A Comparative Analysis of Reinforcement Learning's Effectiveness in Solving the Capacitated Vehicle Routing Problem Against Traditional Methods

Venkatesh Duraiarasan, DA24C021

2025-01-06

Contents

1. Motivation and Problem statement

Capacitated Vehicle Routing Problem (CVRP) is a generalization of the Traveling Salesman Problem (TSP), which is also NP-hard. While the TSP seeks the shortest route for a single salesperson to visit all cities, the CVRP extends this by adding vehicles with limited capacity that must satisfy customer demands. This addition of capacity constraints introduces further complexity.Consequently, exact methods often fail to find feasible solutions for large CVRP instances within a reasonable timeframe. While metaheuristics offer practical alternatives for larger problems, they only provide approximate solutions.

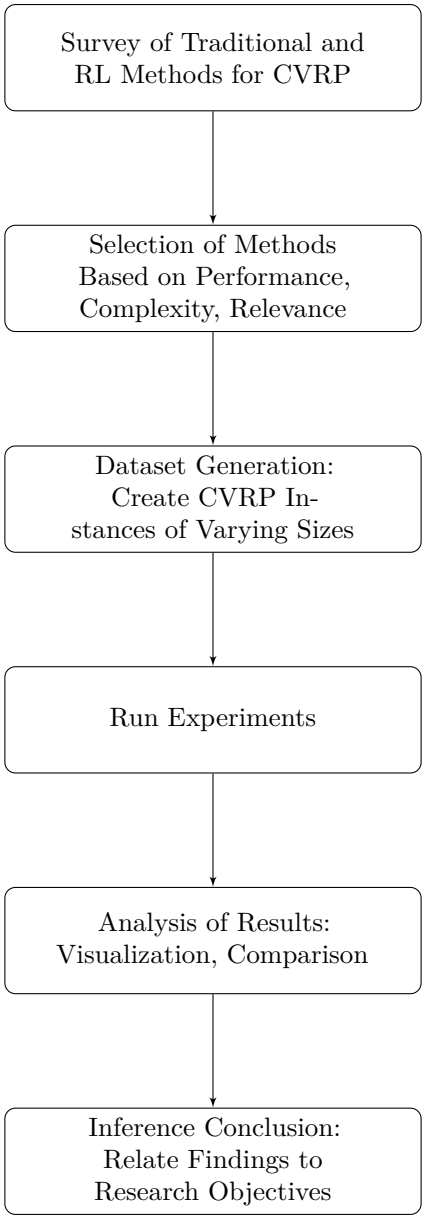
Given the limitations of existing methods, this project investigates Reinforcement Learning (RL) as a potential alternative for solving CVRP and compares its performance to established methods.

2. Project Objectives

The goal of this project is to conduct a comparative analysis of RL's performance and effectiveness in solving the CVRP against traditional methods which includes exact methods and metaheuristics. The specific objectives are:

- To evaluate the effectiveness of RL algorithms in finding near-optimal solutions to the CVRP compared to traditional methods, as measured by solution quality and optimality gap.
- To assess the computational efficiency of RL algorithms in solving CVRP instances of varying sizes and complexities, as measured by inference time and training time
- To identify the strengths and limitations of RL and traditional methods in the context of CVRP.

3. Methodology



4. Survey and Selection of Traditional Methods to solve CVRP

Traditional methods to solve above CVRP can be categorized into two types

- Exact Methods
- Heuristics/ Metaheuristics

The project chooses best methods available from above two categories for comparing with Reinforcement learning based on following criteria

- performance on benchmark instances
- popularity within the research community
- availability of public code implementations or the feasibility of implementation within the project scope.

Based on above criteria following methods were selected for comparison with Reinforcement learning

Category	Method	Reference	Remarks
Exact	MIP - Branch-and-Cut-and-Price (BCP)	Pecin et al. [Pec+14]	Reported as a very strong exact method.[TV14]
Metaheuristics	Hybrid Genetic Search (HGS)	Vidal et al. [Vid+13], Vidal [Vid22b]	It has consistently ranked among the top-performing algorithms for the CVRP, as evidenced by its success in the EURO Meets NeurIPS 2022 Vehicle Routing Competition[Vid+23].
Metaheuristics	Iterated Local Search (ILS)	Lourenço, Martin, and Stützle [LMS03]	A commonly used and effective metaheuristic for the CVRP.

A more comprehensive list of methods considered but not included in the main analysis is provided in Appendix 10.1

5. Survey of Recent Research and Selection of RL method to solve CVRP

This section provides a brief survey of recent research efforts, highlighting different methodological approaches and their contributions.

