King Saud University
College of Computer and Information Sciences
Computer Science Department

كلية علوم الحاسب والمعلومات
قسم علوم الحاسب

Course Title:	Selected topics in Artificial Intelligence
Course Code:	CS569

Assignment 2

Single-solution based metaheuristic to solve Capacitated Vehicle Routing Problem (CVRP)

Student #1: Name, ID

Student #2: Name, ID

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. VRP solutions have specific objectives and limitations in real applications, making VRP have categories or variants. One of the most popular VRP variants is Capacitated Vehicle Routing Problem (CVRP). This problem is considered a fundamental problem in **combinatorial optimization**. It is known to be **NP-hard problem** and a generalization of Traveling Salesman Problem [1]. CVRP can be described as follows, there is a set of geographically dispersed customers where each customer has a certain demand. There is also one centralized depot and the travel costs between all locations are given. To serve customers' demands, there is a fleet of identical vehicles whose base is the depot, and each vehicle has a certain capacity [2]. The main goal of this problem is to find the **minimum cost routes** for the capacitated vehicles to serve customers' demands [1], with the following constraints [3]:

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

2. Methodology (S-metaheuristic)

This section shows the details of the used Single-Solution Based metaheuristics (S-metaheuristics) to solve CVRP, that is **Tabu Search** (TS). It is an iterative, memory-based neighborhood-search method. TS was first proposed in 1986 by Fred Glover [4]. TS works like an ordinary local search iteratively from one solution to another until reaching a stopping criterion. The improvement is that TS uses some strategies to escape from local optima, thus when all neighborhood solutions are non-improving, best neighbor may be accepted. The used acceptance strategy is the **best improvement** that explores the whole neighborhood and selects the best solution among them.

Tabu list is one of the essential elements of TS that help to escape from local optima. It is used to prevent cycling by accepting non-improved solution to move beyond local optima. It is a short-term memory of the recently visited solutions. Therefore, a non-improving neighbor is accepted only if it is not in the tabu list [5]. The used tabu list has static size. Moreover, the used aspiration criteria is to allow solution with an objective value better than the best so far solution, even if it is tabu.

As a heuristic algorithm, TS needs termination criterion in the iterations in order to get the solution in the reasonable time. In practice, obviously, the search has to be stopped at some point. The used stopping criteria is to stop after 1000 iterations without an improvement in the objective function value. Following is the used algorithm of TS:

Algorithm 1 Tabu Search (TS) algorithm

Input: MaxIteration I, TabuListSize T, initial_solution x

```
1. Initialize an empty Tabu list
2. Let best_solution ← x { best_solution is the best so far solution}
3. $x'$ ← x {$x'$ is the current solution}
4. Repeat
5.     Generate a set S of solution N(x) {N(x) is the current neighborhood of x}
6.     Select the best neighborhood $x'' \in$ S
7.     If ( $f(x'') < f$ (best_solution) ) Then
8.         best_solution ← $x''$ {aspiration condition: if current move improves best so far,
           accepts the solution even it is tabu}
9.         $x'$ ← $x''$        {update the current solution}
10.        T ← T + $x''$     {update the tabu list}
11.    Else
12.        If ($x'' \in$ N(x) / T) Then
13.            $x'$ ← $x''$        {update the current solution}
14.            T ← T + $x''$     {update the tabu list}
15. Until (reach MaxIteration I or 1000 iterations without improvement)
16. Return best_solution
```

Output: A best found solution

2.1 Solution Representation

There are some solutions representation methods for CVRP, one of them is permutation based. A permutation of the depot 0 and N customers is used to represent a solution to the problem, where 0 can appear more than once, each time for a new route, while each custom can only occur once. Figure 1 shows an example of one solution for the instant A-n32-k5. As shown, 0 represents depot and there are 31 customers. 5 routes are determined for this solution. One-to-one representation-solution mapping function is used where each solution is represented by a single encoding and each encoding represents a single solution.

```
[[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]]
```

