Learning destroy operators for vehicle routing problems with deep neural networks

André Hottung[a,*], Paula Wong-Chung[b], Kevin Tierney[a,*]

[a] Bielefeld University, Germany
andre.hottung@uni-bielefeld.de kevin.tierney@uni-bielefeld.de
[b] University of British Columbia, Canada
* Corresponding author

Extended abstract submitted for presentation at the 12th Triennial Symposium on Transportation Analysis conference (TRISTAN XII)
June 22-27, 2025, Okinawa, Japan

February 28, 2025

Keywords: neural combinatorial optimization, vehicle routing problems, large neighborhood search, reinforcement learning

1 INTRODUCTION

Learning-based methods promise to simplify and accelerate the creation of new heuristics for combinatorial optimization. However, despite recent advancements, learning-based methods often fail to surpass the state-of-the-art techniques from the operations research (OR) community. For example, while some learning-based construction methods for the capacitated vehicle routing problem (CVRP) have outperformed the LKH3 solver (Helsgaun, 2000), they still struggle to compete with the state-of-the-art HGS solver (Vidal *et al.*, 2012).

In this paper, we introduce a novel learning-based large neighborhood search (LNS) framework that leverages a deep neural network (DNN) trained via reinforcement learning (RL) to perform the destroy operation. While existing methods have integrated learned components into the LNS or ruin-and-recreate frameworks (Shaw, 1998, Schrimpf *et al.*, 2000), prior work has primarily focused on learning the repair procedure (e.g., Chen & Tian (2019)) or identifying promising parts of the solution for improvement (e.g., Li *et al.* (2021)). In contrast, our approach is the first to directly learn the destroy procedure.

Our method iteratively alternates between destroy and repair phases as shown in Figure 1. The destroy phase uses a DNN to sequentially select customers to remove from the current solution, while the repair phase employs a simple, greedy insertion algorithm that reinserts the removed customers at locally optimal positions. Notably, our approach is trained using RL, enabling it to adapt to problem instances without requiring pre-existing solutions.

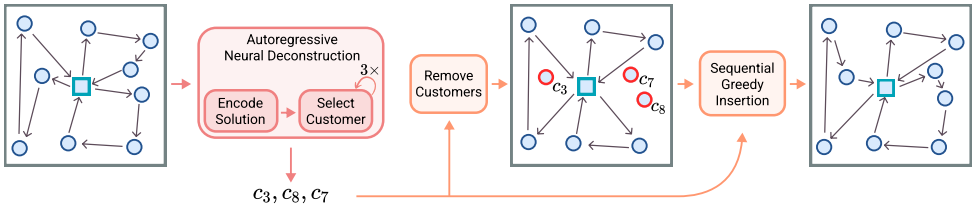

Figure 1 – *Large neighborhood search with a learned destroy operator.*

The strength of our approach lies in its hybrid nature: it combines the nuanced decision-making capabilities of a DNN with the fast evaluation speed of handcrafted OR heuristics. The DNN guides complex destroy operation, while the efficient greedy repair operator allows rapid evaluation of insertion positions. By balancing a learned, strategic destroy procedure with a fast repair heuristic, our method achieves state-of-the-art performance comparable to leading OR approaches like SISRs (Christiaens & Vanden Berghe, 2020) and HGS. Furthermore, our method is highly adaptable to new problem types.While minor adjustments to the repair operator may be required in some cases, the DNN adapts to different problems and distributions on its own.

We evaluate our approach on several challenging problems, including the CVRP, the vehicle routing problem with time windows (VRPTW), and the price-collecting vehicle routing problem (PCVRP). Our method demonstrates substantial gains over existing learned construction techniques and achieves slightly improved performance compared to the best OR methods. To the best of our knowledge, this is the first learning-based approach to reach this level of performance.

2 METHODOLOGY

2.1 Intelligent Destroy Operator

To destroy a solution, we employ a neural network that sequentially selects customers to remove. This process can be viewed as a sequence of interdependent decisions, where each choice impacts the next. More formally, given a feasible solution s to a vehicle routing problem (VRP) instance l involving customers $c_1, \ldots, c_N$, a policy network π_θ, parameterized by θ, selects M customers for removal. At each step $m \in \{1, \ldots, M\}$, the network chooses an action $a_m \in \{1, \ldots, N\}$, determining which customer to remove. The selection is made according to the probability distribution $\pi_\theta(a_m \mid l, s, a_{1:m-1})$, where $a_{1:m-1}$ represents the previously removed customers.

The policy network is based on a transformer architecture, comprising an encoder and a decoder. The encoder generates embeddings for all nodes by incorporating both the problem instance l and the current solution s. These embeddings are then passed to the decoder, which produces the output distribution over the remaining nodes at each of the M decision steps. While our decoder architecture mostly follows that of earlier works, we propose novel encoder components that are focused on learning capable internal representation of VRP solutions.