Applying RL to routing problems was introduced by Bello et al. (2016) in their work, "Neural Combinatorial Optimization with Reinforcement Learning". This work demonstrated the feasibility of using a Pointer Network with REINFORCE algorithm to learn approximate solutions to the Traveling Salesman Problem (TSP). This laid the foundation for subsequent research on applying RL to the more complex CVRP.

Following Bello et al., various RL methods have been proposed for CVRP. Some notable examples are summarized in the table below:

Method	Selected For Experiment (This Study)	Paper/References	Code Implementation/ Code Library	Year
Pointer Network + Asynchronous Advantage Actor-Critic (A3C)	No	Reinforcement Learning for Solving the Vehicle Routing Problem (Nazari et al.)	Paper Code	2018
Attention Model + REINFORCE	Yes	Attention, Learn to Solve Routing Problems (Kool et al.)	Paper Code	2019, 2022
Attention Model + PPO	No	RLOR: A Flexible Framework of Deep Reinforcement Learning for Operation Research (Ching et al.)	Paper Code	2023

Attention Model + REINFORCE performed better than Pointer Network + A3C [KHW19a]. When compared with REINFORCE, PPO has only marginal improvement in performance but significant improvement in training time [WTW23].

The focus of this study is to compare RL with traditional methods. Attention Model + REINFORCE serves as a good baseline to compare the effectiveness of RL against traditional methods. Other complex models like PPO can be tried in future studies.

6. Experimental Evaluation and Analysis

6.1 Experimental Setup

Experiments were conducted for fixed numbers of customer nodes (n=10, 20, 50, 100), with each problem type named VRP10, VRP20, VRP50, and VRP100, respectively. Vehicle capacity was set to [20, 30, 40, 50] corresponding to the node sizes. The following methods were chosen for comparison:

Method Type	**Method**	**Paper/ References**	**Code Implementation/ Code Library**	**Year**
Exact	MIP - Branch & Cut & Price (BCP)	-	BCPCod, Python Wrapper	2021
Metaheuristics	Hybrid Genetic Search (HGS)	Hybrid genetic search for the CVRP	HGS-CVRP, Python wrapper	2022
Metaheuristics	Iterated Local Search	-	Google OR-Tools v9.4	-
RL	Attention Model + REINFORCE	Attention, learn to solve routing problems	Original Implementation, Library - RL4CO	2019, 2022

6.1.1 Dataset Generation

Train Data Instances consisting of node locations (customers) and demands were randomly generated from a uniform random distribution. Specifically, the customer and depot locations were randomly generated in the unit square $[0, 1] \times [0, 1]$. The demand of each node was a discrete number in $\{1, ..., 9\}$, chosen uniformly at random. Node at index 0 is considered as depot node.

For training and validation of the RL model, 100,000 and 10,000 instances were generated, respectively.

Test Data Evaluation/Test set data was also randomly generated as above but test dataset size was set to 1000. Inter-node distances were calculated using the Euclidean norm. To avoid floating-point precision issues and maintain consistency across different solvers, these distances were scaled by a factor of 10,000 and rounded to the nearest integer. Test data sets for VRP10, VRP20, VRP50, VRP100 were generated with same random seed.

The same test dataset was used to evaluate all methods (RL and traditional) to ensure a fair comparison.

6.1.2 Solving instances using Exact Method

MIP-Branch & Cut & Price Instances from the datasets (VRP10, VRP20, VRP The CVRP instances (VRP10, VRP20, VRP50, VRP100) were solved using the Branch & Cut & Price (BCP) algorithm implemented in BCPCod [11], [13]. Since exact method can take forever to solve some instances of VRP50, VRP100, time limit was set as 1000 seconds. The upper bound on the number of vehicles was set to the number of customer nodes, which is a valid but potentially loose upper bound. The solver was accessed through its Python API

6.1.3 Solving instances using RL Method

6.1.3.1 RL Model Training The Attention Model + REINFORCE [KHW19b] was trained separately for each problem type (VRP10, VRP20, VRP50, VRP100). The following hyperparameters were used for training:

- Training set size = 100,000 instances
- Validation set size = 10,000 instances
- Number of epochs = 100
- Trainable parameters = 1.4 Million
- Dimension of input embedding = 128
- Dimension of hidden layers in Encoder/Decoder = 128
- Number of layers in the Encoder network = 3
- Learning rate = 0.0001
- Batch size was adjusted according to the problem type.