Figure 1: Example of one of the solutions for the instant A-n32-k5

2.2 Objective Function

The objective function is essential for evaluate solution quality of any metaheuristic to. The used objective function computes the total route distance for all routes in the solution. It is considered a **self-sufficient objective function** where the definition of the objective function is straightforward. To find the total distance for each route, first the departing distance from the depot to the first node is added, then the distances between all following nodes are added, and finally the returning distance from the last node to the depot is added. The objective function of a solution is the total distances for all routes in the solution. In the TS we are looking for minimisation of this objective function. The following equation shows the used objective function:

$\sum_{r=1}^{v}(d_{\pi(0),\pi(1)} + \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(0)})$, where v is number of vehicles, n is number of customer, π represents a permutation encoding of a route r, and $\pi(0)$ represents the depot.

2.3 Constraint Handling

Vehicle capacity is the most important constraint in CVRP, every vehicle total load (the summation of the vehicle route customers' demands) must not exceed the vehicle capacity. We implement **preserving strategy** for constraint handling in which specific representation and operators will ensure the generation of feasible solutions. A feasible solution where each vehicle load satisfied the capacity is generated as initial solution. Furthermore, neighborhood move operators produce only feasible solutions.

2.4 Initial Solution

A **Nearest Neighbor** (NN) algorithm is one of the most known heuristic algorithms to solve optimization problems as in CVRP. Each route is initialized with randomly chosen node from the problem set and continues until all the vehicle capacities are consumed or all the demands are satisfied. The algorithm requires the number of vehicles, capacity of the vehicles and the node list to conclude with a feasible solution. Following is the used algorithm:

Algorithm 2 Nearest Neighbor (NN) algorithm

Input: Demand list D, Number of vehicles k, Vehicle Capacity (C), List of Un-visited Nodes (N)

```
1. Initialize an empty Routes list
2. v = 0
3. While v < k Do
4.     route = []
5.     capacity = 0
6.     idx= random unvisited node
7.     Append idx to route
8.     capacity = capacity + D[idx]
9.     While there is unvisited node and capacity < C Do
10.            j = Nearest Unvisited Node
11.            If (capacity + D[j] < C) Then
12.                Append j to route
13.                capacity = capacity + D[j]
```

```
14.             Else
15.                 Break
16.     End while
17.     v = v + 1
18.     Append route to Routes list
19. End while
20. Return Routes list
```

Output: A feasible solution where each vehicle total load satisfied the capacity.

2.5 Neighborhood Moves

Neighborhood solutions are created by changing one attribute (e.g. arcs) or a combination of attributes of a given solution. A better neighbourhoods solution replaces the current solution, and the search continues with the new current solution. A random integer number between 0 and 4 is generated by the computer, then one of five move operators: 2-opt (intra-route), Swap (inter-route), Swap (intra-route), Insert, and Inverse is adopted with the same possibility according to the generated random number. All of these move operators produce feasible solution only. The details of these moves are shown in the following Table 1.

Table 1: Neighborhood move operators.

Move	Definition	Example
2-opt (intra-route)	Tour is improved iteratively by replacing two of its edges (i, i + 1) and (j, j + 1) by two other edges (i, j) and (i + 1, j + 1), reversing the orientation of the path. An improvement is given, when the (sub-) travel cost of the new route is smaller.	Initial route: [21, 31, 19, 17, 13, 7] After 2-opt: [21, 31, 17, 19, 13, 7]
Swap (inter-route)	Swap two randomly chosen nodes i and j from two randomly chosen routes r1 and r2.	Initial routes: [21, 31, 19, 17, 13, 7] and [12, 5, 14, 11, 3] After swap: [21, 31, 19, 17, 14, 7] and [12, 5, 13, 11, 3]
Swap (intra-route)	Swap two randomly chosen nodes from one randomly chosen route.	Initial route: [21, 31, 19, 17, 13, 7] After swap:

		[21, 13, 19, 17, 31, 7]
Insert	From one randomly chosen route, two nodes are randomly chosen and insert the back one before the front.	Initial route: [21, 31, 19, 17, 13, 7] After insert: [21, 13, 31, 19, 17, 7]
Inverse	Inverse the subsequence between two different random nodes of a randomly chosen route.	Initial route: [21, 31, 19, 17, 13, 7] After inverse: [21, 13, 17, 19, 31,7]

2.6 Diversification and Intensification strategies

The balance between diversification and intensification is crucial for the efficiency of any metaheuristic. For diversification, TS uses a tabu list to prevent revisiting explored solution while intensification is achieved by using neighboring search to focus on improving the current solution.

3. Motivation

CVRP is NP-hard problem like all VRPs, for this reason, only a small instance size can be solved with exact methods in reasonable time. Metaheuristic methods can solve problems with larger size instances. Moreover, real-life transportation problems can have many customers and companies. It should be able to find optimal or near optimal routes efficiently in short time. Therefore, various metaheuristic algorithms are used. TS is one of the commonly used metaheuristics for solving variety of optimization problems. It is an improvement of the local search algorithm, that can escape from local optima by accepting non-improving solutions when all neighbors are non-improving, and it incorporates a memory of tabu list to avoid cycling between solutions. The tabu list encourages exploration, and this is performed through the whole search process. Our TS algorithm for the CVRP has five neighborhoods move operators. One of these operators is selected randomly to generate each neighborhood solution.

4. Experimental Setup

This section provides a description of the used datasets, parameter tunning, and the experimentations settings.

4.1 Implementation Environment

In this experiment, Python programming language was used for model implementation and the whole experiment is run on the Google Colab platform, with 1 core 2.25 GHz CPU (AMD EPYC 7B12) and 13 GB RAM.

4.2 Parameter Tuning

The **offline parameter tuning** strategy is used in this experiment where parameters are tuned one at a time and their optimal value is determined empirically. The following Table 2 shows the details of the parameter tuning.

Table 2: Values of parameter tuning.

Parameters (Factors)	Levels	Optimal value
Tabu list size	20, 50 ,100 ,200	100
Number of neighborhoods	100, 200, 300, 400, 500	500
Number of iterations (as stopping criterion of TS)	100, 1000, 2000, 5000, 10000, 20000, 50000	50000, and stop after 1000 iteration without improvement

4.3 Datasets

The used set of instances to test a metaheuristic algorithm must be diverse in terms of size, difficulty, and structure. In this experiment, the **constructed instances** are used. Specifically, **three** benchmark instances from CVRPLIB public library of different sizes: *A-n32-k5*, *F-n135-k7*, and *M-n200-k17* [6]. Table 3 shows the customer nodes number, vehicles number, and vehicle capacity for each instance. Figures 2-4 show a plot for these instances.

Table 3: The used CVRP benchmark instances.

