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

**Xiaowei Liu** and Kevin Tierney

July 16, 2026

Department of Business Decisions and Analytics

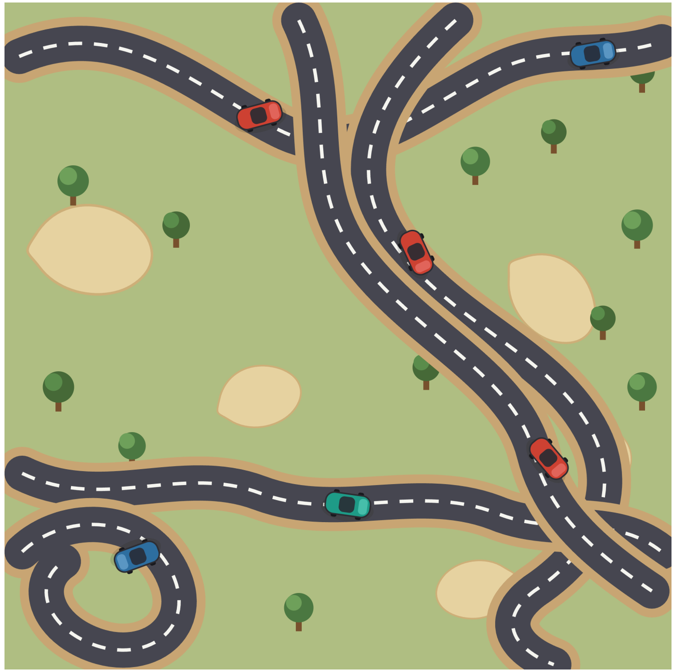
# Automated Heuristic Discovery with VRPAgent

# Background: Routing Problems

## Vehicle Routing Problem (VRP)

Visit $n$ customers with $m$ vehicles such that

- each customer is visited exactly once
- the total travel cost is minimized.



# Background: Routing Problems

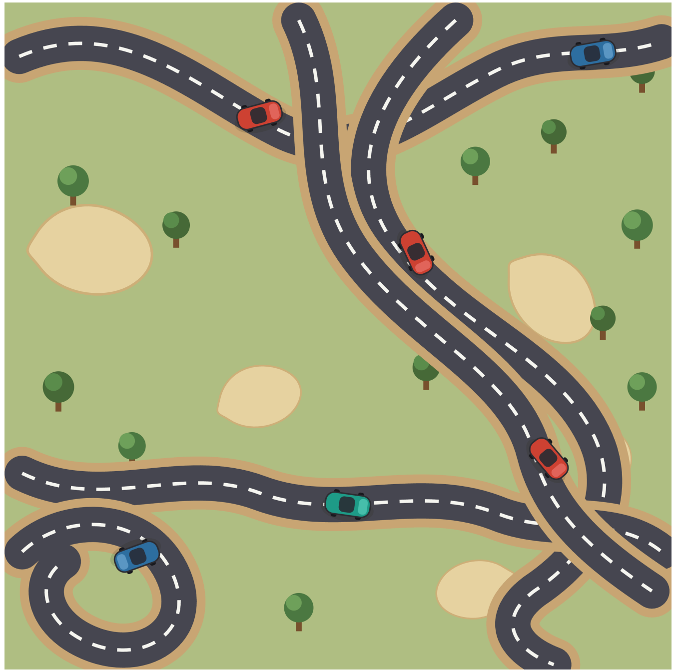
### Vehicle Routing Problem (VRP)

Visit $n$ customers with $m$ vehicles such that

- each customer is visited exactly once
- the total travel cost is minimized.

And its variants:

- **Capacitated VRP (CVRP)**
- **VRP with Time Windows (VRPTW)**
- **Prize-Collecting VRP (PCVRP)**



# Large Neighborhood Search (LNS)

The core idea of the LNS algorithm (Shaw, 1998) is to **“destroy and repair”**.

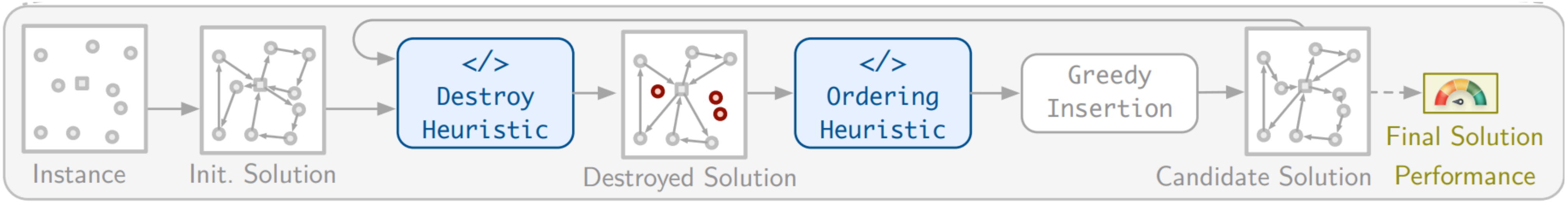
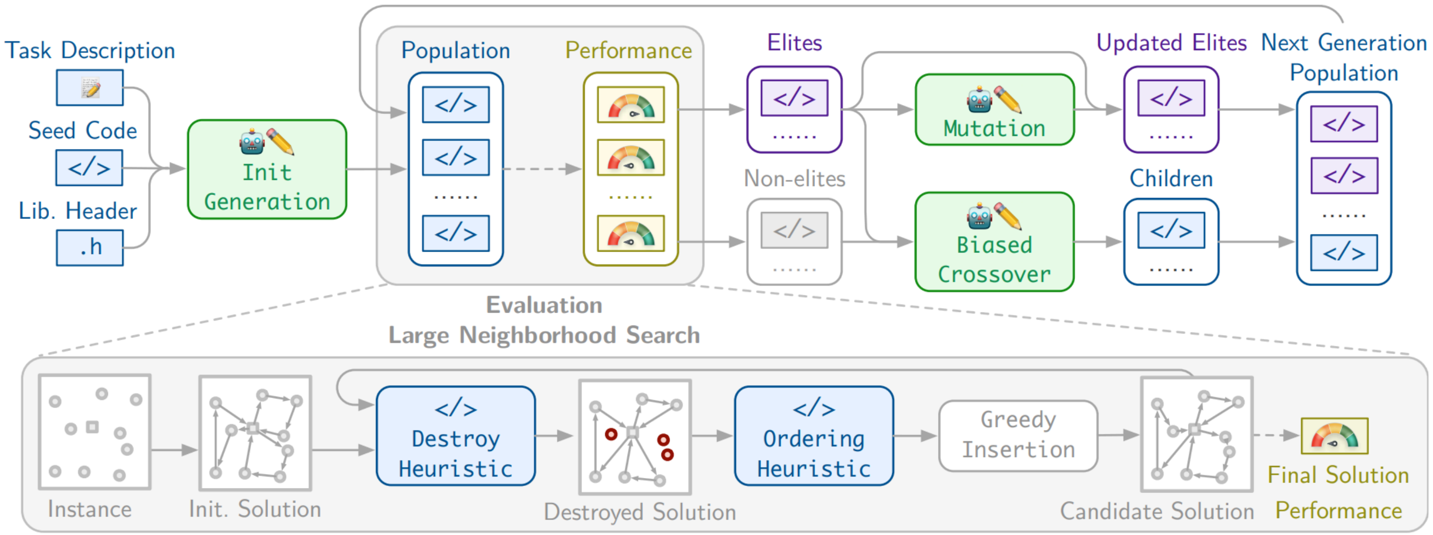



Figure adapted from (Hottung et al., 2025a)