These hyperparameters were chosen based on the values reported in the original paper [KHW19b] and through preliminary experimentation. Latest code implementation with same architecture is available in a python library RL4CO [10]

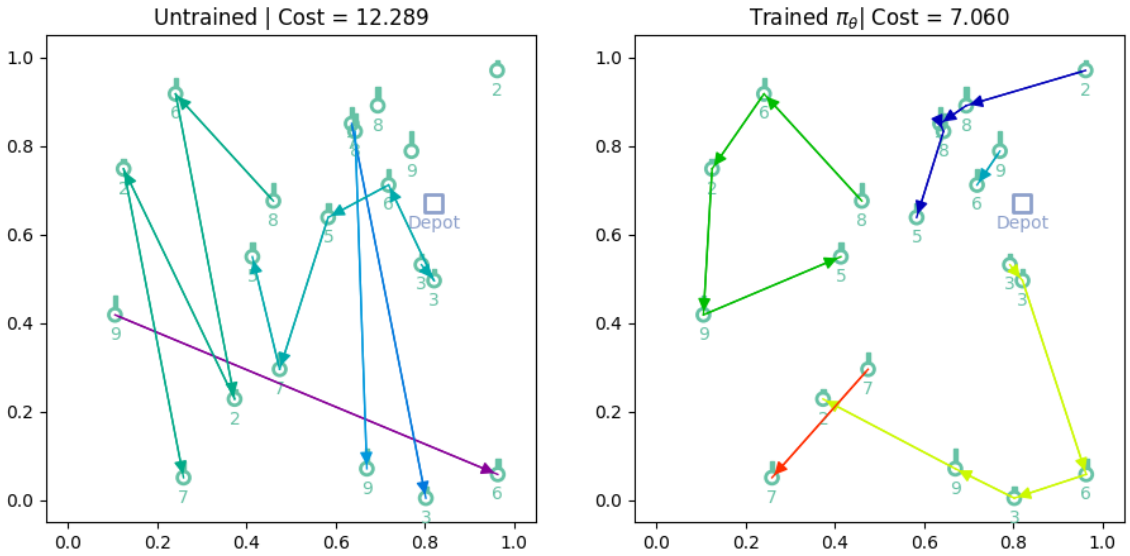


Figure - Example of solving an instance from the VRP20 test set. Left image: Solution before starting RL training. Right image: Solution after 100 epochs.

6.1.3.2 RL Model Solution After training the Attention Model after 100 epoch, the trained model was used to solve each instance in the test set. For each instance, the trained model was used to construct a solution in a single forward pass. The output of the model provides a probability distribution over the possible actions (visiting the next customer), and at each step, the action with the highest probability was chosen (greedy decoding).

6.1.4 Solving instances using Metahueristics For solving using HGS, code implementation/library [14], [15] was used. And for solving using Iterated Local Search method, Google OR Tools was used. Since metaheuristics return better results with more time, time limit for each instance is limited by upper bound. The inference times for HGS and OR-tools are upper bounds limited by the RL inference times. HGS and OR-tools were allowed to run only up to the time taken by RL for each instance. This setup provides a much fairer comparison of solution quality because all methods except BCP are given the same amount of time to find a solution on each instance.

6.1.5 Baselines To evaluate the performance of the RL model, the following baselines were considered:

- **VRP10, VRP20, VRP50:** For these smaller instances, the Branch & Cut & Price (BCP) algorithm was used as the baseline. BCP was able to find optimal solutions for all instances in VRP10 and VRP20 within the 1000-second time limit, resulting in an optimality gap of 0. For VRP50, BCP found optimal solutions for most instances, but failed to find any feasible solution within the time limit for others.
- **VRP100:** Due to the computational limitations of the exact method on larger instances, the Hybrid Genetic Search (HGS) algorithm was used as the baseline for VRP100.

6.1.6 Evaluation Metrics:

Solution Quality : Total Distance traveled by all vehicles for a given instance.

Optimality Gap : The optimality gap is calculated by measuring the mean difference between the distances returned by the RL and BCP methods. For VRP10, VRP20, and VRP50, the optimality gap was found by running the exact algorithm, as it is computationally less demanding than solving VRP100. For VRP100 since BCP method did not return solution for all instances. HGS is used for performance comparison

Inference Time : Inference time is the time taken by a model to provide a valid solution for a particular instance.

