation idea in (Liu et al. 2024e; Lin et al. 2025) to transform (1) into the following optimization objective:

$$\mathcal{F}(H) = \frac{1}{m} \sum_{i \in I} f^*_{H,i}. \tag{2}$$

For clarity, we call $\mathcal{F}(H)$ as **Complementary Performance Index (CPI)**. Taking CPI as the objective function, the AHSD problem could be formally stated as below.

> *AHSD Problem*: Given a target heuristic design task $\mathcal{T}$ with m task instances $I = \{i_1, \ldots, i_m\}$, the aim of AHSD is to automatically generate a heuristic set $H = \{h_1, \ldots, h_k\} \subseteq D$ with $1 < k \ll m$ such that $\mathcal{F}(H)$ is minimized.

When $k = 1$, the AHSD problem is equal to finding a single optimal heuristic with the best average performance across all instances, which is aligned with the objective of most LLM-driven AHD methods. When $k = m$, the AHSD problem is equal to finding the best heuristic for each instance independently. Thus, we consider the non-trivial case of $1 < k \ll m$. Next, we demonstrate some theoretical properties of $\mathcal{F}(H)$.

Theoretical Analysis

Theorem 1 (Monotonicity and Supermodularity)**.** *For any two heuristic sets $U \subseteq V \subseteq D$ with $1 \leq |U| \leq |V|$, $\mathcal{F}(U) \geq \mathcal{F}(V)$ holds. Besides, for any heuristic $h' \in D \backslash V$, $\mathcal{F}(U) - \mathcal{F}(U \cup \{h'\}) \geq \mathcal{F}(V) - \mathcal{F}(V \cup \{h'\})$ holds.*

Proof. We first demonstrate that $f^*_{U,i} \geq f^*_{V,i}$ holds for all instances $i \in \{i_1, \ldots, i_m\}$. For simplicity, we use j^* to denote the index of the best heuristic for i within V, i.e., $j^* = argmin_{1 \leq j \leq |V|} f_i(h_j)$. Therefore, we only need to discuss the following two cases:

1. $1 \leq j^* \leq |U|$. This case means that the best heuristic for i is in U. As $U \subseteq V$, we can derive $f^*_{U,i} = f^*_{V,i}$.
2. $|U| < j^* \leq |V|$. This case means that the best heuristic for i is in $V \backslash U$, showing that $f^*_{U,i} \geq f^*_{V,i}$.

These two cases indicate $f^*_{U,i} \geq f^*_{V,i}$. As $\mathcal{F}(H)$ is a convex combination of $f^*_{H,i}$, we have $\mathcal{F}(U) \geq \mathcal{F}(V)$, showing that $\mathcal{F}(H)$ is monotone.

Let $U' = U \cup \{h'\}$. We then show that $f^*_{H,i}$ is supermodular, i.e., $f^*_{U,i} - f^*_{U',i} \geq f^*_{V,i} - f^*_{V',i}$. For simplicity, we use j^* to denote the index of the best heuristic for i within V', i.e., $j^* = argmin_{1 \leq j \leq |V|+1} f_i(h_j)$. Therefore, we only need to discuss the following three cases:

1. $1 \leq j^* \leq |U|$. This case means that the best heuristic for i is in U. Thus, we can derive $f^*_{U,i} = f^*_{U',i} = f^*_{V,i} = f^*_{V',i}$, showing that $f^*_{U,i} - f^*_{U',i} \geq f^*_{V,i} - f^*_{V',i}$.
2. $|U| < j^* \leq |V|$. This case means that the best heuristic for i is in $V \backslash U$, showing that $f^*_{V,i} = f^*_{V',i}$. Since $f^*_{U,i} \geq f^*_{U',i}$, we have $f^*_{U,i} - f^*_{U',i} \geq f^*_{V,i} - f^*_{V',i}$.
3. $j^* = |V| + 1$. This case means that the best heuristic for i is h', showing that $f^*_{U',i} = f^*_{V',i}$. Since $f^*_{U,i} \geq f^*_{V,i}$, we have $f^*_{U,i} - f^*_{U',i} \geq f^*_{V,i} - f^*_{V',i}$.

