Enhancing Vehicle Routing Solutions: Custom Heuristic and Local Search Algorithms for Capacitated Vehicle Routing Problems

Aji Thomas[1], Sushma Duraphe[2], and Arvind Gupta[3]

[1]Resarch Scholar, Department of Mathematics, Motilal Vigyan Mahavidyalaya, Bhopal, M.P., India, aji.thomas20@gmail.com

[2]Professor, Department of Mathematics, Motilal Vigyan Mahavidyalaya, Bhopal, M.P., India, duraphe sus65@rediffmail.com

[3]Professor, Department of Mathematics, Motilal Vigyan Mahavidyalaya, Bhopal, M.P., India, arvind533@yahoo.com

Abstract

When compared to traditional heuristics algorithms, local search algorithms offer significant computational advantages to solve combinatorial optimisation problems, such as the Vehicle Routing Problem (VRP). However, the actual usefulness of existing searching-based algorithms is generally limited by their inability to handle a fixed number of vehicles. These algorithms often avoid the challenging issue of assigning customers to a fixed number of available vehicles. However, in practical situations, logistic service providers need to find solutions that work with a certain fleet size and can quickly adjust to sudden changes in the number of vehicles or solve the Capacitated Vehicle Routing Problem (CVRP). Therefore, the main goal of this paper is to apply custom heuristic algorithms to solve the CVRP to determine the best vehicle delivery routes while keeping in mind each vehicle's maximum carrying capacity. The problem is initially solved by the heuristics presented in this paper, and subsequently, specific local search algorithms are used to further improve the heuristics' findings. The implementation of two heuristic algorithms includes one that is based on the nearest-neighbour strategy and another that is motivated by clustering nearby customers. Two custom local search algorithms are also introduced. The proposed algorithms are compared to well-used benchmarks that have undergone extensive testing and validation.

Keywords: Capacitated Vehicle Routing Problem, Combinatorial Optimization Problems, Heuristics Algorithms, Local Search Algorithms, Vehicle Routing Problem.

1 Introduction

The CVRP, which acts as a fundamental optimization problem, is extremely important in the logistics and transportation industry. It involves figuring out the best routes for a fleet of vehicles to take to deliver goods or services to a group of customers, all while taking into account the capacity constraints of each vehicle. Across numerous industries, including transportation, package delivery, and supply chain management, effectively addressing the CVRP is crucial. Operational expenses can be decreased, customer happiness can be

increased, and the environmental impact can be reduced by effectively allocating resources and planning routes. However, due to the problem's combinatorial nature and computational complexity, it is difficult to identify the best solution for large-scale examples. In recent times, heuristic algorithms (1),(2) have evolved as useful and effective ways to produce near-optimal solutions within certain time frames in response to the challenges given by the CVRP . These algorithms offer efficient strategies for approximation that prioritise computing effectiveness over optimality, making them suitable for large-scale applications. The development and evaluation of custom heuristic algorithms designed for the CVRP is the primary focus of this paper by proposing novel strategies that can improve the quality of discovered solutions. Additionally, the results generated by the heuristics will be improved by the use of local search algorithms. Therefore, another goal of this paper is to develop heuristic algorithms that can find the quickest routes for each vehicle while respecting vehicle capacity constraints. For this purpose, two heuristic algorithms have been developed.

The first algorithm uses nearest-neighbour strategies, starting from a central depot and gradually choosing which customer to visit. The vehicles' travel distances are kept to a minimum with this algorithm. The second algorithm is modelled after clustering strategies and groups neighbouring customers to create subsets that can be handled as separate subproblems. We may optimise routes inside each cluster by focusing on these sub-problems separately, ultimately cutting down on overall travel distance. The paper will use two locally designed search algorithms in addition to the heuristic algorithms to further improve the solutions obtained. Local search algorithms concentrate on examining nearby solutions through actions like customer reassignment or swapping to incrementally improve an initial result. These algorithms' main goal is to improve the routes suggested by the heuristics and maybe find better solutions that improve performance as a whole. The performance of the custom heuristic algorithms and local search algorithms will be compared to that of widely used benchmarks (3) (4)(5)that have undergone thorough testing and proven usefulness to determine their effectiveness and competitiveness. By comparing our algorithms to established ones, we may assess the effectiveness, performance, and viability of our ideas and obtain important insights into the pros and cons of the suggested algorithms. By providing valuable answers to the CVRP, this paper aims to make a significant contribution to the field of logistics and transportation optimisation. The findings of this research have the potential to improve route planning, resource allocation, and operational efficiency for a variety of industries. Industry can cut costs, improve customer happiness, and lessen their environmental effect by optimising the delivery process while taking vehicle capacity constraints into account. The paper is organised as follows; the related works are broken down into subsections in the next section. The methodology and suggested algorithms are provided in Section 3. The experimental results based on the suggested algorithms are reported in Section 4, and we

wrap up the paper in Section 5 with some conclusions and recommendations for further work.

2 Related Works

2.1 Heuristics

The creation of Pointer Networks (PtrNet) by Vinyals et al. (6) revived the idea of using end-to-end deep learning strategies in the area of developing solutions for combinatorial optimisation problems. PtrNet, which was built using an encoder-decoder architecture and supervised training, was designed with the Travelling Salesman Problem (TSP) in mind. On top of this, Reinforcement Learning (RL) was added by Bello et al. (7) and Khalil et al. (8) to train PtrNet for the TSP. Nazari et al. (9) updated PtrNet to handle the CVRP while ensuring invariance to the order of input sequences, broadening its usefulness. The Transformer design (10) was incorporated by Kool et al. (11) to create solution sequences for the TSP and several CVRP variations. Xin et al. (12) improved attention strategies by using multiple identical decoders without common parameters to foster variation among generated solutions. Recent developments include ”heatmap”, which uses supervised learning to assign probabilities to edges in a graph representation, in addition to sequential approaches. Joshi et al. (13)showed the superiority of auto-regressive building strategies over parallelized beam search using a Graph Convolutional Network for TSP routes. Kool et al. (14) extended this model to the CVRP and used a combination of dynamic programming, policy training, and heatmap preprocessing to improve partial solutions. To handle the building of many routes within the context of the TSP, Kaempfer and Wolf (15) expanded the heatmap strategies, outperforming meta-heuristic solvers made for the TSP with exceptional performance.