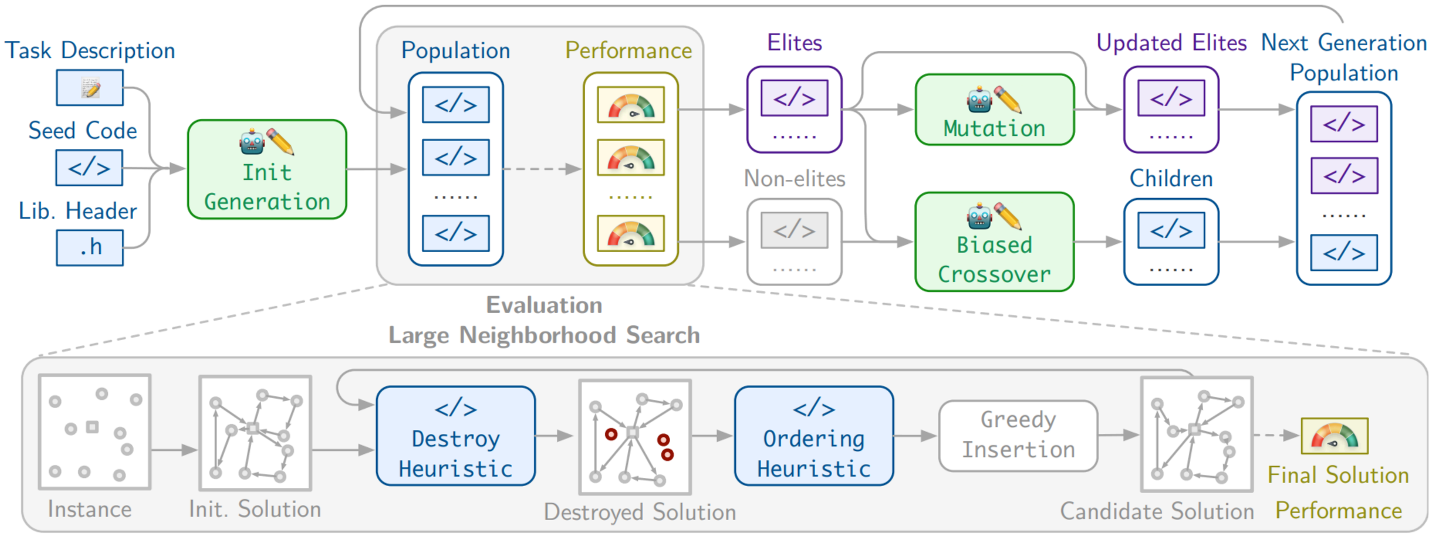
# VRPAgent: Automated Heuristic Discovery by LLMs




VRPAgent pipeline overview. Figure adapted from (Hottung et al., 2025a)

1️⃣ Initial heuristics generation

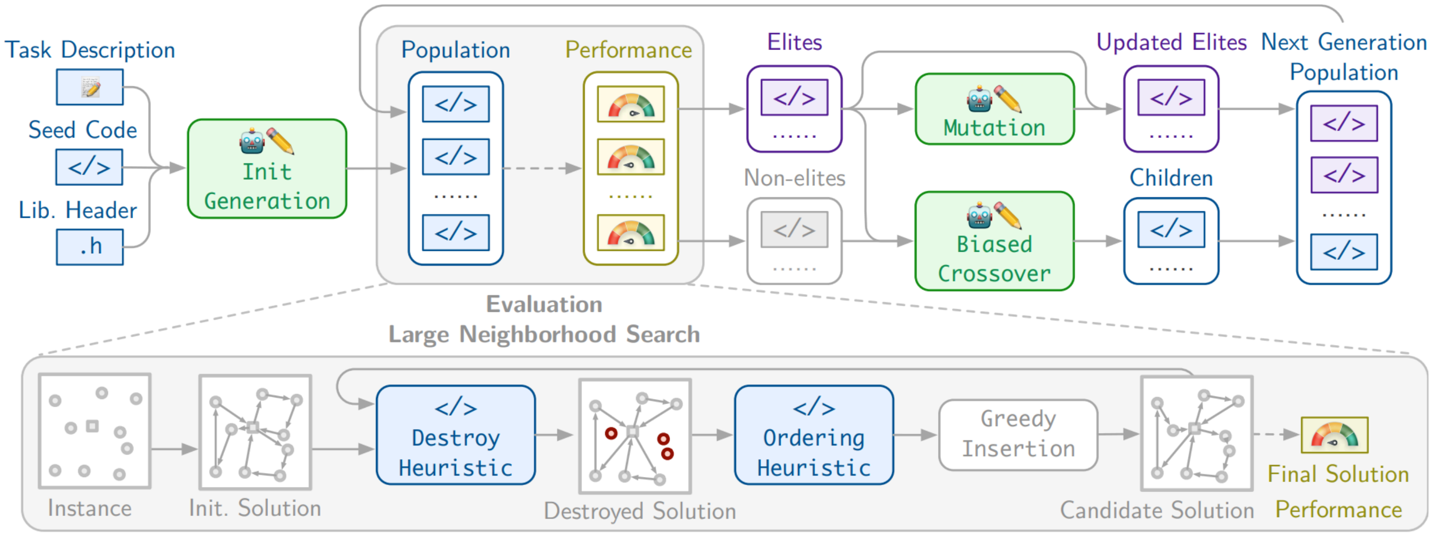
# VRPAgent: Automated Heuristic Discovery by LLMs




VRPAgent pipeline overview. Figure adapted from (Hottung et al., 2025a)

1️⃣ Initial heuristics generation 2️⃣ **Evaluation via LNS**

# VRPAgent: Automated Heuristic Discovery by LLMs

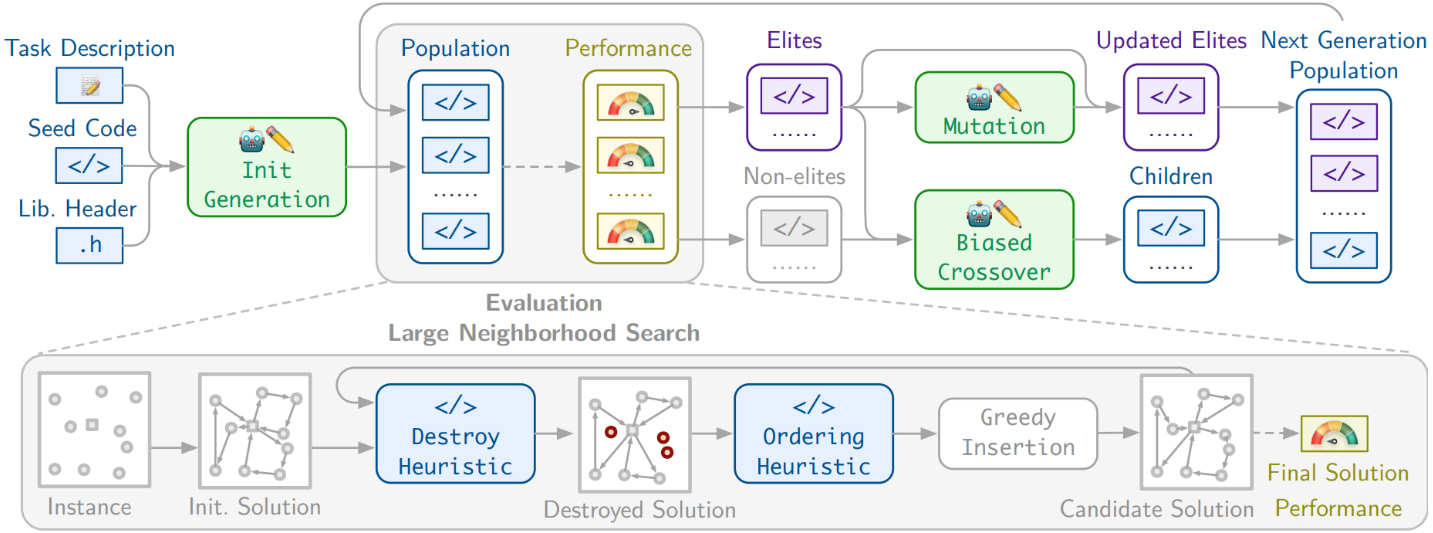



VRPAgent pipeline overview. Figure adapted from (Hottung et al., 2025a)

1️⃣ Initial heuristics generation 2️⃣ Evaluation via LNS 3️⃣ **Genetic Algorithm**

- Improve each elite via *mutation*
- *Crossover* of elites and non-elites

# VRPAgent: Automated Heuristic Discovery by LLMs




VRPAgent pipeline overview. Figure adapted from (Hottung et al., 2025a)