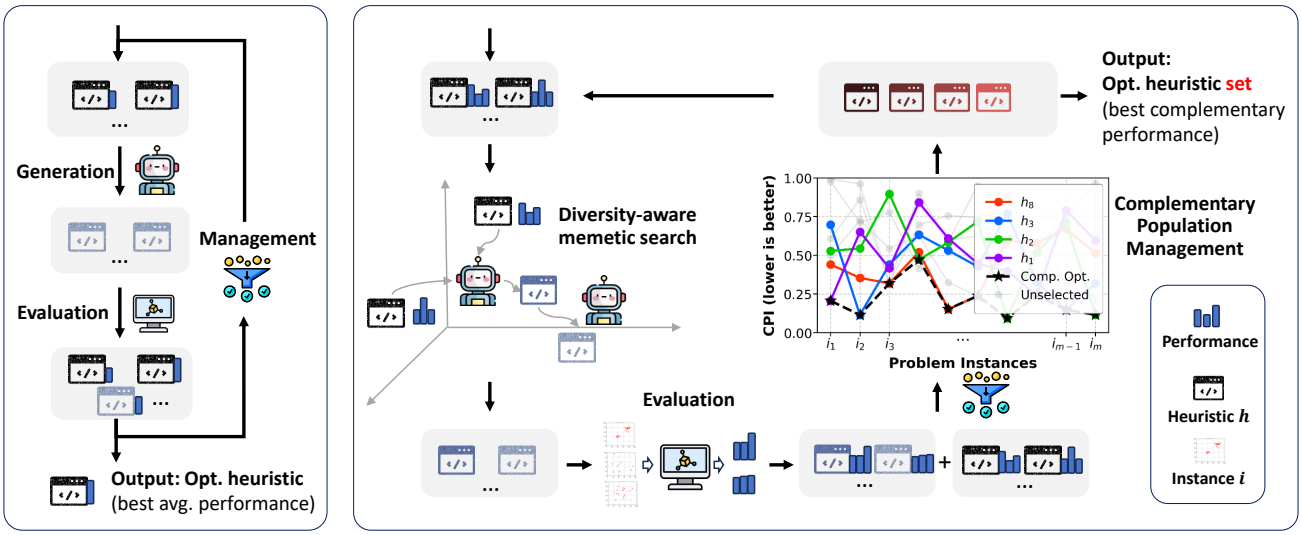
These three cases confirm the supermodularity of $f^*_{H,i}$. Similarly, as $\mathcal{F}(H)$ is a convex combination of $f^*_{H,i}$, $\mathcal{F}(U) - \mathcal{F}(U \cup \{h'\}) \geq \mathcal{F}(V) - \mathcal{F}(V \cup \{h'\})$ holds, showing that $\mathcal{F}(H)$ is supermodular. Thus, Theorem 1 holds.

Evolution of Heuristic Set (EoH-S)

Framework Overview

We propose an evolutionary search framework, named Evolution of Heuristic Set (EoH-S), aimed at automatically designing a set of complementary heuristics. As illustrated in Figure 1, the existing LLM-driven AHD methods aim at generating a single optimal heuristic to optimize the average performance on the target task (Liu et al. 2024b; Romera-Paredes et al. 2024; Ye et al. 2024; Zheng et al. 2025b). In contrast, EoH-S is proposed for AHSD to design a set of heuristics that complement each other on diverse instances.

Following EoH (Liu et al. 2024b, 2023), each heuristic in EoH-S is represented by both a high-level thought description and an executable code implementation (in this paper, Python functions). However, unlike existing approaches that use average performance as fitness, EoH-S maintains an



a) Existing AHD methods b) EoH-S for AHSD

Figure 1: **a) Existing LLM-driven Automated Heuristic Design (AHD) methods** employ an iterative search framework to identify a single optimal heuristic, optimizing average performance. **b) Automated Heuristic Set Design (AHSD)** seeks to generate a set of complementary heuristics, enhancing performance across diverse problem instances. EoH-S adopts an evolutionary framework with diversity-aware memetic search and complementary population management for effective AHSD.

instance-wise performance vector (i.e., scores across m instances) for each heuristic, enabling complementary search and management.

The EoH-S framework operates as follows:

1. **Initialization:** EoH-S begins with an initial population, P_0, consisting of n heuristics $\{h_1, \ldots, h_n\}$. These heuristics are generated by repeatedly prompting LLMs using an initialization prompt $h_i = \text{Init}(LLM, p_i)$. The initialization prompt consists of a task description and a function template. The LLM is instructed first to generate a heuristic thought and then implement its code based on the template. (In this paper, we prompt LLM to generate all the initial heuristics following EoH. However, we can also use one or more existing hand-designed heuristics in the initial population.)
2. **Evolutionary Process:** Start from the initial population, EoH-S iteratively performs the following steps:
   (a) **Memetic Search:** Two strategies are adopted n times with the same probability to create n new heuristics $\{h_{o_1}, \ldots, h_{o_n}\}$ based on existing ones in the current population:
      i. Complementary-aware search (CS): $h_o = \text{CS}(LLM, p_{cs}, h_p)$, where p_{cs} is the prompt for complementary-aware search and h_p denotes the parent heuristics used in the prompt. In this paper, we use two parent heuristics, which are selected according to their complementary behaviour on diverse instances. CS is used to explore diverse heuristics to enhance the complementary performance of the entire heuristic set.
      ii. Local search (LS): $h_o = \text{LS}(LLM, p_{ls}, h_p)$, where p_{ls} is the local search prompt and h_p is one selected heuristic. LS aims at revising one heuristic to search for a new heuristic close to the parent heuristic.
   (b) **Population Management:** The n new heuristics and n heuristics from the current population are combined into a candidate pool of size $2n$: $\hat{P}_{i+1} = P_i \cup \{h_{o_1}, \ldots, h_{o_n}\}$. Then, a Complementary Population Management (CPM) strategy is used to select n heuristics from these $2n$ condidates to form the next population: $P_{i+1} = CPM(\hat{P}_{i+1})$.
3. **Termination:** The process is terminated when a predefined stopping criterion is met, such as reaching the maximum number of evaluated heuristics N_{max}.

