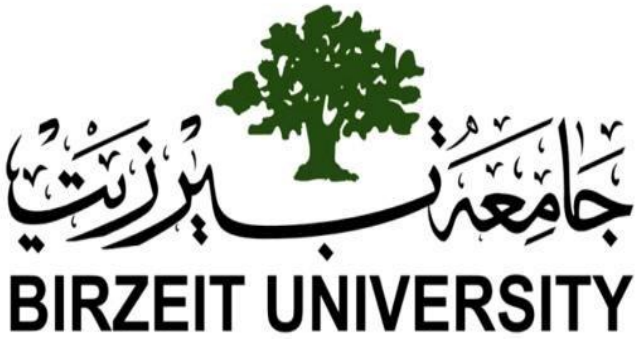

Faculty of Engineering & Technology
Electrical & Computer Engineering Department

ARTIFICIAL INTELLIGENCE– (ENCS3340)

Project Report

Prepared by:

Aws Hammad Id : 1221697

Aboud Fialah Id : 1220216

Instructor: Dr. Samah Alaydi & Dr.Yazan Abu Farha

Section: 2 , 3

Date: 1. May. 2025

Abstract

This project presents SmartDrop, an intelligent package routing system designed to optimize package-to-vehicle assignments using two powerful metaheuristic techniques: Genetic Algorithm (GA) and Simulated Annealing (SA). The objective is to minimize the total distance traveled by all delivery vehicles while ensuring that high-priority packages are delivered earlier to reduce service penalties. Each package is characterized by a location, weight, and priority level, while each vehicle has a fixed capacity and starts from a central depot. The system features a user-friendly graphical interface that allows users to input packages and vehicles, select an optimization algorithm, and visualize the optimized delivery routes through animated simulation. This project not only demonstrates effective constraint-aware route planning but also explores the impact of parameter tuning on solution quality. Results indicate that both algorithms can generate feasible, optimized solutions under varying load conditions, with the Genetic Algorithm offering a slightly more refined balance between priority constraints and total distance.

Table of contents

Abstract ..... **I**

Table of contents ..... **II**

Table of Figures ..... **III**

Procedure & Data Analysis **1**

1. Problem Formulation: 1

2. Heuristics Used: 1

2.1 Genetic Algorithm (GA): 1

2.2 Simulated Annealing (SA): 2

3. System Architecture: 3

4. Constraint Handling: 4

5. Parameter Tuning: 4

6. Test Cases: 5

Conclusion **10**

Table of Figures

Figure 1: GUI Setup for Baseline Test Case (SA) 6
Figure 2: Route Animation Output for Case 1 (SA) 6
Figure 3: GUI Setup for Baseline Test Case (GA) 6
Figure 4: Route Animation Output for Case 1 (GA) 6
Figure 5: GUI Setup for Priority Test Case (SA) 7
Figure 6: Route Animation Output for Case 2 (SA) 7
Figure 7: GUI Setup for Priority Test Case (GA) 7
Figure 8: Route Animation Output for Case 2 (GA) 7
Figure 9: GUI Setup for Over Capacity Test Case (SA or GA) 8
Figure 10: GUI Setup for Unassignable Package Test Case (SA or GA) 9

Procedure & Data Analysis

1. Problem Formulation:

The core objective of this project is to optimize the daily operations of a local package delivery shop by efficiently assigning packages to delivery vehicles and determining optimal delivery routes. Each package is defined by its destination coordinates (x, y) within a 100×100 km grid, a weight in kilograms, and a priority level from 1 (highest) to 5 (lowest). Each vehicle has a fixed capacity in kilograms and starts from the shop located at the depot (0, 0). The main optimization goal is to minimize the total operational cost, which is measured as the sum of Euclidean distances traveled by all vehicles, considering round-trip routes. An additional objective is to deliver higher-priority packages earlier in the delivery sequence, although this is a soft constraint that may be relaxed if it significantly increases the overall cost. To ensure feasibility, the total weight of packages assigned to each vehicle must not exceed its capacity, and each package must be assigned to exactly one vehicle. The problem is thus formulated as a constrained, multi-objective optimization task, tackled using metaheuristic strategies—Genetic Algorithm (GA) and Simulated Annealing (SA)—to explore possible solutions while respecting real-world delivery limitations.