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

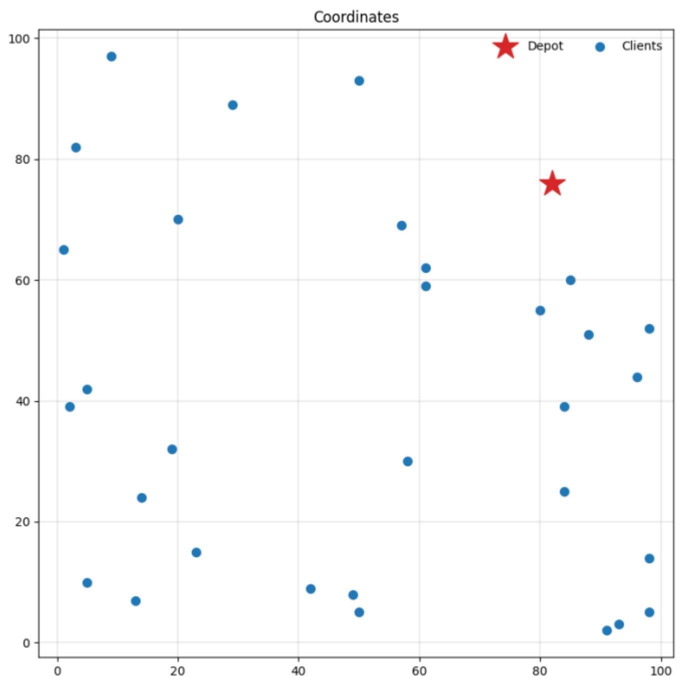

Figure 2: A-n32-k5 Instance

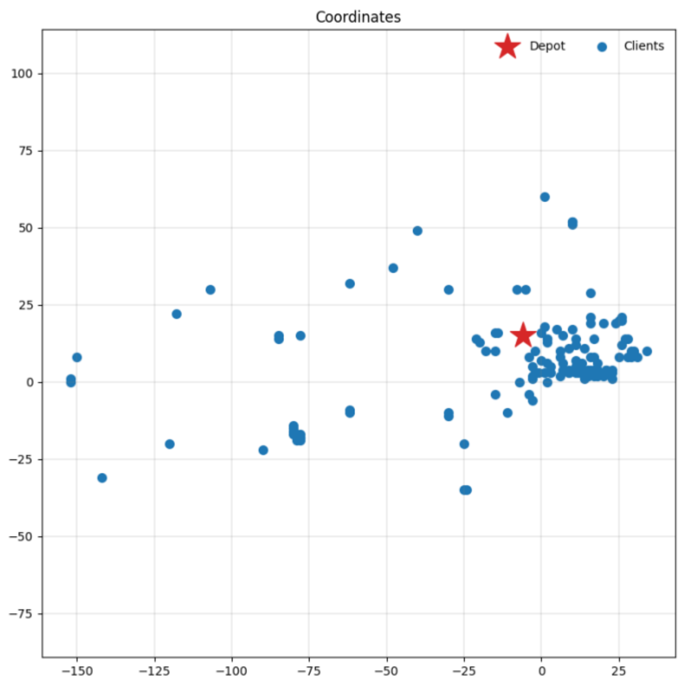

Figure 3: F-n135-k17 Instance

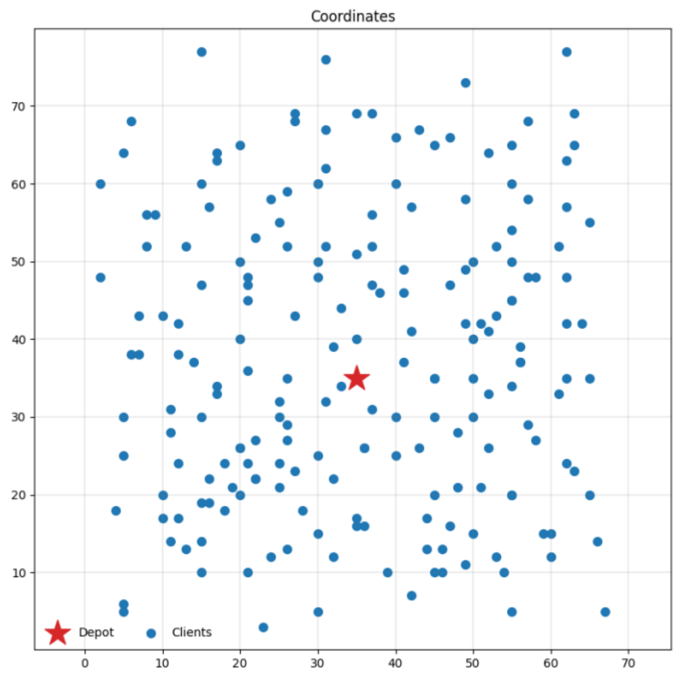

Figure 4: M-n200-k17 Instance

5. Results and Discussion

To measure the performance of the proposed metaheuristics after executing different experiments, some statistical analyses are applied on the obtained results. **Solution quality** indicator that measures the percent deviation of the obtained solution to the **Best-Known Solutions (BKS)** is used. BSKs are obtained from the CVRPLIB [6] for the used three instances A-n32-k5, F-n135-k7, and M-n200-k17. Percent deviation is computed of both the best obtained solution and the average of all obtained solutions for each instance.

