Faculty of Engineering and Technology

Electrical and Computer Engineering Department

ARTIFICIAL INTELLIGENCE– ENCS3340

……

PROJECT#1 REPORT :Delivery optimizer system

……

Instructor: Dr.yazan farha   Prepared by: Danah AbuRayya -1210195

date : 30/4/2025

Abstract

This project introduces a delivery optimization system designed to reduce total travel distance while considering vehicle capacity and package priority. Users can input the number of vehicles, their capacities, and package details. The system assigns packages using a priority-based heuristic and optimizes delivery routes using Simulated Annealing and Genetic Algorithm. Visualization tools show route improvements, demonstrating the system's effectiveness in enhancing delivery efficiency.

Problem Formulation

The problem addressed in this project is the optimization of package delivery operations for a local delivery shop. The goal is to efficiently assign packages to a limited fleet of vehicles while minimizing the total distance traveled by all vehicles. Each package has a defined destination, weight, and priority level, and each vehicle has a fixed maximum carrying capacity.

Objective

The primary objective is to minimize the **total delivery distance** traveled by all vehicles while ensuring that:

- All packages are successfully delivered.
- No vehicle exceeds its maximum weight capacity.
- Higher priority packages (priority = 1 is highest) are scheduled for delivery before lower-priority ones.

Constraints

1. **Vehicle Capacity Limits**
   Each vehicle has a maximum weight it can carry. The total weight of assigned packages must not exceed this limit.
2. **Priority-Based Delivery**
   Packages are prioritized numerically, with lower values representing higher importance. The system gives preference to delivering high-priority packages before others.
3. **All Packages Must Be Assigned**
   Every package in the input dataset must be delivered by one of the available vehicles. The algorithm must ensure full coverage.
4. **Distance Calculation**
   The delivery shop is assumed to be located at the coordinate (0, 0). The Euclidean distance is used to compute travel distances between the shop and package destinations, and between consecutive deliveries.

Heuristics Used

To effectively solve the delivery optimization problem under real-world constraints, we implemented two well-known metaheuristic approaches: **Simulated Annealing (SA)** and **Genetic Algorithm (GA)**. Both heuristics were adapted to respect capacity limits and priority rules while exploring different assignment and routing configurations.

3.1Greedy Heuristic

Before applying any optimization techniques, an initial solution is generated using a greedy heuristic based on package priority and vehicle capacity. This step ensures a feasible starting point for further optimization.

Algorithm Steps:

1. **Sort Packages by Priority**
   Packages are sorted in ascending order of priority (1 is highest), ensuring that high-priority deliveries are considered first.
2. **Assign Packages to Vehicles**
   For each package in sorted order:
   - Iterate through the list of vehicles.
   - Assign the package to the first vehicle that can accommodate it without exceeding its weight capacity.
   - If no vehicle can carry the package, the assignment is skipped (though in practice, input constraints are designed to prevent this).
3. **Route Determination (Per Vehicle)**
   Within each vehicle, the delivery order is set in the sequence of assignment, without initial optimization of the route.

3.2 Simulated Annealing (SA)

Simulated Annealing is inspired by the process of annealing in metallurgy. It starts with an initial solution (a greedy assignment based on priority and vehicle capacity) and iteratively explores neighboring solutions by randomly swapping package assignments or modifying delivery orders.

- At each iteration, a new solution is accepted if it improves the total distance or with a certain probability if it worsens the distance — controlled by a temperature parameter.
- The temperature gradually decreases, reducing the acceptance of worse solutions over time.
- This heuristic helps escape local minima and explore a wide range of possible delivery routes.

Adaptations made:

- Neighbor generation respects vehicle capacity limits.
- Only feasible moves (those that don't violate capacity) are considered.
- High-priority packages remain prioritized in most neighbors.

3.3 Genetic Algorithm (GA)

The Genetic Algorithm is a population-based search inspired by biological evolution. A population of possible solutions is maintained and evolved over several generations using crossover, mutation, and selection operations.

- Each individual (solution) encodes a specific assignment of packages to vehicles and a delivery sequence.
- The fitness function evaluates individuals based on total delivery distance.
- Crossover combines assignments from two parents while preserving feasibility.
- Mutation introduces small random changes to promote diversity and explore new areas of the search space.

Adaptations made:

- Crossover and mutation operations are designed to maintain capacity and ensure complete assignment.
- Elitism is used to preserve the best solutions between generations.

These heuristics offer complementary strengths: SA performs well in local refinement, while GA effectively explores a broader search space through population diversity.

3.4Parameter Tuning Effects

a) Simulated Annealing

- **Key Parameter:** Cooling Rate (α)
- **Effect:**
  - **Higher values (e.g., 0.99):** Slower cooling, better exploration, more accurate results, but slower convergence.
  - **Lower values (e.g., 0.85):** Faster convergence, less optimal.
- **Best Balance Found:** α = 0.98

b) Genetic Algorithm

- **Key Parameter:** Population Size
- **Effect:**

    - **Smaller (e.g., 10):** Fast execution, poor diversity.
    - **Larger (e.g., 100):** Slower but more diverse, better solutions.
- **Best Balance Found:** 50–60 for 100 packages.

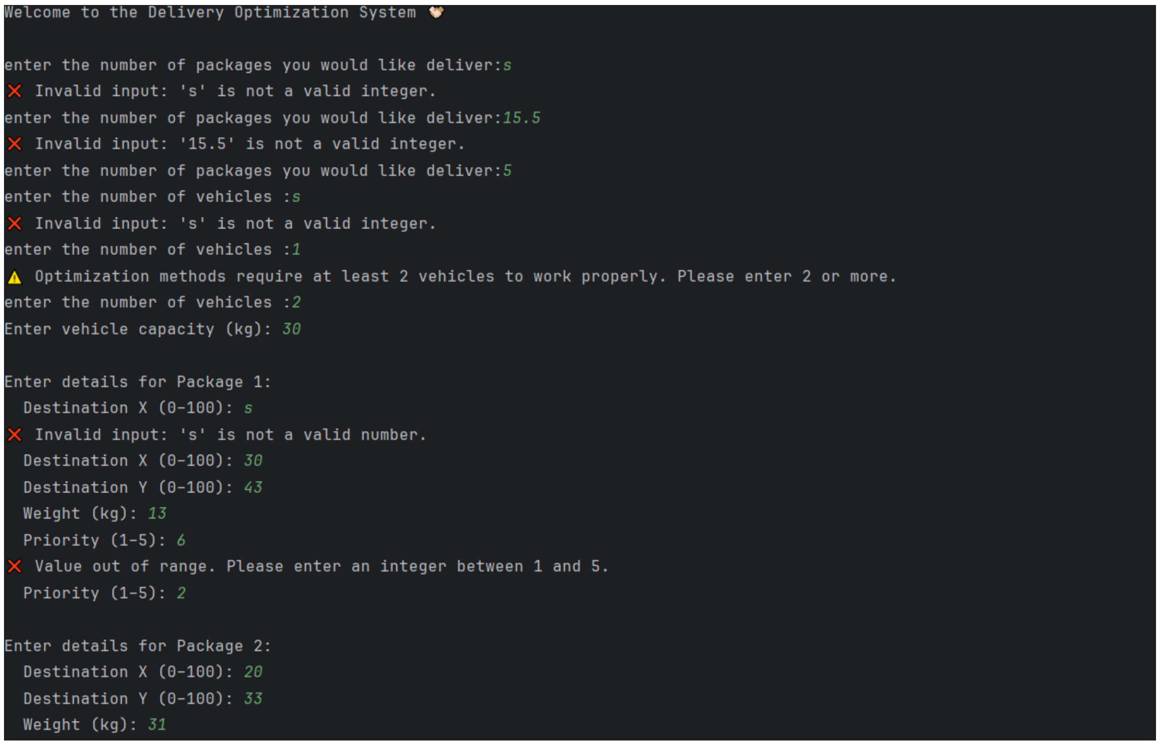
Results & Evaluation Section

4.1Input Validation

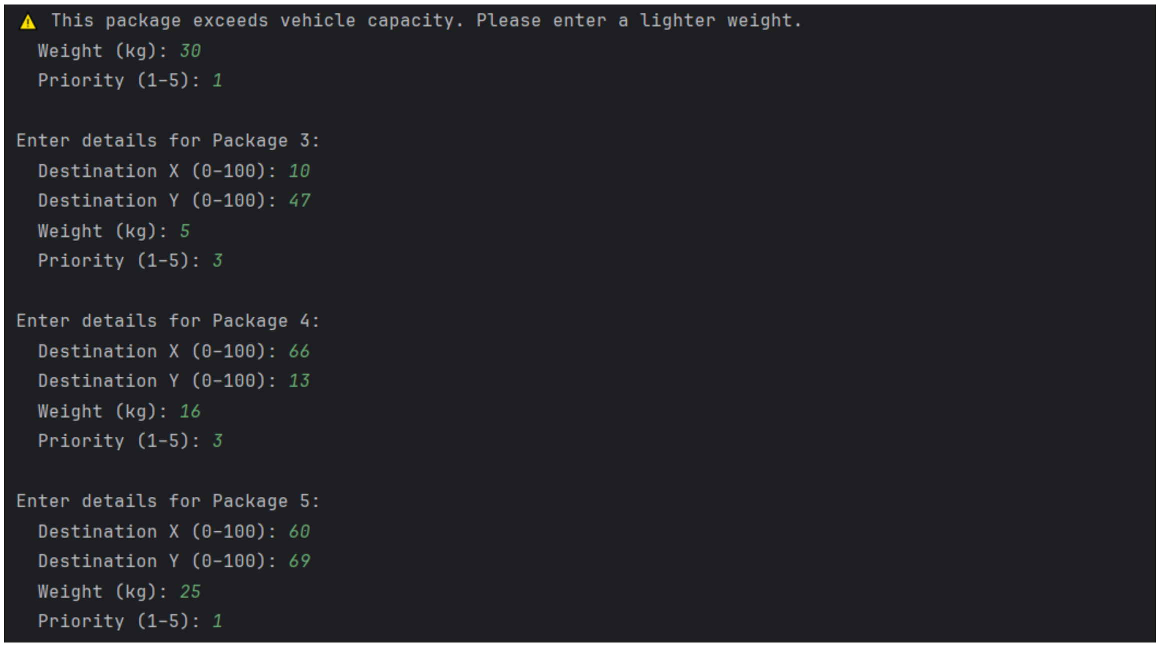
his section of the **Delivery Optimization System** handles comprehensive user input validation and structured package data entry. It ensures only correct data types and valid values are accepted before proceeding with optimization.