2.2 Local Search

The local search uses well-established local search strategies that are frequently used in combinatorial optimisation problems to repeatedly refine an initially supplied solution, in contrast to heuristics that begin creating solutions from scratch. To improve a VRP solution, the NeuRewriter strategy created by Chen and Tian (16)implements a policy made up of ”rule-picking” and ”region-picking” components, outperforming other auto-regressive strategies. Similar to this, Lu et al. (17) offer the “learning to improve” strategies, which, when compared to other machine learning heuristics, achieves state-of-the-art solutions by learning a policy to pick local search operators for iterative solution enhancement. Although this strategy can solve a single instance in more than thirty minutes, its computational feasibility is limited, making it incomparable to most other strategies. A good compromise between solution quality and computing efficiency is achieved by Hottung and Tierney’s (18) Neural Large Neighbourhood Search (NLNS), which is based on attention strategies and incorporates learnt combinatorial optimisation

problems-specific operators into the search process. To provide competitive results, nevertheless, a batch evaluation of a thousand examples is necessary.
Table 1 lists the summary of the aforementioned studies.

Table 1: Summary

Aspect	Heuristics	Local Search
Approach	End-to-end deep learning strategies	Well-established local search strategies
Problem	Travelling Salesman Problem (TSP), Capacitated Vehicle Routing Problem (CVRP)	Vehicle Routing Problem (VRP)
Techniques	Pointer Networks (PtrNet), Reinforcement Learning (RL), Transformer design, Graph Convolutional Network, heatmap, auto-regressive strategies	NeuRewriter, "learning to improve" strategies, Neural Large Neighbourhood Search (NLNS)
Noteworthy Contributions	PtrNet adapted for CVRP, improved attention strategies, auto-regressive building strategies, extended heatmap strategies, outperforming meta-heuristic solvers	NeuRewriter policy, "learning to improve" strategies achieving state-of-the-art solutions, Neural Large Neighbourhood Search (NLNS) with competitive results
Computational Feasibility	Varies across different strategies, some requiring extensive training and batch evaluation, while others have limited computational efficiency	A good compromise between solution quality and computing efficiency achieved by NLNS
Performance	Benchmarking against established algorithms and achieving exceptional results	Outperforming other auto-regressive strategies, state-of-the-art solutions achieved

3 Mathematical Models for Capacitated Vehicle Routing Problem

The Capacitated Vehicle Routing Problem (CVRP) can be mathematically modelled as follows:
Let:

• V = {0, 1, 2, . . . , n} be the set of vertices, where 0 is the depot and {1, 2, . . . , n} are the customers,
• A = {(i, j) : i, j ∈ V, i/= j} be the set of arcs,
• C_{ij} be the travel cost from vertex i to vertex j,
• q_i be the demand of customer i,
• Q be the capacity of each vehicle,
• K be the number of vehicles available.

Decision variable:

$$x_{ij} = \begin{cases} 1 & if\ the\ vehicle\ travels\ from\ vertex\ i\ to\ vertex\ j, \\ 0 & otherwise \end{cases}$$

The objective function to minimize the total travel cost is:

$$Minimize \sum_{(i,j)\in A} c_{ij}x_{ij}$$

Subject to constraints:

$$\sum_{j=1}^{n} x_{0j} = K \quad \text{(vehicles departing depot)} \quad (1)$$

$$\sum_{i=1}^{n} x_{i0} = K \quad \text{(vehicles returning depot)} \quad (2)$$

$$\sum_{j=0, j\neq i}^{n} x_{ij} = 1, \quad \forall i \in \{1, \ldots, n\} \quad (each\ customer\ visited\ exactly\ once) \quad (3)$$

$$\sum_{i=0, i\neq j}^{n} x_{ij} = 1, \quad \forall j \in \{1, \ldots, n\} \quad (each\ customer\ left\ exactly\ once) \quad (4)$$

$$\sum_{i\in S} \sum_{j\in S, j\neq i} x_{ij} \leq |S| - 1, \quad \forall S \subseteq \{1, \ldots, n\}, |S| \geq 2 \quad (subtour\ elimination) \quad (5)$$

$$\sum_{i=1}^{n} \sum_{j=0,}^{n} q_i x_{ij} \leq Q, \quad \forall\ vehicle \quad (vehicle\ capacity\ constraint) \quad (6)$$

4 Methodology

4.1 Algorithm 1

To solve the CVRP, our first algorithm uses the nearest neighbor strategies. The depot, which serves as the initial point for customer requests, is where the algorithm begins. We choose the closest location from the depot and include it in the initial routing. Next, we add the point that is the next closest to the most recent point to the route. This procedure keeps going until the maximum capacity for each instance of the problem is achieved. Until the capacity constraint is met, we add the nearest neighbor between the locations again and again. At some point, all the points in the problem instance are covered, and one or more routes for one or more vehicles are produced. The stages are displayed in Algorithm 1.

The first proposed algorithm has an O(n^2) time complexity. It entails twice iterating through the problem instance's points. The goal of each iteration is to calculate the shortest route between the present location and any other unexplored locations. The matching point is noted as visited and changes to the new current point after the shortest distance has been determined. The process continues until the vehicle is fully loaded, at which time it turns around and heads back to the depot. A new route then begins, and so on until all of the points have been reached.