Moreover, to measure the **computational effort**, the clock time, without input/output and pre-processing/post-processing time is calculated. Additionally, the number of objective function evaluations is calculated to measure the computational effort without relying on the computer system. To measure the **robustness** of the developed TS, 7 different runs on the same instance are performed to take the average of the obtained results. This is required due to the randomness in the algorithm.

Table 4 shows the obtained result for each instance during the 7 runs. The last three rows show the best, worst, and average of the obtained solution. As shown the difference between each of best, worst, and average is reasonable. Table 5 shows a comparison between the obtained solution and BKS for CVRP. For the smallest applied instance, A-n32-k5, the objective of best solution obtained is 807.70 which is 3% worse than BKS. For instance F-n135-k7, the best obtained solution objective is 1281.2 which is 10.3% worse than BKS. For the largest applied instance M-n200-k17, the best obtained solution objective is 1465.56 which is 14.9% worse than BKS. As shown in Figure 5 the TS efficiency decreases with the increase of the instance size, and this is justifiable since the problem complexity increases with its size. Although the basic version of TS is used, the obtained solution is considered acceptable.

Computational time in seconds is shown in Table 6, the average computational time increases with increase of the instance size, the instance F-n135-k7 computational time is longer since it needs more iterations to converge. Table 7 shows the number of objective function evaluation, the table shows large numbers, and this is acceptable since TS performs frequent evaluations to the objective function, to evaluate many neighborhood solutions to select the best one.

Table 4: Tabu search solutions quality.

	Instance		
Run #	**A-n32-k5**	**F-n135-k7**	**M-n200-k17**
1	816.68	1296.12	1465.56
2	842.81	1282.99	1501.58
3	818.57	1353.35	1564.06
4	849.44	1309.37	1553.67
5	807.70	1300.89	1478.67
6	839.93	1281.21	1524.78
7	816.68	1299.16	1494.28
Best	807.70	1281.21	1465.56
Worst	849.44	1353.35	1564.06
Avg.	827.40	1303.30	1511.80

Table 5: Comparison between Tabu search solutions and best-known solutions for CVRP.

Instance	**BKS**	**Tabu search**		**Best**	**Avg.**
		Best	**Avg.**	**Diff (%)**	**Diff (%)**
A-n32-k5	784.00	807.70	827.40	3.0	5.5
F-n135-k7	1162.00	1281.21	1303.30	10.3	12.2
M-n200-k17	1275.00	1465.56	1511.80	14.9	18.6