The system checks both **data type** and **value range** for each input:

- **Number of packages / vehicles**: Must be integers. Vehicles must be ≥2.
- **Vehicle capacity**: Must be a float within a valid positive range.
- **Package details**:
    - **Coordinates (X, Y)**: Float values between 0–100.
    - **Weight**: Float, must not exceed vehicle capacity.
    - **Priority**: Integer between 1–5.



```
Welcome to the Delivery Optimization System 👋

enter the number of packages you would like deliver:s
❌ Invalid input: 's' is not a valid integer.
enter the number of packages you would like deliver:15.5
❌ Invalid input: '15.5' is not a valid integer.
enter the number of packages you would like deliver:5
enter the number of vehicles :s
❌ Invalid input: 's' is not a valid integer.
enter the number of vehicles :1
⚠ Optimization methods require at least 2 vehicles to work properly. Please enter 2 or more.
enter the number of vehicles :2
Enter vehicle capacity (kg): 30

Enter details for Package 1:
  Destination X (0-100): s
❌ Invalid input: 's' is not a valid number.
  Destination X (0-100): 30
  Destination Y (0-100): 43
  Weight (kg): 13
  Priority (1-5): 6
❌ Value out of range. Please enter an integer between 1 and 5.
  Priority (1-5): 2

Enter details for Package 2:
  Destination X (0-100): 20
  Destination Y (0-100): 33
  Weight (kg): 31
```



```
⚠ This package exceeds vehicle capacity. Please enter a lighter weight.
  Weight (kg): 30
  Priority (1-5): 1

Enter details for Package 3:
  Destination X (0-100): 10
  Destination Y (0-100): 47
  Weight (kg): 5
  Priority (1-5): 3

Enter details for Package 4:
  Destination X (0-100): 66
  Destination Y (0-100): 13
  Weight (kg): 16
  Priority (1-5): 3

Enter details for Package 5:
  Destination X (0-100): 60
  Destination Y (0-100): 69
  Weight (kg): 25
  Priority (1-5): 1
```

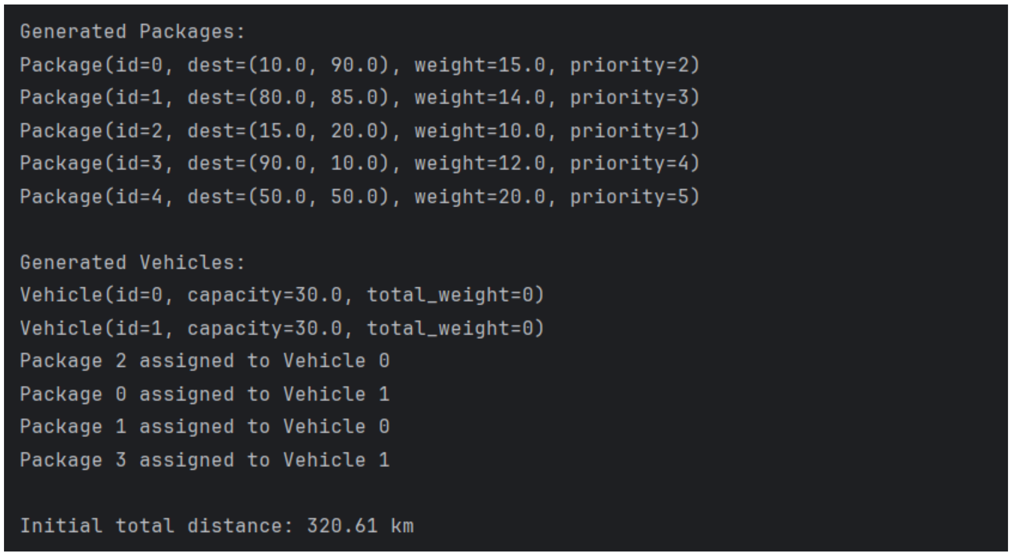
4.2Algorithm testing

For testing the algorithm we consider these values

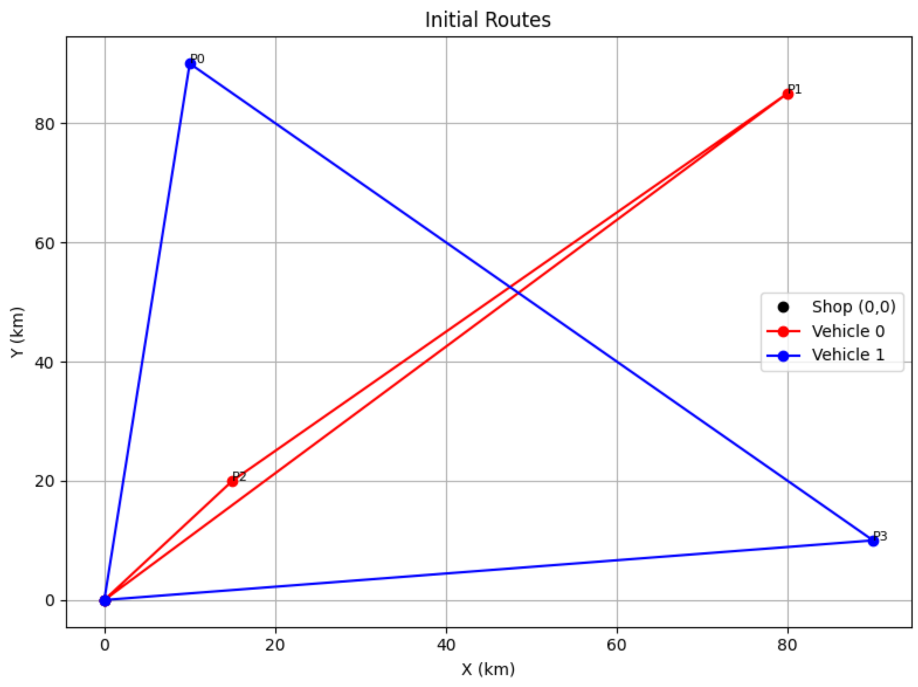
- **Number of packages**: 5
- **Number of vehicles**: 2
- **Vehicle capacity**: 30 kg

ID	Destination (X, Y)	Weight (kg)	Priority
0	(30, 43)	13	2
1	(20, 33)	30	1
2	(10, 47)	5	3
3	(66, 13)	16	3