Algorithm 1

Require: Set of customers C, depot location D, vehicle capacity Q, demand q_i for customer i ∈ C

Ensure: Set of routes satisfying vehicle capacity constraints
1: Unvisited ← C
2: Routes ← ∅
3. **while** Unvisited ≠ ∅ **do**
4: CurrentRoute ← [D]
5: CurrentLoad ← 0
6: CurrentLocation ← D
7: **while** True **do**
8: NearestCustomer ← arg $\min_{i \in Unvisited}$ {Distance(CurrentLocation, i)}
9: **if** CurrentLoad + qNearestCustomer ≤ Q **then**
10: Append NearestCustomer to CurrentRoute
11: CurrentLoad ← CurrentLoad + qNearestCustomer
12: CurrentLocation ← NearestCustomer
13: Remove NearestCustomer from Unvisited
14: **else**
15: Break inner loop
16: **end if**
17: **if** Unvisited = ∅ **then**
18: Break inner loop
19: **end if**
20: **end while**
21: Append D (depot) to CurrentRoute
22: Add CurrentRoute to Routes
23: **end while**
24: **return** Routes

4.2 Algorithm 2

Our second algorithm is influenced by cluster analysis, which is a popular machine learning and data mining strategy and is demonstrated in Algorithm 2. We want to guarantee that locations are clustered together based on their closeness before designing routes for the CVRP using this algorithm.

The following is how Algorithm 2 works:

1. Pick a customer location that is unincluded in any solution route and is located the farthest away from the depot.
2. Until the vehicle's capacity is met, continuously locate the nearest customer point that is not associated with any route, combine these points, and save them as a cluster.
3. Continue performing steps (1) and (2) until all customer points are satisfied.

The main while loop ends after covering all customer points in a certain instance, hence the algorithm's complexity is O(n^2). The while loop also needs O(n) operations for each iteration.

Algorithm 2

Require: Set of customers C, depot location D, vehicle capacity Q, demand q_i for customer i ∈ C
Ensure: Set of routes satisfying vehicle capacity constraints
1: Unassigned ← C
2: Clusters ← ∅
3: **while** Unassigned ≠ ∅ **do**
4: Pick customer u ∈ Unassigned that is farthest from depot D
5: CurrentCluster ← {u}
6: CurrentLoad ← q_u
7: Remove u from Unassigned
8: **while** CurrentLoad < Q and Unassigned ≠ ∅ **do**
9: Find customer v ∈ Unassigned nearest to any customer in CurrentCluster
10: **if** CurrentLoad + q_v ≤ Q **then**
11: Add customer v to CurrentCluster
12: CurrentLoad ← CurrentLoad + q_v
13: Remove v from Unassigned
14: **else**
15: Break inner loop
16: **end if**
17: **end while**
18: Add completed CurrentCluster to Clusters
19: **end while**
20: Generate routes for each cluster, starting and ending at depot D, by solving each cluster individually.
21: **return** Clusters and associated routes

4.3 Algorithm 3

Local search algorithms are used to start the optimisation phase after the initial problem instances have been resolved using heuristics. These algorithms seek to enhance the canonical solutions while reducing overall costs. Two specifically created local search algorithms are presented in our paper, both of which exchange visited nodes between two routes to optimise the solution.

These local search algorithms can be combined with conventional local search strategies for the TSP like 2-opt and 3-opt, which enhance solutions by swapping and reversing the visited nodes' order within a single route.

The produced routes obtained from the heuristics are the main focus of the first local search algorithm, as shown in Algorithm 3. It analyses two routes, looking for potential node swaps while it does so. If a swap shortens at least one route while maintaining the maximum capacity of both routes, it is regarded as viable.

The second local search algorithm uses a similar methodology but modifies it somewhat. To find a better solution, it chooses one route and evaluates it against every other route. With each route, this procedure is repeated to methodically examine potential

upgrades across the whole solution set.

Algorithm 3

Require: Initial routes from heuristic algorithms, vehicle capacity Q
Ensure: Optimised routes with reduced overall cost
(a) Pairwise Node Swap Between Two Routes
for each pair of routes (R_1, R_2) **do**
 for each node u ∈ R_1 **do**
  for each node v ∈ R_2 **do**
   Swap nodes u and v
   Check feasibility (capacity constraints)
   Compute new route lengths after swap
   if swap reduces at least one route length and maintains feasibility **then**
    Accept swap permanently
   else
    Undo swap
   end if
  end for
 end for
end for

(b) Single-to-All Routes Node Swap
for each route R_i **do**
 for each other route R_j, where i ≠ j **do**
  for each node u ∈ R_i **do**
   for each node v ∈ R_j **do**
    Swap nodes u and v
    Check feasibility (capacity constraints)
    Compute new total solution cost after swap
    if swap reduces total solution cost and maintains feasibility then
     Accept swap permanently
    else
     Undo swap
    end if
   end for
  end for
 end for
 return Optimized solution routes

4.4 Algorithm 4

Our fourth algorithm takes a geometric approach to optimizing CVRP solutions by focusing on the shape formed by routes, known as convex hulls. The main idea is to simplify the optimization process by considering exchanges only between nodes located on the outer boundary (convex hull) of each route, which significantly reduces the complexity and effort involved.

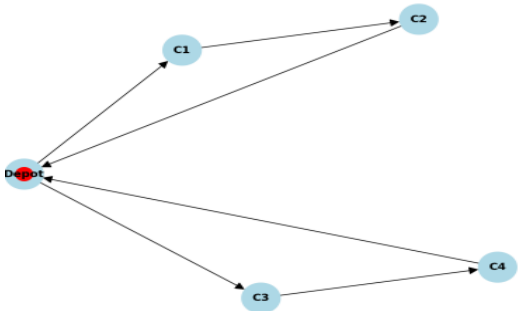
In practice, the algorithm first computes the convex hull for each route, marking potential exchange nodes. It then identifies a central reference point for each route by averaging these convex hull nodes. This central point helps the algorithm quickly determine the most suitable candidates for node exchanges between routes.