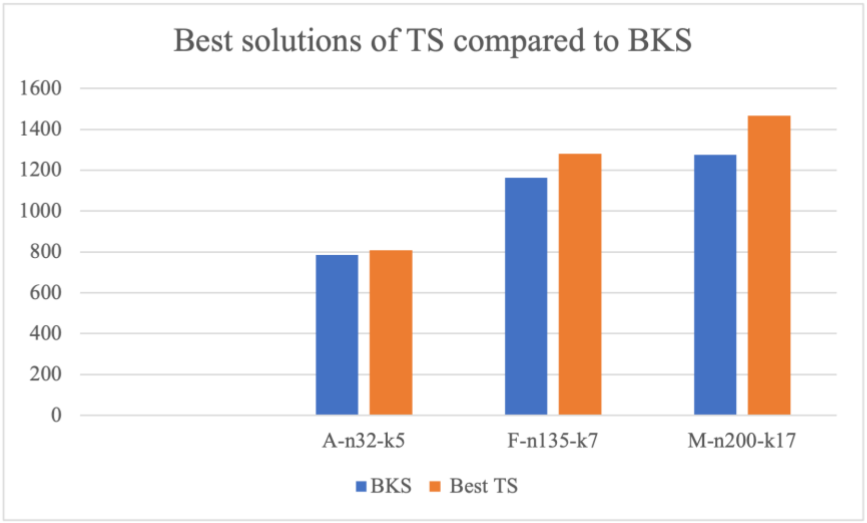

Figure 5: Best solutions of TS compared to BKS.

Table 6: Computational time in seconds.

	Instance		
Run #	**A-n32-k5**	**F-n135-k7**	**M-n200-k17**
1	0:03:08	1:27:50	0:28:48
2	0:02:55	2:01:18	0:21:39
3	0:03:56	1:52:07	0:23:32
4	0:02:57	1:03:59	0:23:17
5	0:03:10	1:41:22	0:19:14
6	0:02:58	1:53:25	0:34:50
7	0:03:02	1:27:12	0:18:46
Best	0:02:55	1:03:59	0:18:46
Worst	0:03:56	2:01:18	0:34:50
Avg.	0:03:10	1:38:10	0:24:18

Table 7: Number of objective function evaluation.

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

The following presents the best obtained solution for each of the three instances. Figures 6-8 show a plot for the best obtained solutions for each instance.

Routes of best solution for A-n32-k5:

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

Routes of best solution for F-n135-k7:

```
[[118, 117, 116, 131, 114, 115, 119, 130, 65, 19, 18, 17, 71, 66],
[91, 25, 26, 28, 30, 31, 59, 60, 61, 62, 50, 51, 52, 53, 102, 103, 104, 101,
35, 36, 99, 100, 98, 97, 105, 57, 58],
[77, 64, 63, 79, 67, 70, 69, 68, 80, 33, 133, 78, 76, 134, 32, 48, 1, 75,47],
[21, 87, 86, 84, 85, 83, 20, 82, 46, 72, 23, 24, 22, 27, 90, 89, 16, 92, 29,
93, 94, 45, 39, 38, 37, 96, 56, 55, 54, 49, 34, 74, 73],
[106, 107, 108, 109, 120, 121, 127, 128, 129],
[95, 40, 44, 43, 3, 41, 42, 2, 4, 5, 6, 7, 8, 9, 10, 12, 11, 14, 88, 15, 13],
[132, 81, 112, 125, 111, 110, 122, 123, 124, 126, 113]]
```

Routes of best solution for M-n200-k17:

```
[[153, 106, 194, 7, 48, 168, 47, 124, 82, 114, 60, 166],
[87, 144, 57, 178, 2, 115, 145, 41, 22, 133, 74, 171, 73],
[96, 104, 99, 93, 85, 91, 193, 100, 37, 98, 151, 92, 95],
[182, 148, 62, 159, 10, 189, 108, 126, 63, 90, 32],
[158, 129, 79, 185, 33, 81, 120, 9, 161, 103],
[12, 109, 177, 150, 80, 68, 3, 77, 116, 196, 76, 184],
[94, 59, 191, 141, 44, 119, 192, 14, 142, 42, 172, 97, 117],
[156, 112, 53, 105, 40, 180, 149, 179, 195, 138, 28],
[146, 27, 167, 52, 127, 190, 88, 31, 162, 101, 70, 30, 122],
[69, 132, 176, 1, 50, 102, 157, 169, 121, 29, 24, 163, 134, 54],
[21, 72, 197, 75, 23, 186, 56, 4, 155, 110, 26],
[118, 5, 84, 83, 199, 125, 45, 174, 8, 18],
[183, 61, 173, 17, 113, 16, 86, 140, 38, 43, 15, 137, 13],
[46, 36, 143, 49, 19, 123, 107, 175, 11, 64, 181, 131, 160],
[154, 130, 165, 55, 25, 170, 67, 39, 187, 139, 198],
[78, 34, 164, 135, 35, 136, 65, 71, 66, 128, 20, 188, 51],
[152, 58, 6, 147, 89, 111]]
```