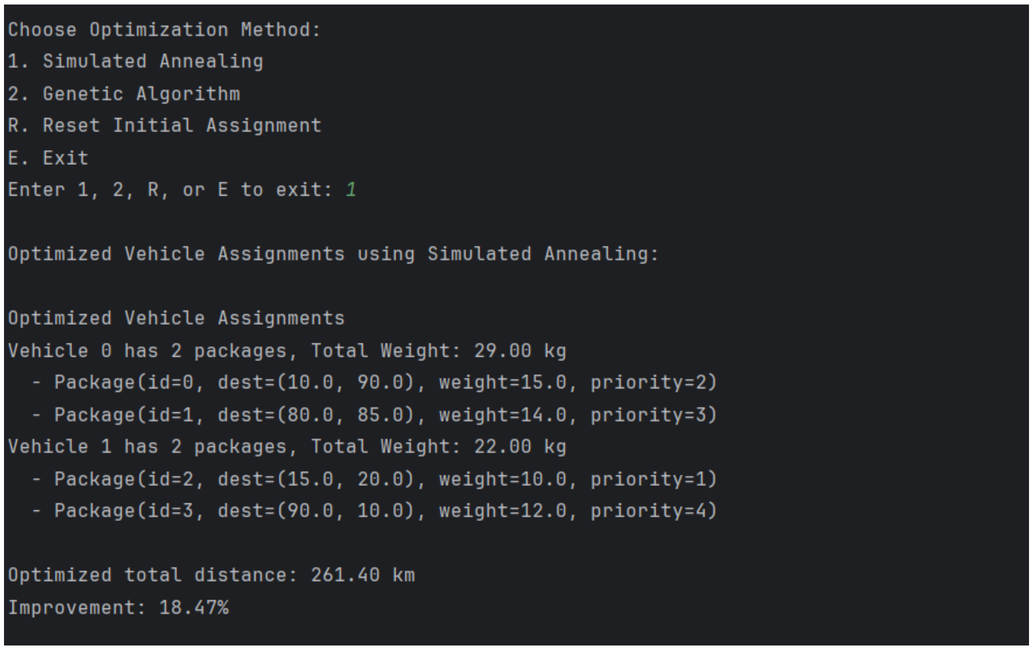
4.2.1Greedy Algorithm



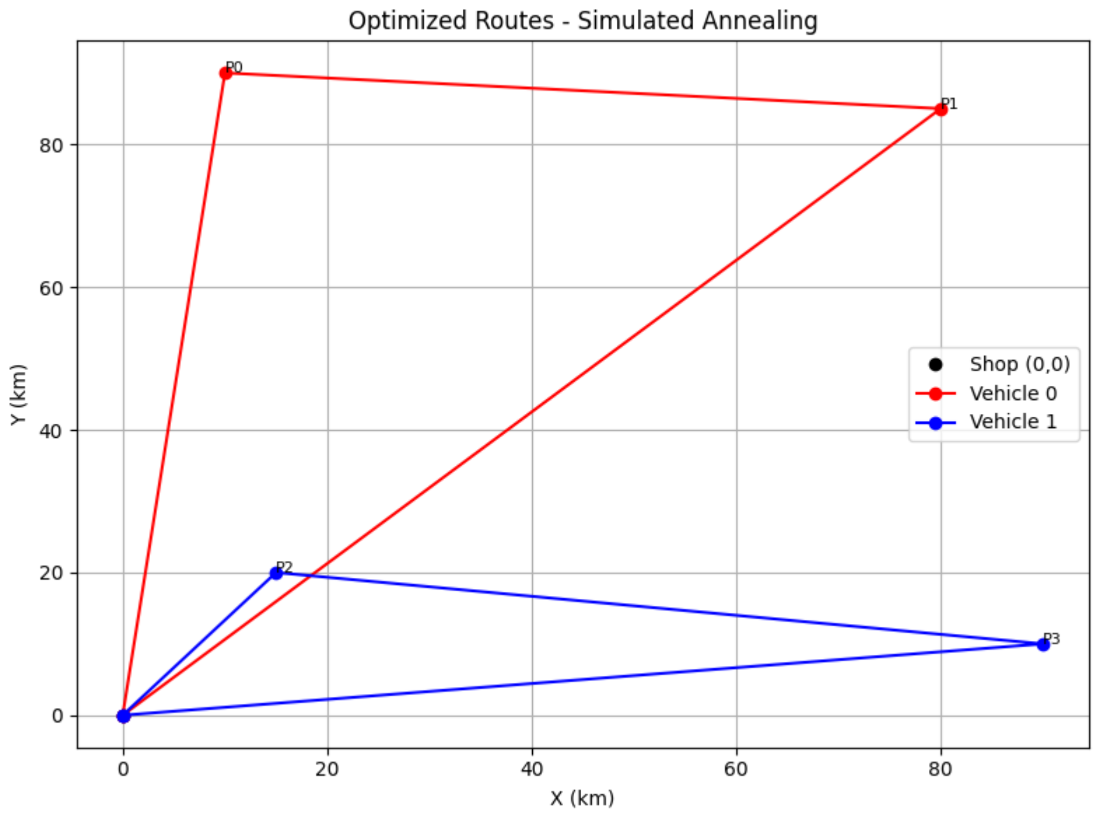
```
Generated Packages:
Package(id=0, dest=(10.0, 90.0), weight=15.0, priority=2)
Package(id=1, dest=(80.0, 85.0), weight=14.0, priority=3)
Package(id=2, dest=(15.0, 20.0), weight=10.0, priority=1)
Package(id=3, dest=(90.0, 10.0), weight=12.0, priority=4)
Package(id=4, dest=(50.0, 50.0), weight=20.0, priority=5)

Generated Vehicles:
Vehicle(id=0, capacity=30.0, total_weight=0)
Vehicle(id=1, capacity=30.0, total_weight=0)
Package 2 assigned to Vehicle 0
Package 0 assigned to Vehicle 1
Package 1 assigned to Vehicle 0
Package 3 assigned to Vehicle 1

Initial total distance: 320.61 km
```



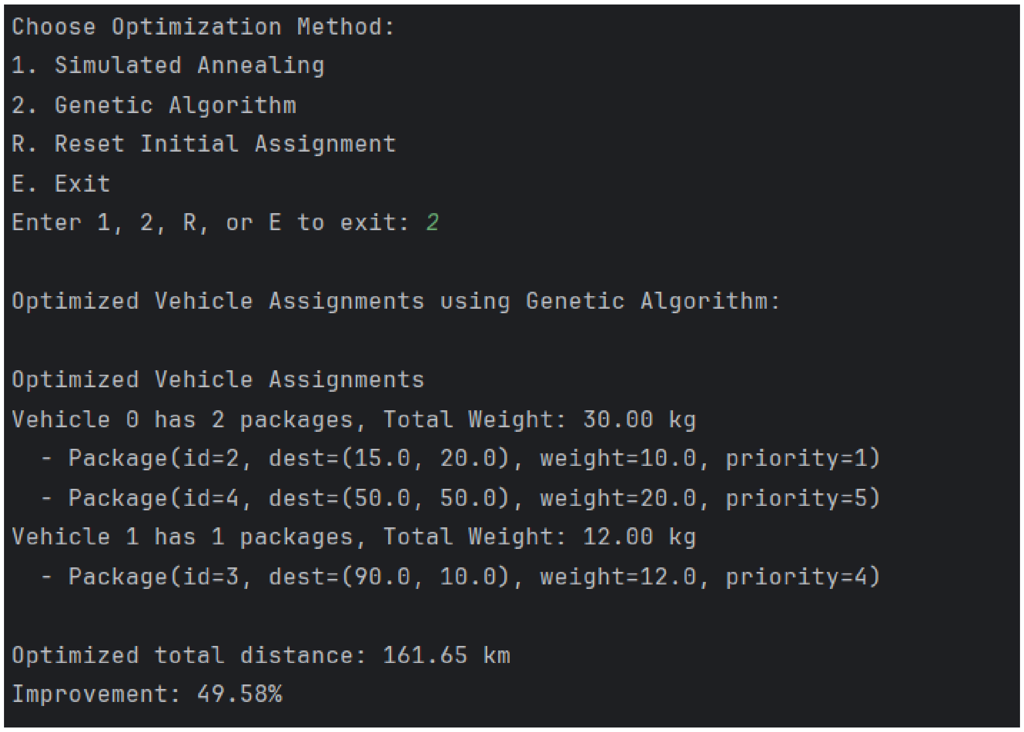
4.2.2Simulated Annealing algorithm



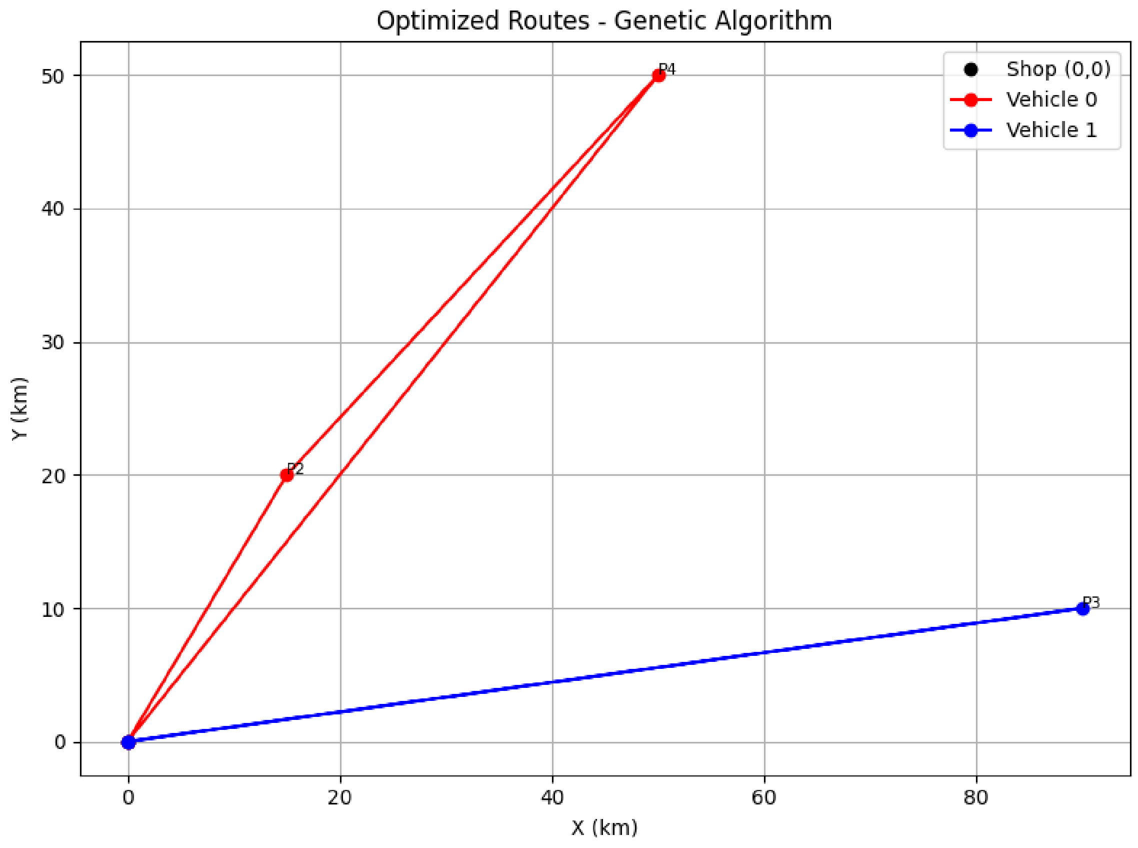
```
Choose Optimization Method:
1. Simulated Annealing
2. Genetic Algorithm
R. Reset Initial Assignment
E. Exit
Enter 1, 2, R, or E to exit: 1

Optimized Vehicle Assignments using Simulated Annealing:

Optimized Vehicle Assignments
Vehicle 0 has 2 packages, Total Weight: 29.00 kg
  - Package(id=0, dest=(10.0, 90.0), weight=15.0, priority=2)
  - Package(id=1, dest=(80.0, 85.0), weight=14.0, priority=3)
Vehicle 1 has 2 packages, Total Weight: 22.00 kg
  - Package(id=2, dest=(15.0, 20.0), weight=10.0, priority=1)
  - Package(id=3, dest=(90.0, 10.0), weight=12.0, priority=4)

Optimized total distance: 261.40 km
Improvement: 18.47%
```