To improve the solution, the algorithm carefully examines three types of node exchanges between route pairs:

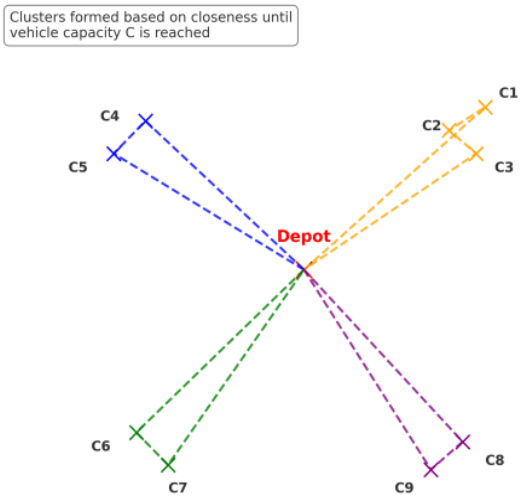
1. Move a node from route A to route B, provided route B can accommodate this node and the move improves the overall solution.
2. Move a node from route B to route A under similar conditions of capacity and solution enhancement.
3. Swap two nodes directly between route A and route B if this exchange meets the capacity constraints and results in a better solution.

After deciding which nodes to exchange, the algorithm considers how to place these nodes optimally within the new routes, ensuring minimal travel distance. Each successful exchange triggers an update of the convex hulls to reflect the new route configuration accurately. Due to the detailed evaluations and convex hull recalculations, the algorithm has a computational complexity of $O(n^3 \log n)$ per iteration.

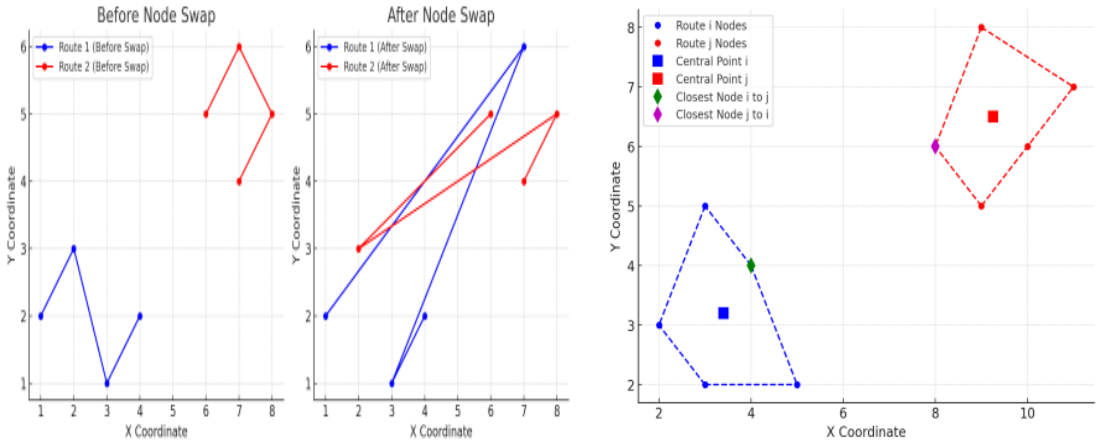
This approach combines intuitive geometry with thorough analysis to efficiently enhance route quality and reduce overall travel costs.



(a) Algorithm 1



(b) Algorithm 2



(c) Algorithm 3 (d) Algorithm 4

Figure 1: Illustrations of algorithms used for solving the CVRP

Algorithm 4

Require: Initial routes obtained from heuristic methods, vehicle capacity Q
Ensure: Improved routes with reduced overall travel costs

Compute convex hull and central points:
for each route R **do**
  Compute the convex hull H(R) of route R
  Calculate central point C(R) as mean of hull points
end for
Local search optimization:
for each pair of routes (R_i, R_j) **do**
  Identify closest node u ∈ R_i to central point C(R_j)
  Identify closest node v ∈ R_j to central point C(R_i)
  Evaluate the following cases:
  Case 1: Move node u from R_i to R_j if it doesn't exceed capacity Q of R_j and improves cost.
  Case 2: Move node v from R_j to R_i if it doesn't exceed capacity Q of R_i and improves cost.
  Case 3: Swap nodes u and v between routes R_i and R_j if it respects capacities Q and improves overall
cost.
  For cases 1 and 2, evaluate possible insertion points in routes to find best ordering after the move.
  if any case results in improvement **then**
  Perform the optimal move or swap
  Recompute convex hull and central points for updated routes
  end if
end for
return Improved set of routes

5 Experimental Analysis

5.1 Benchmarks Description

In the subject of combinatorial optimisation, Set A (4) and Set M (5) are two benchmark instance sets that are frequently utilised. Set A, also referred to as Augerat Set A, is made up of 16 symmetric instances of the TSP, with sizes ranging from 17 to 120 locations. These instances are derived from geographic data, with exact strategies used to arrive at known optimal solutions. There are 16 symmetric instances of the CVRP in Set M, also known as the Christofides-Mingozzi-Toth, which represent various problem sizes. These situations are based on actual vehicle routing circumstances, and the best solutions are given. For TSP and CVRP, respectively, Set A and Set M have both been extensively used in assessing and contrasting the performance of various optimisation algorithms. They have been used as comparison points for evaluating the effectiveness and performance of heuristic and approximate algorithms. A fresh set of 20 large-scale VRPs are also included in the benchmark by Golden et al. (3). The knowledge and improvement of optimization algorithms for TSP and CVRP have benefited greatly from these benchmark sets and studies. They offer a standardised framework for measuring the efficiency of various algorithms and their practical application in resolving optimisation problems encountered in the real world. However, it's important to note that due to computational complexity, we used 20 instances for Golden et al. (3) and 7 instances for Set A, 7 instances for Set B, and 7 instances for Set C.

