



King Saud University

College of Computer and Information Sciences
Computer Science Department

Assignment 3

Population based metaheuristic to solve
Capacitated Vehicle Routing Problem (CVRP)

Course Title: Selected topics in Artificial Intelligence

Course Code: CS569

1 Problem Statement

The **Vehicle Routing Problem (VRP)** is one of the most important and widely studied problems due to its application in delivery and transport logistics[17]. VRP solutions have specific objectives and limitations in real applications, making VRP have categories or variants[18]. One of the most popular VRP variants is **Capacitated Vehicle Routing Problem (CVRP)**[19]. This problem is considered a fundamental problem in combinatorial optimization[20]. It is known to be an **NP-hard** problem and a generalization of the Traveling Salesman Problem[21].

CVRP can be described as follows: there is a set of geographically dispersed customers, where each customer has a certain demand[22]. There is also one centralized depot and the travel costs between all locations are given[23]. To serve customers' demands, there is a fleet of identical vehicles whose base is the depot, and each vehicle has a certain capacity[24]. The main goal of this problem is to find the **minimum cost routes** for the capacitated vehicles to serve customers' demands[25].

The constraints are as follows:

1. Each customer is visited exactly once and served by exactly one vehicle[26].
2. Each vehicle starts at and returns to the depot after serving a subset of customers[27].
3. Each customer's demand is satisfied and the total load on any route does not exceed the vehicle's capacity[28].

2 Methodology (P-metaheuristic)

This section shows the details of the used **Population-Based metaheuristics (P-metaheuristics)** to solve CVRP, which is the **Genetic Algorithm (GA)**[30, 31]. GA is developed by John Holland in Michigan, USA in 1975[32]. GA is an optimization algorithm inspired by the principles of natural evolution and genetics, and it is one of the most common evolutionary algorithms[32].

2.1 Main Concepts in GA

The main concepts in GA are:

- **Chromosome/Genotype:** the representation of a potential solution. It is composed of genes that encode the parameters or features of the solution[34].
- **Population:** a collection of individuals, each representing a potential solution to the optimization problem[35].
- **Fitness Function:** an objective function that evaluates how well a solution performs with respect to the optimization criteria[36].
- **Crossover (Recombination):** the process of exchanging genetic material between two individuals to create new offspring[42]. The main role of crossover is to inherit some characteristics of the two parents to generate the offspring[43]. The crossover rate is commonly in the interval [0.45, 0.95][44].
- **Mutation:** the introduction of small random changes to the genetic material of an individual to maintain diversity in the population[45]. It represents small changes of the selected individual[46]. The Mutation Rate is commonly in the interval [0.001, 0.01][46].
- **Selection:** the process of choosing individuals from the current population to become parents for the next generation[47]. Individuals with a higher fitness value have more chance of being selected[48]. A replacement strategy is needed to decide which individuals will survive to the next generation[49].

As a heuristic algorithm, GA needs a termination criterion[50, 51]. The used GA stopping criterion is stopping after a maximum number of iterations (**4000**)[52].

Algorithm 1 Genetic Algorithm (GA) algorithm

1. population_size \leftarrow desire population size [53]
2. GenerationSize \leftarrow desire generation size [53]
3. population $\leftarrow \{\}$ [53]
4. for population_size time do [53]
5. for($i = 0; i < \text{population_size}/2; i++$) do [53]
6. population \leftarrow population $\cup \{\text{Nearest Neighbor individual}\}$ [53]
7. for($i = \text{population_size}/2; i < \text{population_size}; i++$) do [58]
8. population \leftarrow population $\cup \{\text{Random individual}\}$ [60, 61]
9. Best \leftarrow None [62]
10. Repeat [63]
11. for each individual $P \in$ population do [65]
12. Calculate_fitness(P) [69]
13. if Best = None or Calculate_fitness(P) $>$ Calculate_fitness(Best) then [70]
14. Best $\leftarrow P$ [71]
15. offspring $\leftarrow \{\}$ [73]
16. for population_size/2 do [75]
17. Parents \leftarrow Roulette_wheel_selection(population) [77, 78]
18. children \leftarrow crossover(copy(Parents)) [80, 81]
19. offsprings \leftarrow offsprings \cup mutation(children) [83, 84]
20. population \leftarrow half of population \cup half of offsprings [85]
21. Until reach GenerationSize [86]
22. return Best [87]

