**Faculty of Engineering & Technology Electrical & Computer Engineering Department**

Artificial intelligent ENCS3340

**Project#1**

# Optimization Strategies for Local Package Delivery Operations

------------------------------------------------------------------------

**Team Members:**

| | | |
|---|---|---|
| Sawsana Atari | 1221320 | section: 2 |
| Sara Al-Souqi | 1221618 | section: 3 |

**Instructor**: Dr. Samah Alaydi

**Date**: 4. May. 2025

## 1. Abstract:

This project explores and compares the performance of Genetic Algorithm (GA) and Simulated Annealing (SA) in solving a complex package delivery optimization problem. The objective is to minimize the total travel distance of a fleet of delivery vehicles while satisfying constraints related to vehicle capacities, package weights, and delivery priorities. Both algorithms were implemented and tested under identical problem settings. The comparison focused on solution quality, convergence speed, constraint handling, and scalability. The results indicate that while SA offers faster convergence for smaller problem sizes, GA consistently provides better global optimization and scalability for larger and more complex delivery scenarios.

Each solution is encoded as an array where the index represents a package ID, and the value indicates the assigned vehicle ID. For example, a solution like `[2, 3, 4, 5]` means that the first package is assigned to Vehicle 2, the second to Vehicle 3, and so on. This representation simplifies crossover, mutation, and reassignment logic while maintaining clarity in vehicle-package allocation.

## 2. Table of content

## 3. Theory

### 3.1 Problem Formulation:

**In this project, we simulate the daily operations of a local package delivery shop. Each day, a set of packages must be delivered to various destinations across a 2D area.**

- **Initial State:**

  The initial state represents the situation where:

  - All packages are located at the delivery shop (origin at coordinates (0, 0)).
  - No packages have been assigned to any delivery vehicle.
  - All vehicles are empty and ready for delivery.

- **State Space:**

  The state space includes all possible assignments of packages to vehicles and the possible delivery routes that vehicles can take. A single state in this space defines:

  - Which packages are assigned to which vehicle.
  - The delivery sequence (route) of each vehicle.
  - The current load of each vehicle.

- **Actions (Operators / Successor Function):**

  From a given state, actions include:

  - Assigning a package to a specific vehicle (if the weight constraint is not violated).
  - Determining or modifying the delivery route of a vehicle.
  - Swapping or reordering delivery locations to reduce the total distance traveled.

  Each action transitions the system to a new state with updated vehicle assignments and/or routes.

- **Goal State:**

A goal state is reached when:

- All packages are assigned to delivery vehicles.
- No vehicle exceeds its capacity.
- A complete delivery route is defined for each vehicle.
- The total distance traveled by all vehicles is minimized (or close to minimum under algorithm constraints).

- **Goal Test:**

To check if a state is a valid goal:

- All packages must be delivered.
- No vehicle exceeds its weight capacity.
- All vehicles have complete and feasible delivery routes.
- The solution minimizes the total delivery distance (objective function).

- **Path and Path Cost:**

- **Path**: The sequence of decisions made to assign packages and plan routes.
- **Path Cost**: The total Euclidean distance traveled by all vehicles. This is the primary metric used to evaluate the efficiency of the solution.

- **Solution:**

A solution is a valid path from the initial state to a goal state that satisfies:

- Weight capacity constraints for all vehicles.
- Priority handling (prefer, but not strictly enforce, early delivery of high-priority packages).
- The least total travel distance, determined by route optimization.

## 3.2 Algorithm-Specific Design and Evaluation:

### 3.2.1 Simulated Annealing (SA):

- **Heuristics Used**

- Packages are initially sorted by priority (1 highest – 5 lowest).
- Objective is to minimize total travel distance from origin (0,0) to all destinations.

- Random neighbor generation swaps packages between vehicles to explore better routes.

- **Constraints Handling**
    - **Package Assignment**:
        - Packages are assigned to vehicles only if the vehicle's load plus the new package's weight does not exceed its capacity.
        - If no vehicle can accommodate a package, it is added to a list of **unassigned packages**.
    - **Neighbor Generation**:
        - During package swaps between vehicles, the swap is accepted only if both vehicles' loads remain within capacity.
        - If the swap violates the capacity constraint, it is rejected, and the vehicle's packages are shuffled.