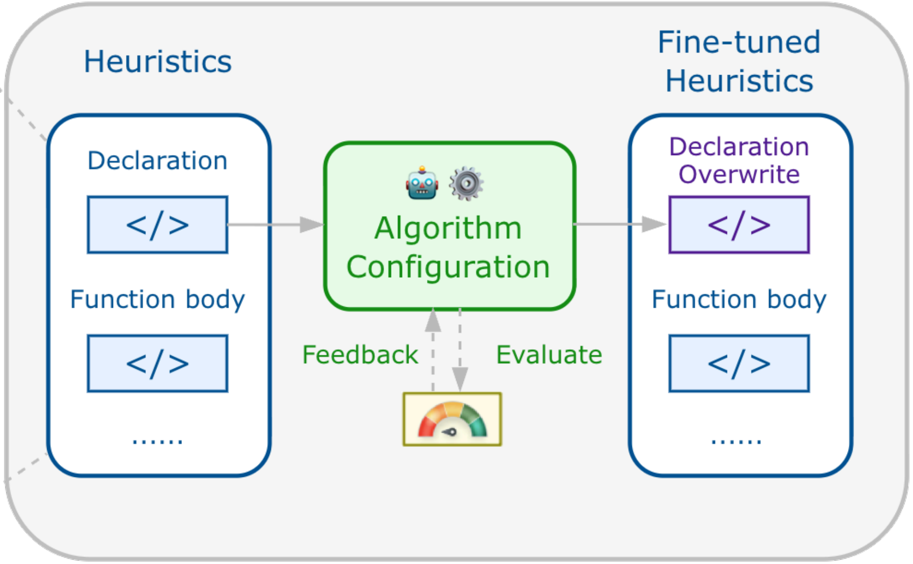
1️⃣ Initial heuristics generation 2️⃣ Evaluation via LNS 3️⃣ Genetic Algorithm 4️⃣ Re-evaluation

# Heuristics Discovered by LLMs

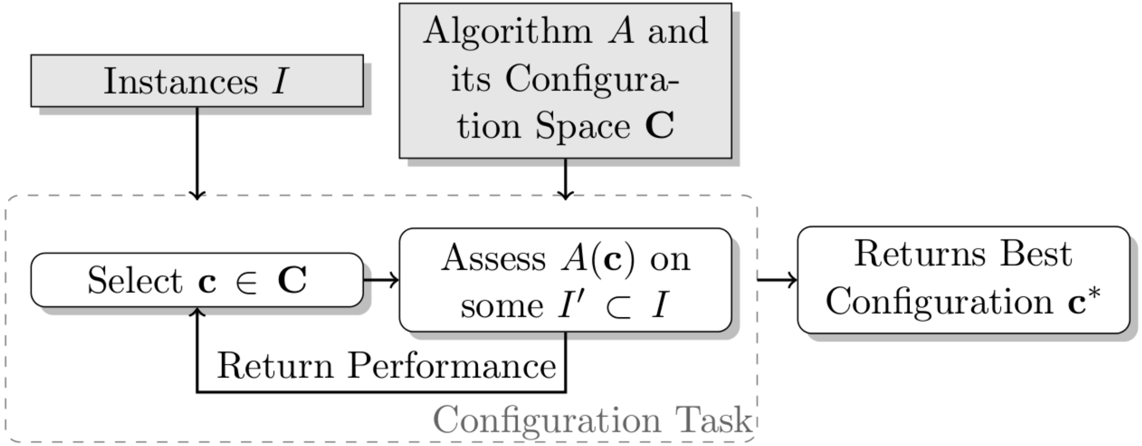
```
// Customer selection function generated by LLM
std::vector<int> select_by_llm_1(const
Solution& sol) {
  const int MIN_CUSTOMERS_TO_REMOVE = 15;
  const int MAX_CUSTOMERS_TO_REMOVE = 25;
  //... Configurations definitions ...

  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());
}
```



# Algorithm Configuration




Workflow of Algorithm Configuration

Figure adapted from AutoML.org.

# Examples of Parameters in Heuristics

- Float parameters

  Example: `prob_tour_neighbor` and `prob_geo_neighbor`.

- Integer parameters

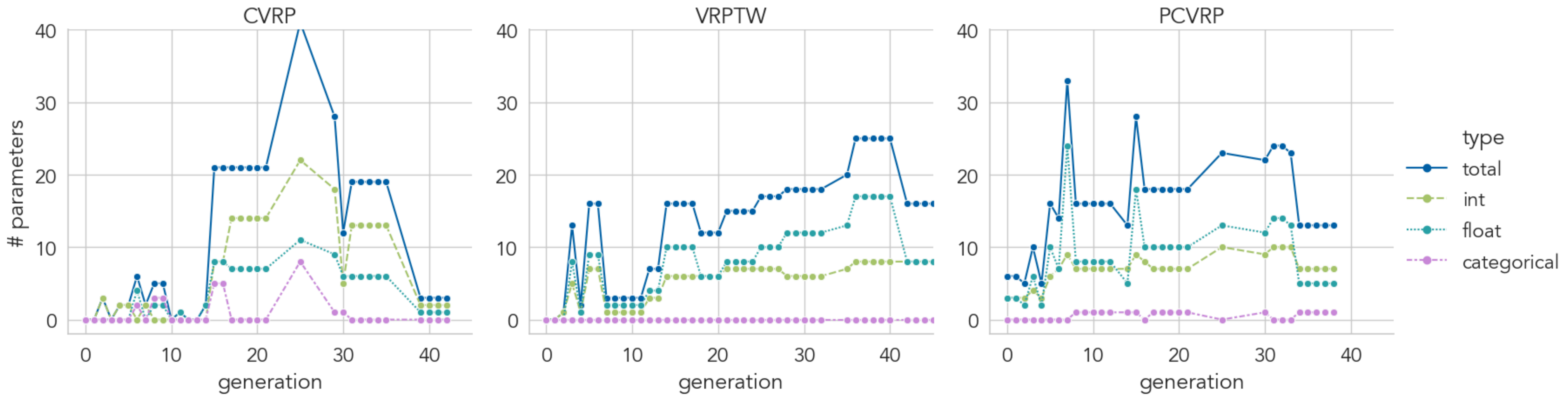
  Example: `min_customers_to_remove` and `max_customers_to_remove`.

- “Categorical” parameters (discarded)

  Most of them are dependencies, e.g., in the form of `y = x + 1`.

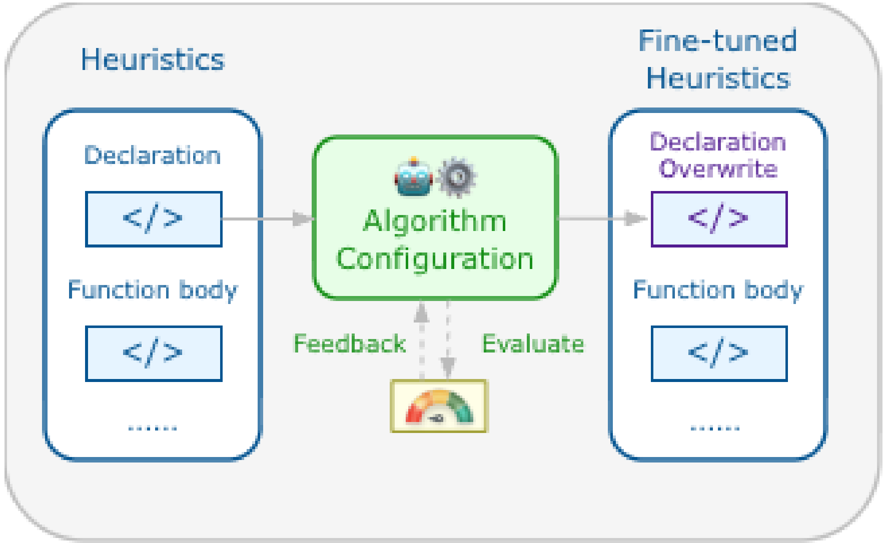
# Examples of Parameters in Heuristics

The number of parameters increases in early generations and decreases later.



# Configuration Setup

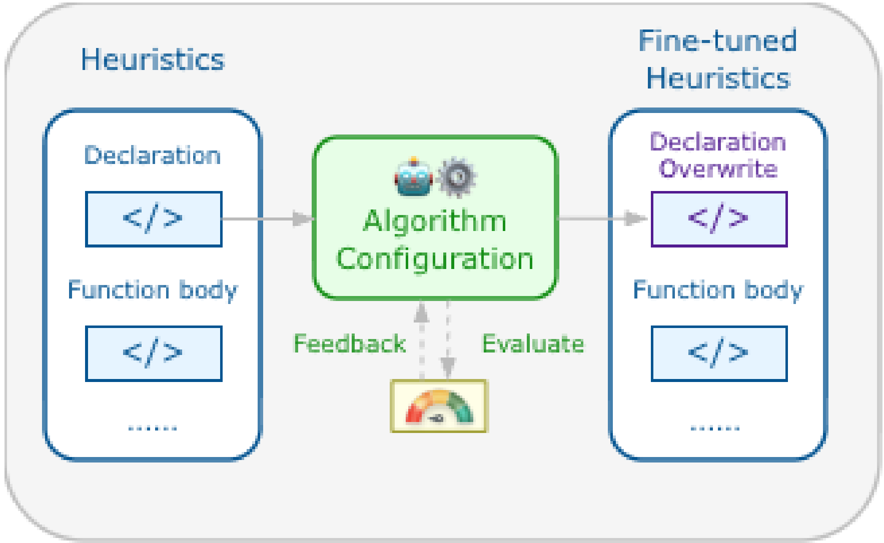
- Configurator: We use the `irace` package (López-Ibáñez et al., 2016) with a total budget of **3500** experiments on **256** training instances (never fully utilized).



