Modelling, Simulation and Optimization in Vehicle Routing Problem (VRP) Project CA

Rian Dwi Putra
MSc Data Analytics
National College of Ireland
Dublin, Ireland
x22108637@student.ncirl.ie

Abstract— *The Vehicle Routing Problem (VRP) is a critical optimization problem in transportation, logistics, and supply chain management, concerning the efficient distribution of goods and services to geographically dispersed customers using a fleet of vehicles. This paper provides an overview of the VRP, its significance, and the challenges it poses in real-world applications. This study also reviews the prominent solution approaches for the VRP, ranging from exact methods, such as integer programming and branch-and-bound, to heuristic and metaheuristic algorithms. This paper serves as a comprehensive introduction to the VRP, its challenges, and the state-of-the-art methods for tackling this complex optimization problem.*

Keywords— Vehicle Routing Problem, Travel Salesman Problem, Modelling, Simulation and Optimization

INTRODUCTION

Modelling, simulation, and optimization are interconnected methods used in various fields to understand, predict, and improve the performance of systems, processes, and phenomena.

Modelling: This is the process of creating a representation (usually mathematical) of a real-world system, process, or phenomenon. Models can be classified into different types, such as physical models, statistical models, or computer models, depending on the application. The goal of modelling is to simplify complex systems and capture their essential features, allowing us to understand and predict their behavior under different conditions.

Simulation: Simulation refers to the process of using a model to imitate the behavior of a real-world system over time. This can be done using mathematical equations, computer algorithms, or physical replicas. Simulations are often used to test hypotheses, assess performance under various conditions, or predict future outcomes. They can be classified as deterministic (where the same input always produces the same output) or stochastic (where randomness is incorporated into the model).

Optimization: Optimization involves finding the best solution to a problem, usually by minimizing or maximizing an objective function. In the context of modelling and simulation, optimization seeks to find the best set of input parameters or decision variables that produce the most desirable outcome, subject to certain constraints. Optimization techniques include linear programming, nonlinear programming, genetic algorithms, and gradient-based methods, among others.

In summary, modelling, simulation, and optimization are interrelated methods used to understand, predict, and improve real-world systems. They have applications in various fields such as engineering, finance, economics, operations research, and computer science. By developing models and running simulations, researchers can gain insights into system behavior, identify potential problems or opportunities, and optimize solutions for improved performance.

This study will only focus on Modelling, Simulation and Optimization in Vehicle Routing Problem.

Combinatorial optimization problem known as the Vehicle Routing Problem (VRP) arises in transportation, logistics, and distribution. The VRP's main objective is to find the best routes for a fleet of vehicles to take to deliver goods to a set of customers at the lowest possible cost or distance. The problem can be thought of as an extension of the well-known traveling salesman problem (TSP), in which the goal is to find the shortest route that goes to a set of locations and then comes back to the starting point.

In its simplest form, the VRP considers a single depot (or starting point) from which a fleet of vehicles departs to serve a set of customers with known demand. Each vehicle has a limited capacity, and the objective is to determine the best routes for the vehicles so that all customer demands are met, without exceeding the vehicle capacity, and the total cost (or distance) is minimized.

Data & Research Question

The experiments presented in this study are based on Project Descriptor that has been shared in Moodle and its related Jupyter Notebook file.

And these are some of research questions that may raise based on those files:

1. How can we incorporate data-driven approaches into the modelling, simulation, and optimization of VRP to leverage historical data, improve prediction accuracy, and adapt to evolving circumstances?
2. How can we develop robust and resilient vehicle routing strategies that can cope with uncertainties, disruptions, or changing conditions in supply chains and distribution networks?

Related Work

The Vehicle Routing Problem (VRP) is a combinatorial optimization problem that arises in the fields of transportation, logistics, and supply chain management. The main objective of the VRP is to determine the optimal routes for a fleet of vehicles to deliver goods to a set of geographically dispersed customers, while minimizing the total cost or distance traveled, subject to various constraints.

In its simplest form, the VRP consists of the following components:

1. A central depot or multiple depots, where vehicles are based.
2. A set of customers, each with specific demands for goods or services.
3. A fleet of vehicles, each with a limited carrying capacity.
4. A road network or distance matrix, representing the distances or travel times between customers and depots.