- **Effects of Parameter Tuning**
    - **Initial Temperature**:
      The initial temperature determines how likely the algorithm is to accept worse solutions at the beginning.
        - **Effect**:
            - **High temperature**: Increases exploration, allowing the algorithm to escape local minima but leads to longer runtime.
            - **Low temperature**: Speeds up convergence but risks getting stuck in suboptimal solutions.
    - **Cooling Rate**:
      The cooling rate controls how quickly the temperature decreases during the annealing process.
        - **Effect**:
            - **High cooling rate (close to 1)**: Slows down cooling, allowing more time for exploration, but may result in longer computation time.
            - **Low cooling rate**: Speeds up convergence but may lead to premature stopping, missing the optimal solution.
    - **Maximum Iterations**:

      This parameter defines how many iterations the algorithm runs before stopping.

        - **Effect**:

- **High iterations**: Allows more time to explore the solution space, improving solution quality but increases computational cost.
- **Low iterations**: May stop prematurely, providing fewer opportunities to find the best solution.

### 3.2.2 Genetic Algorithm (GA):

- **Heuristics Used**

  - **Package Prioritization**:
    Packages are initially sorted by priority (1 = highest, 5 = lowest). Higher-priority packages are considered earlier when assigning and sorting routes.
  - **Random Initialization with Bias**:
    Each chromosome is initialized by randomly assigning packages to vehicles. Each vehicle's route is then sorted by priority to encourage more efficient delivery patterns.
  - **Route Optimization**:
    After assignment, two route sequences are evaluated:
    1. **Priority-sorted route**
    2. **Greedy nearest-neighbor route**
       If the greedy route is at least 20% shorter, it replaces the priority route.
  - **Fitness Function**:
    The algorithm minimizes total distance traveled, penalizes overloaded vehicles (by a factor of 1000), and lightly penalizes skipped/unassigned packages (penalty of 10 per package). Fitness is calculated as:

$$\text{Fitness} = \frac{1}{distace + overload\ penalty + skip\ penalty + 1}$$

- **Constraints Handling**

  - **Vehicle Capacity Constraint**:
    - A package is assigned to a vehicle only if its weight does not exceed the vehicle's remaining capacity.
    - If the initially selected vehicle cannot handle a package, other vehicles are checked in order.

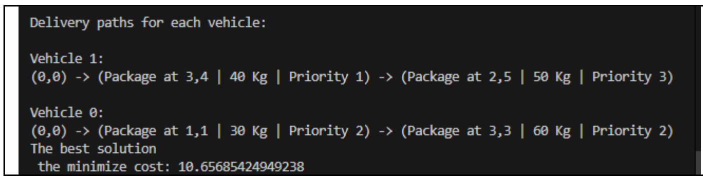
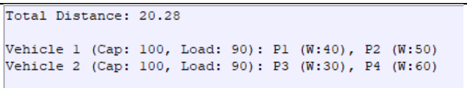
        - If no suitable vehicle is found, the package is marked as skipped.
    - **Neighbor Generation**:
        - **Mutations** randomly reassign a package to a different vehicle (if multiple vehicles exist).
        - **Crossovers** use single-point crossover to combine parent genes into new offspring.
        - After crossover or mutation, the route is rebuilt while maintaining capacity constraints.
        - Packages violating capacity limits after crossover are reassigned or skipped.
    - **Skipped Packages**
        - A `skipped_map` keeps track of which vehicle couldn't take certain packages.
        - These skipped packages contribute a **minor penalty** to fitness, encouraging full assignment.

- **Effects of Parameter Tuning**
    - **Population Size (100)**:
        - **Effect**:
            - Small sizes converged early. Medium sizes (70–85) balanced speed and diversity. Very large sizes slowed without consistent gains. Best population: **70–85** for efficiency and performance.
    - **Mutation Rate (0.05)**:
        - **Effect**:
            - Low rates caused stagnation. Medium (0.04–0.07) gave optimal balance. High rates disrupted good solutions. Best mutation: **0.05–0.07** for stable, diverse, and improving generations.
    - **Number of Generations (500)**:
        - **Effect**:
            - More generations allow deeper search of the solution space but increase computation time.