6.1.7 Hardware Details: The BCP, HGS, and ILS experiments were run on the local CPU-only machine. RL training and inference were performed on both the local CPU+GPU machine and the Kaggle cloud notebook.
Local CPU only machine specification: 9th Generation Intel Core i5-9300H Processor
Local CPU +GPU machine specification: GeForce GTX 1650 + 9th Generation Intel Core i5-9300H Processor
Cloud Notebook - GPU+CPU specification (Kaggle Provided): 1x Tesla P100 + Intel(R) Xeon(R) CPU @ 2.00GHz

6.2 Performance Comparison

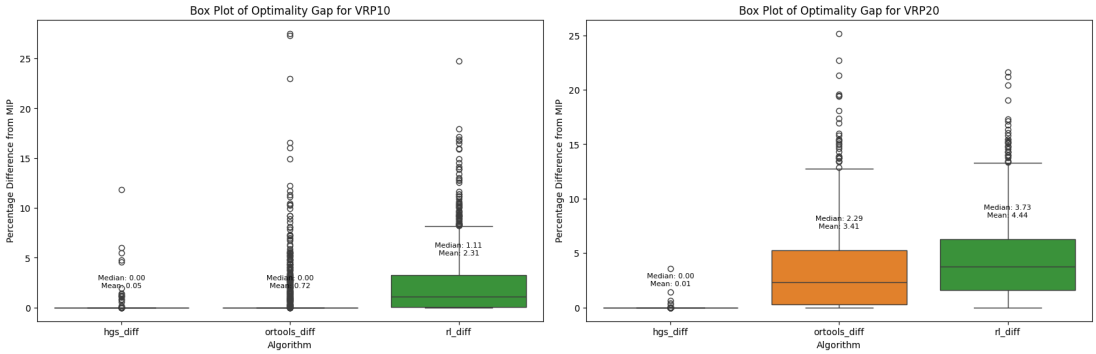
6.2.1 Solution Quality and Optimality Gap Analysis: Below is the comparative result of solution quality and optimality gap for different problem type.

Smaller Instances

Problem	Method	Solution Quality (Avg Distance)	Optimality Gap%
VRP 10	Exact - MIP - BCP	4.5383	-
	Meta heuristics - HGS	4.5406	0.04%
	Meta heuristics - OR-tools - ILS	4.5713	0.71%
	RL	4.6363	**2.31%**
VRP 20	Exact - MIP - BCP	6.1529	-
	Meta heuristics - HGS	6.1533	0.01%
	Meta heuristics - OR-tools - ILS	6.3622	3.40%
	RL	6.4265	**4.45%**
VRP 50	Exact - MIP - BCP	10.4072	-
	Meta heuristics - HGS	10.4073	0.00%

Problem	Method	Solution Quality (Avg Distance)	Optimality Gap%
	Meta heuristics - OR-tools - ILS	11.4569	10.09%
	RL	11.0096	**5.79%**

The Reinforcement Learning (RL) approach consistently yielded the largest optimality gaps across all problem sizes, with gaps of 2.31%, 4.45%, and 5.79% for VRP10, VRP20, and VRP50, respectively. This implies, in its current implementation, it underperforms compared to both the exact BCP method and the advanced metaheuristic HGS. It even lags behind the basic ILS implementation in OR-tools.



HGS consistently found solutions that were very close to the optimal solution. The interquartile range (IQR) is very small. For RL, the box is much wider than HGS's, indicating greater variability in solution quality. There are many outliers, implying in some cases, RL solutions were significantly worse than the optimal.

Medium Instances

Problem	Method	Solution Quality (Avg Distance)	Performance Gap%
VRP 100	Meta heuristics - HGS	15.7196	-
	Meta heuristics - OR-tools - ILS	18.4258	17.22%
	RL	16.8476	**7.18%**

For medium instances such as VRP100, the RL method's performance, while not as good as HGS, is better than what was extrapolated from smaller instances, and better than OR-tools ILS. Poor performance of OR-Tools might be due to shorter time limit set for inference.

6.2.2 Runtime Analysis:

Problem	Method	Inference Time (Mean) CPU Only (in secs)	Model Training Time GPU (CPU) (in hrs)
VRP 10	Exact - MIP - BCP	**0.56**	-
	Meta heuristics - HGS	= 1 (fixed)ˆ	-
	Meta heuristics - OR-tools	= 1 (fixed)ˆ	-
	RL	**0.67**	**0.55** (7.1)
VRP 20	Exact - MIP - BCP	**4.55**	-
	Meta heuristics - HGS	= 2 (fixed)ˆ	-
	Meta heuristics - OR-tools	= 2 (fixed)ˆ	-
	RL	**1.66**	**1.09** (15.1)
VRP 50	Exact - MIP - BCP	**69.17**	-
	Meta heuristics - HGS	= 5 (fixed)ˆ	-
	Meta heuristics - OR-tools	= 5 (fixed)ˆ	-
	RL	**5.14**	**4.15**
VRP 100	Exact - MIP - BCP	>**600**	-
	Meta heuristics - HGS	= 17 (fixed)ˆ	-
	Meta heuristics - OR-tools	= 17 (fixed)ˆ	-
	RL	**17.39**	**9.7**