2.2 Solution Representation

One solution representation method for CVRP is **permutation-based**[93]. A permutation of the depot (0) and N customers is used to represent a solution, where 0 can appear more than once for a new route, and each customer can only occur once[94]. A one-to-one representation-solution mapping function is used[97].

Example of a Solution (Instance A-n32-k5): The solution is determined to have 5 routes for 31 customers (0 represents the depot)[95, 96].

```
[  
  [0, 27, 24, 14, 26, 30, 16, 0],  
  [0, 21, 31, 19, 17, 13, 7, 0],  
  [0, 8, 18, 22, 9, 15, 10, 25, 5, 29, 20, 0],  
  [0, 2, 3, 23, 28, 4, 11, 6, 0],  
  [0, 12, 1, 0]  
]
```

2.3 Fitness Function

The used fitness function is the **inverse of the objective function**, which computes the total route distance for all routes in the solution[113].

The **objective function** is the total distance for all routes in the solution[116]. The total distance for each route is calculated by: (departing distance from depot to first node) + (distances between all following nodes) + (returning distance from last node to the depot)[115]. The goal is minimization of this objective function[117].

The objective function F is given by:

$$F = \sum_{r=1}^v \left(d_{\pi(0), \pi(1)} + \sum_{i=1}^{n-1} d_{\pi(i), \pi(i+1)} + d_{\pi(n), \pi(0)} \right) [118]$$

where v is the number of vehicles, n is the number of customers, π represents a permutation encoding of a route r , and $\pi(0)$ represents the depot[118, 119].

The **fitness function** is:

$$\frac{1}{F} [120]$$

2.4 Constraint Handling

The most important constraint is the **vehicle capacity**: the total load of a vehicle (summation of route customers' demands) must not exceed the vehicle capacity[122].

The GA implements a **preserving strategy** for constraint handling, which ensures the generation of feasible solutions through specific representation and operators[123].

- The initial population consists of a set of initial feasible solutions where each vehicle load satisfies the capacity[124].
- After crossover, the produced nodes in the permutation are arranged in routes according to vehicle capacity[125].
- For mutation, only feasible solutions will be generated because the Swap (intra-route) operator is used[126].

2.5 Initial Population

The initial population is a collection of potential solutions (individuals/chromosomes)[128].

- **50%** of the population size is generated using the **Nearest Neighbor (NN)** algorithm[129].
- The other **50%** is generated **randomly** to improve the diversification of the algorithm[136].

In the NN algorithm, each route is initialized with a randomly chosen node and continues until all vehicle capacities are consumed or all demands are satisfied[137].

Algorithm 2 Nearest Neighbor (NN) algorithm

1. Input: Demand list D , Number of vehicles k , Vehicle Capacity (C), List of Unvisited Nodes (N) [141, 142]
2. Initialize an empty Routes list [142]
3. $v = 0$ [142]
4. While $v < k$ Do [142]
 5. route = [] [142]
 6. capacity = 0 [142]
 7. idx = random unvisited node [142]
 8. Append idx to route [142]
 9. capacity = capacity + $D[\text{idx}]$ [142]
 10. While there is unvisited node and capacity $< C$ Do [142]
 11. j = Nearest Unvisited Node [142]
 12. If (capacity + $D[j] < C$) Then [142]
 13. Append j to route [142]
 14. capacity = capacity + $D[j]$ [142]
 15. Else [142]
 16. Break [142]
 17. End while [142]
 18. $v = v + 1$ [142]
 19. Append route to Routes list [142]
20. End while [142]
21. Return Routes list [142]

The output is a feasible solution where each vehicle's total load satisfies the capacity[146].