2.2 Fast Repair Operator

The greedy repair operator reinserts the customers removed by the policy, one by one, following a given order. At each step, a customer is placed in the position with minimal insertion costs, considering all feasible insertion points across the current tours. During this process, various constraints are respected, such as vehicle capacity limits. If no feasible insertion can be found, a new tour is created for the customer. The order in which customers are reinserted plays a critical role in the overall solution quality. We explore two strategies: reinserting customers either in the order dictated by the neural network or in random order. Allowing the neural network to control the reinsertion order enables it to influence the repair process, potentially identifying sorting rules that lead to improved solutions. Our repair operator is inspired by earlier large neighborhood approaches (Christiaens & Vanden Berghe, 2020), but we intentionally keep it as simple as possible. For example, unlike Christiaens & Vanden Berghe (2020), we do not rely on handcrafted sorting rules, focusing instead on leveraging the neural network's learned policies.

2.3 Training

The policy DNN learns a destroy strategy in a RL training loop. At each iteration, new problem instances are generated dynamically. For each instance, an initial feasible solution is created and subjected to cycles of destruction and repair. After each repair phase, a reward is computed

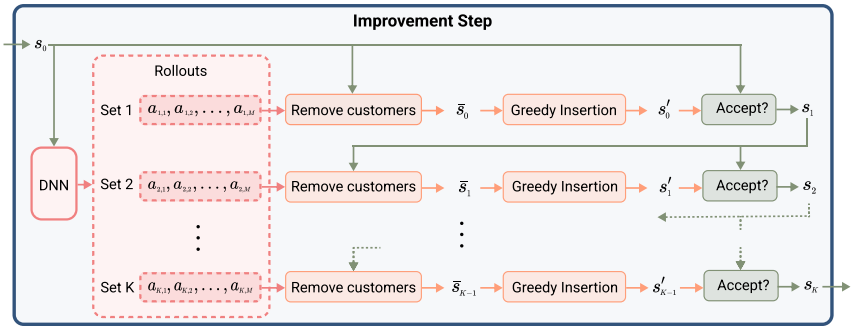

Figure 2 – *GPU-accelerated improvement step.*

based on the cost reduction relative to the original solution. To promote exploration, rewards are limited to be non-negative, encouraging the network to take risks. The rewards guide the learning process via the REINFORCE algorithm, which calculates gradients based on policy performance. Network weights are adjusted to increase the likelihood of selecting customers whose removal and reinsertion result in improved solutions. Over time, this iterative process enables the policy network to refine its ability to identify effective destroy strategies.

2.4 Search

The trained destroy policy can be integrated into any large neighborhood search framework. However, to run efficiently, the neural network requires a GPU, and we have designed a novel improvement step procedure that fully leverages the parallel computing capabilities of GPUs.

As illustrated in Figure 2, the GPU-optimized improvement step begins with an initial solution s_0, which is passed to the policy DNN. The DNN generates K rollouts rollouts of the policy, i.e., K sequences of M actions that specify the customers to be removed. By calculating these K rollouts in parallel on the GPU, we maximize computational efficiency. After the rollout calculations, the rollouts are sequentially applied to generate new candidate solutions. For example, s_0 is destroyed according to the actions from the first rollout, yielding an intermediate solution $\bar{s_0}$, which is then repaired into s'_0. An acceptance criterion determines whether s'_0 or s_0 should be retained, resulting in s_1. This process is repeated for K iterations, resulting in significant cost reductions from the initial input s_0 to the final output s_K.

3 RESULTS & DISCUSSION

We evaluate our method on the CVRP, the VRPTW, and the PCVRP, with 100, 500, and 1000 customers. For the CVRP, we benchmark our method against the state-of-the-art handcrafted methods HGS (Vidal *et al.*, 2012) and SISRs. For the VRPTW and PCVRP, we compare to SISRs and the multi-variant VRP solver PyVRP (Wouda *et al.*, 2024), which is based on the HGS framework. In each case, we evaluate our approach on 100 test instances generated using the instance generator from Queiroga *et al.* (2022). We perform a separate training run for each problem type and instance size. During testing, our method runs on an Nvidia A100 GPU and a single CPU core, while all baseline methods use only a single CPU core.

Figure 3 illustrates the costs of each approach over different runtimes. Compared to SISRs, our method consistently performs better across all three problems. When compared to HGS and PyVRP, our approach shows superior performance on PCVRP and VRPTW instances with 500 and 1000 nodes. For the VRPTW with 100 customers and CVRP scenarios with 500 or more customers, our method outperforms HGS/PyVRP for shorter runtime intervals. Our approach is dominated by HGS only in a single setting, namely the CVRP with 100 customers.