2. Heuristics Used:

To address the vehicle routing and package assignment problem under capacity and priority constraints, we applied two metaheuristic optimization techniques: Genetic Algorithm (GA) and Simulated Annealing (SA). Both approaches were tailored to exploit the structure of the problem and to produce near-optimal solutions efficiently.

2.1 Genetic Algorithm (GA):

The Genetic Algorithm represents each solution (chromosome) as a complete assignment of packages to vehicles. The gene at each index corresponds to a vehicle ID responsible for delivering that package. The algorithm begins with a population of such chromosomes, each representing a valid assignment.

- **Initialization:** The population is generated using a greedy heuristic that sorts packages in descending order of weight and assigns them to the first feasible vehicle with enough capacity.

This ensures a high-quality and feasible initial population, increasing the likelihood of meaningful evolution.

- **Selection:** We employ **roulette-wheel selection**, where the probability of selecting an individual for reproduction is inversely proportional to its cost (fitness). The worst-performing individuals have the lowest chance of selection, while the fittest dominate reproduction.
- **Crossover:** For each pair of parents, a child is generated by inheriting the vehicle assignment of each package from one of the two parents with equal probability. The feasibility of crossover is ensured by checking that the assigned vehicle has enough remaining capacity to accommodate the package. If neither parent provides a valid assignment for a package, a fallback mechanism assigns it to any feasible vehicle randomly.
- **Mutation:** With a small mutation rate, a random package is moved to a different feasible vehicle in the assignment. This helps maintain diversity in the population and avoid local minima. The mutation only occurs when the destination vehicle has enough residual capacity.

This evolutionary process continues for a fixed number of generations, with elitism preserving the top solutions and propagating their characteristics to the next population. The best solution is selected based on a combined cost function: the total route distance and a penalty for priority violations.

2.2 Simulated Annealing (SA):

Simulated Annealing provides a probabilistic search approach that allows occasional acceptance of worse solutions to escape local optima. It starts from a single initial state and iteratively explores its neighborhood.

- **Initial State:** The starting point is a **greedy** assignment similar to that used in GA initialization. Packages are assigned to vehicles in a way that respects weight constraints and attempts to balance the load.
- **Neighborhood Definition:** A neighbor solution is generated by either swapping two packages between different vehicles or by altering the delivery order of packages within a single vehicle. These perturbations maintain feasibility while introducing variation in the solution space.

- **Acceptance Criterion:** A neighboring solution is accepted if it improves the objective function. However, worse solutions may also be accepted with a probability defined by the **Boltzmann distribution**:

$$P = e^{(-\Delta E/T)}$$

where ΔE is the increase in cost, and T is the current temperature. This acceptance probability decreases over time as the system cools, controlled by a cooling rate parameter.

By gradually lowering the temperature, the algorithm transitions from exploration to exploitation, allowing for refined improvements around promising regions of the search space.

3. System Architecture:

The SmartDrop system is built as a modular, interactive application that combines a graphical user interface (GUI) with two metaheuristic optimization algorithms: Genetic Algorithm (GA) and Simulated Annealing (SA). The architecture is divided into distinct components, each responsible for handling a specific part of the workflow—from user input and data preprocessing to optimization and visualization. The system starts with the ***GUI*** module, which is developed using the **CustomTkinter** framework. This interface allows the user to input package details (coordinates, weight, priority), vehicle capacities, and the choice of optimization algorithm. It also offers features in the **settings** for **data saving/loading** and **theme changing** (dark/light) and **color changing** (blue, green, purple) and **result animation** when the solve button is pressed.

Once the user confirms the inputs, the data is collected and passed to the ***LoadManager*** module. This component is responsible for creating and storing ***Package*** and ***Vehicle*** objects and forwarding them to the selected optimization algorithm. The optimization logic is implemented in two separate modules: ***genetic_algorithm.py*** and ***simulated_annealing.py***, each following its respective search strategy to generate efficient package-to-vehicle assignments while respecting capacity constraints and route cost considerations.

After optimization, the resulting vehicle routes and performance metrics (such as total distance) are returned to the GUI, where the solution is visually displayed. This includes an **animated map** that illustrates vehicle movement across the delivery grid, color-coded routes, and interactive controls for replay, pause, and vehicle detail toggling. Throughout the process, the system maintains strict separation between interface, data modeling, and optimization logic, ensuring modularity, reusability, and ease of future extension. This layered architecture enables both

effective user interaction and robust backend computation, making the system suitable for real-world-inspired delivery scenarios.