4.2.4 Genetic Algorithm

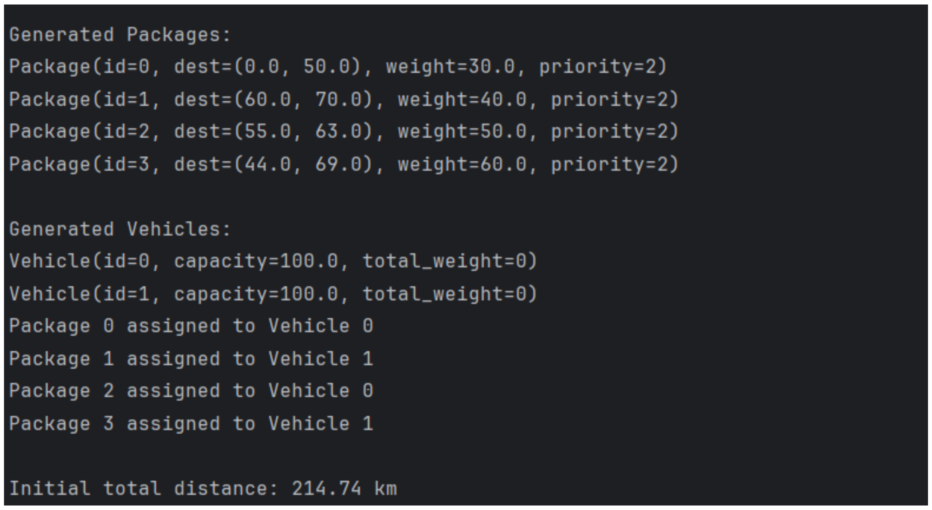


```
Choose Optimization Method:
1. Simulated Annealing
2. Genetic Algorithm
R. Reset Initial Assignment
E. Exit
Enter 1, 2, R, or E to exit: 2

Optimized Vehicle Assignments using Genetic Algorithm:

Optimized Vehicle Assignments
Vehicle 0 has 2 packages, Total Weight: 30.00 kg
  - Package(id=2, dest=(15.0, 20.0), weight=10.0, priority=1)
  - Package(id=4, dest=(50.0, 50.0), weight=20.0, priority=5)
Vehicle 1 has 1 packages, Total Weight: 12.00 kg
  - Package(id=3, dest=(90.0, 10.0), weight=12.0, priority=4)

Optimized total distance: 161.65 km
Improvement: 49.58%
```



Test Case 1: Basic Feasibility Test

- **Objective**: Ensure that the system can assign packages to vehicles without exceeding their capacities.
- **Input**:
    - **Vehicles**: 2 vehicles, each with a capacity of 100 kg.
    - **Packages**: 4 packages with the following weights: 30 kg, 40 kg, 50 kg, and 60 kg.
- **Expected Outcome**:
    - Packages are distributed among vehicles such that no vehicle carries more than 100 kg.
    - All packages are assigned to vehicles.
- **Validation Criteria**:
    - Total weight per vehicle ≤ 100 kg.
    - No unassigned packages remain.

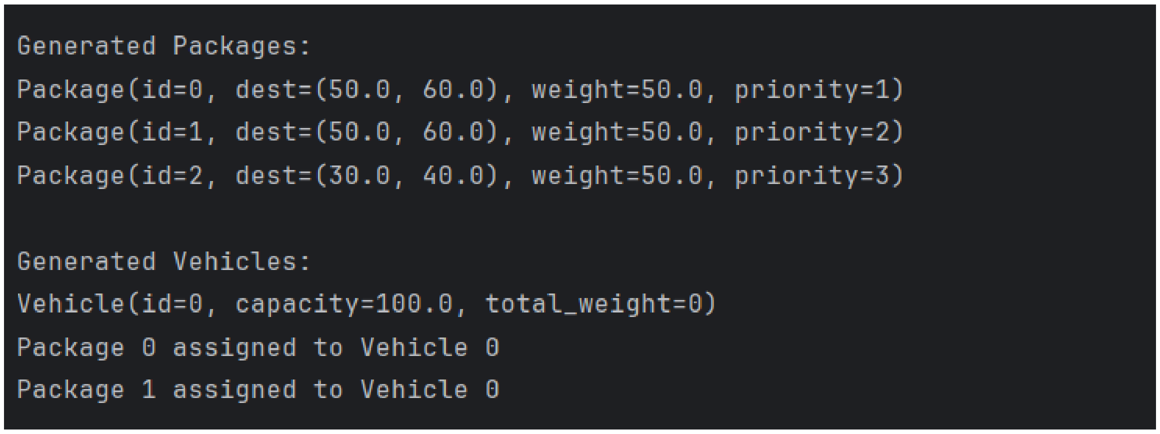


```
Generated Packages:
Package(id=0, dest=(0.0, 50.0), weight=30.0, priority=2)
Package(id=1, dest=(60.0, 70.0), weight=40.0, priority=2)
Package(id=2, dest=(55.0, 63.0), weight=50.0, priority=2)
Package(id=3, dest=(44.0, 69.0), weight=60.0, priority=2)

Generated Vehicles:
Vehicle(id=0, capacity=100.0, total_weight=0)
Vehicle(id=1, capacity=100.0, total_weight=0)
Package 0 assigned to Vehicle 0
Package 1 assigned to Vehicle 1
Package 2 assigned to Vehicle 0
Package 3 assigned to Vehicle 1

Initial total distance: 214.74 km
```

Test Case 2: Priority Handling Test

- **Objective**: Verify that higher-priority packages are delivered before lower-priority ones when possible.
- **Input**:
    - **Vehicles**: 1 vehicle with a capacity of 100 kg.
    - **Packages**:
        - Package A: 50 kg, Priority 1

        - Package B: 50 kg, Priority 2
        - Package C: 50 kg, Priority 3
- **Expected Outcome**:
    - Packages A and B are selected for delivery due to higher priority.
    - Package C is deferred or unassigned due to capacity constraints.

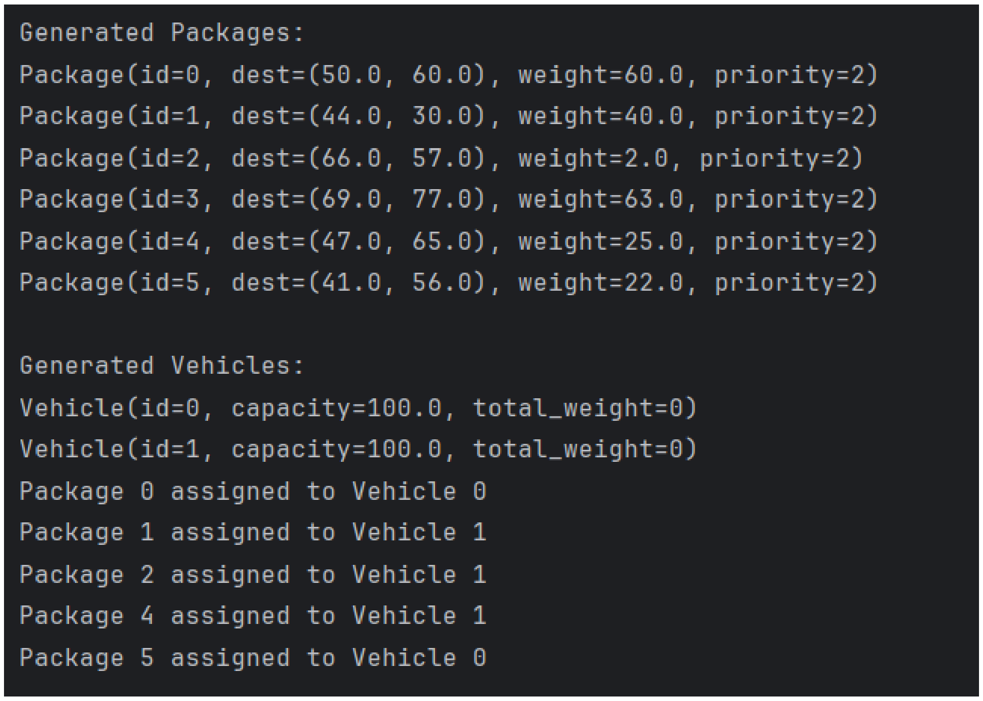


```
Generated Packages:
Package(id=0, dest=(50.0, 60.0), weight=50.0, priority=1)
Package(id=1, dest=(50.0, 60.0), weight=50.0, priority=2)
Package(id=2, dest=(30.0, 40.0), weight=50.0, priority=3)

Generated Vehicles:
Vehicle(id=0, capacity=100.0, total_weight=0)
Package 0 assigned to Vehicle 0
Package 1 assigned to Vehicle 0
```

Test Case 3: Distance Optimization Test

- **Objective**: Ensure that the system minimizes the total distance travelled by all vehicles.
- **Input**:
    - **Vehicles**: 2 vehicles, each with a capacity of 100 kg.
    - **Packages**: 6 packages located at varying distances from the depot.
- **Expected Outcome**:
    - Packages are assigned and routed to minimize the combined distance travelled by both vehicles.