# Configuration Setup

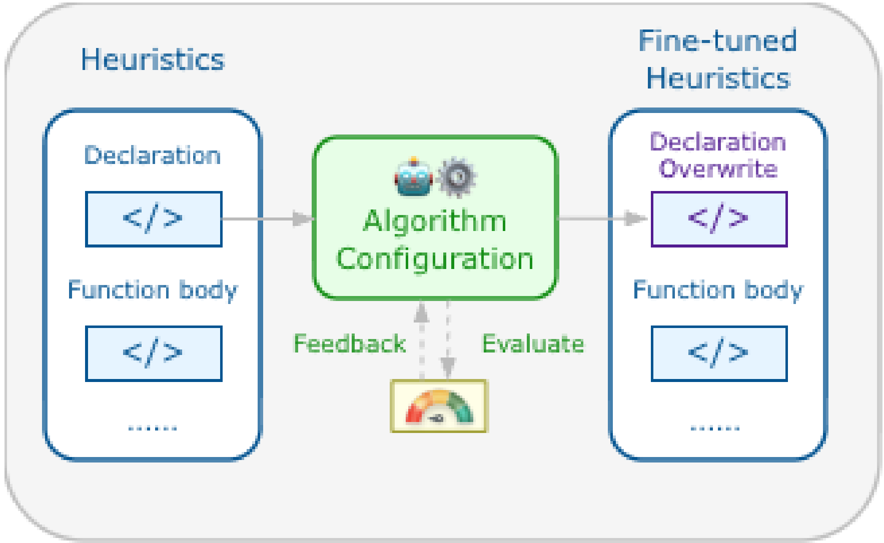
- Configurator: We use the `irace` package (López-Ibáñez et al., 2016) with a total budget of **3500** experiments on **256** training instances (never fully utilized).
- On average, it takes ~11 hours (VRPTW) and ~20 hours (CVRP and PCVRP) to fine-tune.

* Each LNS search takes **20** seconds and `irace` can use **16** cores.



# Configuration Setup

- Configurator: We use the `irace` package (López-Ibáñez et al., 2016) with a total budget of **3500** experiments on **256** training instances (never fully utilized).
- On average, it takes ~11 hours (VRPTW) and ~20 hours (CVRP and PCVRP) to fine-tune.
- Finally, we evaluate on 64 training + 64 testing instances (500 customers).



*Can LLM-generated heuristics be fine-tuned?*

*Can LLM-generated heuristics be fine-tuned?*

Yes.

*Can LLM-generated heuristics be fine-tuned?*

Yes.

*Can the best discovered LLM-generated heuristics be fine-tuned?*

*Can LLM-generated heuristics be fine-tuned?*

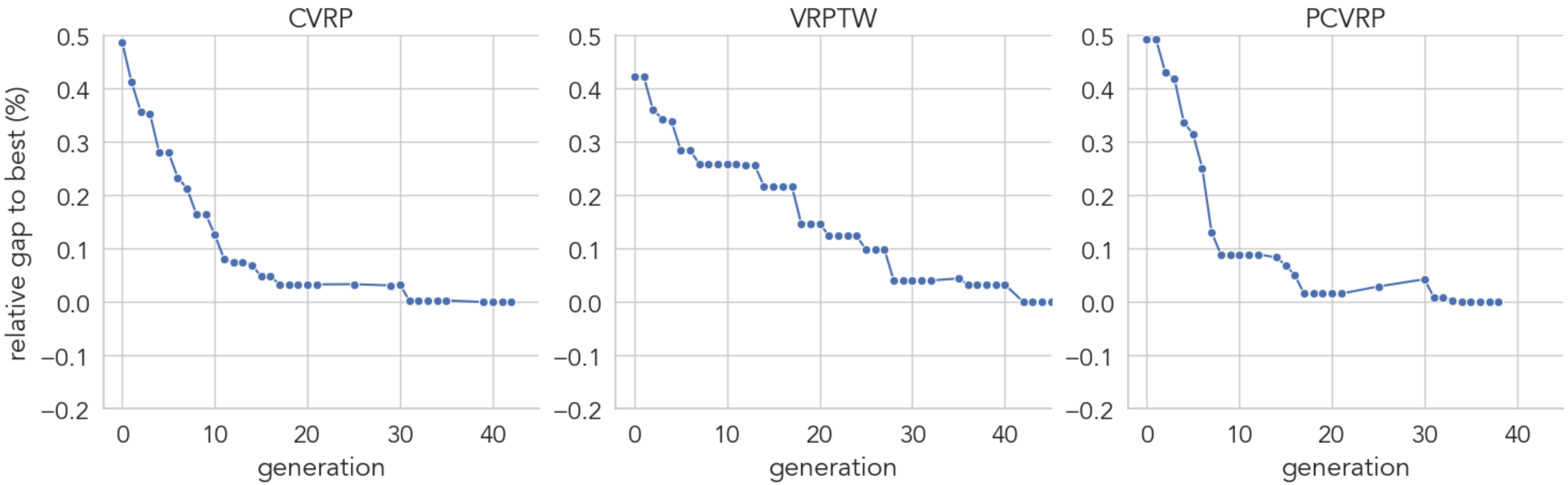
Yes.

*Can the best discovered LLM-generated heuristics be fine-tuned?*

Yes.

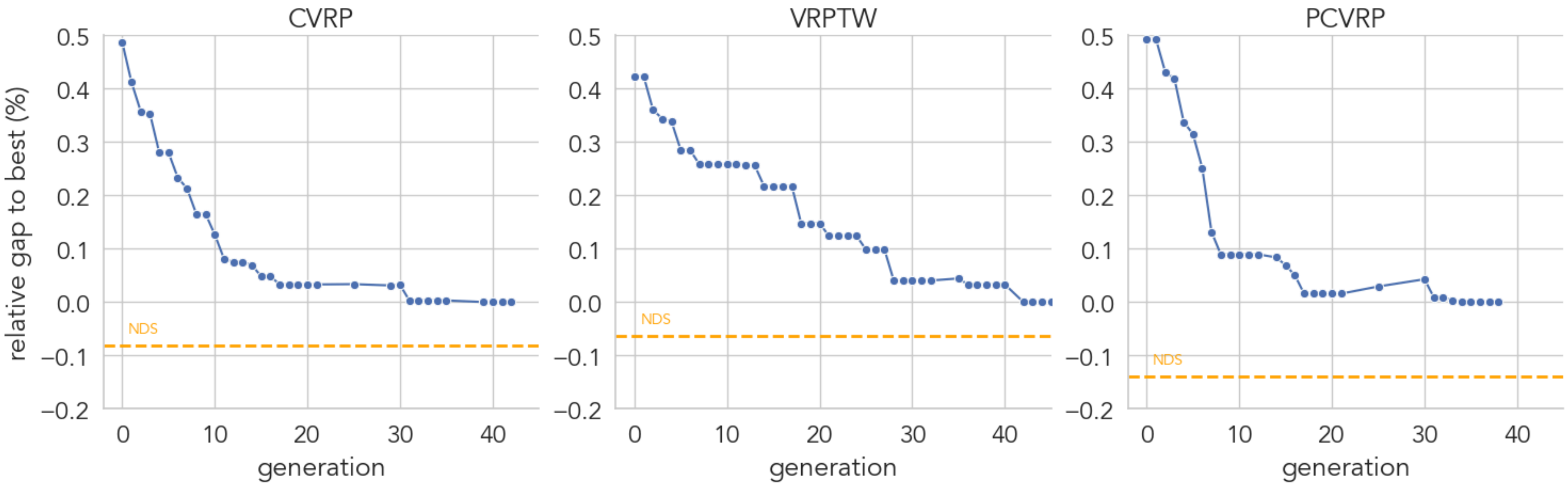
# Fine-tuning Best Heuristics

* *"Best"*: Evaluated on 64 instances, see (Hottung et al., 2025a).