## 3.3 Test cases & Results:

- **Test Case 1: Basic Feasibility Test**

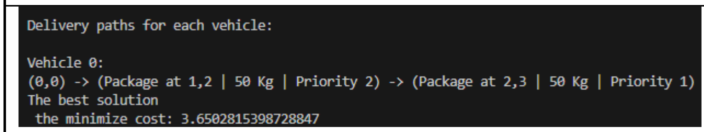
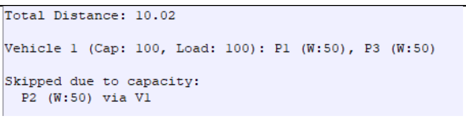
| SA | GA |
|---|---|

|  |  |
|---|---|
|  |  |

```
Delivery paths for each vehicle:

Vehicle 1:
(0,0) -> (Package at 3,4 | 40 Kg | Priority 1) -> (Package at 2,5 | 50 Kg | Priority 3)

Vehicle 0:
(0,0) -> (Package at 1,1 | 30 Kg | Priority 2) -> (Package at 3,3 | 60 Kg | Priority 2)
The best solution
 the minimize cost: 10.65685424949238
```

```
Total Distance: 20.28

Vehicle 1 (Cap: 100, Load: 90): P1 (W:40), P2 (W:50)
Vehicle 2 (Cap: 100, Load: 90): P3 (W:30), P4 (W:60)
```

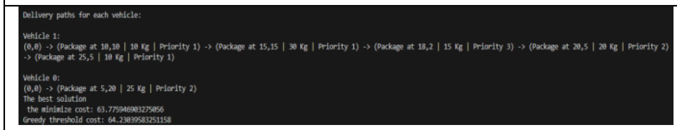
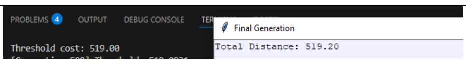
- **Test Case 2: Priority Handling Test**

| SA | GA |
|---|---|
|  |  |

```
Delivery paths for each vehicle:

Vehicle 0:
(0,0) -> (Package at 1,2 | 50 Kg | Priority 2) -> (Package at 2,3 | 50 Kg | Priority 1)
The best solution
 the minimize cost: 3.6502815398728847
```

```
Total Distance: 10.02

Vehicle 1 (Cap: 100, Load: 100): P1 (W:50), P3 (W:50)

Skipped due to capacity:
  P2 (W:50) via V1
```

- **Test Case 3: Distance Optimization Test**

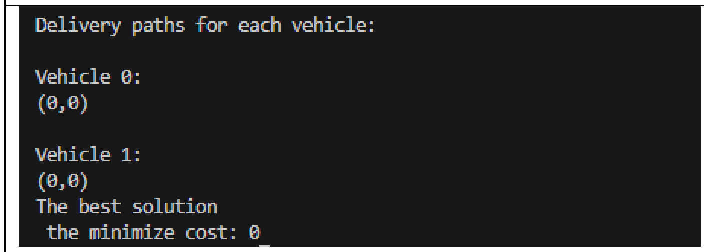
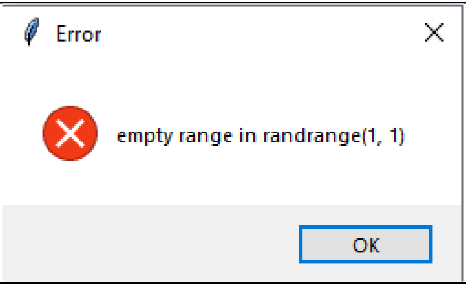
| SA | GA |
|---|---|
|  |  |

```
Delivery paths for each vehicle:

Vehicle 1:
(0,0) -> (Package at 10,10 | 10 Kg | Priority 1) -> (Package at 15,15 | 30 Kg | Priority 1) -> (Package at 18,2 | 15 Kg | Priority 3) -> (Package at 20,5 | 20 Kg | Priority 2)
-> (Package at 25,5 | 10 Kg | Priority 1)

Vehicle 0:
(0,0) -> (Package at 5,20 | 25 Kg | Priority 2)
The best solution
 the minimize cost: 63.775946903275056
Greedy threshold cost: 64.23039583251158
```

```
Threshold cost: 519.00
```

```
Total Distance: 519.20
```