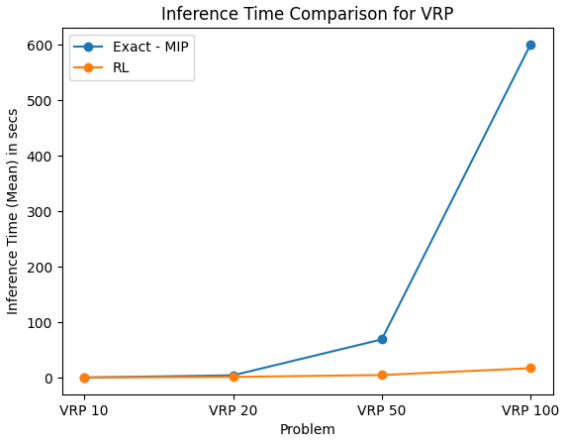
ˆ The inference time limit for HGS and OR-tools was set to be the same as the inference time taken by the RL method for each respective problem instance.

RL's inference time scales relatively well with problem size. While inference is fast, RL requires substantial training time, especially for larger problems. This is a trade-off to consider.

As expected, BCP is very fast for small instances but quickly becomes impractical as the problem size grows.

Even with the imposed time limits, HGS was able to find significantly better solutions than both OR-tools and RL (as seen in the previous solution quality analysis). HGS likely could have found even better solutions if given more time.

The time limit hinders OR-tools' ability to find better solutions. Given the same amount of time as RL, it still performs considerably worse, especially as the problem size increases. The default OR-tools ILS implementation is not very efficient at quickly finding good solutions.



The inference time for RL scales almost linearly with problem size where as for Exact BCP method almost exponential growth is evident.

6.3 Sensitivity Analysis - For RL method only

6.3.1 Impact of Training Data:

Training Sample Size: The effect of varying the amount of training data on the RL model's performance for the VRP10, 20 problem.

VRP10 Training set size	**Mean Cost (Test set, 1000 samples)**	**Optimality Gap**	**Training time With GPU (min)**	**Training time With CPU (hrs)**
1000	5.2760	17.24%	00.48	0.3
10000	4.7512	5.58%	04.68	1.0
20000	4.6939	4.31%	09.36	2.0
50000	4.6377	3.06%	22.56	4.8
100000	4.6363	2.30%	33.36	7.1

VRP20 Training set size	**Mean Cost (Test set, 1000 samples)**	**Optimality Gap**	**Training time With GPU (min)**	**Training time With CPU (hrs)**
1000	7.3341	19.20%	00.9	-
10000	6.8755	11.74%	8.76	-
20000	6.6432	7.97%	17.82	-

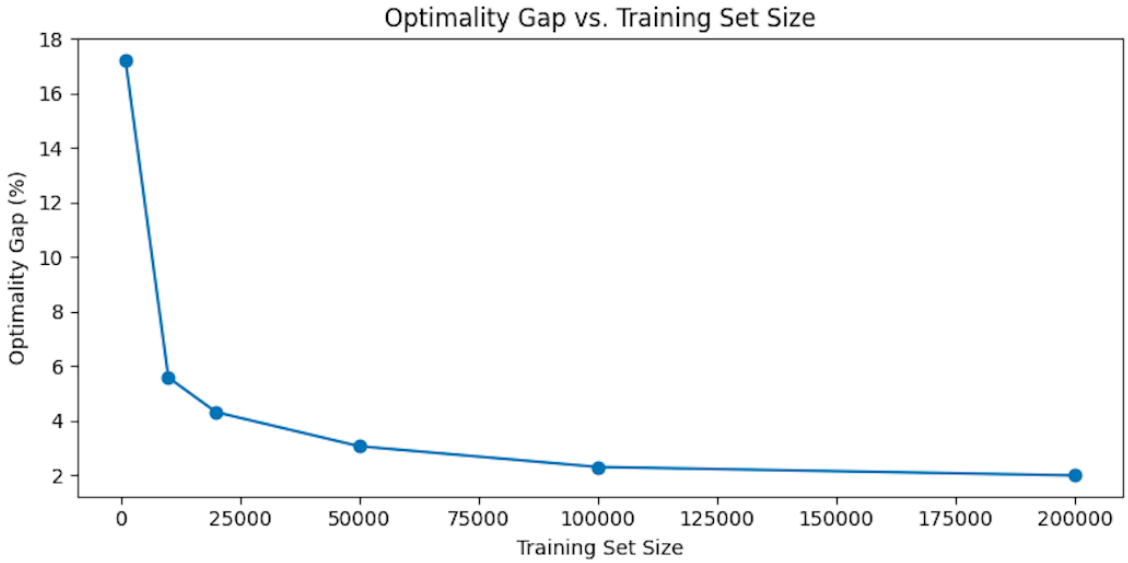
VRP20 Training set size	Mean Cost (Test set, 1000 samples)	Optimality Gap	Training time With GPU (min)	Training time With CPU (hrs)
50000	6.5065	5.75%	41.28	-
100000	6.4265	4.45%	65.40	15.1

