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

جامعة الملك سعود
King Saud University
كلية علوم الحاسب والمعلومات
قسم علوم الحاسب

Course Title: Selected topics in Artificial Intelligence

Course Code: CS569

Assignment 3

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

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 (P-metaheuristic)

This section shows the details of the used Population- Based metaheuristics (P-metaheuristics) to solve CVRP, that is **Genetic algorithm** (GA). GA is developed by John Holland [4] in Michigan, USA in 1975. GA is an optimization algorithm inspired by the principles of natural evolution and genetics, and it is one of the most common evolutionary algorithms. Following are the main concepts in GA:

- **Chromosome/Genotype**: the representation of a potential solution. It is composed of genes that encode the parameters or features of the solution.
- **Population**: a collection of individuals, each representing a potential solution to the optimization problem.
- **Fitness Function**: an objective function that evaluates how well a solution performs with respect to the optimization criteria.

- **Crossover (Recombination)**: the process of exchanging genetic material between two individuals to create new offspring. The main role of crossover is to inherit some characteristics of the two parents to generate the offspring. Crossover rate represents the proportion of parents on which a crossover operator will act, and it is commonly in the interval [0.45, 0.95] [5].
- **Mutation**: the introduction of small random changes to the genetic material of an individual to maintain diversity in the population. It represents small changes of the selected individual. The Mutation Rate represents the probability to mutate each element (gene) of the representation (chromosome) and it is commonly in the interval [0.001, 0.01] [5].

- **Selection**: the process of choosing individuals from the current population to become parents for the next generation. A fitness value is assigned to each individual (based on its objective function value) and individuals with higher fitness value have more chance of being selected. A replacement strategy is needed to decide which individuals will survive to the next generation, replacement may be generational where new offspring replaced the old population or by elitism of selecting best individuals from both the parents and the offspring [5].

As a heuristic algorithm, GA needs termination criterion in order to get the solution in the reasonable time. In practice, obviously, the search has to be stopped at some point. The used GA stopping criteria is stopping after a maximum number of iterations (4000). Following is the used GA:

Algorithm 1 Genetic Algorithm (GA) algorithm

```
1. population_size ← desire population size
2. GenerationSize ← desire generation size
3. population ← {}
4. for population_size time do
5.     for (i=0; I < population_size/2; i++) do
6.         population ← population ∪ {Nearest Neighbor individual}
```

```
7.    for (i= population_size/2; I < population_size; i++) do
8.          population ← population ∪ {Random individual}
9. Best ← None
10. Repeat
11.    for each individual P ∈ population do
12.        Calculate_fitness (P)
13.        if Best = None or Calculate_fitness (P) > Calculate_fitness (Best) then
14.            Best ← p
15.    offspring ← {}
16.    for population_size/2 do
17.         Parents ← Roulette_wheel_selection(population)
18.         children ← crossover(copy(Parents))
19.         offsprings ← offsprings ∪  mutation (children)
20.    population ← half of population ∪ half of offsprings
21. Until reach GenerationSize
22. return Best
```

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 Fitness Function

The fitness function is essential for evaluate solution quality of any P-metaheuristic. The used fitness function is the inverse of the objective function, that computes the total route distance for all routes in the solution. The objective function 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:

$F = \sum_{r=1}^{v}(d_{\pi(0),\pi(1)} + \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(0)})$, 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, and the fitness function is: $\frac{1}{F}$.

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. The used GA implements **preserving strategy** for constraint handling in which specific representation and operators will ensure the generation of feasible solutions. A set of initial feasible solutions where each vehicle load satisfied the capacity is generated as initial population. Furthermore, the produce nodes in the permutation after the cross over will be arrange in routes according to vehicle capacity. For the mutation, only feasible solution will be generated because Swap (intra-route) (section 2.7) operator is used.

2.4 Initial Population

A population of potential solutions, represented as individuals or chromosomes, is created. Each individual represents a possible solution to the CVRP. In this work 50% of the population size are generated using **Nearest Neighbor** (NN) algorithm which is one of the most known heuristic

algorithms to solve optimization problems as in CVRP. The other 50% of the population size are generated randomly to improve the diversification of the algorithm.