5.2 Result Analysis

Although creating our unique solutions has been difficult, our experiments and algorithms have demonstrated encouraging solutions. We do, however, admit some design challenges, with the nearest neighbour strategies serving as an example. The "leftovers" from the allocation procedure can be blamed for the later routes larger lengths between locations even though the earlier routes function well. In developing these algorithms, we took care to avoid showing favoritisms for currently used strategies.

Instance	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
A-n32-k05	861	989	866	981
A-n33-k05	741	669	746	662
A-n33-k06	761	811	765	812
A-n34-k05	882	805	886	772
A-n36-k05	883	802	866	802
A-n37-k05	743	662	746	662
A-n37-k06	1132	962	1132	942
A-n38-k05	823	732	826	732
A-n39-k05	884	882	884	882

A-n39-k06	884	992	884	995
A-n44-k06	1015	953	1013	953
A-n45-k06	1015	973	1013	977
A-n45-k07	1014	1113	1014	1167
A-n46-k07	1007	1025	1003	1037
A-n48-k07	1074	1042	1074	1041
A-n53-k07	1147	1055	1147	1037
A-n54-k07	1257	1215	1253	1205
A-n55-k09	1155	1135	1153	1130
A-n60-k09	1417	1395	1413	1396
A-n61-k09	1180	1605	1179	1606
A-n62-k08	1377	1365	1373	1366
A-n63-k09	1488	1465	1485	1465
A-n63-k10	1508	1445	1502	1445
A-n64-k09	1392	1324	1392	1325
A-n65-k09	1278	1264	1272	1265
A-n69-k09	1159	1162	1159	1162
A-n80-k10	1909	1924	1902	1925

Table 2: Results on Set A

We tried to use our imagination to come up with fresh strategies for tackling problems. Although we were influenced by several ideas, we didn't explicitly use any algorithms that already existed. This strategy has improved our ability to solve problems and given us insightful information about practical strategies. On some occasions, though, we are making headway, and our unique approach is producing encouraging findings. Based on their costs for resolving various problem instances, the four algorithms (Algorithm 1, Algorithm 2, Algorithm 3, and Algorithm 4) are compared in Table 2. The instances of the problem are designated A-nXX-kYY, where XX stands for the number of customer points and YY for the number of vehicles (or routes) in the instance. The table shows the cost figures that each algorithm produced for each problem instance. The cost value shows the overall travel distance or expense necessary to reach every customer location while taking the CVRP's limitations into account. The algorithm performs better in terms of minimising the total distance or cost necessary to serve all customer points the lower the cost value is. By comparing the results of the various algorithms across a variety of problem instances, one may see which algorithm typically achieves lower cost values, indicating superior solutions. We must evaluate the cost values of each algorithm for each problem instance to decide which algorithm is the best among Algorithms 1, 2, 3, and 4. In terms of minimising the overall distance or cost necessary to serve all customer points, the algorithm with the lowest cost value typically implies greater performance. We can examine each algorithm's performance based on the table provided: In some cases, such as A-n32-k05, A-n34-k05, A-n36-k05, A-n38-k05, A-n39-k05, A-n48-k07, A-n53-k07, A-n61-k09, A-n62-k08, A-n64-k09,

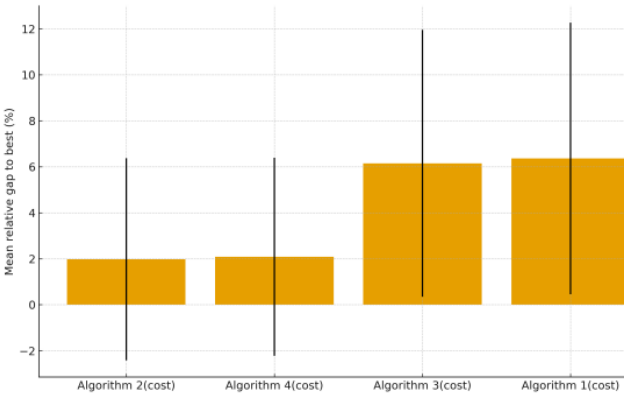
A-n69- k09, and A-n65-k09, Algorithm 1 performs reasonably well. In these cases, it achieves reduced cost values in comparison to Algorithms 2, 3, and 4. It might not, however, be the overall most effective algorithm. In multiple cases, including A-n33-k05, A-n37-k05, A-n45-k07, A-n46-k07, A-n55-k09, and A-n80-k10, Algorithm 2 consistently achieves lower cost values.

Algorithm 2 is therefore superior to the other algorithms in these circumstances. In situations like A-n33- k06, A-n39-k06, and A-n54-k07, Algorithm 3 performs admirably. In these cases, it regularly outperforms Algorithms 1, 2, and 4 in terms of cost values, making it the preferable option. In none of the examples given, Algorithm 4 outperforms the other algorithms in terms of performance. It regularly performs similarly to

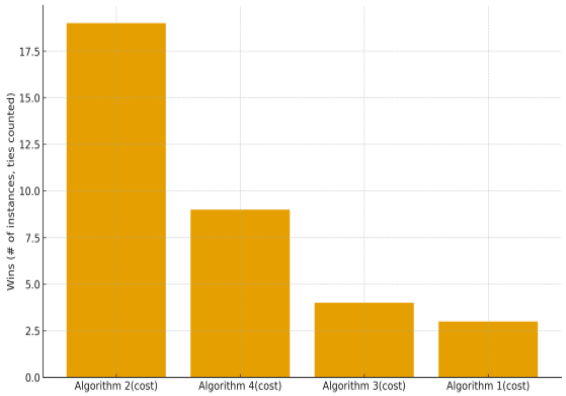
Method	Mean Cost	Mean Gap (%)	Std Gap (%)	Wins (#)
Algorithm 1	1151.2962	6.3602	5.9057	3
Algorithm 2	1108.3703	1.9778	4.3939	19
Algorithm 3	1148.2962	6.1481	5.8051	4
Algorithm 4	1109.7407	2.0813	4.3088	9

Table 3: Set A: Aggregate summary.