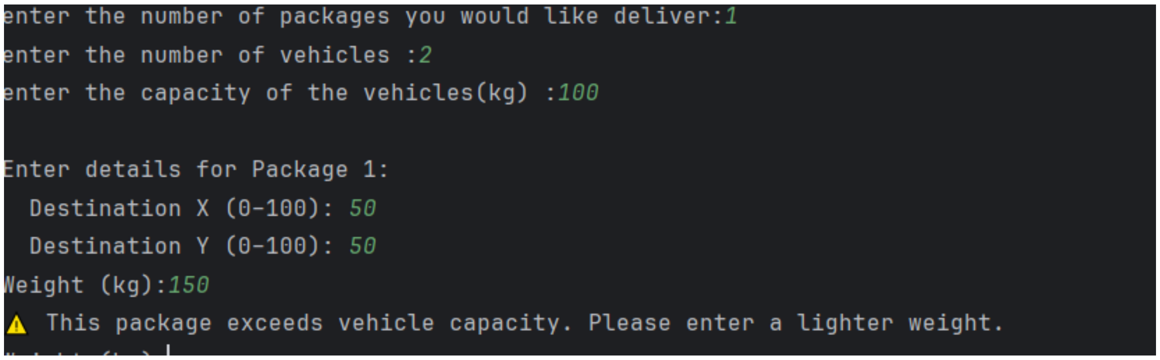
- **Validation Criteria**:Total distance travelled is less than or equal to a predefined threshold based on optimal routing calculations



```
Generated Packages:
Package(id=0, dest=(50.0, 60.0), weight=60.0, priority=2)
Package(id=1, dest=(44.0, 30.0), weight=40.0, priority=2)
Package(id=2, dest=(66.0, 57.0), weight=2.0, priority=2)
Package(id=3, dest=(69.0, 77.0), weight=63.0, priority=2)
Package(id=4, dest=(47.0, 65.0), weight=25.0, priority=2)
Package(id=5, dest=(41.0, 56.0), weight=22.0, priority=2)

Generated Vehicles:
Vehicle(id=0, capacity=100.0, total_weight=0)
Vehicle(id=1, capacity=100.0, total_weight=0)
Package 0 assigned to Vehicle 0
Package 1 assigned to Vehicle 1
Package 2 assigned to Vehicle 1
Package 4 assigned to Vehicle 1
Package 5 assigned to Vehicle 0
```

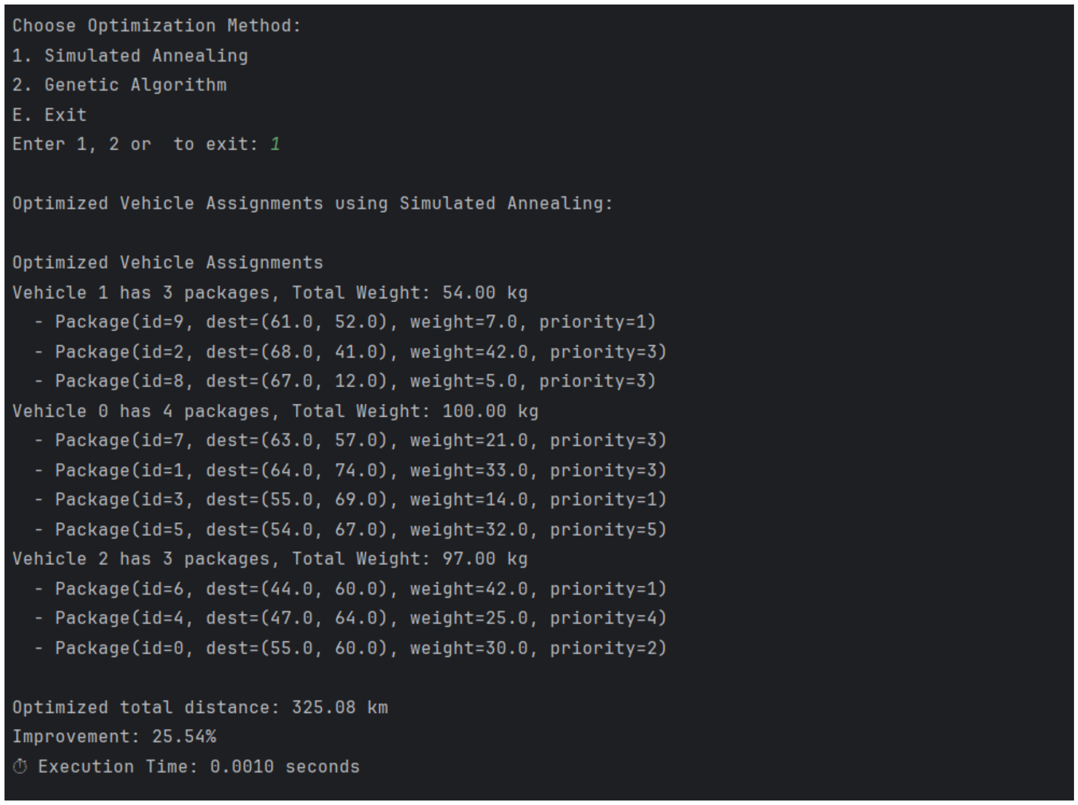
Test Case 4: Edge Case - Overcapacity Package

- **Objective**: Test the system's behaviour when a package exceeds the capacity of all available vehicles.
- **Input**:
  - **Vehicles**: 2 vehicles, each with a capacity of 100 kg.
  - **Packages**: 1 package weighing 150 kg.
- **Expected Outcome**:
  - The system identifies that the package cannot be delivered due to weight constraints.
  - Appropriate handling or notification is provided.
- **Validation Criteria**:
  - Package is not assigned to any vehicle.
  - System logs or reports the issue clearly.

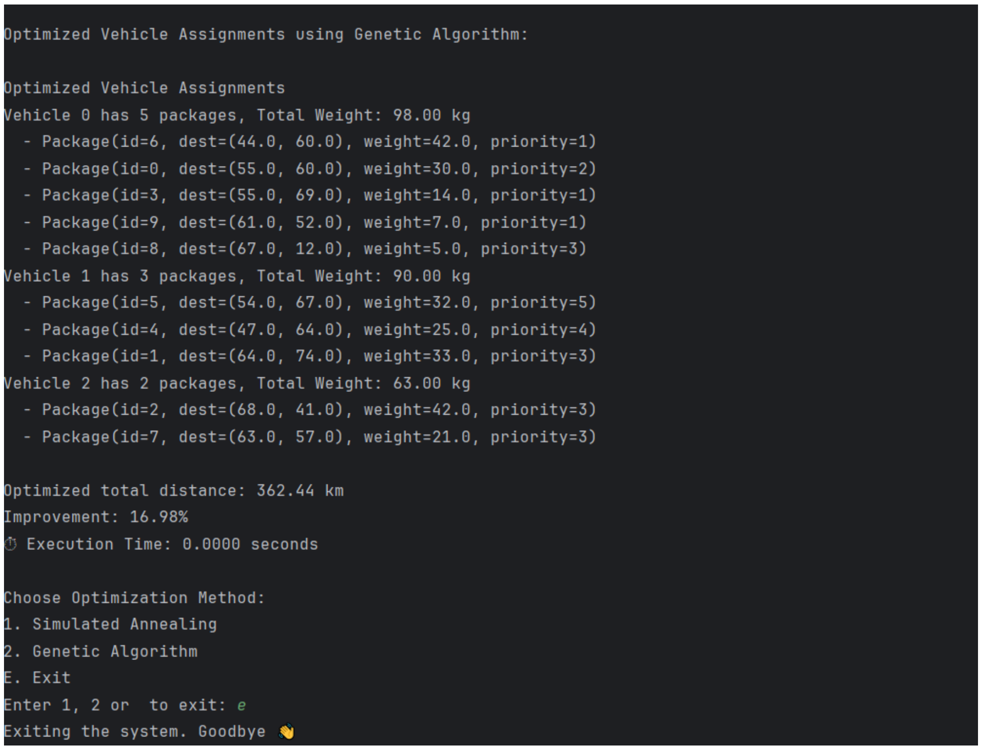

```
enter the number of packages you would like deliver:1
enter the number of vehicles :2
enter the capacity of the vehicles(kg) :100

Enter details for Package 1:
  Destination X (0-100): 50
  Destination Y (0-100): 50
Weight (kg):150
⚠ This package exceeds vehicle capacity. Please enter a lighter weight.
```

Test Case 5: Simulated Annealing vs. Genetic Algorithm Comparison

- **Objective**: Compare the performance of both algorithms in terms of solution quality and computation time.
- **Input**:
  - **Vehicles**: 3 vehicles, each with a capacity of 100 kg.
  - **Packages**: 10 packages with varying weights and priorities.
- **Expected Outcome**:
  - Both algorithms produce valid solutions.
  - Performance metrics (e.g., total distance, computation time) are recorded for comparison.
- **Validation Criteria**:
  - Solutions meet all constraints.
  - Comparative analysis highlights strengths and weaknesses of each algorithm.



```
Choose Optimization Method:
1. Simulated Annealing
2. Genetic Algorithm
E. Exit
Enter 1, 2 or  to exit: 1

Optimized Vehicle Assignments using Simulated Annealing:

Optimized Vehicle Assignments
Vehicle 1 has 3 packages, Total Weight: 54.00 kg
  - Package(id=9, dest=(61.0, 52.0), weight=7.0, priority=1)
  - Package(id=2, dest=(68.0, 41.0), weight=42.0, priority=3)
  - Package(id=8, dest=(67.0, 12.0), weight=5.0, priority=3)
Vehicle 0 has 4 packages, Total Weight: 100.00 kg
  - Package(id=7, dest=(63.0, 57.0), weight=21.0, priority=3)
  - Package(id=1, dest=(64.0, 74.0), weight=33.0, priority=3)
  - Package(id=3, dest=(55.0, 69.0), weight=14.0, priority=1)
  - Package(id=5, dest=(54.0, 67.0), weight=32.0, priority=5)
Vehicle 2 has 3 packages, Total Weight: 97.00 kg
  - Package(id=6, dest=(44.0, 60.0), weight=42.0, priority=1)
  - Package(id=4, dest=(47.0, 64.0), weight=25.0, priority=4)
  - Package(id=0, dest=(55.0, 60.0), weight=30.0, priority=2)

Optimized total distance: 325.08 km
Improvement: 25.54%
⏱ Execution Time: 0.0010 seconds
```



