Configuring LLM-Generated Heuristics via Instance-Specific Deep Reinforcement Learning

Xiaowei Liu, Kevin Tierney
University of Vienna

Introduction

While large language models (LLMs) have shown promise in automated heuristic discovery, configuring parameters of generated heuristics remains challenging. This poster proposes an instance-specific **Deep Reinforcement Learning (DRL)** approach to configure **LLM-generated heuristic operators**. Our method integrates LLM-based heuristic refinement with local search to minimize parameter changes and is evaluated on Vehicle Routing Problems (VRPs).

Background

VRPAgent

VRPAgent (Hottung et al., 2025) is a new framework that uses LLMs to design heuristic operators for a large neighborhood search (LNS). It employs a genetic algorithm for heuristic discovery featuring elitism with biased crossover for improved exploitation.

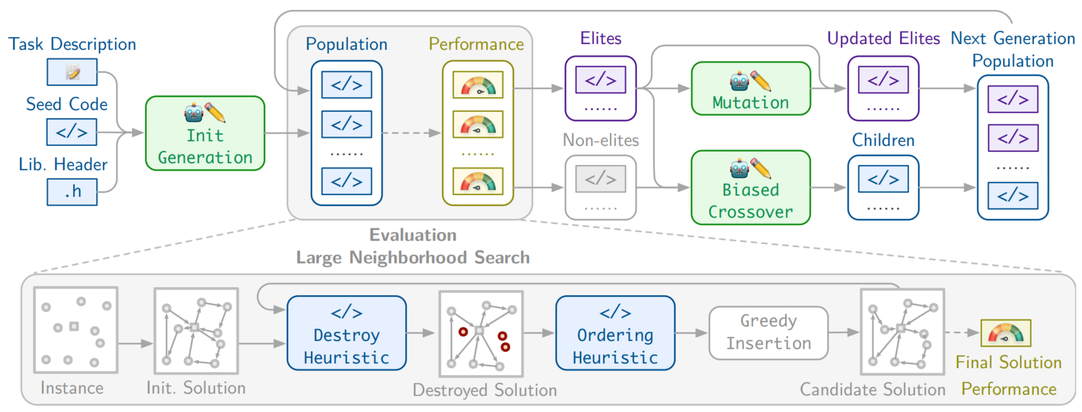


Figure 1: VRPAgent overview (adapted from Hottung et al., 2025).

OPTICAT

OPTICAT (Schede et al., 2025) is an instance-specific Algorithm Configuration (AC) method based on DRL. The DRL agent proposes configurations, which are evaluated through target algorithm execution; the agent is then updated based on the resulting feedback using an augmented Proximal Policy Optimization (PPO) algorithm.

Idea

Can automated AC for LLM-driven heuristic discovery achieve superior performance compared to the state-of-the-art?

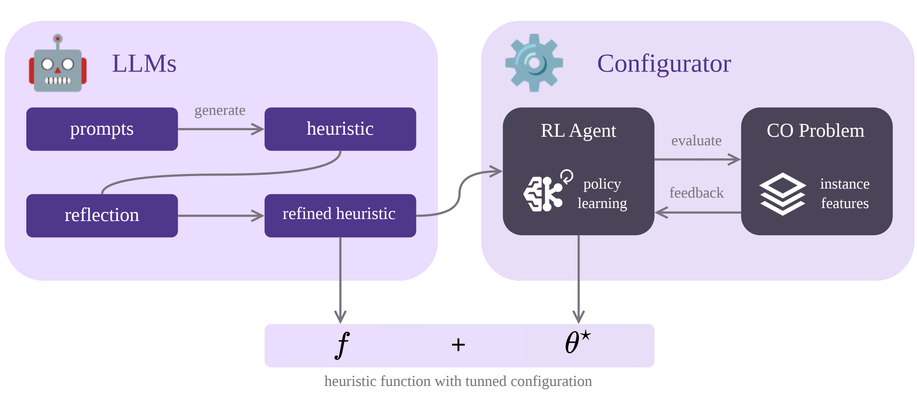


Figure 2: An illustrative pipeline: AC for LLM-based metaheuristics.