In NN algorithm, 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 Parents Selection

GA need a selection method to determine which solutions will be selected as parents to participate in reproduction process. Roulette Wheel Selection is used in this work, the following are the steps to perform this selection method:

1. Calculate the fitness of each individual in the population using a fitness function. Higher fitness values indicate better solutions.

2. Normalize the fitness values to convert them into probabilities. This is typically done by dividing each individual's fitness by the sum of all fitness values in the population. The normalized fitness values represent the probability of selection.

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

3. Generate a random number between 0 and 1. This random number determines the location where the roulette wheel stops spinning.

Selection:

4. Select the individual whose slice contains the randomly generated number. Individuals with larger fitness values have larger slices, and therefore, a higher chance of being selected.

5. Repeat the process to select the desired number of individuals for the next generation. However, it is possible for the same individual to be selected more than once, especially if it has a higher fitness.

2.6 Crossover Operator

For permutation representation several alternative crossover operators exist such as Order Crossover and Partially Mapped Crossover. The crossover operator used in the GA is Order Crossover. The used Order Crossover method steps are as follow:

1. Two crossover points are randomly selected from parent 1
2. The part between the two points will be copied from parent 1 to the offspring 1.
3. From parent 2, start at the **first** node and pick the elements that are not already selected from

parent 1.

4. Two crossover points are randomly selected from parent 2.

5. The part between the two points will be copied from parent 1 to the offspring 2.

6. From parent 1, start at the **first** node and pick the elements that are not already selected from parent 1.

Example:

The cross points are 5 and 25 for offspring1:

```
parent1 [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]
parent2 [30, 15, 29, 14, 23, 19, 4, 22, 13, 2, 11, 16, 31, 9, 28, 12, 5,
27, 25, 18, 10, 21, 1, 20, 26, 6, 24, 7, 3, 8, 17]
Offspring1 [19, 4, 22, 11, 31, 9, 15, 10, 13, 7, 16, 26, 30, 12, 28, 23,
3, 2, 6, 14, 24, 29, 5, 20, 27, 25, 18, 21, 1, 8, 17]
```

The cross points are 0 and 13 for offspring2:

```
parent1 [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]
parent2 [30, 15, 29, 14, 23, 19, 4, 22, 13, 2, 11, 16, 31, 9, 28, 12, 5,
27, 25, 18, 10, 21, 1, 20, 26, 6, 24, 7, 3, 8, 17]
Offspring2 [30, 15, 29, 14, 23, 19, 4, 22, 13, 2, 11, 16, 31, 8, 18, 9,
10, 7, 26, 12, 28, 3, 6, 24, 5, 20, 27, 1, 21, 17, 25]
```