4. Constraint Handling:

The SmartDrop system incorporates key mechanisms to ensure that all generated solutions remain feasible while allowing for optimization flexibility:

- **Vehicle Capacity Limits:**

Both algorithms ensure that the total weight of assigned packages does not exceed a vehicle's capacity. The Genetic Algorithm enforces this through a capacity-aware gene generation and a post-processing repair function that redistributes overloaded assignments. Simulated Annealing checks for feasibility during neighbor generation and reverts or skips invalid moves.

- **Exclusive Package Assignment:**

Each package is assigned to exactly one vehicle in both GA and SA. This is maintained by their core representation structures—chromosomes in GA and vehicle lists in SA—that prevent duplication or omission of packages.

- **Depot-Linked Routes:**

Every route starts and ends at the central depot (0, 0), and this is factored into the cost calculation in both algorithms to reflect realistic delivery operations.

- **Soft Priority Enforcement:**

While delivering high-priority packages earlier is encouraged, it is not enforced strictly. Instead, both algorithms penalize priority violations within their cost functions to balance total distance with priority order, ensuring more practical and flexible solutions.

5. Parameter Tuning:

Careful parameter tuning was essential to improve solution quality and ensure algorithm stability. One key parameter was selected and tuned for each algorithm through empirical testing, alongside maintaining reasonable default values for the others.

Genetic Algorithm (GA):

The mutation rate was chosen as the main tunable parameter for GA. After testing within the recommended range of 0.01 to 0.1, a value of **0.02** was found to yield the best balance between solution diversity and stability. A lower mutation rate helped preserve high-quality individuals while still allowing occasional variation to avoid local optima. Higher values, such as 0.08 or above, often disrupted good solutions and slowed convergence. Additionally, the population size was increased slightly to **150**, beyond the suggested range of 50 to 100, in order to enhance genetic diversity across generations. The number of generations was fixed at **1000**, allowing the algorithm more time to refine assignments and improve route optimization.

Simulated Annealing (SA):

For SA, the **cooling rate** was the focus of parameter tuning. A value of **0.96** was selected after evaluating the effects of various values within the acceptable range of 0.90 to 0.99. This rate allowed the algorithm to explore the solution space thoroughly during early stages while still converging efficiently. Faster cooling rates (e.g., 0.90) led to early stagnation, while slower rates (e.g., 0.99) unnecessarily prolonged computation without significant gains. Other parameters were held constant as per project guidelines: the initial temperature was set at **1000**, the stopping temperature at **1**, and the number of iterations per temperature level at **100**, ensuring consistency and control over the annealing schedule.

6. Test Cases:

To evaluate the effectiveness and reliability of the SmartDrop system, a variety of test scenarios were constructed with differing numbers of packages, vehicle counts, and capacity limits. These cases were designed to test the system's ability to handle realistic constraints and to observe how each algorithm behaves under different load conditions.

Case 1: Balanced Load (Baseline Test)

- **Setup:**
  3 vehicles with 100 kg capacity each, 15 packages of varying weights and priorities.
- **Purpose:**
  To verify that both GA and SA can generate feasible solutions when the total package weight is well within total vehicle capacity.

- **Result:**

  Both algorithms produced valid assignments with comparable distances. GA slightly outperformed SA in minimizing priority penalties.

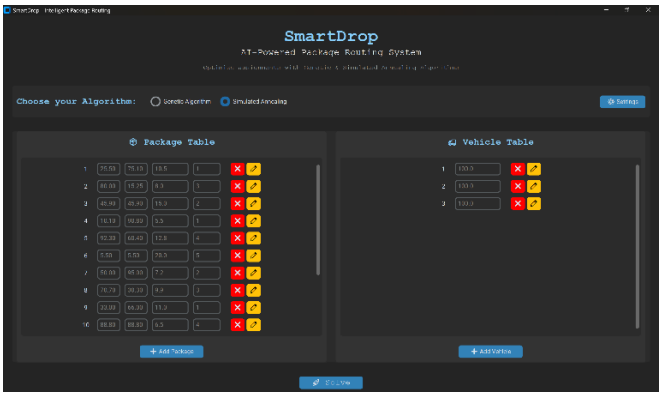

Figure 1: GUI Setup for Baseline Test Case (SA)