- **Test Case 4: Edge Case - Overcapacity Package**

| SA | GA |
|---|---|
|  |  |

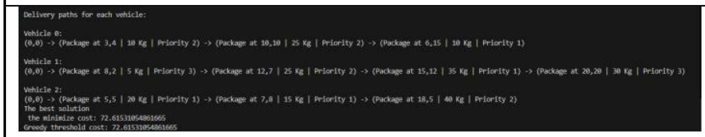
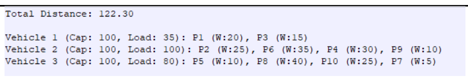
```
Delivery paths for each vehicle:

Vehicle 0:
(0,0)

Vehicle 1:
(0,0)
The best solution
 the minimize cost: 0
```

empty range in randrange(1, 1)

- **Test Case 5: Simulated Annealing vs. Genetic Algorithm Comparison**

| Package ID | Weight (kg) | Priority | Coordinates (x, y) |
|---|---|---|---|
| 0 | 20 | 1 | (5, 5) |
| 1 | 25 | 2 | (10, 10) |
| 2 | 15 | 1 | (7, 8) |

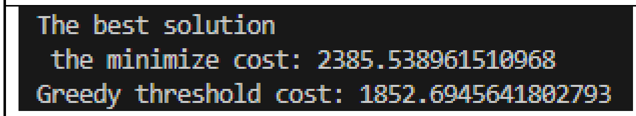
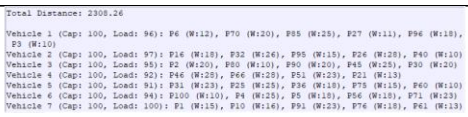
| | | | |
|---|---|---|---|
| 3 | 30 | 3 | (20, 20) |
| 4 | 10 | 2 | (3, 4) |
| 5 | 35 | 1 | (15, 12) |
| 6 | 5 | 3 | (8, 2) |
| 7 | 40 | 2 | (18, 5) |
| 8 | 10 | 1 | (6, 15) |
| 9 | 25 | 2 | (12, 7) |

| SA | GA |
|---|---|
|  |  |

```
Delivery paths for each vehicle:

Vehicle 0:
(0,0) -> (Package at 3,4 | 10 Kg | Priority 2) -> (Package at 10,10 | 25 Kg | Priority 2) -> (Package at 6,15 | 10 Kg | Priority 1)

Vehicle 1:
(0,0) -> (Package at 8,2 | 5 Kg | Priority 3) -> (Package at 12,7 | 25 Kg | Priority 2) -> (Package at 15,12 | 35 Kg | Priority 1) -> (Package at 20,20 | 30 Kg | Priority 3)

Vehicle 2:
(0,0) -> (Package at 5,5 | 20 Kg | Priority 1) -> (Package at 7,8 | 15 Kg | Priority 1) -> (Package at 18,5 | 40 Kg | Priority 2)
The best solution
 the minimize cost: 72.61531054861665
Greedy threshold cost: 72.61531054861665
```

```
Total Distance: 122.30

Vehicle 1 (Cap: 100, Load: 35): P1 (W:20), P3 (W:15)
Vehicle 2 (Cap: 100, Load: 100): P2 (W:25), P6 (W:35), P4 (W:30), P9 (W:10)
Vehicle 3 (Cap: 100, Load: 80): P5 (W:10), P8 (W:40), P10 (W:25), P7 (W:5)
```