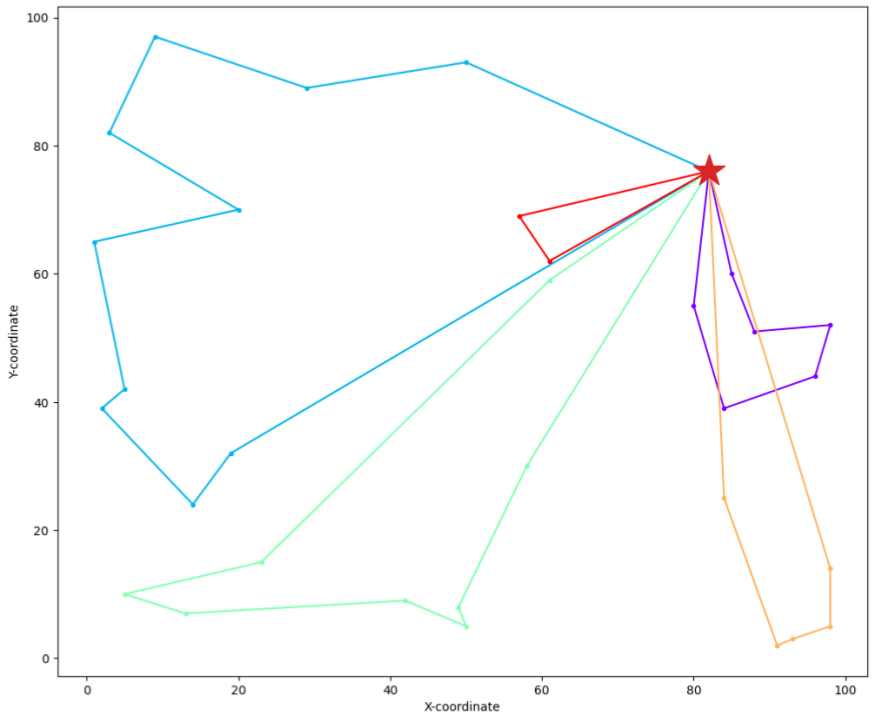

Figure 6: Best solution for instance A-n32-k5.