Algorithms 1, 2, and 3, or at greater cost values. Algorithms 2 and 3 perform better than the other algorithms in terms of minimising the overall cost, according to this analysis, and show more promise. It's crucial to remember that this evaluation is based on the specific instances given; further testing on more examples would be necessary for a thorough grasp of the algorithms' performance. The cost values from running four algorithms (Algorithm 1, Algorithm 2, Algorithm 3, and Algorithm 4) on various instances identified as CMT01, CMT02, CMT03, CMT04, CMT05, CMT11, and CMT12 are represented in Table 4. Algorithm 1 obtains a cost of 861, Algorithm 2 achieves a cost of 982, Algorithm 3 achieves a cost of 864, and Algorithm 4 achieves a cost of 982, for instance, in the instance CMT01. The cost values for each algorithm for the remaining instances (CMT02, CMT03, CMT04, CMT05, CMT11, and CMT12) are similarly shown in Table 4. Each algorithm's cost values are used as a gauge of how well it performs in terms of minimising the overall distance or cost necessary to serve the customer points in each instance. We can determine which algorithm performs better or worse in terms of optimising the solution by comparing the cost numbers between the algorithms. On instances CMT02, CMT11, and CMT12, Algorithm 2 outperform the other algorithms, while Algorithm 3 outperforms Algorithms 1 and 4 on instances CMT01, CMT03, CMT04, CMT05, and CMT12. There is no clear winner in the comparison between Algorithm 1 and Algorithm 4.



Set A: Mean relative gap (%).



Set A: Wins (ties counted)

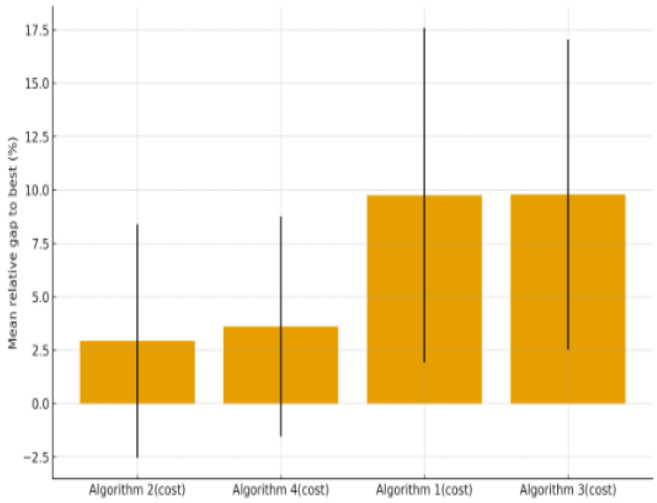
Instance	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
CMT01	861	982	864	982
CMT02	741	660	744	664
CMT03	761	810	764	814
CMT04	881	770	884	774
CMT05	861	800	864	808
CMT11	742	660	744	668
CMT12	1142	940	1123	948

Table 4: Results on Set M / CMT

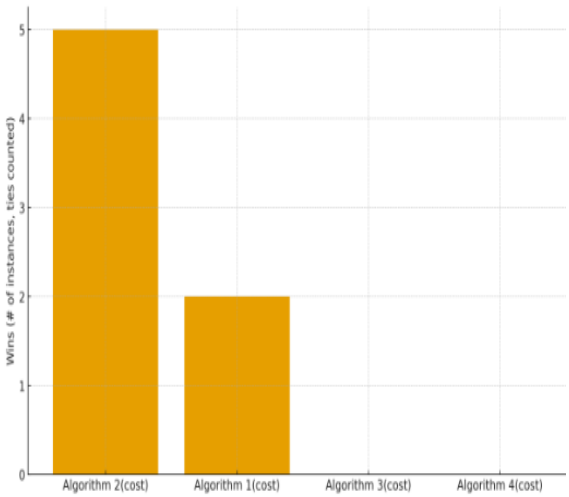
The cost values produced by four algorithms (Algorithms 1, 2, 3, and 4) on various instances designated as Golden 01, Golden 02, Golden 03, and so on are shown in Table ??. Each instance is associated with a particular dataset or problem situation. For instance, in Golden 01, Algorithms 1 and 2 costs 861, 988, 861, and 982 respectively, whereas Algorithms 3 and 4 cost 861. In Golden 02, for example, Algorithm 1 costs 741, Algorithm 2 costs 668, Algorithm 3 costs 741, and Algorithm 4 costs 662. In Golden 03, for example, Algorithm 1 costs 761, Algorithm 2 costs 818, Algorithm 3 costs 761, and Algorithm 4 costs 812. In Golden 04, for example, Algorithm 1 costs 881, Algorithm 2 costs 774, Algorithm 3 costs 881, and Algorithm 4 costs 772. The cost values for each algorithm are provided in the table 6as it does for the remaining instances. We can observe the following things by examining the table: For each instance, the cost values for Algorithms 1 and 3 are the same.

Method	Mean Cost	Mean Gap (%)	Std Gap (%)	Wins (#)
Algorithm 1	855.5714	9.7467	7.8287	2
Algorithm 2	803.1428	2.9274	5.4614	5
Algorithm 3	855.2857	9.7814	7.2652	0
Algorithm 4	808.2856	3.6009	5.1488	0

Table 5: Set M/CMT: Aggregate summary



Set M/CMT: Mean relative gap (%).