2.6 Parents Selection

The **Roulette Wheel Selection** method is used[152].

1. Calculate the fitness (f_i) of each individual[153].
2. Normalize the fitness values to convert them into probabilities (p_i)[154, 156]:

$$p_i = \frac{f_i}{\sum_{j=1}^n f_j} \quad [157]$$

3. Generate a random number between 0 and 1[158].
4. Select the individual whose probability slice contains the randomly generated number (larger fitness \Rightarrow larger slice \Rightarrow higher chance of selection)[160, 161].
5. Repeat the process to select the desired number of individuals; the same individual can be selected more than once[162, 163].

2.7 Crossover Operator

The crossover operator used for permutation representation is **Order Crossover**[165, 166].

Order Crossover Steps:

1. Two crossover points are randomly selected from **parent 1**[167].
2. The part between the two points is copied from parent 1 to the **offspring 1**[168].
3. From **parent 2**, start at the first node and pick the elements that are not already selected (copied from parent 1) to fill the remaining parts of offspring 1[169, 170].
4. Two crossover points are randomly selected from **parent 2**[175].
5. The part between the two points will be copied from **parent 1** to the **offspring 2**[176].
6. From **parent 1**, start at the first node and pick the elements that are not already selected (copied from parent 1) to fill the remaining parts of offspring 2[179, 180]. (Note: Step 6 repeats parent 1/parent 1, which appears to be a transcription error in the original text, likely meant to be "From parent 2, start at the first node and pick elements..." for a standard Order Crossover approach, but the source explicitly states parent 1/parent 1 for offspring 2's filling sequence [179, 180]).

2.8 Mutation

The mutation operator used is **Swap (intra-route)**: swap two randomly chosen nodes from one randomly chosen route[251].

Example (Selected route is 3, selected nodes are 2 and 4):

Before mutation:

```
[[4, 11, 8, 18, 22, 9, 15, 10], [13, 7, 16, 26, 30, 12],  
[28, 23, 3, 2, 6, 14, 24], [29, 5, 20, 27, 1, 21, 31], [17, 19, 25]]
```

After mutation (nodes 2 and 4 swapped in route 3):

```
[[4, 11, 8, 18, 22, 9, 15, 10], [13, 7, 16, 26, 30, 12],  
[28, 23, 3, 6, 14, 2, 24], [29, 5, 20, 27, 1, 21, 31], [17, 19, 25]]
```

2.9 Replacement

The replacement strategy used is replacing approximately **50% of the old population** with the new offsprings[262, 267].

2.10 Diversification and Intensification strategies

- **Diversification** (Exploration): Starting with a diverse initial population helps with exploration[273]. Mutation operators generally generate worse offspring, which increases diversification[274].
- **Intensification** (Exploitation): Crossover operators mostly produce children that are at least better than their worse parent, increasing the intensification of the algorithm[275, 276].

3 Motivation

CVRP is an NP-hard problem, so exact methods can only solve small instances in a reasonable time[278]. Metaheuristic methods are required for larger instances and real-life transportation problems to find optimal or near-optimal routes efficiently[279, 280]. GA is one of the commonly used P-metaheuristics that is attractive for solving optimization problems because it can obtain good solutions in reasonable time[281]. The work also aims to compare the results of GA and Tabu Search (TS)[282].

4 Experimental Setup

4.1 Implementation Environment

The experiment was implemented using the **Python** programming language and run on the **Google Colab** platform, with a 1 core 2.25 GHz CPU (AMD EPYC 7B12) and 13 GB RAM[293].

4.2 Parameter Tuning

The **offline parameter tuning** strategy was used, where parameters were tuned one at a time to empirically determine their optimal value[295].