The results highlight the effectiveness of our approach in learning powerful destroy strategies

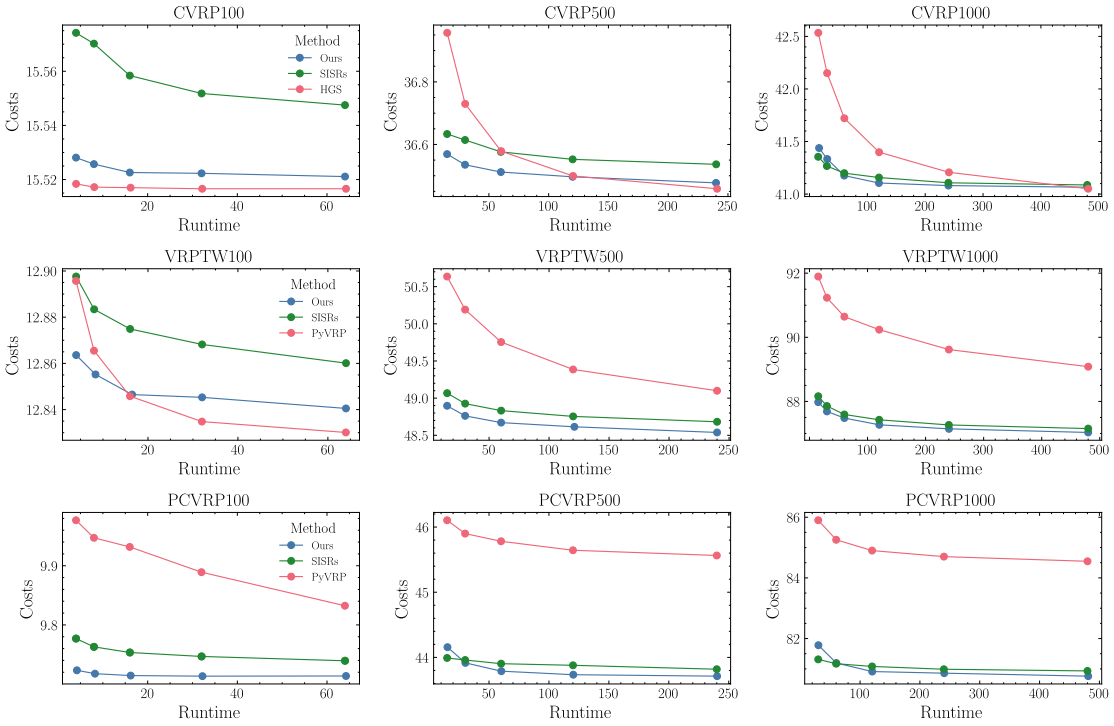

Figure 3 – *Performance evaluation: costs vs. runtime.*

through reinforcement learning. It is worth noting that our method requires a GPU at test time, which incurs additional computational costs compared to CPU-only solutions. However, these costs may be offset by reduced development time, particularly for routing problems lacking well-established approaches. Moving forward, we aim to extend our approach to more complex routing problems with additional constraints, better aligning with real-world applications. Additionally, we plan to get a better understanding of the learned destroy strategies to assess whether these strategies can be efficiently implemented using CPU-only algorithms.

References

Chen, Xinyun, & Tian, Yuandong. 2019. Learning to perform local rewriting for combinatorial optimization. *Pages 6278–6289 of: Advances in Neural Information Processing Systems*.

Christiaens, Jan, & Vanden Berghe, Greet. 2020. Slack induction by string removals for vehicle routing problems. *Transportation Science*, **54**(2), 417–433.

Helsgaun, Keld. 2000. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, **126**, 106–130.

Li, Sirui, Yan, Zhongxia, & Wu, Cathy. 2021. Learning to delegate for large-scale vehicle routing. *Advances in Neural Information Processing Systems*, **34**, 26198–26211.

Queiroga, Eduardo, Sadykov, Ruslan, Uchoa, Eduardo, & Vidal, Thibaut. 2022. 10,000 optimal CVRP solutions for testing machine learning based heuristics. *In: AAAI-22 Workshop on Machine Learning for Operations Research (ML4OR)*.

Schrimpf, Gerhard, Schneider, Johannes, Stamm-Wilbrandt, Hermann, & Dueck, Gunter. 2000. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, **159**(2), 139–171.

Shaw, Paul. 1998. Using constraint programming and local search methods to solve vehicle routing problems. *Pages 417–431 of: CP*. Springer.

Vidal, Thibaut, Crainic, Teodor Gabriel, Gendreau, Michel, Lahrichi, Nadia, & Rei, Walter. 2012. A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research*, **60**(3), 611–624.

Wouda, Niels A, Lan, Leon, & Kool, Wouter. 2024. PyVRP: A high-performance VRP solver package. *INFORMS Journal on Computing*.