Memetic Search

EoH-S adopts a complementary-aware memetic search to effectively explore new heuristics. It integrates two reproduction strategies: **Complementary-aware Search (CS)** and **Local Search (LS)**.

Complementary-aware Search (CS) CS encourages exploring complementary heuristics through both 1) selecting two parent heuristics h_p and 2) the prompt p_{cs}. Specifically, two parent heuristics h_p are selected based on their instance-wise performance across m instances. The complementary nature between two heuristics h_a and h_b is measured using Manhattan distance d_{h_a,h_b}. The Manhattan distance measures the sum of the absolute differences between corresponding elements, quantifying the total performance difference between two heuristics across the same set of problem instances. It is computed as:

$$\mathrm{d}_{h_a,h_b} = \sum_{j=1}^{m} |f_{i_j}(h_a) - f_{i_j}(h_b)|,$$

where $F(h_a) = \{f_{i_1}(h_a), \ldots, f_{i_m}(h_a)\}$ and $F(h_b) = \{f_{i_1}(h_b), \ldots, f_{i_m}(h_b)\}$ denote the performance scores of heuristics a and b on m instances, respectively. We select the pair of heuristics in the current population P with the largest distance:

$$(h_{p1}, h_{p2}) = argmax_{h_a,h_b \in P} \mathrm{d}_{h_a,h_b}.$$

In the prompt p_{cs}, we inform LLM that we have this complementary pair of heuristics and instruct LLM to create new heuristics distinct from existing designs.

Local Search (LS) In contrast to CS, LS focuses on refinement of existing heuristics instead of exploring new ones. The one parent heuristic h_p is selected according to a weighted random selection where better-ranked functions have higher probabilities (Liu et al. 2024b). The rank is sorted using the average performance on m instances. This strategy promotes exploitation by producing heuristics that preserve the parents' strengths while improving performance through local adjustments.

We generate n new heuristics in memetic search using the two reproduction strategies with equal probability. Following reproduction, candidate heuristics are evaluated on a set of diverse instances.

Complementary Population Management

EoH-S employs a **Complementary Population Management (CPM)** mechanism to select the n heuristics from $2n$ candidate pool to form the next population. Unlike conventional LLM-driven AHD methods that select heuristics based solely on average performance, CPM leverages instance-wise performance vectors to maintain population complementarity across problem instances. It greedily select n heuristics from $2n$ candidate heuristics to minimize the $\mathcal{F}(H)$ value on m instances.

Specifically, the CPM is performed as follows: First, select the heuristic with the best average performance on m instances from $2n$ heuristics to be the first one. Then, iteratively select the next heuristic to minimize the delta CPI until n heuristics have been selected. Given a current heuristic set $H_k = \{h_1, \ldots, h_k\}$ and a candidate heuristic h_i, the delta CPI is defined as:

$$\Delta\text{CPI}(h_i \mid H_k) = \sum_{j=1}^{m} \min\left(f_{i_j}(h_i) - f^*_{H_k,i_j}, 0\right), \quad (3)$$

where $f_{i_j}(h_i)$ is the performance score of h_i on instance i_j (lower is better), and $f^*_{H_k,i_j} = \min_{h \in H_k} f_{i_j}(h)$ is the best score in H_k for instance i_j.

The CPM operates as follows:

1. **Initialize**: Select the heuristic with best average performance from $2n$ candidates as h_1.
2. **Iterative Selection**:
   (a) For current $H_k = \{h_1, \ldots, h_k\}$, compute $\Delta\text{CPI}(h_i \mid H_k)$ for each remaining h_i in $2n - k$ candidates.
   (b) Select $h_{i^*} = argmin_{i \in 2n-k} \Delta\text{CPI}(h_i \mid H_k)$ and update $H_{k+1} = H_k \cup \{h_{i^*}\}$.
   (c) Update reference scores: $f^*_{H_{k+1},i_j} = \min(f^*_{H_k,i_j}, f_{i_j}(h_{i^*}))$.
3. **Terminate** when $|H_k| = n$.

Experimental Studies

Tasks and Instances

We investigate the following three tasks:

- Online Bin Packing (OBP) involves packing items into the minimum number of fixed-capacity bins as items arrive sequentially. The heuristic is used to select the assigned bin for each incoming item. The performance is measured as the relative gap to the lower bound of the optimal number of bins computed as in (Martello and Toth 1990). For heuristic evaluation during automated design (training), I consists of 128 Weibull instances (Romera-Paredes et al. 2024) with a bin capacity of 100 and the number of items ranging from 200 to 2000 (2k). For testing, we use six sets with larger capacities 200,500 and more items 1k, 5k, 10k, each set consists of 5 instances following existing works (Romera-Paredes et al. 2024; Liu et al. 2024b)
- Traveling Salesman Problem (TSP) requires finding the shortest route visiting all cities exactly once before returning to the start. We design a step-by-step construction heuristic. The heuristic is used to iteratively select the next city to visit. The average relative gap from the baseline solution generated by LKH (Helsgaun 2017) is used for performance measurement. For training, I consists of 128 instances with the number of cities ranging from 10 to 200, sampled in $[0, 1]$ using a Gaussian distribution in clusters following Bi et al. (2022). For testing, we use instances in uniform distribution with the number of cities ranging from 50 to 500.
- Capacitated Vehicle Routing Problem (CVRP) extends TSP by incorporating vehicle capacity constraints and customer demands. The goal is minimizing total travel distance while ensuring all customer demands are met without exceeding vehicle capacities. We design a step-by-step construction heuristic. The heuristic is to select the next node. The average relative gap from the baseline solution generated by LKH (Helsgaun 2017) is used for performance measurement. For training, I consists of 256 instances with the number of nodes from 20 to 200 and the capacities from 10 to 150. For testing, we use instances with node sizes ranging from 50 to 500. For both training and testing, the nodes are sampled in $[0, 1]$ using a uniform distribution. We do not consider Gaussion distribution here because the different capacities and sizes have already introduced diversities.

Compared Methods and Settings

We compare state-of-the-art LLM-driven AHD methods, including **Random** (i.e., repeated prompt LLMs to generate heuristic without an iterative search framework (Zhang et al.

Methods	Training (c100)				Testing (c200-500)				
	n200-500	n500-1k	n1k-2k	n1k_c200	n1k_c500	n5k_c200	n5k_c500	n10k_c200	n10k_c500
First Fit	0.0652	0.0318	0.0387	0.0240	**0.0124**	0.0173	0.0075	0.0163	0.0055
Best Fit	0.0627	0.0310	0.0372	0.0220	**0.0124**	0.0161	0.0070	0.0154	0.0050
Random*	0.3638	0.2988	0.1489	0.1939	1.3508	0.0376	0.2816	0.0189	0.1402
1+1 EPS*	0.2482	0.2180	0.1246	0.0569	0.2390	0.0101	0.0439	0.0055	0.0224
FunSearch*	0.2510	0.2000	0.1326	0.0827	0.6169	0.0193	0.2260	0.0120	0.1211
EoH*	0.1216	0.1169	0.0750	0.0190	**0.0124**	0.0044	0.0050	0.0041	0.0042
MEoH*	0.0991	0.0749	0.0547	0.1037	0.0149	0.0417	0.0035	0.0326	0.0015
CALM*	0.0805	0.0300	0.0470	0.0120	**0.0124**	0.0048	0.0030	0.0034	0.0025
ReEvo*	0.0877	0.0344	0.0556	0.0120	**0.0124**	0.0074	0.0040	0.0068	0.0025
MCTS-AHD*	0.0974	0.0360	0.0596	0.0150	**0.0124**	0.0046	0.0040	0.0034	0.0022
FunSearch	0.0623	0.0298	0.0367	0.0213	**0.0124**	0.0159	0.0070	0.0152	0.0051
FunSearch Top10	0.0608	0.0289	0.0361	0.0200	**0.0124**	0.0153	0.0065	0.0123	0.0038
EoH	0.0619	0.0307	0.0371	0.0240	0.0133	0.0182	0.0075	0.0170	0.0059
EoH Top10	0.0618	0.0299	0.0368	0.0216	**0.0124**	0.0158	0.0065	0.0152	0.0051
ReEvo	0.0604	0.0298	0.0362	0.0217	0.0133	0.0162	0.0068	0.0155	0.0053
ReEvo Top10	0.0580	0.0285	0.0349	0.0203	**0.0124**	0.0125	0.0063	0.0108	0.0049
EoH-S	**0.0515**	**0.0230**	**0.0314**	**0.0113**	**0.0124**	**0.0033**	**0.0025**	**0.0014**	**0.0010**

Table 1: Evaluation of heuristics designed by different methods on OBP instances. We use 128 Weibull instances with sizes ranging from 200 to 2k for training and six different sets (n1k_c200, n1k_c500, n5k_c200, n5k_c500, n10k_c200, n10k_c500) for testing, where n and c represent the number of items and the bin capacity, respectively. Each method is run three times and we report the average performance. We directly use the best-performing heuristics from their original papers for the methods with $*$. The best values are **in bold** with a grey background and the second-best values are in a light-grey background.

2024)), **EoH** (Liu et al. 2023, 2024b), **FunSearch** (Romera-Paredes et al. 2024), **ReEvo** (Ye et al. 2024), **1+1 EPS** (Zhang et al. 2024), **MEoH** (Yao et al. 2025), **MCTS-AHD** (Zheng et al. 2025b), and **CALM** (Huang et al. 2025).

We evaluate these methods under two settings:

- **Direct comparison**: We directly compare the best-performing heuristics reported in the original papers for the online bin packing problem. These heuristics are all designed for the Weibull instances with slightly different distributions.
- **Controlled comparison**: For some representative methods (EoH, FunSearch, and ReEvo), we conduct training using identical instances to EoH-S, enabling a fair performance comparison. Three independent runs are performed for these methods on all three tasks.