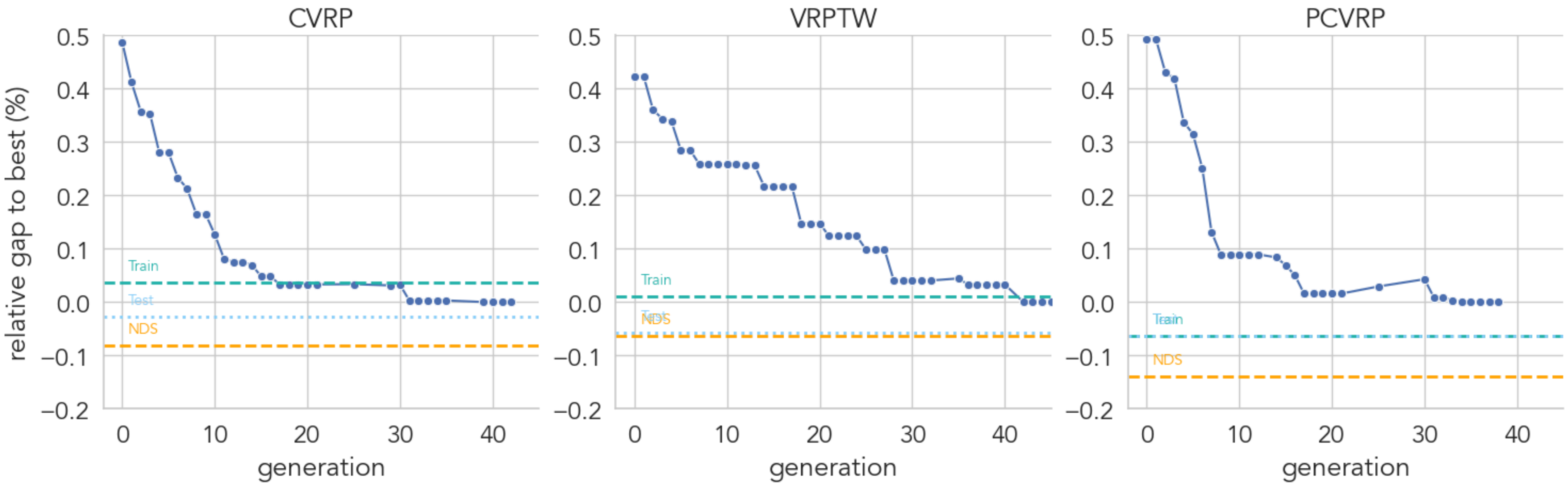
# Fine-tuning Best Heuristics

*NDS: Neural Deconstruction Search baseline (Hottung et al., 2025b).



# Fine-tuning Best Heuristics

*Train and Test: Fine-tuned parameters + best discovered heuristic function on training & testing instances



# Fine-tuning Best Heuristics

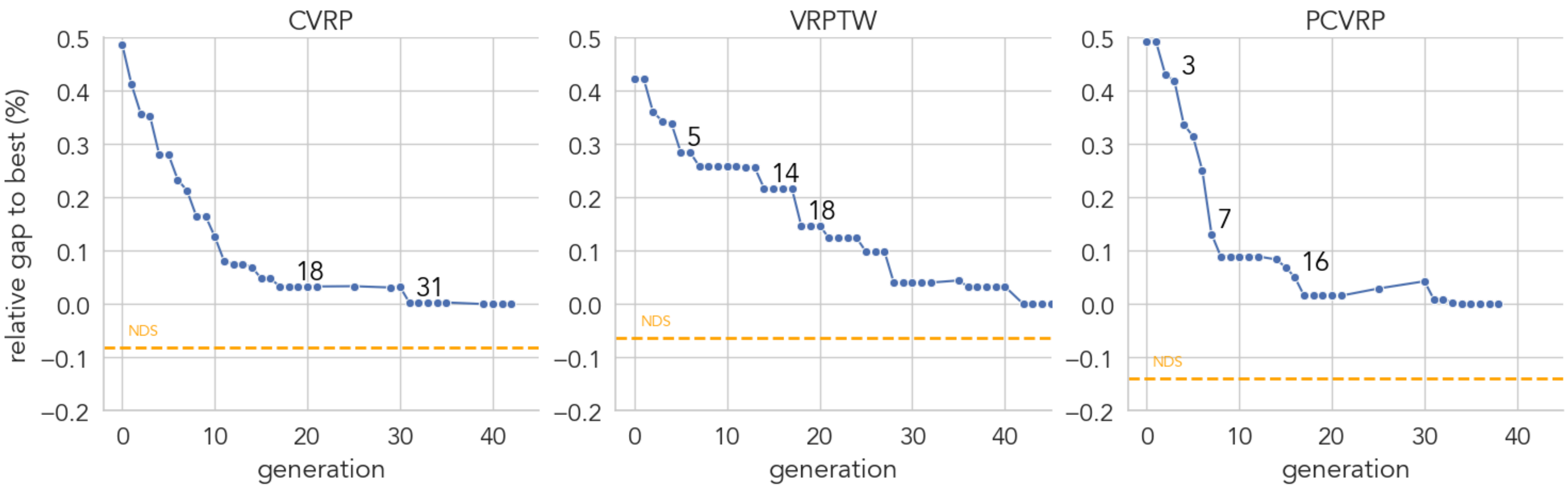
- The gaps are very *small*.
- However, even with fine-tuned parameters, the best discovered heuristic only approaches, but does not beat the NDS baseline.