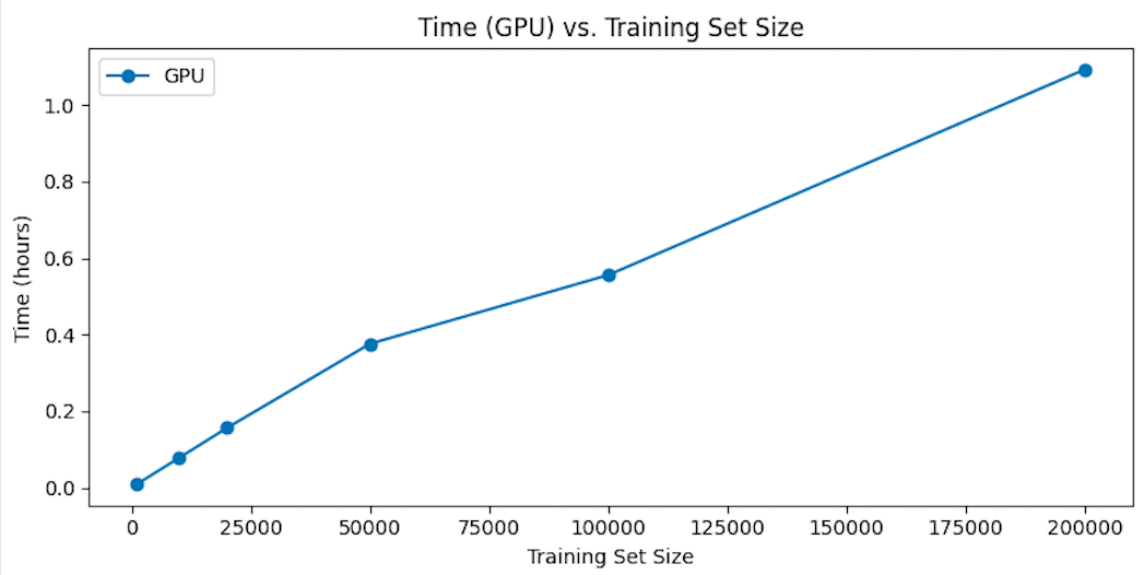
Solution Quality (VRP10)

As the training set size increases, the mean cost decreases, and the optimality gap shrinks implying RL model's performance improves with more training data. The most significant improvement occurs between 1000 and 10000 samples (17.24% to 5.58% optimality gap).The improvement continues as we increase the training set size to 20000, 50000, and 100000. The improvement from 100000 to 200000 is relatively small (from 2.30% to 1.99%).

Training Time (VRP10)

The GPU and CPU training time increases roughly linearly with the training set size.





7. Discussion

7.1 Summary of Finding

Solution Quality:

For smaller Instances (VRP10, VRP20, VRP50), the RL method consistently exhibited the largest optimality gaps compared to other methods. HGS performed exceptionally well, finding near-optimal or optimal solutions in these instances. The exact BCP method, as expected, provided optimal solutions but with rapidly increasing computation time as problem size grew.

For medium Instances (VRP100), RL showed improved relative performance compared to smaller instances, outperforming the basic ILS in OR-tools but still lagging behind HGS.

Runtime:

Inference Time: RL demonstrated relatively fast and scalable inference times, increasing almost linearly with problem size. This contrasts sharply with the exponential growth of BCP's computation time.

Training Time: RL required substantial training time, particularly for larger problems. This represents a significant trade-off compared to heuristic methods that do not require training.

HGS and OR-tools: When given a fixed inference time limit (matching RL's inference time), HGS consistently found better solutions

Sensitivity to Training Data:

Increasing the amount of training data for the RL model significantly improved its performance, reducing both the mean cost and the optimality gap.

7.2 Strengths of RL for CVRP

Based on the findings, the following strengths of RL for CVRP are identified:

Once trained, RL models can generate solutions very quickly. This makes them suitable for real-time or near-real-time applications where rapid decision-making is crucial, such as dynamic routing scenarios or online delivery platforms.