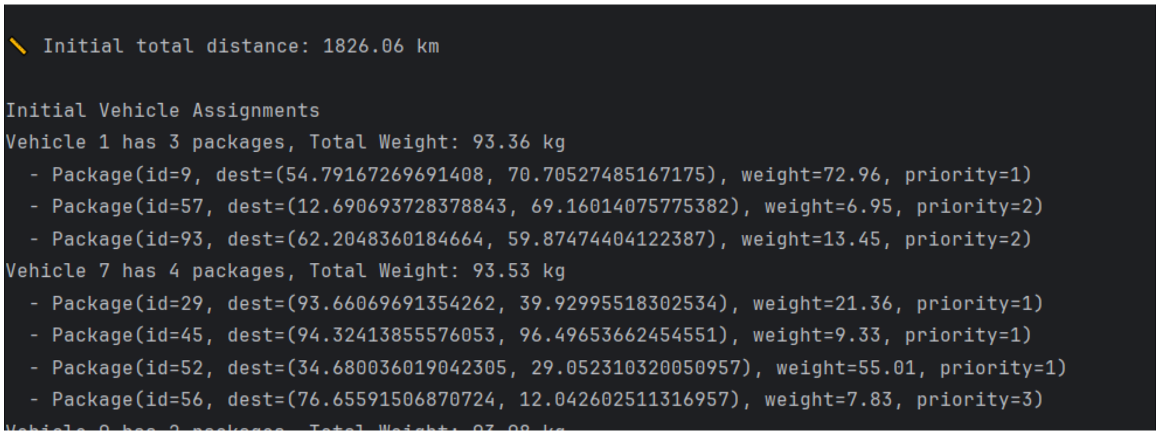
```
Optimized Vehicle Assignments using Genetic Algorithm:

Optimized Vehicle Assignments
Vehicle 0 has 5 packages, Total Weight: 98.00 kg
  - Package(id=6, dest=(44.0, 60.0), weight=42.0, priority=1)
  - Package(id=0, dest=(55.0, 60.0), weight=30.0, priority=2)
  - Package(id=3, dest=(55.0, 69.0), weight=14.0, priority=1)
  - Package(id=9, dest=(61.0, 52.0), weight=7.0, priority=1)
  - Package(id=8, dest=(67.0, 12.0), weight=5.0, priority=3)
Vehicle 1 has 3 packages, Total Weight: 90.00 kg
  - Package(id=5, dest=(54.0, 67.0), weight=32.0, priority=5)
  - Package(id=4, dest=(47.0, 64.0), weight=25.0, priority=4)
  - Package(id=1, dest=(64.0, 74.0), weight=33.0, priority=3)
Vehicle 2 has 2 packages, Total Weight: 63.00 kg
  - Package(id=2, dest=(68.0, 41.0), weight=42.0, priority=3)
  - Package(id=7, dest=(63.0, 57.0), weight=21.0, priority=3)

Optimized total distance: 362.44 km
Improvement: 16.98%
⏱ Execution Time: 0.0000 seconds

Choose Optimization Method:
1. Simulated Annealing
2. Genetic Algorithm
E. Exit
Enter 1, 2 or  to exit: e
Exiting the system. Goodbye 👋
```

Note :the execution time is .0000 for genetic algorithm mean it less than 1ms and I fix the code to measure for more 2 decimal

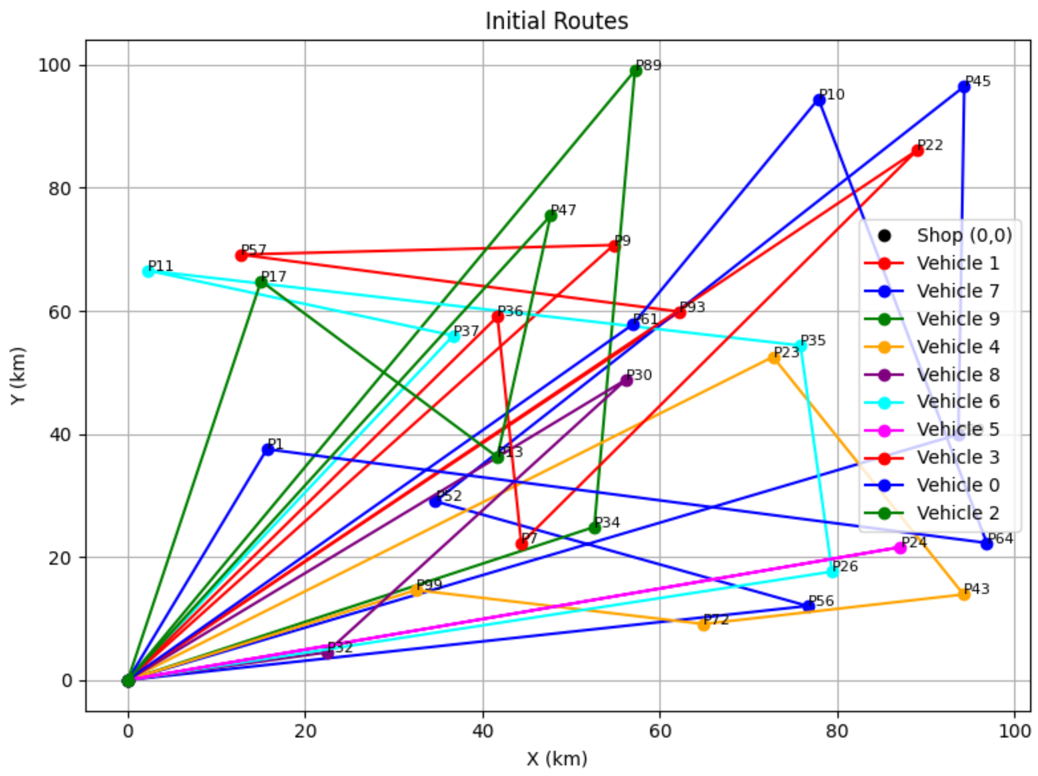
Test Case 6: Scalability Test

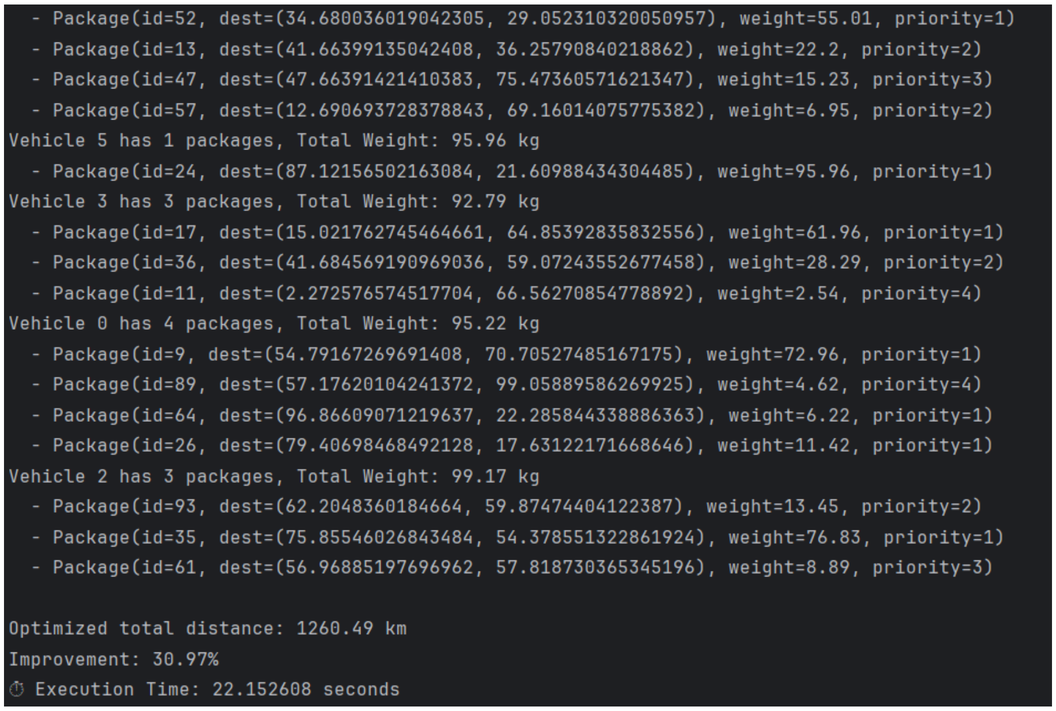
- **Objective**: Assess the system's performance with a large number of packages and vehicles.
- **Input**:
    - **Vehicles**: 10 vehicles, each with a capacity of 100 kg.
    - **Packages**: 100 packages with random weights and priorities.
- **Expected Outcome**:
    - System processes all packages efficiently.
    - All constraints are satisfied.
- **Validation Criteria**:
    - Reasonable computation time (e.g., under 5 minutes).
    - Valid assignments without capacity violations.