*Can early generations be tuned to beat later generations?*

*Can early generations be tuned to beat later generations?*

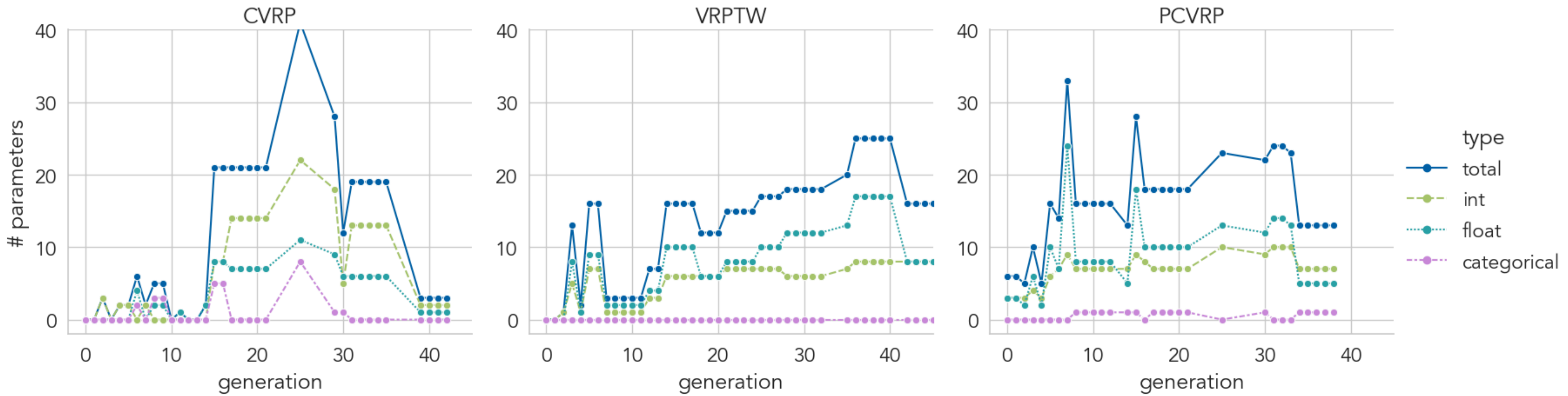
Well, it depends…

# Selecting Heuristics with Early Generations



# Selecting Heuristics with Early Generations

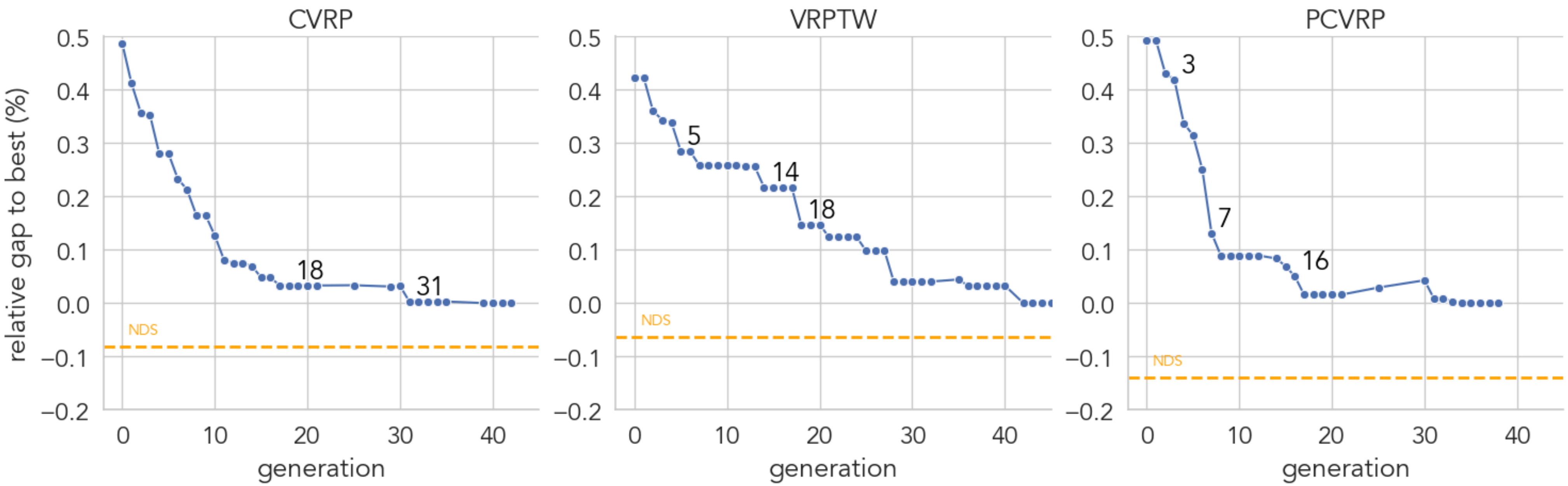
Recall the # parameters of the discovered heuristics.



# Selecting Heuristics with Early Generations

Similar parameter structure (CVRP 18 vs. 31).

Different parameter structure (VRPTW and PCVRP).



# Fine-tuning Heuristics with Early Generations

We take one run (seed 71) of `irace` and compare the fine-tuned heuristics against the best code with default parameters. Wilcoxon test p-values are reported ($\alpha = 0.05$).

| Problem (dataset) | Heuristic | % Gaps to (best + default) | $p$-value | Significant? |
|---|---|---|---|---|
| CVRP (test) | gen 18 | −0.056 | 0.126 | N |
| CVRP (test) | gen 31 | −0.068 | 0.066 | N |

# Fine-tuning Heuristics with Early Generations

We take one run (seed 71) of `irace` and compare the fine-tuned heuristics against the best code with default parameters. Wilcoxon test p-values are reported ($\alpha = 0.05$).

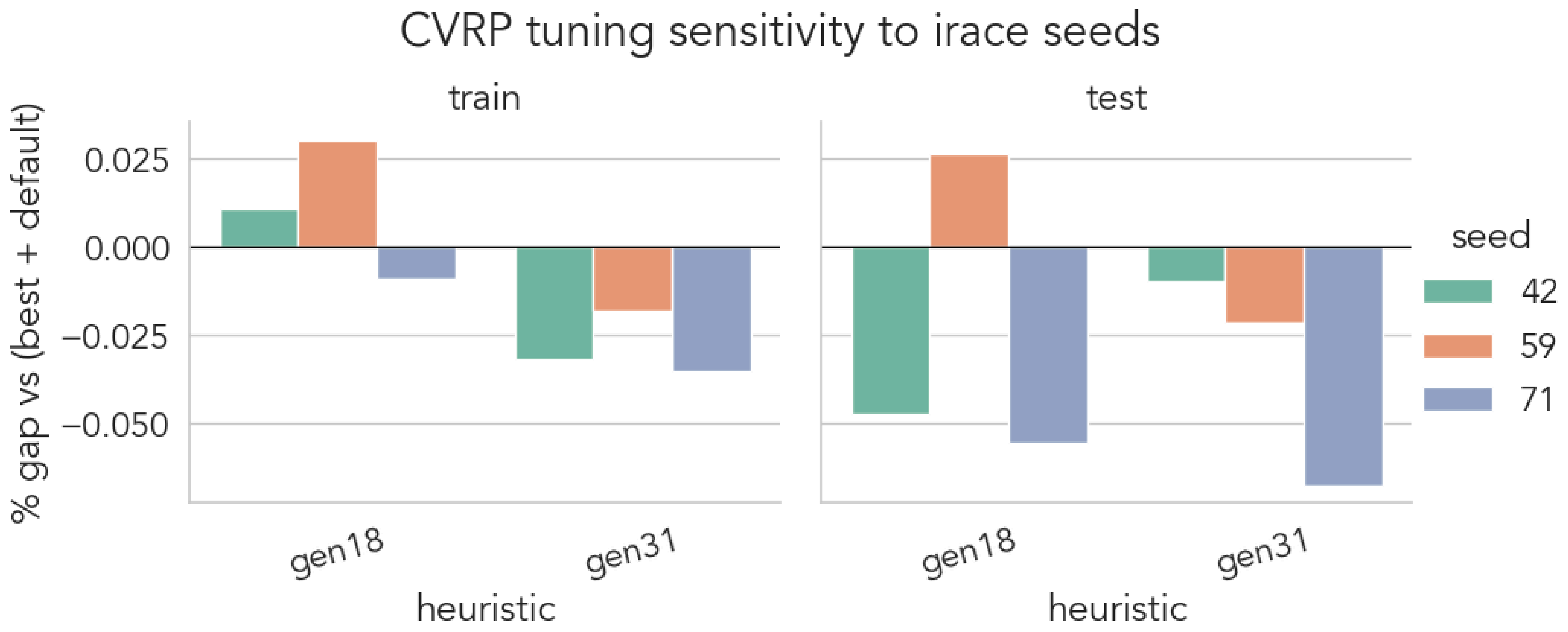
| Problem (dataset) | Heuristic | % Gaps to (best + default) | $p$-value | Significant? |
|---|---|---|---|---|
| VRPTW (test) | gen 5 | −0.008 | - | - |
| VRPTW (test) | gen 14 | 0.063 | - | - |
| VRPTW (test) | gen 18 | 0.135 | - | - |

# Fine-tuning Heuristics with Early Generations

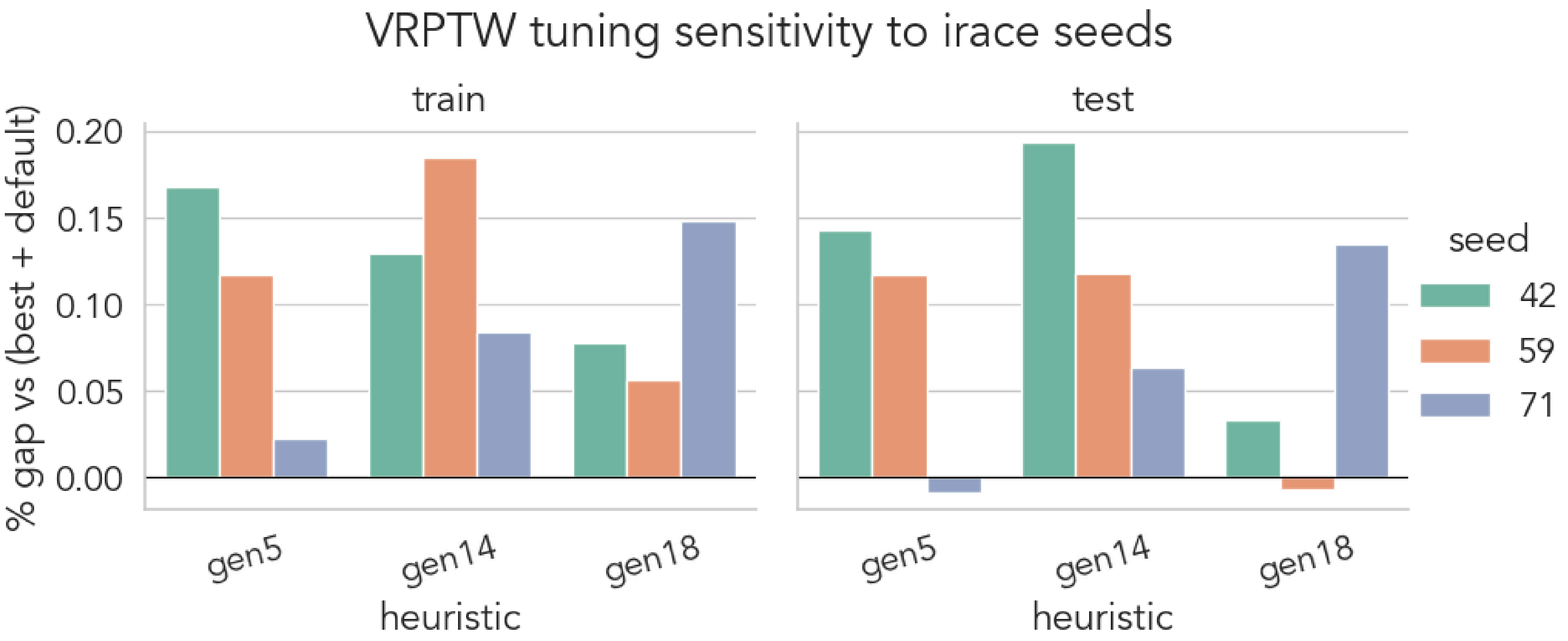
We take one run (seed 71) of `irace` and compare the fine-tuned heuristics against the best code with default parameters. Wilcoxon test p-values are reported ($\alpha = 0.05$).