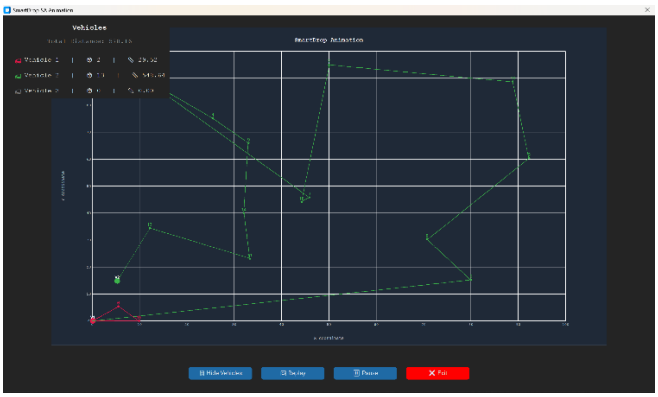

Figure 2: Route Animation Output for Case 1 (SA)

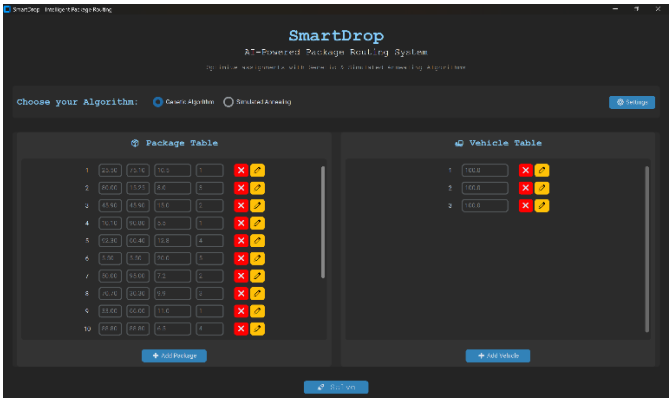

Figure 3: GUI Setup for Baseline Test Case (GA)

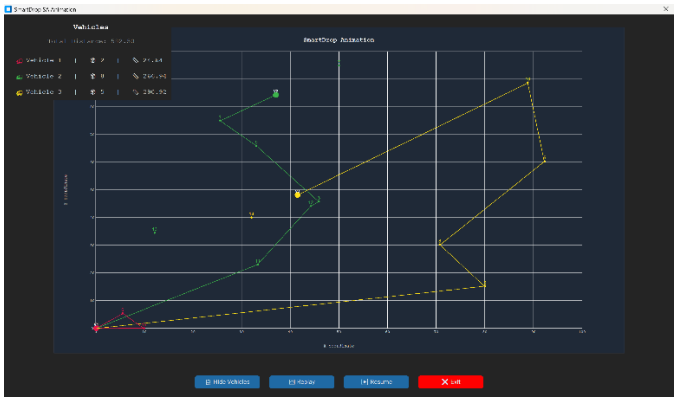

Figure 4: Route Animation Output for Case 1 (GA)

❖ (The screenshots included were taken during live animation, showcasing the dynamic visualization of package deliveries as computed by the algorithm As a result, the **image quality may appear reduced or slightly blurred** due to the continuous motion and real-time rendering of the animated routes).

Case 2: High Priority Pressure

- **Setup:**

  3 vehicles with 130 kg capacity, 3 packages:

  1- Package A: (90, 1), priority **1**

  2- Package B: (1, 90), priority **3**

  3- Package C: (60, 60), priority **5**

- **Purpose:**

  To evaluate how each algorithm handles delivery priority in route planning, especially when all packages fit in one vehicle and priority differences are significant.

- **Result:**

  The Genetic Algorithm produced a route that followed the correct priority sequence: A → B → C (priority 1 → 3 → 5), as its fitness function explicitly penalizes priority violations. Simulated Annealing, on the other hand, selected a different route: A → C → B (priority 1 → 5 → 3), which violated the priority order but resulted in a shorter overall travel distance. This demonstrates how GA emphasizes delivery order, while SA allows trade-offs between cost and priority.