Set M/CMT: Wins (ties counted).

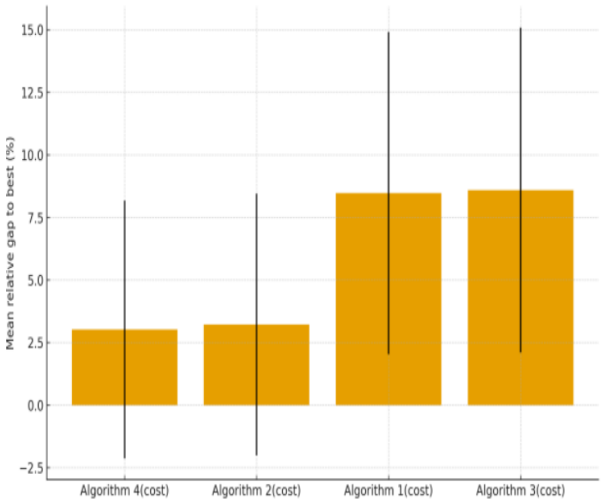
Instance	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
Golden_01	861	988	861	982
Golden_02	741	668	741	662
Golden_03	761	818	761	812
Golden_04	881	774	881	772
Golden_05	861	804	861	802
Golden_06	742	664	742	662
Golden_07	944	1114	944	944
Golden_08	864	861	864	862
Golden_09	744	663	744	662
Golden_10	764	813	764	804
Golden_11	884	773	884	772
Golden_12	861	804	861	802
Golden_13	744	663	744	663
Golden_14	1114	943	1114	944
Golden_15	866	893	866	893
Golden_16	746	663	746	663
Golden_17	769	813	767	773
Golden_18	880	773	887	773
Golden_19	860	803	867	804
Golden_20	860	803	867	804

Table 6: Results on Golden et˜al.

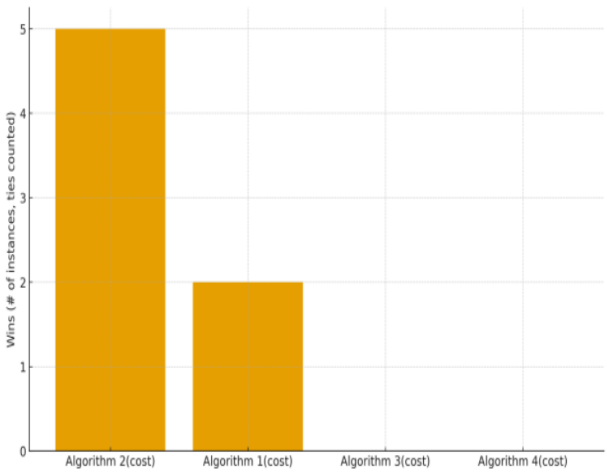
Method	Mean Cost	Mean Gap (%)	Std Gap (%)	Wins (#)
Algorithm 1	846.00	8.4706	6.4446	5

Method	Mean Cost	Mean Gap (%)	Std Gap (%)	Wins (#)
Algorithm 2	807.00	3.2130	5.2358	7
Algorithm 3	846.95	8.5900	6.4908	6
Algorithm 4	805.55	3.0220	5.1514	12

Table 7: Golden: Aggregate summary.



Golden: Mean relative gap (%).



Golden: Wins (ties counted).

This shows that in the provided instances, these two algorithms produce the same solutions. In every instance, Algorithm 2 consistently outperforms Algorithm 4 in terms of cost values. Depending on the particular instance, Algorithm 1's (or Algorithm 3's) performance can be compared to Algorithm 2's. Algorithm 1 (or Algorithm 3) may cost less in some situations, whereas Algorithm 2 may perform better in others. We may infer from this benchmark that Algorithm 2 generally outperforms Algorithm 4, while Algorithm 1 (or Algorithm 3) generally performs similarly to Algorithm 2 in some situations but may perform slightly differently in others. The exact needs and characteristics of the problem instances under consideration would determine whether Algorithm 1 (or Algorithm 3) or Algorithm 2 should be used.

5.3 Size–Based Result Analysis

Size bands used. We grouped instances by number of customers n into three bands that match our datasets: small (Augerat Set A; $n < 100$), medium (CMT; roughly $100 \le n \le 200$ with a couple < 100), and large (Golden; $n > 200$).

Small (Set A)

From table 3 it is clear Algorithm 2 is dominant on small instances, achieving the lowest mean gap and the most wins; Algorithm 4 is the next best. Algorithms 1 and 3 trail with noticeably larger gaps.

Takeaway. Prefer **Algorithm 2** on small CVRPs. If you want a close second with similar stability, **Algorithm 4** is reasonable. **Algorithm 1** and **Algorithm 3** are generally less competitive here.

Medium (CMT)

From table 5Across the mixed CMT set, Algorithm 2 again leads. Algorithm 1 snags two wins on the smaller CMT cases (CMT01, CMT03), but Algorithm 3 and Algorithm 4 record zero wins (though Alg. 4's gaps remain low and consistent).

Takeaway. Use **Algorithm 2** as the default on medium-sized CVRPs. **Algorithm 1** can be competitive on very small/structured CMT cases, but overall **Algorithm 4** is the steadier runner-up (smaller variability) while **Algorithm 3** rarely leads.

Large (Golden)

From table 7 On large Golden instances, Algorithm 4 is best by both mean gap and number of wins. Algorithm 2 is a close second. Algorithms 1 and 3 lag behind (often tied with each other on costs), with noticeably larger mean gaps.

Takeaway. Prefer **Algorithm 4** on large CVRPs; **Algorithm 2** is the best alternative where hull-exchange moves are unavailable.