Here are a few key works in the areas of Vehicle Routing Problem (VRP) and Traveling Salesman Problem (TSP)

- Toth, P., & Vigo, D. (Eds.). (2020). The vehicle routing problem. Society for Industrial and Applied Mathematics. [1] This book provides a comprehensive overview of the VRP, its variants, and state-of-the-art solution approaches, including exact methods, heuristics, and metaheuristics. It is a valuable resource for researchers and practitioners in the field of vehicle routing.
- Laporte, G., & Semet, F. (2020). Classical heuristics for the capacitated VRP. In The Vehicle Routing Problem (pp. 109-128). Society for Industrial and Applied Mathematics. [2] This paper reviews classical heuristics for the Capacitated VRP, such as the Clarke-Wright Savings Algorithm, the Sweep Algorithm, and various insertion heuristics, providing insights into their performance and applicability.
- Dorigo, M., & Stützle, T. (2014). Ant Colony Optimization. MIT Press. [3] In this book, the authors present a comprehensive overview of Ant Colony Optimization (ACO) algorithms, which have been successfully applied to various combinatorial optimization problems, including the TSP and VRP.

The goal is to find the best set of routes for the vehicles to visit all customers exactly once, satisfy their demands, and return to the depot, while adhering to capacity constraints and minimizing the overall cost, which is usually represented by the total distance or travel time.

Based on those papers, there are several variants of the VRP, including:

• The Issue of Capacitated Vehicle Routing: The goal is to meet capacity constraints while minimizing total distance or cost for vehicles with limited capacity.

• Time Windows Vehicle Routing Issue: Customers have predetermined time windows within which they must be served, which complicates the routing problem even more.

• Numerous Station Vehicle Steering Issue: Because there are multiple depots from which vehicles can begin and end their routes, the decisions regarding routing are more difficult.

• Pickup and delivery vehicle routing issue: Goods must be picked up and delivered by vehicles, and the order in which they are delivered and picked up often has additional restrictions.

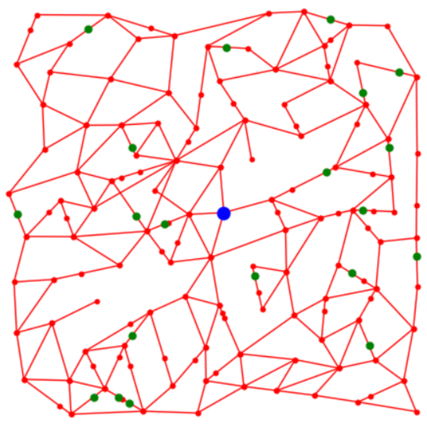
• Problem with Vehicle Routing in a Heterogeneous Fleet: The armada of vehicles has various limits, paces, or expenses, settling on the steering choices more mind boggling.

• Problem with Dynamic Vehicle Routing: Real-time or adaptive routing strategies are required because the problem data (such as customer demands, locations, or time windows) fluctuate over time.

The VRP and its variants are typically solved using optimization algorithms and heuristics, such as linear programming, integer programming, metaheuristics (e.g., genetic algorithms, simulated annealing, or ant colony optimization), or local search techniques. Due to its practical importance and computational complexity, the VRP has been widely studied in operations research, logistics, and transportation science, with ongoing research aimed at developing more efficient and effective solution methods.

METHODOLOGY

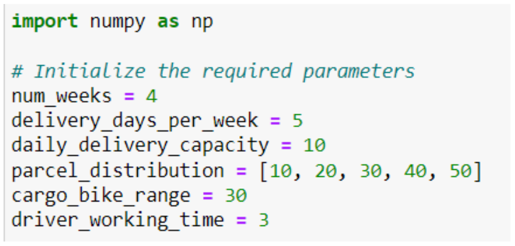
This map will be used for simulating the distribution of parcel.



```
[(0.6255, 0.0725), (0.2975, 0.2509), (0.7128, 0.659), (0.853, 0.128
6349, 0.5404), (0.2484, 0.5803), (0.7785, 0.0558), (0.3562, 0.1899)
64, 0.7068), (0.5326, 0.9258), (0.3907, 0.5468), (0.2551, 0.3104),
0.7062), (0.1971, 0.3915), (0.9406, 0.626), (0.5513, 0.6662), (0.25
865), (0.5761, 0.858), (0.4095, 0.2474), (0.9478, 0.5086), (0.1487,
```