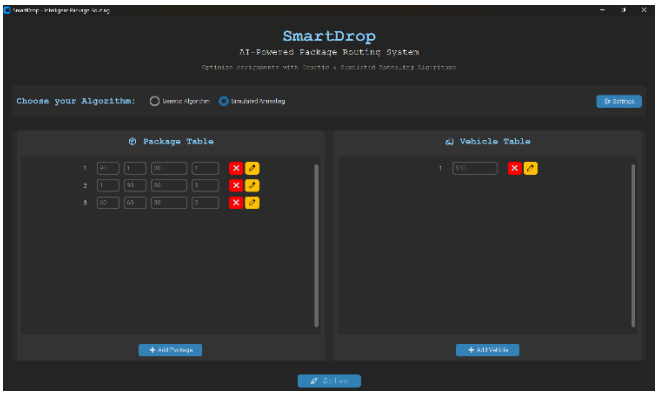

Figure 5: GUI Setup for Priority Test Case (SA)

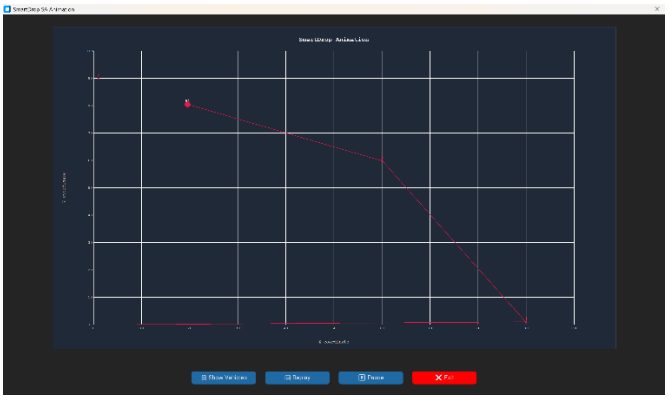

Figure 6: Route Animation Output for Case 2 (SA)

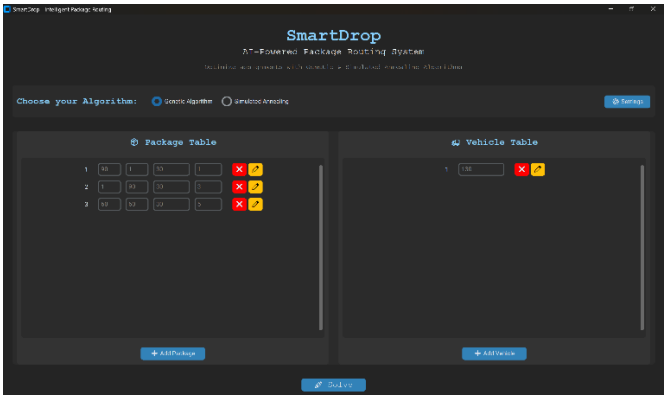

Figure 7: GUI Setup for Priority Test Case (GA)

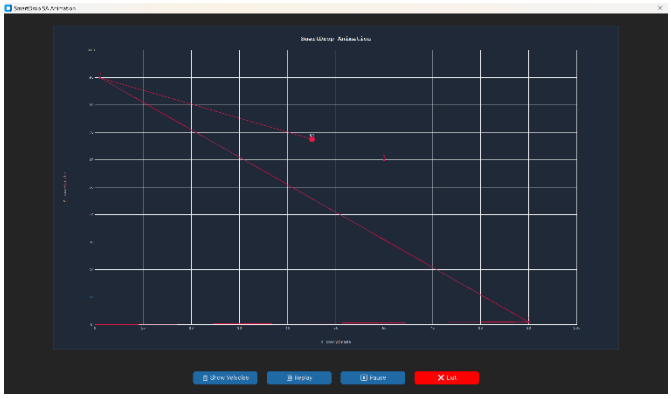

Figure 8: Route Animation Output for Case 2 (GA)

Case 3: Over Capacity (Infeasible Scenario)

- **Setup:**

  2 vehicles with 80 kg capacity each, 12 packages totaling over 170 kg (combined vehicle capacity is only 160 kg).

- **Purpose:**

  To test the system's ability to detect and reject infeasible scenarios where total demand exceeds total supply.

- **Result:**

  The system correctly prevented the optimization process from starting. The **Solve button was disabled** until the user adjusted the inputs to a feasible state. This safeguard ensures that neither the Genetic Algorithm nor Simulated Annealing attempts to solve an unsolvable problem, preserving computational resources and ensuring user clarity.