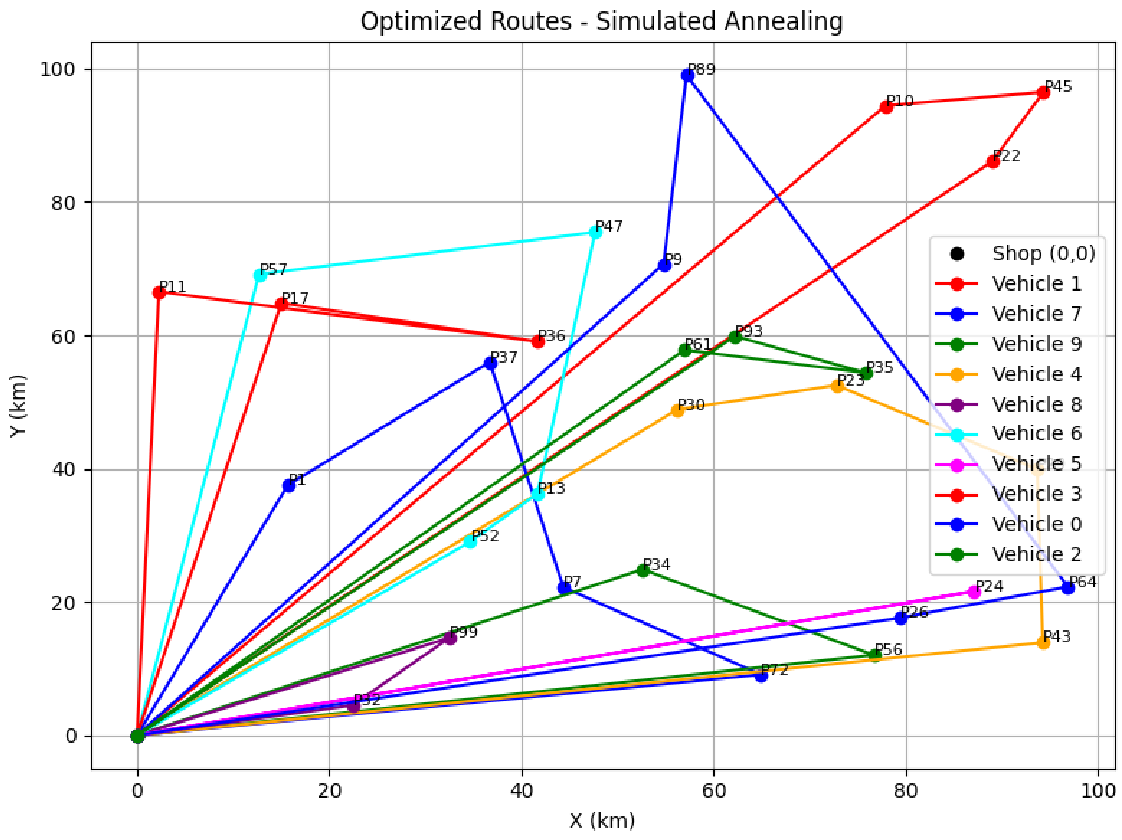
    Note : for this test I have generated I special driver file to randomly make a package to save time and effort and only to test this case

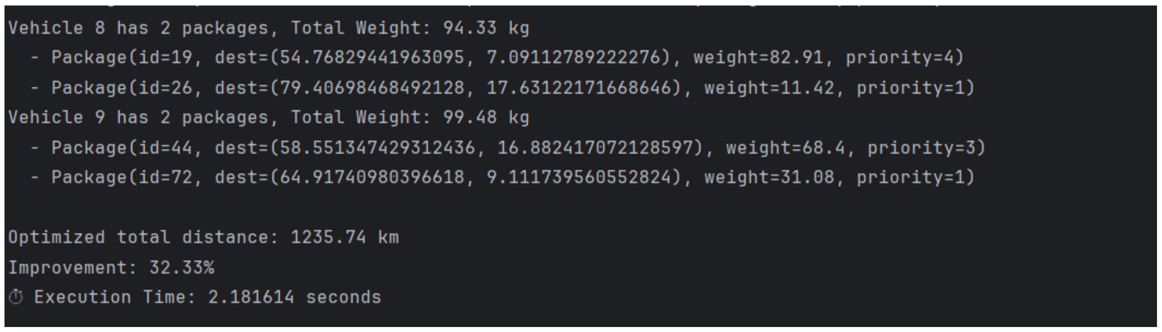


```
📏 Initial total distance: 1826.06 km

Initial Vehicle Assignments
Vehicle 1 has 3 packages, Total Weight: 93.36 kg
  - Package(id=9, dest=(54.79167269691408, 70.70527485167175), weight=72.96, priority=1)
  - Package(id=57, dest=(12.690693728378843, 69.16014075775382), weight=6.95, priority=2)
  - Package(id=93, dest=(62.2048360184664, 59.87474404122387), weight=13.45, priority=2)
Vehicle 7 has 4 packages, Total Weight: 93.53 kg
  - Package(id=29, dest=(93.66069691354262, 39.92995518302534), weight=21.36, priority=1)
  - Package(id=45, dest=(94.32413855576053, 96.49653662454551), weight=9.33, priority=1)
  - Package(id=52, dest=(34.680036019042305, 29.052310320050957), weight=55.01, priority=1)
  - Package(id=56, dest=(76.65591506870724, 12.042602511316957), weight=7.83, priority=3)
```

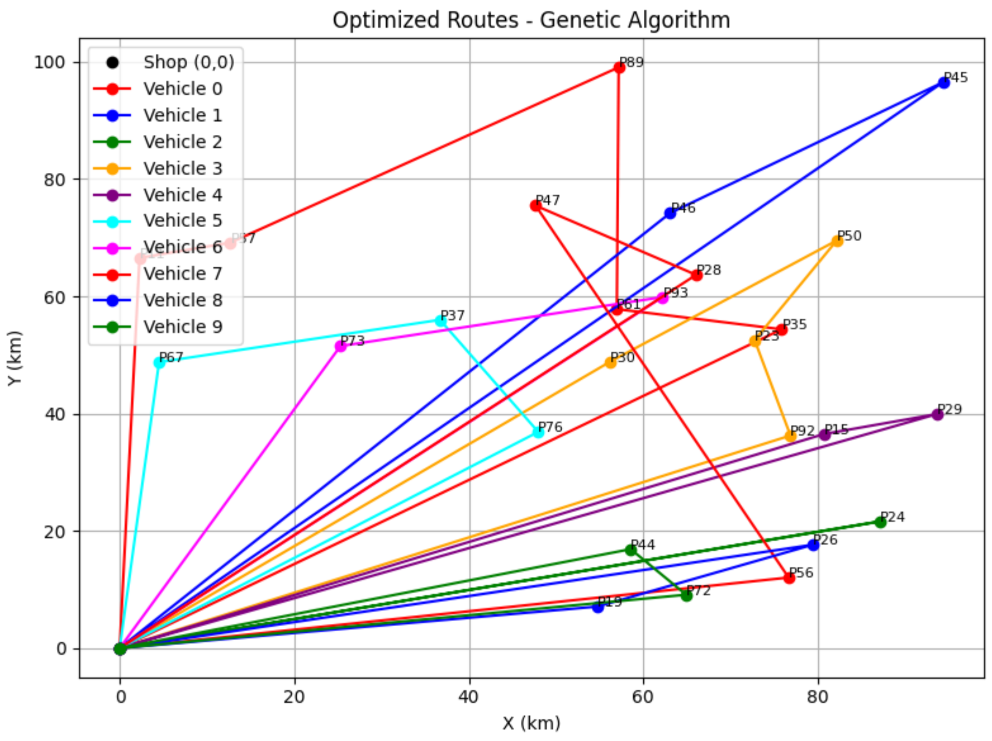



```
  - Package(id=52, dest=(34.680036019042305, 29.052310320050957), weight=55.01, priority=1)
  - Package(id=13, dest=(41.66399135042408, 36.25790840218862), weight=22.2, priority=2)
  - Package(id=47, dest=(47.66391421410383, 75.47360571621347), weight=15.23, priority=3)
  - Package(id=57, dest=(12.690693728378843, 69.16014075775382), weight=6.95, priority=2)
Vehicle 5 has 1 packages, Total Weight: 95.96 kg
  - Package(id=24, dest=(87.12156502163084, 21.60988434304485), weight=95.96, priority=1)
Vehicle 3 has 3 packages, Total Weight: 92.79 kg
  - Package(id=17, dest=(15.021762745464661, 64.85392835832556), weight=61.96, priority=1)
  - Package(id=36, dest=(41.684569190969036, 59.07243552677458), weight=28.29, priority=2)
  - Package(id=11, dest=(2.272576574517704, 66.56270854778892), weight=2.54, priority=4)
Vehicle 0 has 4 packages, Total Weight: 95.22 kg
  - Package(id=9, dest=(54.79167269691408, 70.70527485167175), weight=72.96, priority=1)
  - Package(id=89, dest=(57.17620104241372, 99.05889586269925), weight=4.62, priority=4)
  - Package(id=64, dest=(96.86609071219637, 22.285844338886363), weight=6.22, priority=1)
  - Package(id=26, dest=(79.40698468492128, 17.63122171668646), weight=11.42, priority=1)
Vehicle 2 has 3 packages, Total Weight: 99.17 kg
  - Package(id=93, dest=(62.2048360184664, 59.87474404122387), weight=13.45, priority=2)
  - Package(id=35, dest=(75.85546026843484, 54.378551322861924), weight=76.83, priority=1)
  - Package(id=61, dest=(56.96885197696962, 57.818730365345196), weight=8.89, priority=3)

Optimized total distance: 1260.49 km
Improvement: 30.97%
⏱ Execution Time: 22.152608 seconds
```