Cross–size guidance

• **Small** ($n < 100$): **Algorithm 2** → lowest gaps and most wins; **Algorithm 4** is next best. (**Alg. 1** and **Alg. 3** are generally weaker.)

• **Medium** ($100 \leq n \leq 200$): **Algorithm 2** remains best overall. **Algorithm 1** can win on very small/structured cases; **Algorithm 4** is a stable runner-up; **Algorithm 3** rarely leads.

• **Large** ($n > 200$): **Algorithm 4** is best; **Algorithm 2** is a close second. **Algorithms 1** and **3** tend to trail, often tying each other.

6 Conclusion and Future work

The paper concludes that the custom heuristic algorithms created for the CVRP show promise in locating the best delivery routes while considering vehicle capacity restrictions. The two implemented heuristic algorithms—clustering nearby customers and using the nearest neighbour strategies —offer preliminary solutions to the problem. Then, these heuristics are improved with custom local search algorithms. It is seen that the custom heuristic algorithms function effectively and provide computational improvements over traditional operations research solvers and heuristics through comparison with widely used algorithms that have been tested and validated. They show that they can manage a fixed number of vehicles, which is essential in real-world scenarios where logistic service providers need to find solutions that fit their particular fleet size. Future research can concentrate on enhancing the custom heuristic algorithms and investigating additional heuristic and local search algorithm combinations. The algorithms can be improved to produce even better outcomes in terms of the quality of the solutions and computational effectiveness. The algorithms can also be evaluated for performance and scalability on larger datasets or in real-world situations. To improve the algorithms' capacity for learning and provide them with the flexibility to adjust to dynamic changes in the number of vehicles or other parameters, deep reinforcement learning strategies may also be integrated. This would increase the approach's ability to adapt to changes in logistical

operations that occur in real-time. The work opens up new directions for CVRP research and development by highlighting the possibility of personalised heuristic algorithms for solving the problem.

References

[1] G. S. Kashyap, A. E. I. Brownlee, O. C. Phukan, K. Malik, and S. Wazir, "Roulette-Wheel SelectionBased PSO Algorithm for Solving the Vehicle Routing Problem with Time Windows." https:// arxiv.org/abs/2306.02308v1, June 2023. arXiv preprint. Accessed: Jul. 04, 2023.

[2] G. Rahmanifar, M. Mohammadi, A. Sherafat, M. Hajiaghaei-Keshteli, G. Fusco, and C. Colombaroni, "Heuristic approaches to address vehicle routing problem in the IoT-based waste management system," Expert Systems with Applications, vol. 220, p. 119708, June 2023.

[3] B. L. Golden, E. A. Wasil, J. P. Kelly, and I.-M. Chao, "The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results," in Fleet Management and Logistics (T. G. Crainic and G. Laporte, eds.), pp. 33–56, Kluwer Academic Publishers, 1998.

[4] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberan, D. Naddef, and G. Rinaldi, "Computational ´ results with a branch-and-cut code for the capacitated vehicle routing problem," Technical Report 949-M, Universite Joseph Fourier, Grenoble, 1995.

[5] N. Christofides, A. Mingozzi, and P. Toth, "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations," Mathematical Programming, vol. 20, pp. 255–282, Dec 1981.

[6] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in Advances in Neural Information Processing Systems 28 (NIPS 2015), pp. 2692–2700, 2015.

[7] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in 5th International Conference on Learning Representations (ICLR 2017), Workshop Track, Nov 2017. Accessed: Jul. 04, 2023.

[8] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in Advances in Neural Information Processing Systems (NIPS 2017), pp. 6349–6359, 2017.

[9] M. Nazari, A. Oroojlooy, M. Taka´c, and L. V. Snyder, "Reinforcement learning for solving the vehicle routing problem," in Advances in Neural Information Processing Systems (NeurIPS 2018), pp. 9839–9849, 2018.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems (NIPS 2017), pp. 5999–6009, 2017.

[11] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!," in 7th InternationalConference on Learning Representations (ICLR 2019), 2019. Accessed: Jul. 04, 2023.

[12] L. Xin, W. Song, Z. Cao, and J. Zhang, "Multi-decoder attention model with embedding glimpse for solving vehicle routing problems," in Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021), pp. 12042–12049, May 2021.
[13] C. K. Joshi, T. Laurent, and X. Bresson, "An efficient graph convolutional network technique for the travelling salesman problem." https://arxiv.org/abs/1906.01227v2, June 2019. Accessed: Jul.04, 2023.
[14] W. Kool, H. van Hoof, J. Gromicho, and M. Welling, "Deep policy dynamic programming for vehicle routing problems," in Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR 2022) (P. Schaus, ed.), vol. 13292 of Lecture Notes in Computer Science, pp. 190–213, Springer, Feb 2022.
[15] Y. Kaempfer and L. Wolf, "Learning the multiple traveling salesmen problem with permutation invariant pooling networks." https://arxiv.org/abs/1803.09621v2, Mar 2018. Accessed: Jul. 04, 2023.
[16] X. Chen and Y. Tian, "Learning to perform local rewriting for combinatorial optimization," in Advances in Neural Information Processing Systems (NeurIPS 2019), pp. 6278–6289, 2019. Accessed: Jul. 04,2023.
[17] H. Lu, X. Zhang, and S. Yang, "A learning-based iterative method for solving vehicle routing problems," in 8th International Conference on Learning Representations (ICLR 2020), 2020. Accessed: Jul. 04,2023.
[18] A. Hottung and K. Tierney, "Neural large neighborhood search for the capacitated vehicle routing problem," in 24th European Conference on Artificial Intelligence (ECAI 2020), vol. 325 of Frontiers in Artificial Intelligence and Applications, pp. 443–450, IOS Press, Aug 2020.