Parameters (Factors)	Levels	Optimal value
Generation	500, 1000, 4000, 5000	4000
Population Size	50, 70, 100	100
Crossover Rate	0.8, 0.85, 0.9	0.9
Mutation Rate	0.08, 0.09, 0.1	0.1

Table 1: Values of parameters tuning. [297]

4.3 Datasets

Three benchmark instances from the **CVRPLIB** public library were used: **A-n32-k5**, **F-n135-k7**, and **M-n200-k17**[301].

Instance	Nodes number	Vehicles number	Capacity
A-n32-k5	31	5	100
F-n135-k7	134	7	2210
M-n200-k17	199	17	200

Figure 1: A-n32-k5 Instance.

Figure 2: F-n135-k7 Instance.

Figure 3: M-n200-k17 Instance.

Table 2: The used CVRP benchmark instances. [304]

5 Results and Discussion

5.1 Performance Measurement

- **Solution Quality:** Measured by the percent deviation of the obtained solution from the **Best-Known Solutions (BKS)** from CVRPLIB[367, 368].
- **Computational Effort:** Measured by the clock time (without I/O and pre/post-processing) [370] and the number of objective function evaluations (independent of the computer system)[371, 372].
- **Robustness:** Measured by performing 7 different runs on the same instance and taking the average of the obtained results, due to the randomness in the algorithm[372, 373].

The GA results are compared with the results achieved in Assignment 2 using **Tabu Search (TS)**[374].

5.2 Solution Quality Comparison (GA vs. TS)

The difference between the best, worst, and average results for GA is considered reasonable, similar to TS[377]. The GA efficiency decreases as the instance size increases, which is expected due to the problem's complexity increasing with size[382].

- **A-n32-k5:** Best GA objective is 849.30 (8.3% worse than BKS of 784.00)[379, 406].
- **F-n135-k7:** Best GA objective is 1301.39 (12% worse than BKS of 1162.00)[380, 406].
- **M-n200-k17:** Best GA objective is 1542.37 (21% worse than BKS of 1275.00)[381, 406].

TS outperforms GA in terms of solution quality for all instances, based on the best obtained results and their averages[384]. However, GA achieved a better worst obtained result for the F-n135-k7 instance[385].

Run #	Tabu search			Genetic Algorithm		
	A-n32-k5	F-n135-k7	M-n200-k17	A-n32-k5	F-n135-k7	M-n200-k17
1	816.68	1296.12	1465.56	849.30	1307.10	1559.47
2	842.81	1282.99	1501.58	854.72	1310.20	1571.91
3	818.57	1353.35	1564.06	854.18	1315.90	1547.80
4	849.44	1309.37	1553.67	854.72	1315.96	1571.56
5	807.70	1300.89	1478.67	856.25	1301.39	1559.19
6	839.93	1281.21	1524.78	854.74	1303.85	1542.37
7	816.68	1299.16	1494.28	856.25	1303.23	1561.01
Best	807.70	1281.21	1465.56	849.30	1301.39	1542.37
Worst	849.44	1353.35	1564.06	856.25	1315.96	1571.91
Avg.	827.40	1303.30	1511.80	854.31	1308.23	1559.04

Table 1: Tabu search and Genetic Algorithm solutions quality. (*Bold indicate best result)

Table 3: Tabu search and Genetic Algorithm solutions quality. [395]

Instance	BKS	Tabu search		Best Diff (%)		Genetic Algorithm		Avg. Diff (%)	
		Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
A-n32-k5	784.00	807.70	827.40	3.0	5.5	849.30	856.04	8.3	9.2
F-n135-k7	1162.00	1281.21	1303.30	10.3	12.2	1301.39	1308.23	12.0	12.6
M-n200-k17	1275.00	1465.56	1511.80	14.9	18.6	1542.37	1559.04	21.0	22.3

Table 2: Comparison between Tabu search solutions, Genetic Algorithm solutions and best-known solutions for CVRP. (*Bold indicate best result)

Figure 4: Best solutions of TS and GA compared to BKS.