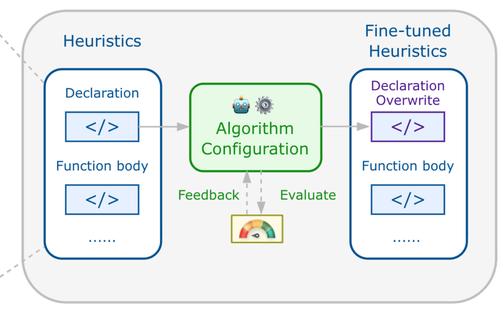
Configuring Discovered Heuristics

The configurations of LLM-generated heuristics are "guesses" and can be optimized in a one-size-fits-all approach (e.g., irace, López-Ibáñez et al., 2016) or an instance-specific manner. OPTICAT can handle these mixed discrete and continuous configuration spaces to further improve performance with an additional set of instances.

```
// Customer selection function generated by LLM
std::vector<int> select_by_llm_1(const Solution& sol) {
  const int MIN_CUSTOMERS_TO_REMOVE = 15;
  const int MAX_CUSTOMERS_TO_REMOVE = 25;
  //... Configurations definitions ...

  std::unordered_set<int> selectedCustomers;
  int numCustomersToRemove =
getRandomNumber(MIN_CUSTOMERS_TO_REMOVE,
MAX_CUSTOMERS_TO_REMOVE);
  //... Main code block logic ...

  return std::vector<int>(selectedCustomers.begin(),
selectedCustomers.end());
}
```



Research Question #1: *Can LLMs generate better heuristics by performing crossover with previously fine-tuned heuristics compared to the VRPAgent baseline?*

Proposed Framework

Instead of using a generic algorithm, we propose using OPTICAT to evaluate the runtime and quality of generated heuristics. This crossover is designed to preserve the structure of the configuration space. Finally, accepted heuristics proceed to the next iteration.

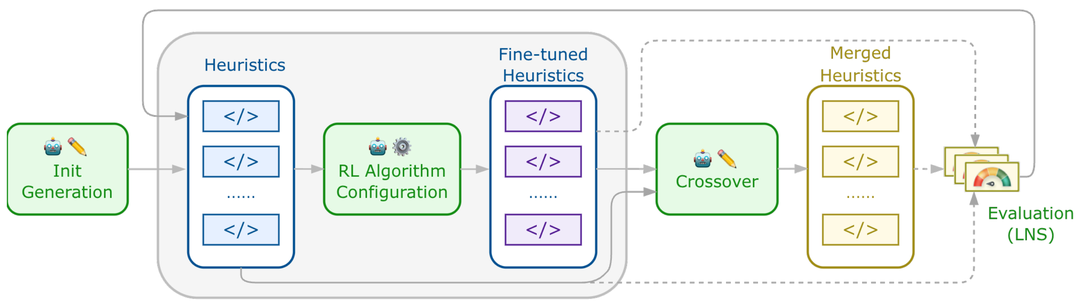


Figure 4: Overview of the proposed framework.

Research Question #2: *Can heuristics generated in early iterations, when tuned via AC, outperform those from later iterations with default configurations?*

Acknowledgments

Many thanks to Dr André Hottung for useful discussions.

References

Hottung et al. (2025). VRPAgent: LLM-Driven Discovery of Heuristic Operators for Vehicle Routing Problems. *Preprint*.

Schede et al. (2025). Deep reinforcement learning for instance-specific algorithm configuration. *GECCO 2025*, 1190-1198.

López-Ibáñez et al. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Oper. Res. Perspect*, 3, 43-58.

Contact: xiaowei.liu@univie.ac.at