All experiments are conducted on LLM4AD platform[1] (Liu et al. 2024d) using the DEEPSEEK-V3 (Liu et al. 2024a) API with default parameter settings. We set the maximum number of heuristic evaluations to $N_{\max} = 2{,}000$ for all tasks and fix the population size at $n = 10$ for EoH-S, EoH, and ReEvo to maintain consistency (FunSearch uses a dynamic population size). The experiments run on one Intel i7-9700 CPU, and the automated heuristic design process completes in about three hours for all methods across different tasks.

[1]https://github.com/Optima-CityU/LLM4AD

It is worth noting that although EoH-S explicitly design a set of complementary heuristics, it does not introduce additional running time because we use the same maximum number of evaluations for all methods.

Results on Training & Testing Instances

The results on training and testing instances of diverse distributions and sizes are summarized in Table 1 (OBP) and Table 2 (TSP and CVRP), where the best values are in bold with a grey background and the second-best values are in a light-grey background. The values are the gap to baseline results (lower bound for OBP (Romera-Paredes et al. 2024) and LKH for TSP and CVRP (Helsgaun 2017)). We report the average results over three independent runs. The methods with $*$ denote the best heuristic from their original papers. For EoH-S, we use 10 heuristics in the final population. For fair comparison, we also compare the top 10 heuristics from FunSearch, EoH, and ReEvo (denoted as FunSearch Top10, EoH Top10, and ReEvo Top10, respectively).

As indicated in Table 1, EoH-S consistently outperforms all baseline methods on both training instances (c100 with varying item counts) and testing instances (c200-500 with different problem sizes), achieving the lowest gap values in 8 out of 9 configurations and matching the best performance in the remaining one. In contrast, existing LLM-driven AHD methods from the literature struggle to generalize across different distributions, with some even underperforming first-fit and best-fit heuristics.

Methods	Training	Testing			
		50	100	200	500
FunSearch	0.156	0.144	0.149	0.169	0.185
FunSearch Top10	0.124	0.091	0.105	0.124	0.157
EoH	0.152	0.142	0.154	0.168	0.194
EoH Top10	0.127	0.079	0.123	0.134	0.175
ReEvo	0.155	0.167	0.183	0.223	0.221
ReEvo Top10	0.117	0.092	0.123	0.157	0.170
EoH-S	**0.097**	**0.040**	**0.065**	**0.090**	**0.111**

Methods	Training	Testing			
		50	100	200	500
FunSearch	0.294	0.315	0.365	0.296	0.241
FunSearch Top10	0.245	0.267	0.309	0.247	0.214
EoH	0.276	0.274	0.299	0.261	0.221
EoH Top10	0.230	0.172	0.232	0.218	0.198
ReEvo	0.265	0.274	0.296	0.256	0.203
ReEvo Top10	0.240	0.214	0.249	**0.173**	0.178
EoH-S	**0.173**	**0.135**	**0.188**	0.180	**0.169**

Table 2: Evaluation of heuristics designed by different methods on TSP (upper) and CVRP (lower). We use 128 TSP instances with sizes ranging from 10 to 200 and 256 CVRP instances with sizes ranging from 20 to 200 for testing, and four different sets with sizes ranging from 50 to 500 for testing. Each method is run three times, and we report the average performance. The best values are **in bold** with a grey background and the second-best values are in a light-grey background.

Table 2 shows that EoH-S performs consistently well on TSP and CVRP. On TSP, EoH-S significantly reduces the optimality gap by 50-60% compared to the second-best method across all test instances of varying sizes (50-500 nodes). For CVRP, EoH-S again achieves the lowest optimality gap on both training and testing instances.

Results on Benchmark Instances

Table 3 presents a comprehensive evaluation of our proposed EoH-S method against existing approaches (EoH and ReEvo) across multiple benchmark datasets, including BPPLib (Delorme, Iori, and Martello 2018), TSPLib (Reinelt 1991), and CVRPLib (Uchoa et al. 2017). Our proposed EoH-S method consistently outperforms both baseline methods across all benchmark instances. On the BPPLib datasets, EoH-S achieves superior results, with particularly notable improvement on the Scholl_H instance (0.095 compared to 0.138 for both baselines).

For the TSPLib benchmark, EoH-S demonstrates substantial performance gains with a score of 0.093, representing a 43.6% improvement over the second-best result (0.165 from ReEvo Top 10). On the CVRPLib benchmarks, EoH-S achieves improvements ranging from 5.8% (on set E) to 26.8% (on set B) compared to the second-best results. These results clearly demonstrate the effectiveness and robustness of our proposed approach in designing complementary heuristics.

Benchmarks	EoH		ReEvo		EoH-S
	Top 1	Top 10	Top 1	Top 10	
BPPLib Sch_1	0.153	0.153	0.153	0.153	**0.152**
BPPLib Sch_2	0.142	0.142	0.142	0.142	**0.141**
BPPLib IRUP	0.077	0.075	0.079	0.074	**0.072**
BPPLib NonIRUP	0.077	0.076	0.080	0.075	**0.073**
BPPLib Scholl_H	0.138	0.138	0.138	0.138	**0.095**
TSPLib	0.184	0.173	0.220	0.165	**0.093**
CVRPLib A	0.326	0.277	0.329	0.261	**0.231**
CVRPLib B	0.374	0.310	0.352	0.250	**0.183**
CVRPLib E	0.334	0.268	0.305	0.257	**0.242**
CVRPLib F	0.539	0.493	0.687	0.547	**0.427**
CVRPLib M	0.461	0.377	0.442	0.372	**0.299**
CVRPLib P	0.273	0.206	0.270	0.188	**0.169**
CVRPLib X	0.270	0.237	0.270	0.224	**0.196**

Table 3: Results on OPPLib, TSPLib, and CVRPLib Benchmarks. The best values are **in bold** with a grey background and the second-best values are in a light-grey background.