| Problem (dataset) | Heuristic | % Gaps to (best + default) | $p$-value | Significant? |
|---|---|---|---|---|
| PCVRP (test) | gen 3 | 0.254 | - | - |
| PCVRP (test) | gen 7 | −0.045 | 0.075 | N |
| PCVRP (test) | gen 16 | 0.042 | - | - |

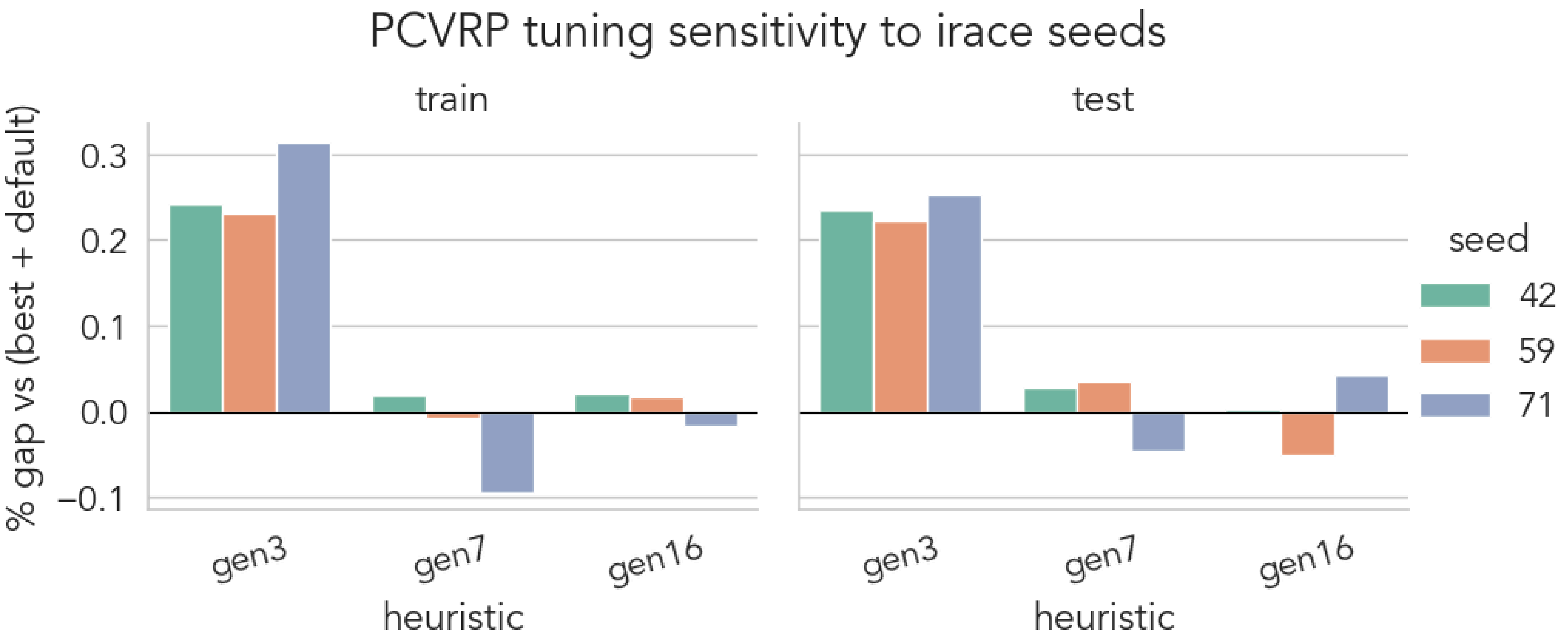
# Sensitivity Analysis of `irace` Runs



# Sensitivity Analysis of `irace` Runs



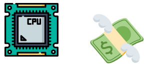
# Sensitivity Analysis of `irace` Runs



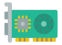
# Future work

# Efficient Heuristic Discovery

Local **LLM** query:

- 
- 
- 

**GA** on instances:

- 
- 
- 

# Efficient Heuristic Discovery

Local **LLM** query:

- CPU 💸
- 🖴 💸💸💸
- ⏳ 💸

**GA** on instances:

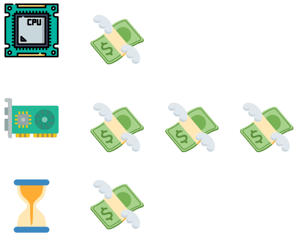
- CPU 💸💸💸
- 🖴
- ⏳ 💸💸

**LLM** + **GA**:

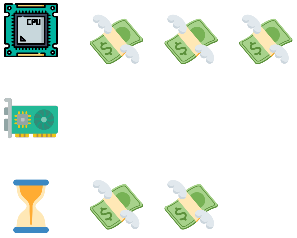
- CPU 💸💸💸💸
- 🖴 💸💸💸
- ⏳ 💸💸💸

# Efficient Heuristic Discovery

Local **LLM** query:



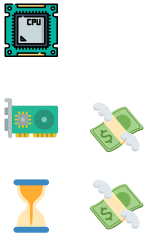
GA on instances:



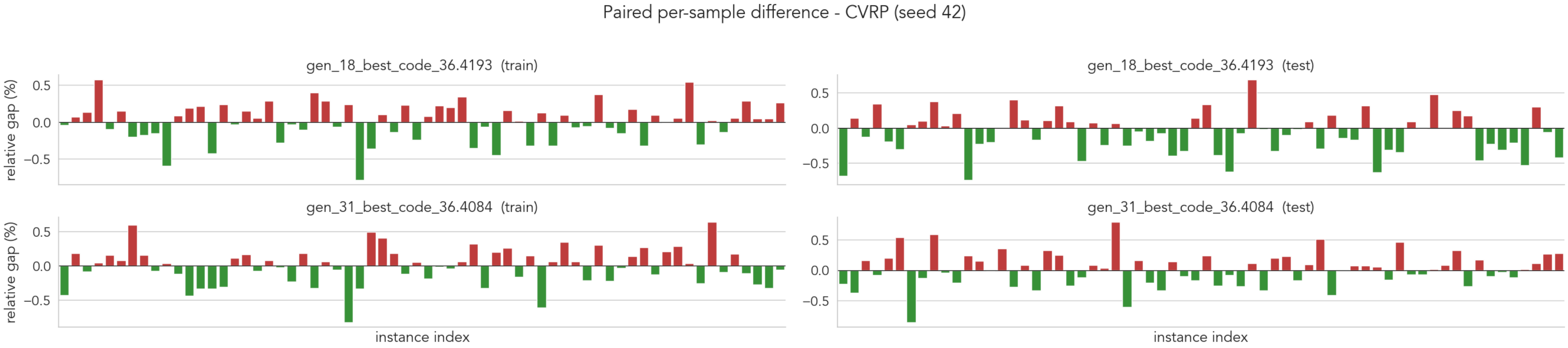
LLM + GA:

RL training stage:

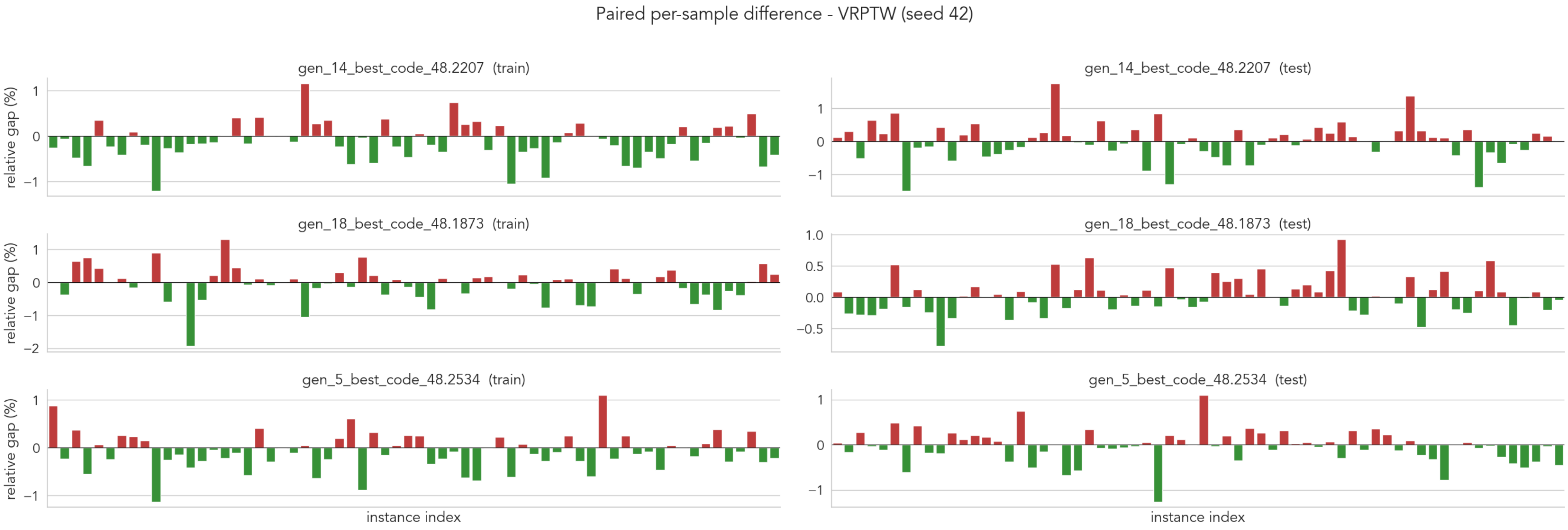
After convergence:



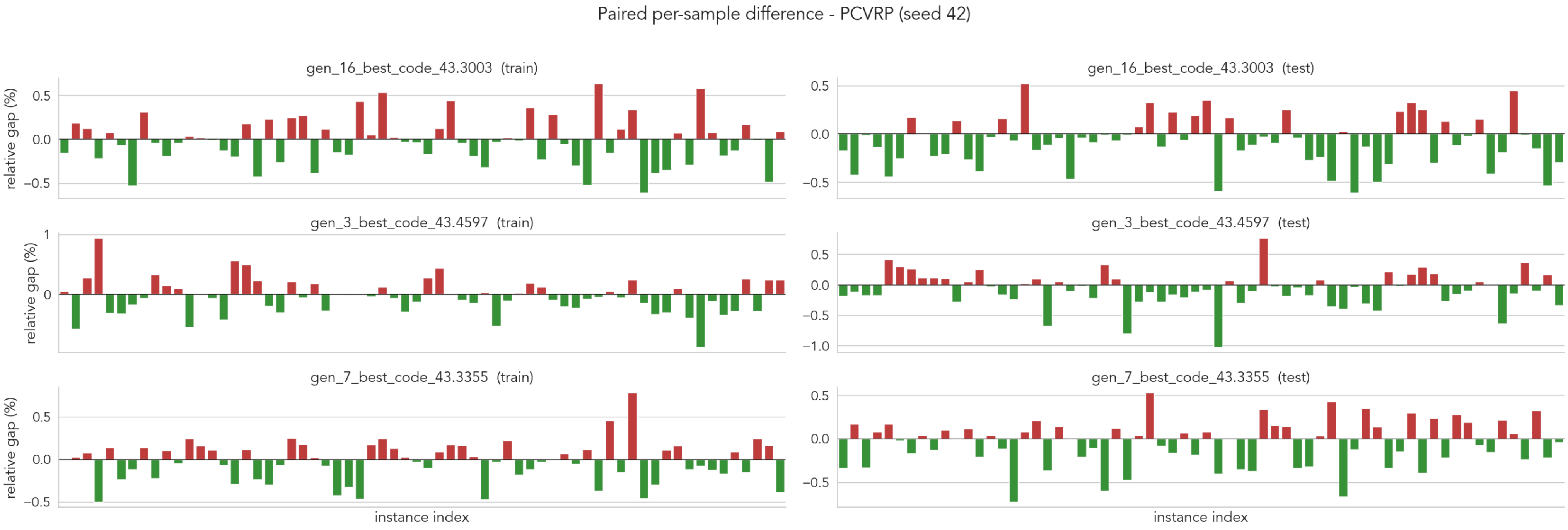
# Instance-specific Algorithm Configuration



# Instance-specific Algorithm Configuration

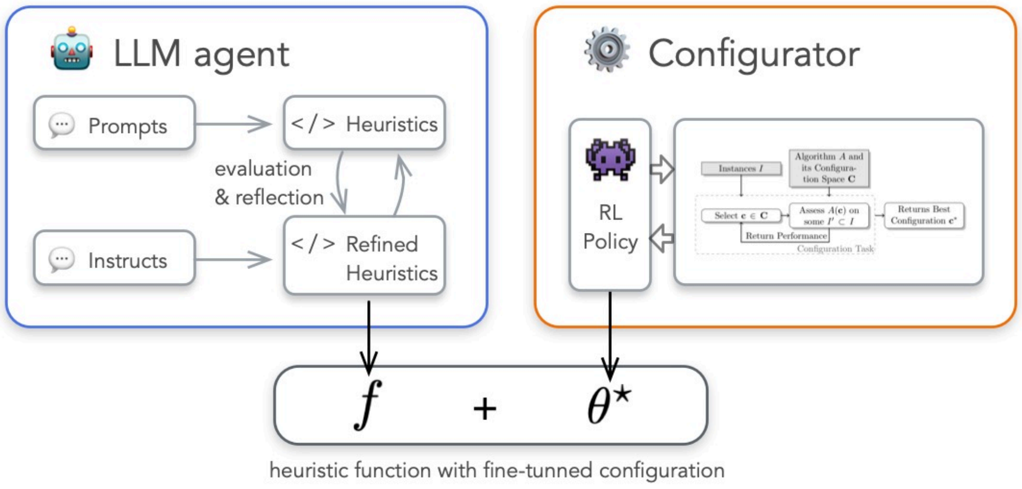


# Instance-specific Algorithm Configuration



# Instance-specific Algorithm Configuration

Using reinforcement learning (RL) for instance-specific algorithm configuration, e.g., OPTICAT (Schede et al., 2025).



Our research question is:

Can **automated algorithm configuration** for LLM-driven heuristic discovery achieve superior performance compared to the state-of-the-art, Our research question is:

Can **automated algorithm configuration** for LLM-driven heuristic discovery achieve superior performance compared to the state-of-the-art,

while addressing *transferability* and *generalization* of the parameter space?

# Takeaways

1. Motivation: LLM-driven heuristic discovery with GA is powerful but costly.

# Takeaways

1. Motivation: LLM-driven heuristic discovery with GA is powerful but costly.
2. Combining GA and AC may achieve better performance.
   - Early generations: ❌
   - “Middle” generations: ✅

# Takeaways

1. **Motivation: LLM-driven heuristic discovery with GA is powerful but costly.**
2. **Combining GA and AC may achieve better performance.**
   - Early generations: ❌
   - "Middle" generations: ✅
3. **Instance-specific deep RL may reduce the overall cost.**

Special thanks to André Hottung for discussions.

VRPAgent/2510.07073

Slides/Personal site

# Bibliography


Hottung, A., Berto, F., Hua, C., Zepeda, N. G., Wetzel, D., Römer, M., Ye, H., Zago, D., Poli, M., Massaroli, S., Park, J., & Tierney, K. (2025a). VRPAgent: LLM-Driven Discovery of Heuristic Operators for Vehicle Routing Problems. *Arxiv*.

Hottung, A., Wong-Chung, P., & Tierney, K. (2025b). Neural Deconstruction Search for Vehicle Routing Problems. *Transactions on Machine Learning Research*. https://openreview.net/forum?id=bCmEP1Ltwq

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43–58. https://doi.org/10.1016/j.orp.2016.09.002

Schede, E., Seiler, M., Tierney, K., & Trautmann, H. (2025). Deep reinforcement learning for instance-specific algorithm configuration. *GECCO*, 1190–1198.

Shaw, P. (1998). Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. *CP*, *1520*, 417–431.