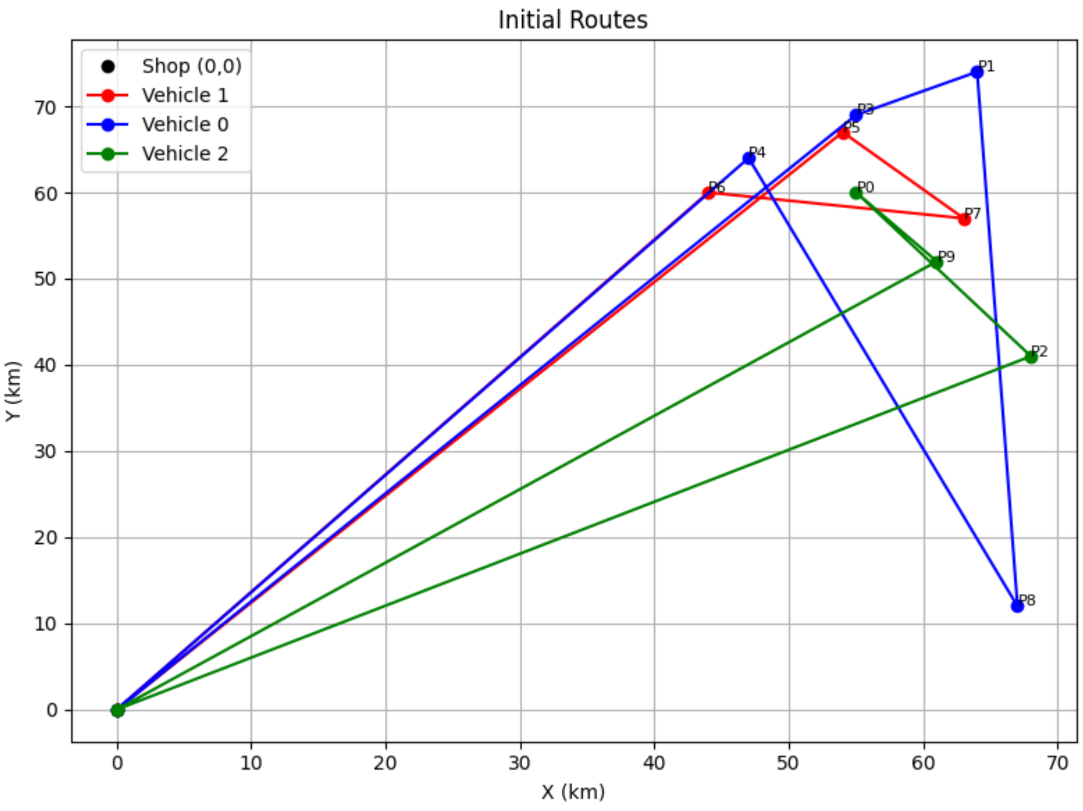
```
Vehicle 8 has 2 packages, Total Weight: 94.33 kg
  - Package(id=19, dest=(54.76829441963095, 7.09112789222276), weight=82.91, priority=4)
  - Package(id=26, dest=(79.40698468492128, 17.63122171668646), weight=11.42, priority=1)
Vehicle 9 has 2 packages, Total Weight: 99.48 kg
  - Package(id=44, dest=(58.551347429312436, 16.882417072128597), weight=68.4, priority=3)
  - Package(id=72, dest=(64.91740980396618, 9.111739560552824), weight=31.08, priority=1)

Optimized total distance: 1235.74 km
Improvement: 32.33%
⏱ Execution Time: 2.181614 seconds
```



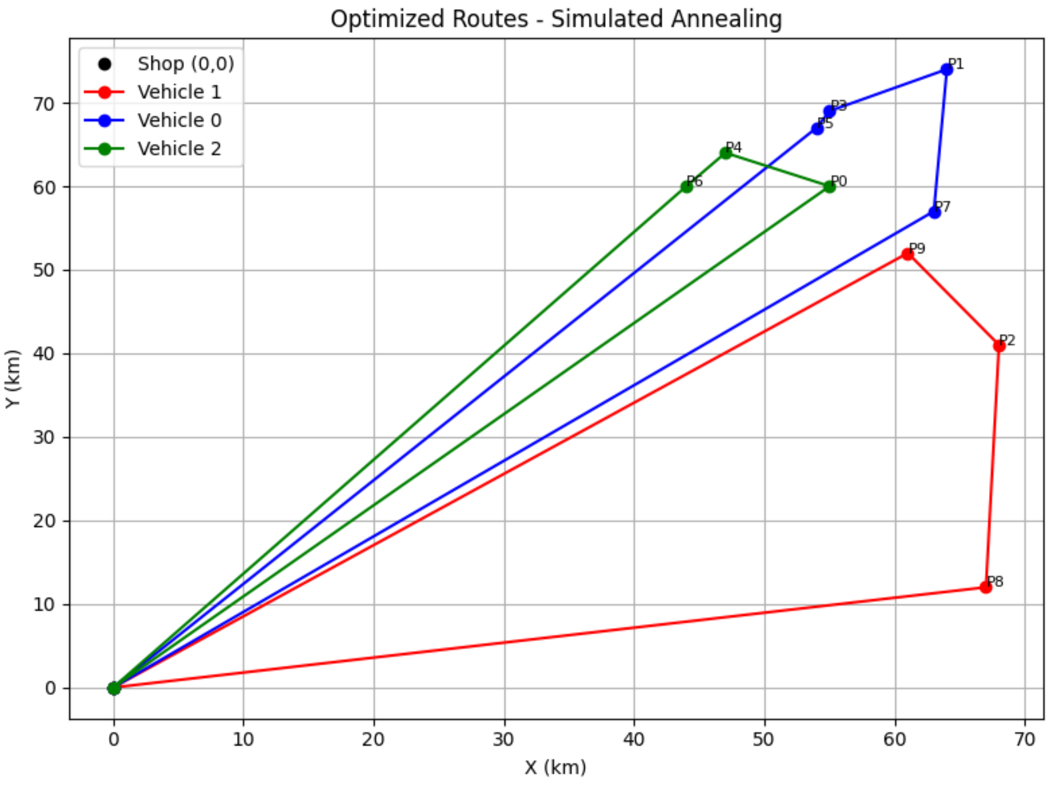
Test Case 7: User Interface Display Test

- **Objective**: Verify that the user interface correctly displays the solution.
- **Input**:
    - Use any of the above scenarios.
- **Expected Outcome**:
    - UI presents vehicle routes, package assignments, and relevant metrics clearly.
- **Validation Criteria**:
    - All information is accurate and user-friendly.
    - No display errors or inconsistencies.

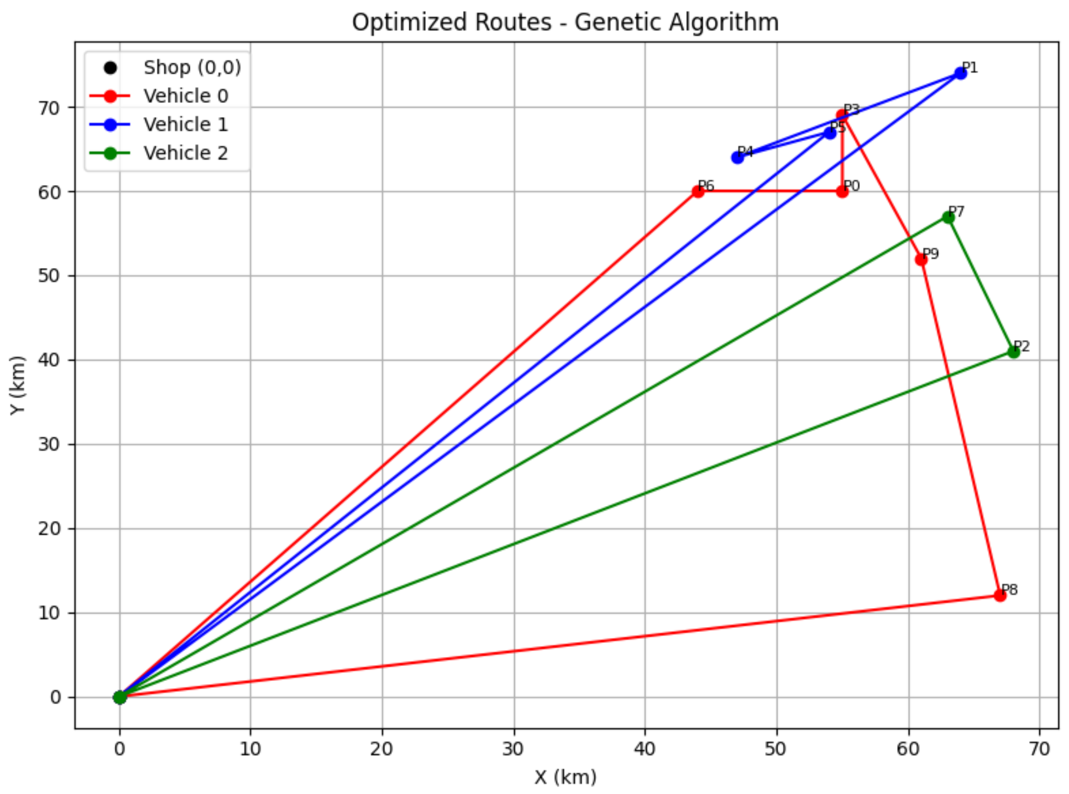
Note I used test #5



Initial Routes
Shop (0,0)
Vehicle 1
Vehicle 0
Vehicle 2
P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
X (km)
Y (km)



Optimized Routes - Simulated Annealing
Shop (0,0)
Vehicle 1
Vehicle 0
Vehicle 2
P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
X (km)
Y (km)



Metric	Simulated Annealing	Genetic Algorithm
Execution Time	⏱ Generally faster on small problems	⏱ Slower due to population handling
Solution Quality	✓ Good but may settle for local optima	✓ Often produces more optimized results
Scalability	⚠ Struggles with large-scale problems	✓ Handles large datasets better
Tuning Complexity	🔧 Simple (mainly temperature control)	🔧 Complex (population size, crossover...)