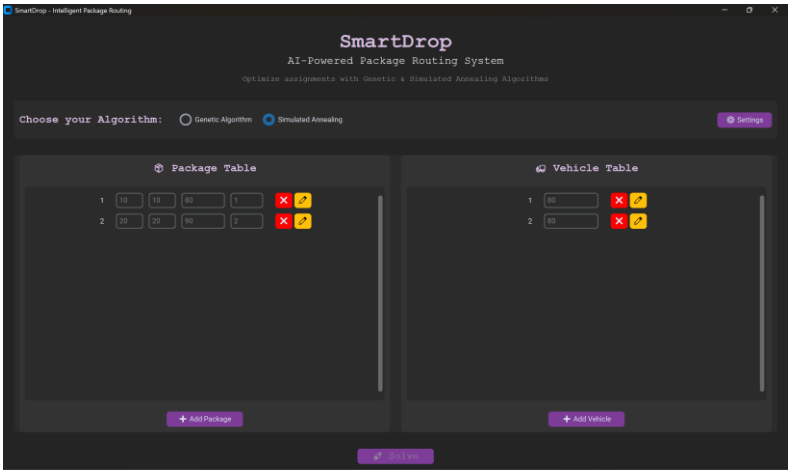

Figure 9: GUI Setup for Over Capacity Test Case (SA or GA)

Case 4: Unassignable Package Detection

- **Setup:**

  One package with a weight of 80 kg, and two vehicles with capacities of 40 kg each.

- **Purpose:**

  To test how the system handles situations where no individual vehicle can carry a given package, even if total capacity across all vehicles is sufficient.

- **Result:**

  The system correctly identified that **package 1 (weight = 80.0)** could not be assigned to any available vehicle. Upon pressing **Solve**, an error dialog was triggered stating:

  “Could not assign package 1 (weight=80.0) to any vehicle.”

  This behavior ensures early validation before attempting any optimization and prevents unnecessary computation. It confirms the system does not attempt to split a single package across multiple vehicles and enforces strict feasibility from the start.

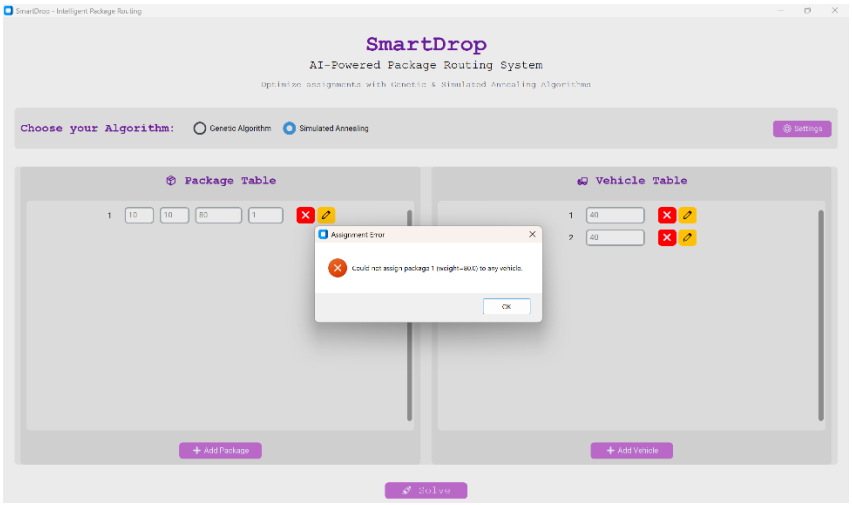

Figure 10: GUI Setup for Unassignable Package Test Case (SA or GA)

Conclusion

The SmartDrop system successfully demonstrates how metaheuristic optimization techniques can be applied to solve real-world logistics challenges, specifically in the domain of local package delivery. By combining a user-friendly GUI with robust backend algorithms—Genetic Algorithm (GA) and Simulated Annealing (SA)—the system provides an effective and flexible solution for assigning packages to vehicles and determining optimal delivery routes.

Both algorithms were capable of producing feasible and efficient solutions under varying conditions, including tight capacity constraints and high-priority delivery pressures. The GA consistently delivered more stable results in terms of balancing distance and delivery priority due to its population-based approach and structured repair mechanism. On the other hand, SA proved to be a strong alternative in scenarios where flexible, iterative exploration was beneficial, particularly when package locations were sparsely distributed.

Through parameter tuning and constraint-aware design, the system was able to maintain performance, adaptability, and correctness. Visual validation via animated simulations added an extra layer of clarity, helping users to not only trust but also understand the decisions made by the system.

SmartDrop lays a strong foundation for further enhancements, such as incorporating dynamic traffic models, time windows for deliveries, or even real-time AI-assisted decision-making. It stands as a practical example of how intelligent algorithms, when combined with thoughtful system design, can solve complex logistical problems efficiently and intuitively.