Complementary Performance

We evaluate the complementary performance of heuristic sets generated by different methods. Figure 2 compares the performance (averaged over three independent runs) of heuristic sets as the number of heuristics increases (on a logarithmic scale from 1 to 100). The CPI measures the complementary performance on all problem instances.

For EoH-S, we use the final population of heuristics with 10 heuristics. For EoH, FunSearch, and ReEvo, we select the top 100 heuristics in terms of fitness from all candidates evaluated during the search process. For example, when the set size is 1, we report the performance of the single best heuristic (in terms of average performance, i.e., fitness). For a set size of 10, we consider the full heuristic set for EoH-S and the top 10 heuristics for other methods. The CPI drop observed when expanding from a single heuristic to a larger set reflects the degree of complementarity, i.e., how effectively additional heuristics contribute to solving diverse problem instances.

Our results demonstrate three key findings: First, the heuristic set designed by EoH-S consistently outperforms those designed by the compared methods. Second, even when utilizing 100 heuristics, the competing methods struggle to match the performance achieved by just 10 heuristics designed by EoH-S. Third, each heuristic in the EoH-S set makes a meaningful contribution to the overall performance. The significantly steeper performance curve of EoH-S compared to other methods indicates that all heuristics in the set provide greater complementary benefits.

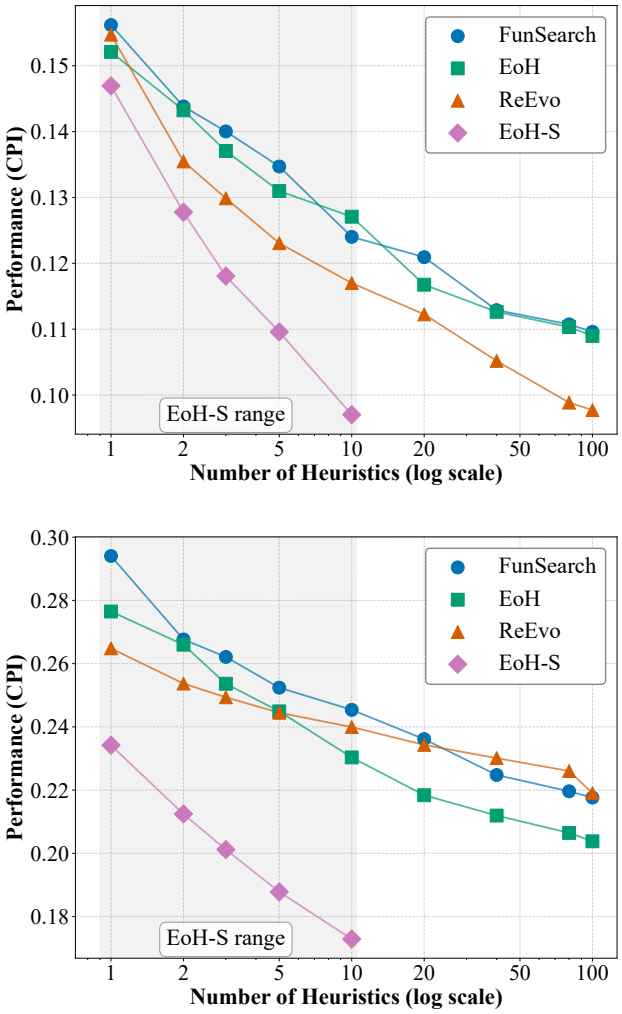


Figure 2: Complementary Performance Index (CPI) comparison across methods with varying numbers of heuristics. For EoH-S, we select heuristics from the final population (maximum 10), while for other methods, we select the best 100 heuristics from the entire search history.

Ablation Studies

We conduct ablation studies on OBP to evaluate the contribution of each key component and the impact of population size. The following variants of EoH-S are examined:

- EoH-S w/o LS: Excludes the LS operator of EoH-S.
- EoH-S w/o CS: Excludes the CS operator of EoH-S.
- EoH-S w/o CPM: Disables the CPM mechanism and instead uses a population management approach in existing works (retaining the best-performing heuristics based on average performance).
- EoH-S n5–n20: Tests EoH-S with population sizes ranging from 5 to 20.

The experimental setup matches the main experiments, with 2,000 total samples per run and a population size 10.