Table 4: Comparison between Tabu search solutions, Genetic Algorithm solutions and best-known solutions for CVRP. [405]

5.3 Computational Time and Objective Function Evaluation

- **Computational Time:** The average computational time of GA is **lower** compared to TS, thus GA outperforms TS in this criterion[392]. The long computational time for F-n135-k7 is due to the small number of vehicles compared to the number of nodes, resulting in longer routes[391].
- **Objective Function Evaluation:** GA shows a constant number of evaluations (400,000 for each run) because it evaluates the whole population at each generation for selection and finding the best individual[393, 435]. GA is considered more stable in this metric than TS, and GA is better for all instances except the best case of A-n32-k5[394].

Run #	Tabu Search			Genetic Algorithm		
	A-n32-k5	F-n135-k7	M-n200-k17	A-n32-k5	F-n135-k7	M-n200-k17
1	0:03:08	1:27:50	0:28:48	0:00:51	0:29:52	0:10:08
2	0:02:55	2:01:18	0:21:39	0:00:43	0:28:27	0:10:23
3	0:03:56	1:52:07	0:23:32	0:00:40	0:27:35	0:10:01
4	0:02:57	1:03:59	0:23:17	0:00:41	0:27:21	0:10:15
5	0:03:10	1:41:22	0:19:14	0:00:41	0:27:45	0:09:58
6	0:02:58	1:53:25	0:34:50	0:00:42	0:28:08	0:10:59
7	0:03:02	1:27:12	0:18:46	0:00:42	0:28:22	0:09:45
Best	0:02:55	1:03:59	0:18:46	0:00:40	0:27:21	0:09:45
Worst	0:03:56	2:01:18	0:34:50	0:00:51	0:29:52	0:10:59
Avg.	0:03:10	1:38:10	0:24:18	0:00:43	0:28:13	0:10:13

Table 3: Computational time in seconds for Tabu search and genetic algorithm. (*Bold indicate best result)

Table 5: Computational time in seconds for Tabu search and genetic algorithm. [425]

Run #	Tabu search			Genetic Algorithm		
	A-n32-k5	F-n135-k7	M-n200-k17	A-n32-k5	F-n135-k7	M-n200-k17
1	379357	1401036	664468	400000	400000	400000
2	388088	2083798	738333	400000	400000	400000
3	485945	1421682	720146	400000	400000	400000
4	389232	1117927	640540	400000	400000	400000
5	396196	1481537	1155677	400000	400000	400000
6	401504	1901465	625546	400000	400000	400000
7	383833	1214799	988878	400000	400000	400000
Best	379357.00	1117927.00	625546.00	400000.00	400000.00	400000.00
Worst	485945.00	2083798.00	1155677.00	400000.00	400000.00	400000.00
Avg.	403450.71	1517463.43	790512.57	400000.00	400000.00	400000.00

Table 4: Number of objective function evaluation for Tabu search and genetic algorithm. (*Bold indicate best result)

Table 6: Number of objective function evaluation for Tabu search and genetic algorithm. [431]

5.4 Best Obtained Solutions and Routes

The following presents the best obtained solution for each of the three instances, including their plotted routes.

Figure 5: Best solution for instance A-n32-k5.

Routes of best solution for A-n32-k5: [439]

```
[[21, 31, 19, 17, 13, 7],
 [27, 24, 14, 26, 30, 16],
 [8, 18, 22, 9, 15, 10, 25, 5, 29, 20],
 [6, 3, 2, 23, 28, 4, 11],
 [12, 1]]
```

Figure 6: Best solution for instance F-n135-k7.