RL's inference time scales favorably with increasing problem size, especially compared to exact methods. This indicates potential for applying RL to larger, more complex CVRP instances where traditional methods become computationally intractable.

7.3 Weaknesses and Limitations of RL for CVRP

In its current implementation, the RL approach generally produced solutions of lower quality compared to metaheuristics like HGS.

Training RL models, especially for larger problem instances, can be computationally expensive and time-consuming. This could be a barrier to adoption in situations where model retraining needs to be frequent.

7.4 Comparison with Traditional Methods: When to Use RL?

RL's fast inference makes it suitable for dynamic routing environments where decisions must be made quickly. RL's ability to learn and adapt makes it a good choice when dealing with fluctuating customer demands, locations, or vehicle capacities. RL's performance depends on having enough high-quality training data.

8. Conclusion

RL offers a new approach for tackling CVRP, particularly in dynamic and complex scenarios. While it currently lags behind highly optimized metaheuristics in terms of solution quality, its strengths in inference speed, scalability, and adaptability make it a potential alternative tool to consider.

9. References

Books

[0] Paolo Toth and Daniele Vigo. *Vehicle Routing: Problems, Methods, and Applications*. 2nd ed. United States: SIAM, 2014.

[1] Lawrence V. Snyder and Zuo-Jun Max Shen. *Fundamentals of Supply Chain Theory*. United Kingdom: Wiley, 2011.

Papers

[2] N. Errami et al. *VRPSolverEasy: a Python library for the exact solution of a rich vehicle routing problem*. Tech. rep. Technical report HAL-04057985, 2023.

[3] Thibaut Vidal et al. "The EURO Meets NeurIPS 2022 Vehicle Routing Competition". In: *Proceedings of Machine Learning Research* 220 (2023), pp. 35–49.

[4] Ching Pui Wan, Jason Min Tung Li, and Jason Wang. "RLOR: A Flexible Framework of Deep Reinforcement Learning for Operation Research". In: *arXiv preprint arXiv:2303.13117* (2023).

[5] Thibaut Vidal. "Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood". In: *Computers and Operations Research* 140 (2022), p. 105643.

[6] Wouter Kool, Herke van Hoof, and Max Welling. "Attention, learn to solve routing problems!" In: *International Conference on Learning Representations (ICLR)*. 2019. URL: https://openreview.net/forum?id=ByxBFsRqYm.

[7] D. Pecin et al. "Improved branch-and-cut-and-price for capacitated vehicle routing". In: *Integer Programming and Combinatorial Optimization*. Springer. Berlin, 2014, pp. 393–403.

[8] Thibaut Vidal et al. "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows". In: *Computers & Operations Research* 40.1 (2013), pp. 475–489.

[9] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. "Iterated local search". In: *Handbook of metaheuristics*. Boston, MA: Springer, 2003, pp. 320–353.

Code and Websites

[10] Kool, Wouter. *Latest implementation of selected RL method (2024)*. https://rl4.co/. Accessed: 2024-01-10. 2024.

[11] BapCoD Team. *Implementation for MIP - BCP*. https://bapcod.math.u-bordeaux.fr/. Accessed: 2024-01-10. 2023.

[12] Google Optimization Team. *OR-Tools*. https://pypi.org/project/ortools/. Accessed: 2024-01-10. 2023.

[13] VRPSolverEasy Team. *VRPSolverEasy: Python wrapper for Exact method - BCP*. https://github.com/inria-UFF/VRPSolverEasy. Accessed: 2024-01-10. 2023.

[**14**] Thibaut Vidal. *HGS-CVRP: A modern implementation of the Hybrid Genetic Search for the CVRP*. https://github.com/vidalt/HGS-CVRP. Accessed: 2024-01-10. 2022.

[**15**] Changhyun Kwon. *PyHygese: Python wrapper for the Hybrid Genetic Search solver for Capacitated Vehicle Routing Problems*. https://github.com/chkwon/PyHygese. Accessed: 2024-01-10. 2020.

[**16**] Wouter Kool, Herke van Hoof, and Max Welling. *Original implementation of selected RL method (2019)*. https://github.com/wouterkool/attention-learn-to-route. Accessed: 2024-01-10. 2019.