No.	Settings	Gap
1	First Fit	0.0413
2	Best Fit	0.0399
3	EoH-S w/o LS	0.0336
4	EoH-S w/o CS	0.0335
5	EoH-S w/o CPM	0.0373
6	EoH-S n5	0.0333
7	EoH-S n15	0.0339
8	EoH-S n20	0.0337
9	EoH-S	0.0326

Table 4: Ablation study of key components and settings. The first two methods are hand-designed heuristics. No. 3 to 5 are EoH-S configurations that without two search operators (LS and CS) and the CPM. The remaining variants (No. 6–8) are EoH-S with different population sizes (5–20).

Table 4 presents the average gap to the lower bound across 128 OBP training instances.

The first two are two hand-designed heuristics. The No. 3 to No. 5 are three variants of EoH-S that assess the importance of the search operators and CPM. Results show that all components contribute to performance, with CPM providing the most significant improvements. The remaining variants (No. 6–8) demonstrate that EoH-S consistently outperforms existing methods across population sizes (5–20), with a size of 10 proving optimal for this task. Results demonstrate the contribution of different components and the robustness of EoH-S across different settings.

Conclusion and Future Works

In this work, we introduced Automated Heuristic Set Design (AHSD), a new methodology for LLM-driven AHD, which generates a small yet complementary set of heuristics for diverse problem instances. We demonstrated that the objective function of AHSD is monotone and supermodular, providing a theoretical foundation for efficient heuristic set optimization. We proposed EoH-S, which is extended from EoH by integrating a complementary population management and a memetic search for effectively evolving a set of high-quality and complementary heuristics. Extensive experiments across three AHD tasks validated the superiority of EoH-S, achieving up to 60% performance improvements over state-of-the-art AHD methods. Notably, the designed heuristics demonstrated strong generalization capabilities.

Our work advances LLM-driven automated heuristic design by shifting from single-best heuristic design to complementary heuristic set design. Future research directions include exploring heuristic collaboration strategies for further performance gains, integrating an automated heuristic selection procedure to select the most promising heuristic for each instance instead of using all heuristics in the set, and conducting additional application studies.

Acknowledgements

The work described in this paper was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (GRF Project No. CityU11215622), and Research Grants Council of the Hong Kong Special Administrative Region, China (GRF Project No. CityU11217325).

References

Bi, J.; Ma, Y.; Wang, J.; Cao, Z.; Chen, J.; Sun, Y.; and Chee, Y. M. 2022. Learning generalizable models for vehicle routing problems via knowledge distillation. *Advances in Neural Information Processing Systems*, 35: 31226–31238.

Çetinkaya, İ. O.; Büyüktahtakın, İ. E.; Shojaee, P.; and Reddy, C. K. 2025. Discovering heuristics with Large Language Models (LLMs) for mixed-integer programs: Single-machine scheduling. *Computers & Operations Research*, 107325.

Da Ros, F.; Soprano, M.; Di Gaspero, L.; and Roitero, K. 2025. Large language models for combinatorial optimization: A systematic review. *arXiv preprint arXiv:2507.03637*.

Dat, P. V. T.; Doan, L.; and Binh, H. T. T. 2025. HSEVO: Elevating automatic heuristic design with diversity-driven harmony search and genetic algorithm using llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 26931–26938.

Delorme, M.; Iori, M.; and Martello, S. 2018. BPPLIB: a library for bin packing and cutting stock problems. *Optimization Letters*, 12(2): 235–250.

Gomes, C. P.; and Selman, B. 2001. Algorithm portfolios. *Artificial Intelligence*, 126(1-2): 43–62.

Helsgaun, K. 2017. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12: 966–980.

Hu, Q.; and Zhang, Q. 2025. Partition to Evolve: Niching-enhanced Evolution with LLMs for Automated Algorithm Discovery. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Hu, S.; Lu, C.; and Clune, J. 2025. Automated Design of Agentic Systems. In *The Thirteenth International Conference on Learning Representations*.

Huang, Z.; Wu, W.; Wu, K.; Wang, J.; and Lee, W.-B. 2025. CALM: Co-evolution of algorithms and language model for automatic heuristic design. *arXiv preprint arXiv:2505.12285*.

Li, A.; Wu, C.; Ge, Z.; Chong, Y. H.; Hou, Z.; Cao, L.; Ju, C.; Wu, J.; Li, H.; Zhang, H.; et al. 2025a. The FM Agent. *arXiv preprint arXiv:2510.26144*.

Li, R.; Wang, L.; Sang, H.; Yao, L.; and Pan, L. 2025b. LLM-assisted automatic memetic algorithm for lot-streaming hybrid job shop scheduling with variable sublots. *IEEE Transactions on Evolutionary Computation*.

Lin, X.; Liu, Y.; Zhang, X.; Liu, F.; Wang, Z.; and Zhang, Q. 2025. Few for many: Tchebycheff set scalarization for many-objective optimization. In *Proceedings of the International Conference on Learning Representations*.

Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Liu, F.; Tong, X.; Yuan, M.; Lin, X.; Luo, F.; Wang, Z.; Lu, Z.; and Zhang, Q. 2024b. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the International Conference on Machine Learning*, 32201–32223.

Liu, F.; Tong, X.; Yuan, M.; and Zhang, Q. 2023. Algorithm evolution using large language model. *arXiv preprint arXiv:2311.15249*.