2.7 Mutation

Swap (intra-route) is used in which swap two randomly chosen nodes from one randomly chosen route. Following is an example where the selected route is 3 and the 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
[[4, 11, 8, 18, 22, 9, 15, 10], [13, 7, 16, 26, 30, 12], [28, 23, 3, 2,
6, 14, 24], [29, 5, 1, 27, 20, 21, 31], [17, 19, 25]]
```

2.8 Replacement

Replacement is concerned with the survivor selection from both the parent and the offspring populations. The replacement strategy used in this work is **replacing a ~50% of the old**

population by the new offsprings.

2.9 Diversification and Intensification strategies

Diversification and intensification are two key aspects in the context of genetic algorithms (GAs) and optimization algorithms in general. They play a crucial role in balancing exploration and exploitation within the search space to efficiently find optimal solutions.

In this work, starting with a diverse initial population helps set a foundation for exploration in different regions. Moreover, mutation operators generally generate worse offspring which will increase the diversification of the algorithm. On the other hand, crossover operator most of the time produces children that are at least better than their worse parent, but on average seldom surpass the fitness of the better parent [6]. For that, crossover will increase the intensification of the algorithm.

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. The search method should be able to find optimal or near optimal routes efficiently in short time. Therefore, various metaheuristic algorithms are used. GA is one of the commonly used P-metaheuristics, it is able to obtain good solutions in reasonable time, thus it is attractive for solving variety of optimization problems. Also, we aim to compare the obtained results of GA and Tabu Search (TS).

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 1: Values of parameters tuning.

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

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* [7]. 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 2: 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

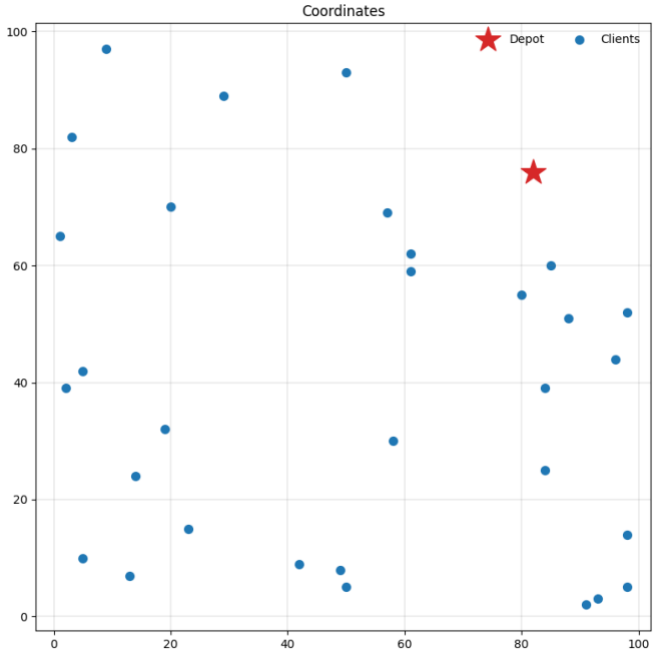

Figure 2: A-n32-k5 Instance.

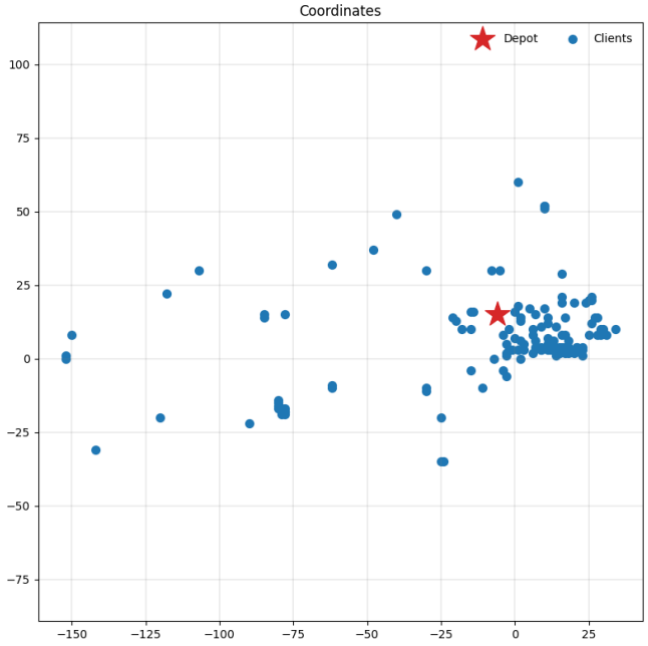

Figure 3: F-n135-k17 Instance.

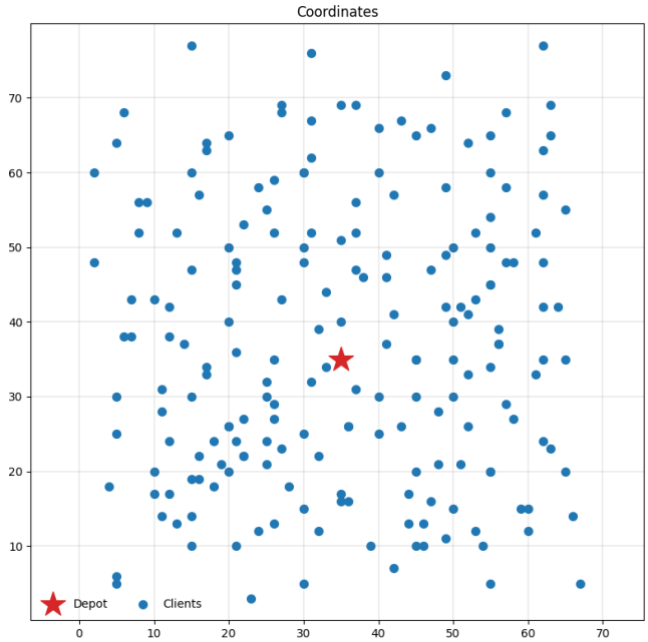

Figure 4: M-n200-k17 Instance.

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 [7] 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 GA, 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.

