LLM-Driven Automated Algorithm Design for Large-scale VRP: A Method for the CVRPLib BKS Challenge

Zhuoliang Xie[1], Hongyu Zhu[1], Rongsheng Chen[1], Zhenkun Wang*[1], Ruihao Zheng[1], Binbin Chen[2], and Xiang Xu[1]

[1]School of Automation and Intelligent Manufacturing, Southern University of Science and Technology, China
[2]Bytedance

Abstract

This document describes the methodology employed by our team for the CVRPLib Best Known Solution Challenge (BKS) Challenge (XL benchmark set). We propose a novel framework that leverages Large Language Models (LLMs) to automatically design and optimize key components of heuristic algorithms. Given the computationally expensive nature of the 1,000 to 10,000 customer instances, our approach focuses on using LLMs to refine local search operators and decomposition strategies within a high-performance open-source base solver. Preliminary tests show that our method can effectively discover superior configurations for large-scale Capacitated Vehicle Routing Problems (CVRP).

1 Introduction

The Capacitated Vehicle Routing Problem (CVRP) is a fundamental combinatorial optimization challenge with extensive industrial applications. The CVRPLib BKS introduces the newly released *XL benchmark set*, which contains 100 instances with dimensions ranging from 1,000 to 10,000 customers. These instances pose significant challenges to existing exact and heuristic methods due to the exponential growth of the search space and the stringent balance required between computational efficiency and solution quality.

Traditionally, the design of high performance heuristics, such as those within the Ruin-and-Recreate framework, relies heavily on human expertise and trial-and-error. However, manual design often struggles to adapt to the diverse topological characteristics and the massive scale of the XL dataset. To bridge this gap, we propose an LLM-driven Automated Algorithm Design (AAD) framework.

Our approach leverages the code-generation and logical reasoning strengths of Large Language Models (LLMs) to automate the design of key algorithmic components. Specifically, we focus on evolving the ruin operators within a state-of-the-art Adaptive Iterated Local Search II (AILS II) backbone. By treating heuristic code as a genotype that can be evolved through an LLM-based crossover and mutation process, we enable the discovery of non-intuitive, problem-specific removal strategies. This work represents a shift from manual tuning to automated evolution, providing a robust and scalable solution for large-scale CVRP optimization.

*Corresponding author: wangzk3@sustech.edu.cn

2 Methodology

2.1 basic framework

The method we propose is a hybrid of the current sota heuristic combined with LLM. The backbone algorithm is Adaptive Iterated Local Search II (AILS II) [2].

2.2 LLM-driven AHD

In the AILS-II framework, we identify and automatically design the key part in iterated local search. Specifically, we design the heuristic for the ruin operator, i.e., the heuristic determines which nodes are removed given a current solution and features in each iteration.

As illustrated in Figure 1, we adopt EoH [1] for Automatic Heuristic Design (AHD) of the ruin heuristic. EoH maintains and evolves a population of heuristics. In each evolution population, four steps are performed: 1) prompt construction, 2) heuristic generation, 3) evaluation, and 4) population update.

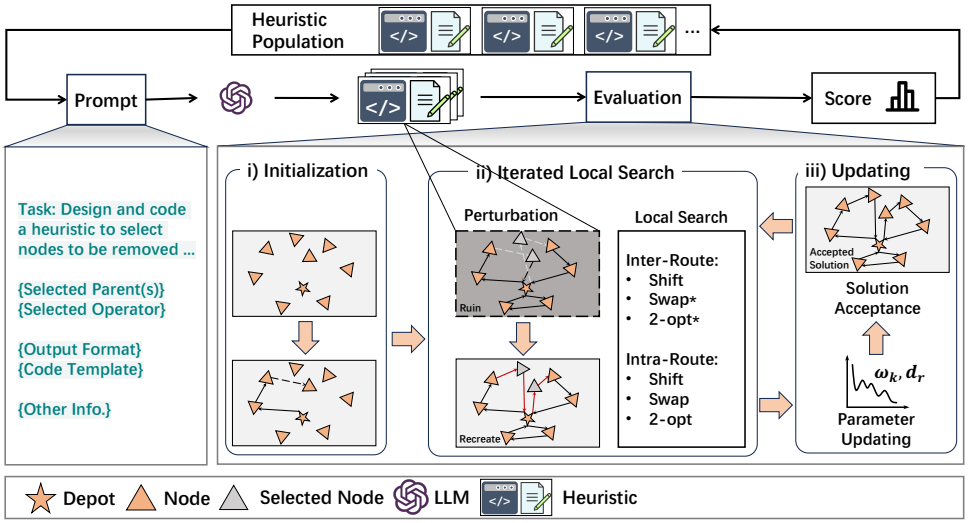

Figure 1: The AILS II-AHD pipeline: (1) LLM-driven evolutionary computation generates ruin heuristics; (2) each candidate heuristic is evaluated via AILS, including initialization, iterated local search and updating; (3) the population is updated based on fitness of the heuristic.

Prompt Construction The first step involves constructing a prompt that includes the task description, the required code template format, and other relevant details. The initial parent seed heuristic is manually designed to serve as a starting point. During subsequent iterations, parent heuristics are selected from the population based on a probability distribution defined in Equation 1.

$$q_i = \frac{f^{s_i}}{\sum_{k \in n^s} f^{s_k}}. \tag{1}$$

Heuristic Generation Once the prompt is constructed, it is sent to the LLM to generate offspring heuristics. The LLM employs four distinct operators to create new heuristics, each tailored to introduce specific variations or improvements.

Evaluation To evaluate the newly generated heuristics, each is integrated into the original evaluation method by replacing the corresponding component. The heuristic is then tested on a carefully selected set of instances, and its fitness is calculated as the average performance across all instances. This rigorous evaluation process ensures that only high-performing heuristics are retained, maintaining the quality and effectiveness of the population.

Population Update After evaluation, the population is updated with the offspring heuristics. To maintain diversity and ensure robust exploration, heuristics with better fitness are retained in the population.

3 Computational Setup

The final solutions are computed on the following hardware:

- **CPU:** Cluster of 100 $\times$ Intel Xeon Platinum processors (2.1 GHz, 24 Cores each)
- **RAM:** 500 GB
- **OS:** CentOS Linux 7

We utilize the full 24 days of the competition to refine the solutions, dedicating approximately 240 hours of computation time per instance for the final run.

4 Conclusion

This document outlines our team's approach for the CVRPLib BKS Challenge. By combining the robustness of AILSII with the creative potential of Large Language Models, we aim to uncover high-quality solutions for the new XL benchmark set.

References

[1] Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Forty-first International Conference on Machine Learning*, 2024.

[2] Vinícius R Máximo, Jean-François Cordeau, and Mariá CV Nascimento. Ails-ii: An adaptive iterated local search heuristic for the large-scale capacitated vehicle routing problem. *INFORMS Journal on Computing*, 36(4):974–986, 2024.