Liu, F.; Yao, Y.; Guo, P.; Yang, Z.; Zhao, Z.; Lin, X.; Tong, X.; Yuan, M.; Lu, Z.; Wang, Z.; et al. 2024c. A systematic survey on large language models for algorithm design. *arXiv preprint arXiv:2410.14716*.

Liu, F.; Zhang, R.; Xie, Z.; Sun, R.; Li, K.; Lin, X.; Wang, Z.; Lu, Z.; and Zhang, Q. 2024d. Llm4ad: A platform for algorithm design with large language model. *arXiv preprint arXiv:2412.17287*.

Liu, Y.; Lu, C.; Lin, X.; and Zhang, Q. 2024e. Many-objective cover problem: Discovering few solutions to cover many objectives. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, 68–82.

Martello, S.; and Toth, P. 1990. Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics*, 28(1): 59–70.

Mo, S.; Wu, K.; Gao, Q.; Teng, X.; and Liu, J. 2025. AutoSGNN: Automatic propagation mechanism discovery for spectral graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 19493–19502.

Novikov, A.; Vũ, N.; Eisenberger, M.; Dupont, E.; Huang, P.-S.; Wagner, A. Z.; Shirobokov, S.; Kozlovskii, B.; Ruiz, F. J.; Mehrabian, A.; et al. 2025. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

Reinelt, G. 1991. TSPLIB—A traveling salesman problem library. *ORSA Journal on Computing*, 3(4): 376–384.

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. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625(7995): 468–475.

Sartaj, H.; and Ali, S. 2025. Search-Based Software Engineering in the Landscape of AI Foundation Models. *arXiv preprint arXiv:2505.19625*.

Senkerik, R.; Viktorin, A.; Kadavy, T.; Kovac, J.; Janku, P.; Pekar, L.; Guzowski, H.; Smolka, M.; Byrski, A.; and Pluhacek, M. 2025. Open and Closed Source Models for LLM-Generated Metaheuristics Solving Engineering Optimization Problem. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, 372–385. Springer.

Shi, Y.; Zhou, J.; Song, W.; Bi, J.; Wu, Y.; and Zhang, J. 2025. Generalizable heuristic generation through large

language models with meta-optimization. *arXiv preprint arXiv:2505.20881*.

Sim, K.; Renau, Q.; and Hart, E. 2025. Beyond the hype: Benchmarking llm-evolved heuristics for bin packing. In *Proceedings of the International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, 386–402. Springer.

Tang, K.; Liu, S.; Yang, P.; and Yao, X. 2021. Few-shots parallel algorithm portfolio construction via co-evolution. *IEEE Transactions on Evolutionary Computation*, 25(3): 595–607.

Uchoa, E.; Pecin, D.; Pessoa, A.; Poggi, M.; Vidal, T.; and Subramanian, A. 2017. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3): 845–858.

van Stein, N.; and Bäck, T. 2024. Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*.

Xie, Z.; Liu, F.; Wang, Z.; and Zhang, Q. 2025. LLM-driven neighborhood search for efficient heuristic design. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 1–8. IEEE.

Yao, S.; Liu, F.; Lin, X.; Lu, Z.; Wang, Z.; and Zhang, Q. 2025. Multi-objective evolution of heuristic using large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 27144–27152.

Yao, Y.; Liu, F.; Cheng, J.; and Zhang, Q. 2024. Evolve cost-aware acquisition functions using large language models. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, 374–390. Springer.

Ye, H.; Wang, J.; Cao, Z.; Berto, F.; Hua, C.; Kim, H.; Park, J.; and Song, G. 2024. ReEvo: Large language models as hyper-heuristics with reflective evolution. *Advances in Neural Information Processing Systems*, 37: 43571–43608.

Ye, H.; Xu, H.; Yan, A.; and Cheng, Y. 2025. Large language model-driven large neighborhood search for Large-scale MILP problems. In *Proceedings of the International Conference on Machine Learning*.

Zhang, R.; Liu, F.; Lin, X.; Wang, Z.; Lu, Z.; and Zhang, Q. 2024. Understanding the importance of evolutionary search in automated heuristic design with large language models. In *International Conference on Parallel Problem Solving from Nature*, 185–202. Springer.

Zhang, Y.; Cheng, R.; Yi, G.; and Tan, K. C. 2025. A Systematic Survey on Large Language Models for Evolutionary Optimization: From Modeling to Solving. *arXiv preprint arXiv:2509.08269*.

Zheng, K.; Wang, Y.; Liu, F.; Zhang, Q.; and Song, W. 2025a. CST-LLM: Enhancing Airfoil Parameterization Method with Large Language Model. *Aerospace Science and Technology*, 110548.

Zheng, Z.; Xie, Z.; Wang, Z.; and Hooi, B. 2025b. Monte Carlo tree search for comprehensive exploration in LLM-based automatic heuristic design. In *Proceedings of the International Conference on Machine Learning*.

Zhou, X.; Wu, X.; Feng, L.; Lu, Z.; and Tan, K. C. 2025. Design principle transfer in neural architecture search via large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 23000–23008.