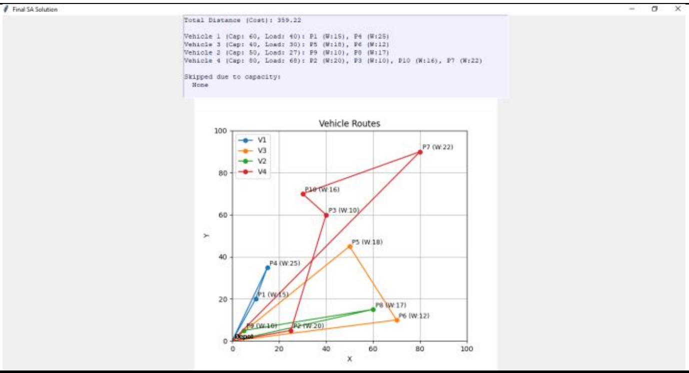
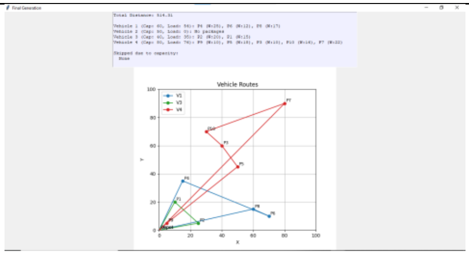
- **Test Case 6: Scalability Test**

| SA | GA |
|---|---|
|  |  |

```
The best solution
 the minimize cost: 2385.538961510968
Greedy threshold cost: 1852.6945641802793
```

```
Total Distance: 2308.26

Vehicle 1 (Cap: 100, Load: 96): P6 (W:12), P70 (W:20), P85 (W:25), P27 (W:11), P96 (W:18),
 P3 (W:10)
Vehicle 2 (Cap: 100, Load: 97): P16 (W:18), P32 (W:26), P95 (W:15), P26 (W:28), P40 (W:10)
Vehicle 3 (Cap: 100, Load: 95): P2 (W:20), P80 (W:10), P90 (W:20), P45 (W:25), P30 (W:20)
Vehicle 4 (Cap: 100, Load: 92): P46 (W:28), P66 (W:28), P51 (W:23), P21 (W:13)
Vehicle 5 (Cap: 100, Load: 91): P31 (W:23), P25 (W:25), P36 (W:18), P75 (W:15), P60 (W:10)
Vehicle 6 (Cap: 100, Load: 94): P100 (W:10), P4 (W:25), P5 (W:18), P56 (W:18), P71 (W:23)
Vehicle 7 (Cap: 100, Load: 100): P1 (W:15), P10 (W:16), P91 (W:23), P76 (W:18), P61 (W:13)
```

- **Test Case 7: User Interface Display Test**

| SA | GA |
|---|---|
|  |  |

```
Total Distance (Cost): 359.22

Vehicle 1 (Cap: 60, Load: 40): P1 (W:15), P4 (W:25)
Vehicle 3 (Cap: 40, Load: 30): P5 (W:18), P6 (W:12)
Vehicle 2 (Cap: 50, Load: 27): P9 (W:10), P8 (W:17)
Vehicle 4 (Cap: 80, Load: 68): P2 (W:20), P3 (W:10), P10 (W:16), P7 (W:22)

Skipped due to capacity:
  None
```

```
Total Distance: 314.31

Vehicle 1 (Cap: 60, Load: 54): P4 (W:25), P6 (W:12), P8 (W:17)
Vehicle 2 (Cap: 50, Load: 0): No packages
Vehicle 3 (Cap: 40, Load: 35): P2 (W:20), P1 (W:15)
Vehicle 4 (Cap: 80, Load: 76): P9 (W:10), P5 (W:18), P3 (W:10), P10 (W:16), P7 (W:22)

Skipped due to capacity:
  None
```

## 4. Conclusion:

In this project, we found that both Genetic Algorithm (GA) and Simulated Annealing (SA) can solve the package delivery problem well, even with limits on vehicle capacity and delivery priority. Simulated Annealing worked fast and gave good results for small problems, so it is useful when time is short, and the problem is simple. But because it only changes one solution at a time, it might miss better options. On the other hand, the Genetic Algorithm worked better for bigger and more complicated problems. It tries many solutions at once and can find better routes over time. It also handles rules and limits more easily and can run faster on multiple computers. So, if the delivery problem is big or needs very good results, GA is the better choice. If the problem is small and needs quick answers, SA is still a good option.