10. Appendices

10.1 Survey of methods to solve CVRP

Method Type	Method Name	Method Description	Reference
Exact Methods	Branch-and-Bound	Explores a tree of possible solutions, pruning branches that can't lead to a better solution than the current best.	Toth & Vigo (2002)
	Branch-and-Cut	Combines Branch-and-Bound with the cutting-plane method to tighten the linear programming relaxation.	Lysgaard et al. (2004)
	Branch-and-Price (Column Generation)	Combines Branch-and-Bound with column generation, iteratively generating promising columns (routes).	Pecin et al. (2014)
	Dynamic Programming	Breaks down the problem into smaller overlapping subproblems and stores their solutions.	Christofides et al. (1981)
Classical Heuristics	Savings Algorithm	Iteratively merges routes based on the "savings" in distance.	Clarke & Wright (1964)
	Sweep Algorithm	Forms routes by "sweeping" a ray around the depot and adding customers to a route until capacity is reached.	Gillett & Miller (1974)
	Cluster First, Route Second	First clusters customers into groups, then solves a TSP for each cluster.	-
	Petal Algorithm	Involves a set partitioning formulation, with columns representing routes that look like a petal of a flower.	-
Classical Improvement Heuristics	2-opt, 3-opt, k-opt	Exchanges edges within the same route to make it shorter.	-
	Or-opt	Removes a sequence of consecutive vertices and inserts it in another position in the route.	-
	λ-interchange	Exchanges sets of customers between routes.	Osman (1993)
Metaheuristics	Tabu Search	Uses a "tabu list" to prevent revisiting recently explored solutions, escaping local optima.	Taillard (1993)
	Simulated Annealing	Accepts worse solutions with a certain probability that decreases over time, escaping local optima.	Kirkpatrick et al. (1983), Osman (1993)
	Genetic Algorithms	Maintains a population of solutions and uses operators like selection, crossover, and mutation to evolve better solutions.	Holland (1975), Baker & Ayechew (2003)
	Ant Colony Optimization	Uses artificial ants to construct solutions, leaving "pheromone trails" that guide the search.	Dorigo et al. (1996), Bullnheimer et al. (1999)

10.2 General Definition and Formulation of CVRP as an Optimization Problem

10.2.1 Assumptions Given a set $N = \{0, \ldots, n\}$ of nodes (Customers) and a **symmetric distance** matrix c_{ij} that satisfies the triangle inequality. Node 0 is the depot. Let $N^- = N \setminus \{0\}$ be the set of customer nodes. Each node $i \in N^-$ has a demand $d_i \geq 0$, with $d_0 = 0$.
For $S \subseteq N^-$, let

$$d(S) = \sum_{i \in S} d_i. \tag{1}$$

- There are K vehicles available at depot
- Vehicles are all identical, each with a capacity of C.
- Demand should be less than the given vehicle capacity ($d_i C$ for all iN)

10.2.2 Formulation The three index formulation to model CVRP was proposed by Fisher and Jaikumar (1981) and below version of formulation can be found from Snyder and Shen, 2019, pg. 467 [1] :

$$\text{minimize} \quad \sum_{k=1}^{K} \sum_{i,j \in N} c_{ij} x_{ijk} \tag{1}$$

$$\text{subject to} \quad \sum_{k=1}^{K} y_{ik} = 1 \qquad \forall i \in N^- \tag{2}$$

$$\sum_{k=1}^{K} y_{0k} = K \tag{3}$$

$$\sum_{i \in N} x_{ihk} = \sum_{j \in N} x_{hjk} = y_{hk} \qquad \forall h \in N, \forall k = 1, \ldots, K \tag{4}$$

$$\sum_{i \in N} d_i y_{ik} \leq C \qquad \forall k = 1, \ldots, K \tag{5}$$

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1 \qquad \forall k = 1, \ldots, K, \forall S \subseteq N^- : |S| \geq 2 \tag{6}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i, j \in N^-, \forall k = 1, \ldots, K \tag{7}$$

$$y_{ik} \in \{0, 1\} \qquad \forall i \in N^-, \forall k = 1, \ldots, K \tag{8}$$

Variables
Variable K is the number of vehicles.
Variable N is the number of nodes including depot as Node 0.
Variable $x_{ijk} = 1$ if vehicle k travels from node i to node j and 0 othewise.
Variable c_{ij} the distance (or transportation cost, travel time, etc.) from node i to node j.
Variable d_i is demand at node i

Objective function
The objective function (1) calculates the total route length (cost) of all vehicles.

Constraints
Constraints (2) require every nondepot node to be served by exactly one route.
Constraint (3) requires the depot to be contained on K routes. (This constraint can be removed if K is an upper bound on the number of vehicles but not all K vehicles need to be used.)
Constraints (4) require y_{hk} to equal 1 if and only if vehicle k traverses exactly one arc into h and one arc out of h.
Constraints (5) ensure the vehicle capacity is not exceeded.
Constraints (6) are subtour-elimination constraints
Equation (7)–(8) are integrality constraints.