Routes of best solution for F-n135-k7: [447]

```
[[26, 25, 21, 91, 22, 24, 23, 72, 73, 74, 76, 134, 32, 48, 1, 75, 47, 77, 64,
 78, 63, 79, 34, 49, 62, 52, 51, 50, 53, 102, 103, 56, 57, 105, 104, 101, 100,
 98],
 [6, 5, 4, 9, 7, 11, 12, 10, 2, 42, 41, 40, 44, 43, 39, 38, 96, 97, 99, 3, 18,
 118, 46],
 [80, 33, 68, 69, 70, 67, 133, 66, 71, 17, 14, 88, 92, 28, 30, 31, 59, 60, 61,
 45, 94, 93, 29],
 [83, 85, 84, 86, 87, 89, 90, 16, 13, 15, 54, 55, 58, 27],
 [120, 109, 108, 107, 106, 114, 115],
 [111, 125, 112, 126, 127, 128, 129, 113, 119, 117, 131, 116, 132],
 [82, 20, 19, 130, 65, 124, 123, 122, 110, 121, 81]]
```

Figure 7: Best solution for instance M-n200-k17.

Routes of best solution for M-n200-k17: [466]

```
[[129, 79, 185, 33, 81, 120, 9, 161, 103, 188],
 [116, 196, 76, 184, 77, 3, 158, 68, 150, 80, 177, 109],
 [82, 48, 124, 47, 168, 123, 19, 107, 175, 11, 126, 63, 90],
 [171, 74, 73, 21, 198, 110],
 [92, 151, 59, 99, 104, 96, 94, 183, 6, 147, 89, 166],
 [172, 42, 142, 14, 119, 192, 44, 141, 191, 91, 193, 100, 37, 98],
 [176, 1, 132, 69, 162, 101, 70, 30, 122, 51, 20, 128, 160, 131, 170],
 [5, 118, 60, 83, 199, 114, 8, 174, 125, 45, 17],
 [164, 34, 78, 169, 121, 29, 24, 163, 134, 54, 130, 165, 55, 25, 32],
 [194, 106, 153, 7, 182, 148, 62, 159, 10, 189, 108],
 [186, 56, 139, 187, 39, 23, 67, 4, 155, 179],
 [85, 93, 61, 173, 84, 113, 16, 86, 140, 38, 43],
 [152, 58, 40, 180, 26, 149, 195, 12, 154, 138],
 [137, 13, 117, 95, 97, 87, 144, 57, 178, 2, 115, 145, 41, 15, 53],
 [65, 136, 35, 135, 71, 66, 181, 64, 49, 143, 36, 46, 18, 52],
 [190, 127, 31, 88, 167, 27, 146, 156, 112, 105],
 [102, 157, 50, 111, 28]]
```

6 References

1. P. Toth and D. Vigo, The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, 2002. doi: 10.1137/1.9780898718515. [587]
2. Z. Borcinova, 'Two models of the capacitated vehicle routing problem', Croat. Oper. Res., vol. 8, pp. 463-469, Dec. 2017, doi: 10.17535/corr.2017.0029. [588, 589]
3. G. Laporte, 'What you should know about the vehicle routing problem', Nav. Res. Logist. NRL, vol. 54, no. 8, pp. 811-819, 2007, doi: 10.1002/nav.20261. [590, 591]
4. J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, 1992. [592, 593]
5. Talbi and E.-G. Talbi, Metaheuristics: From Design to Implementation, vol. 74. 2009. doi: 10.1002/9780470496916. [594]
6. A. Scheibenpflug and S. Wagner, 'An Analysis of the Intensification and Diversification Behavior of Different Operators for Genetic Algorithms', in Computer Aided Systems Theory - EUROCAST 2013, R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 364-371. doi: 10.1007/978-3-642-53856-8_46. [595, 596]
7. 'CVRPLIB All Instances'. Accessed: Nov. 14, 2023. [Online]. Available: <http://vrp.galgos.inf.puc-rio.br/index.php/en/> [597]