To evaluate the obtained results from GA, a comparison is done with the achieved results in assignment 2 using TS. Table 3 shows the obtained result of for each instance during the 7 runs for both algorithms. 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 of GA is reasonable, as what was maintained with TS. Table 4 shows a comparison between the obtained solution and BKS for CVRP. The objective of the best obtained solution by GA are the following, for the smallest applied instance, A-n32-k5, the objective is 849.30 which is 8.3% worse than BKS. For instance F-n135-k7, the best obtained solution objective is 1301.39 which is 12% worse than BKS. For the largest applied instance M-n200-k17, the best obtained solution objective is 1542.37 which is 21% worse than BKS. As shown in Figure 5 the GA efficiency decreases with the increase of the instance size, and this is justifiable since the problem complexity increases with its size. Although a simple GA is used, the obtained solution is considered acceptable. In comparison with TS, for all instances the best obtained results and their averages are found by TS. Therefore, TS outperforms GA in terms of solution quality. Noting that GA is better in the worse obtained result for the instance F-n135-k7.

Computational time in seconds is shown in Table 5, the average computational time of GA increases with increase of the instance size, the instance F-n135-k7 computational time is the longest since the number of vehicles is small compared to the number of nodes, thus the length of each route is long compared with other two instances, and it needs more computational time. In general, computational time of GA is lower compared to TS, therefore, GA outperforms TS in this evaluation criteria.

Table 6 shows the number of objective function evaluation, the table shows constant number, and this is expected since GA needs to perform evaluations to the objective function to the whole population, to be able to select parents, and to evaluate all individuals to select the best one at each generation. Compared to TS, GA may be considered more stable in objective function evaluation, TS is better for the best case of smallest instance (A-n32-k5) while GA is better for all other cases.

Table 3: Tabu search and Genetic Algorithm solutions quality.

	Instance					
	Tabu search			**Genetic Algorithm**		
Run #	**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

Bold indicate best result

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

Instance	BKS	Tabu search		Best Diff (%)	Avg. Diff (%)	Genetic Algorithm		Best Diff (%)	Avg. Diff (%)
		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

Bold indicate best result

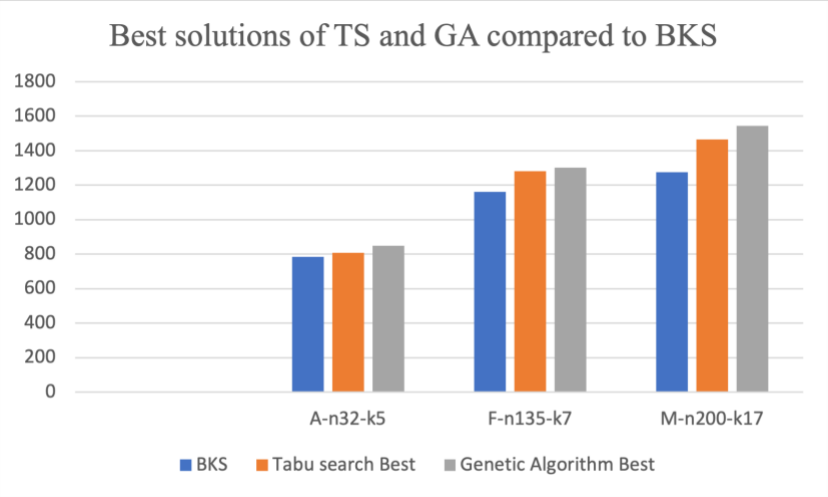

Figure 5: Best solutions of TS compared to BKS.

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

	Tabu Search			Genetic Algorithm		
Run #	**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**

Bold indicate best result

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

	Instance					
	Tabu search			Genetic Algorithm		
Run #	**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**

Bold indicate best result

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:

```
[[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]]
```

Routes of best solution for F-n135-k7:

```
[[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, 8, 7, 11, 12, 10, 2, 42, 41, 3, 40, 44, 43, 39, 38, 96, 97, 99,
36, 35, 37, 95, 45, 94, 93, 29],
[80, 33, 68, 69, 70, 67, 133, 66, 71, 17, 18, 118, 46],
[83, 85, 84, 86, 87, 89, 90, 16, 13, 15, 14, 88, 92, 28, 30, 31, 59, 60, 61,
54, 55, 58, 27],
[120, 109, 108, 107, 106, 114, 115],
[111, 125, 112, 126, 127, 128, 129, 113, 124, 123, 122, 110, 121, 81],
[82, 20, 19, 130, 65, 119, 117, 131, 116, 132]
```

Routes of best solution for M-n200-k17:

```
[[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, 133, 22, 75, 197, 72, 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, 32],
[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, 170],
[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]]
```

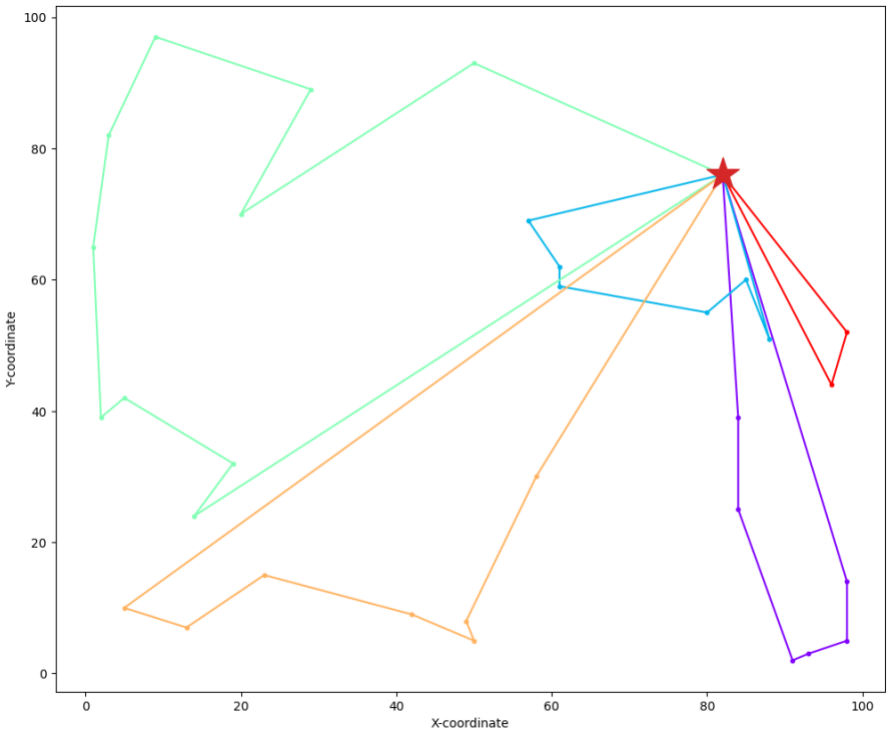

Figure 6: Best solution for instance A-n32-k5.

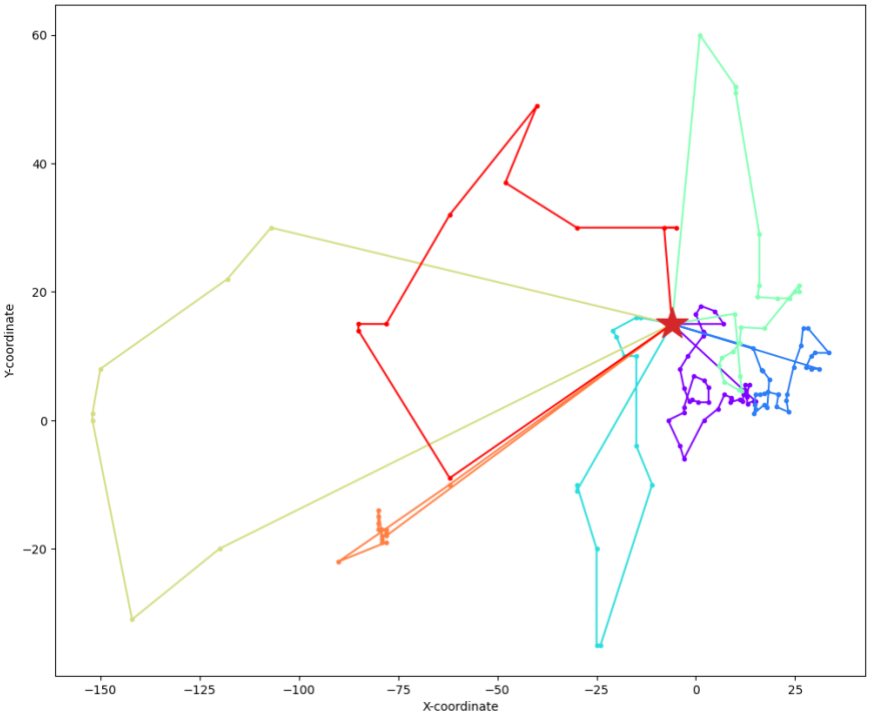

Figure 7: Best solution for instance F-n135-k7.

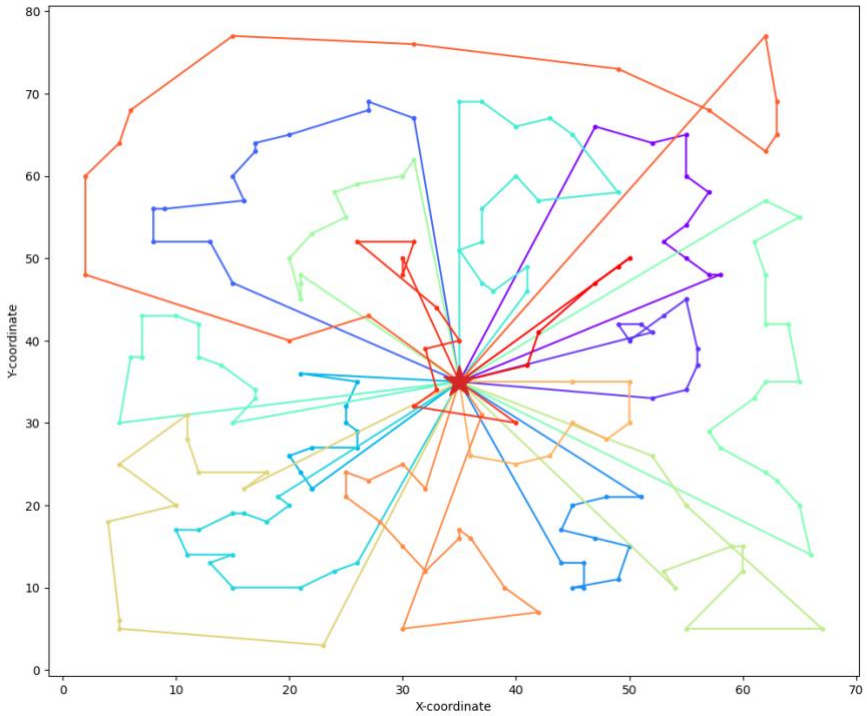

Figure 8: Best solution for instance M-n200-k17.

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] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.

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

[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.

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