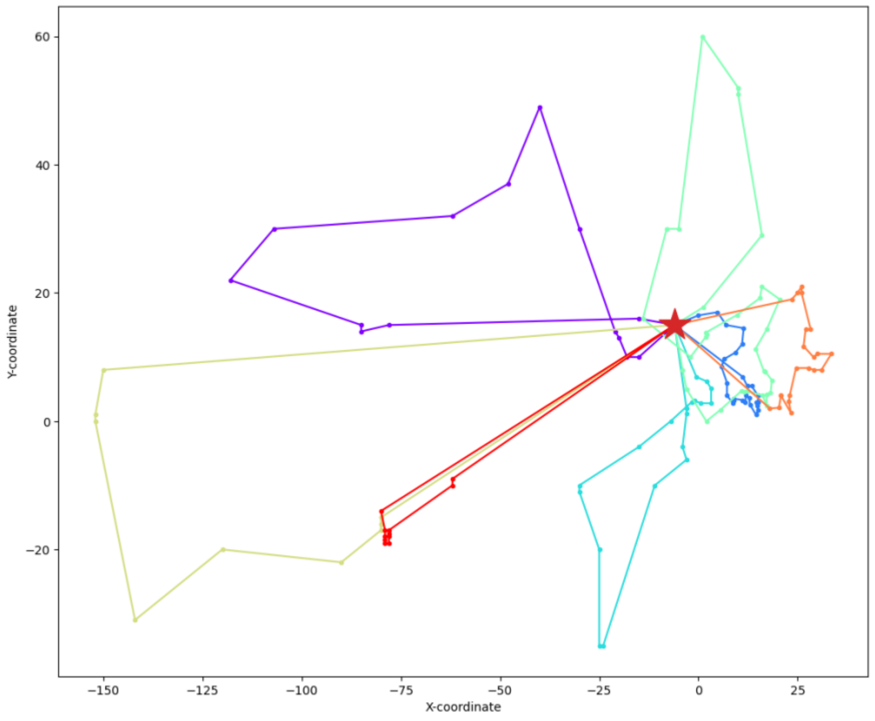

Figure 7: Best solution for instance F-n135-k7.

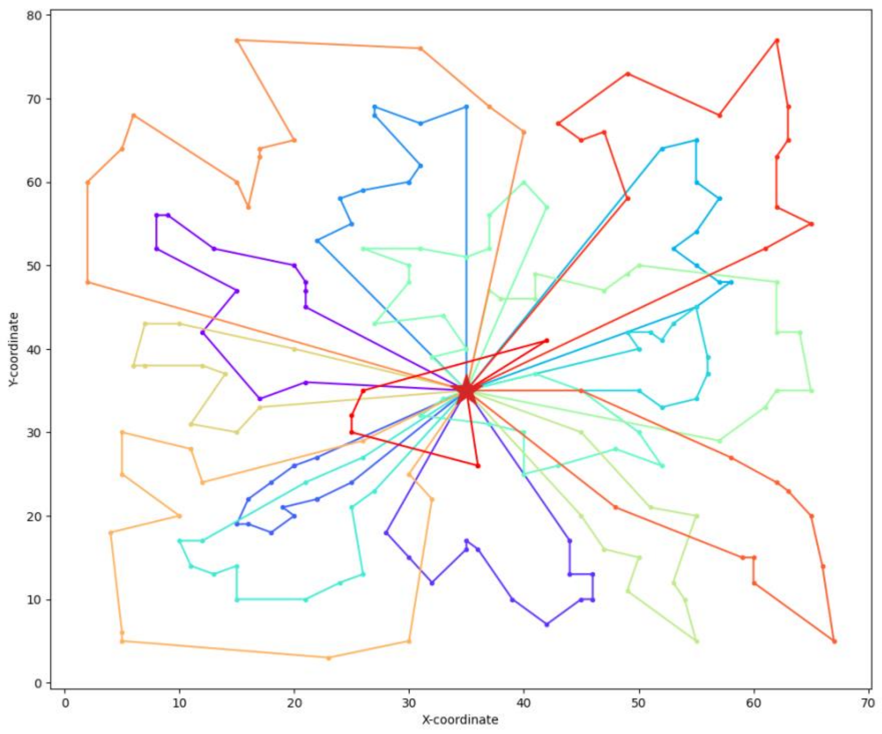

Figure 8: Best solution for instance M-n200-k17.

6. References

[1] P. Toth and D. Vigo, *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002. doi: 10.1137/1.9780898718515.

[2] Z. Borcinova, 'Two models of the capacitated vehicle routing problem', *Croat. Oper. Res. Rev.*, vol. 8, pp. 463–469, Dec. 2017, doi: 10.17535/crorr.2017.0029.

[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.

[4] F. Glover, 'Future paths for integer programming and links to artificial intelligence', *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, Jan. 1986, doi: 10.1016/0305-0548(86)90048-1.

[5] Talbi and E.-G. Talbi, *Metaheuristics: From Design to Implementation*, vol. 74. 2009. doi: 10.1002/9780470496916.

[6] 'CVRPLIB - All Instances'. Accessed: Nov. 14, 2023. [Online]. Available: http://vrp.galgos.inf.puc-rio.br/index.php/en/