Figure 1

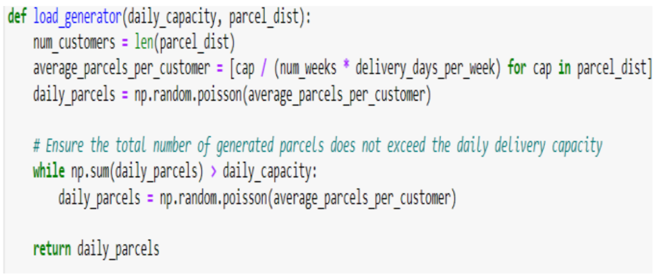
This code will initialize the required parameters for making the simulation works. And parcel distribution refers to the parameter used to determine the distribution of parcels across customers. The distribution is modeled using a Poisson process, which is a stochastic process often used to model random events occurring over time or space. The Poisson distribution has a single parameter, which represents the average number of events (in this case, parcels) occurring in a fixed interval (in this case, per customer).



```python
import numpy as np

# Initialize the required parameters
num_weeks = 4
delivery_days_per_week = 5
daily_delivery_capacity = 10
parcel_distribution = [10, 20, 30, 40, 50]
cargo_bike_range = 30
driver_working_time = 3
```

Figure 2

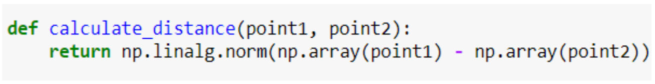
Then we build load generator function, which refers to a function that simulates the arrival of parcels at the distribution point. The function generates the number of parcels arriving at the distribution point for each customer on a given day. It does so by considering the daily delivery capacity (N) and parcel distribution (P) parameters.



```python
def load_generator(daily_capacity, parcel_dist):
    num_customers = len(parcel_dist)
    average_parcels_per_customer = [cap / (num_weeks * delivery_days_per_week) for cap in parcel_dist]
    daily_parcels = np.random.poisson(average_parcels_per_customer)

    # Ensure the total number of generated parcels does not exceed the daily delivery capacity
    while np.sum(daily_parcels) > daily_capacity:
        daily_parcels = np.random.poisson(average_parcels_per_customer)

    return daily_parcels
```

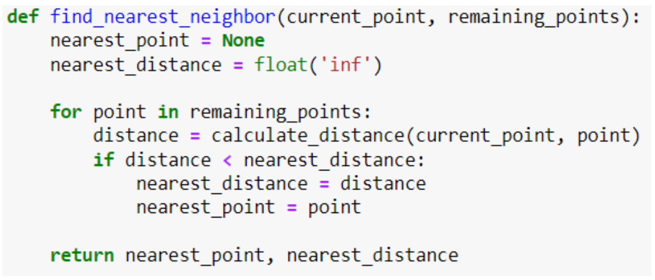
Figure 3

Then these codes will implement greedy algorithm by finding its nearest neighbor and calculate the distance.



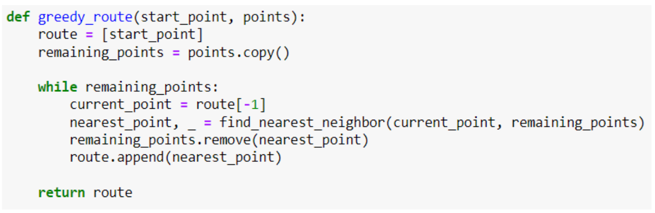
```
def calculate_distance(point1, point2):
    return np.linalg.norm(np.array(point1) - np.array(point2))
```

Figure 4



```
def find_nearest_neighbor(current_point, remaining_points):
    nearest_point = None
    nearest_distance = float('inf')

    for point in remaining_points:
        distance = calculate_distance(current_point, point)
        if distance < nearest_distance:
            nearest_distance = distance
            nearest_point = point

    return nearest_point, nearest_distance
```

Figure 5

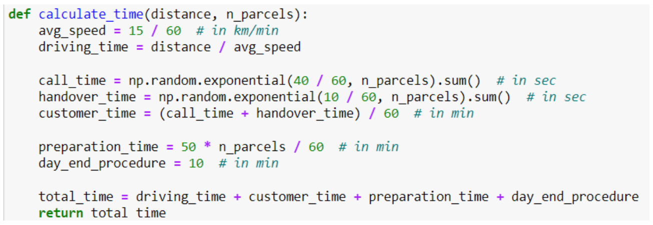


```
def greedy_route(start_point, points):
    route = [start_point]
    remaining_points = points.copy()

    while remaining_points:
        current_point = route[-1]
        nearest_point, _ = find_nearest_neighbor(current_point, remaining_points)
        remaining_points.remove(nearest_point)
        route.append(nearest_point)

    return route
```

Figure 6

A greedy route, in the context of routing and delivery problems, usually refers to a simple and fast approach to finding a delivery route by iteratively choosing the best available option at each step.

After finding the shortest path for every route, this code will apply related constraint to the simulation. The following things need to be calculated while doing the simulation: driving time, call time, handover time, customer time, preparation time and total time.

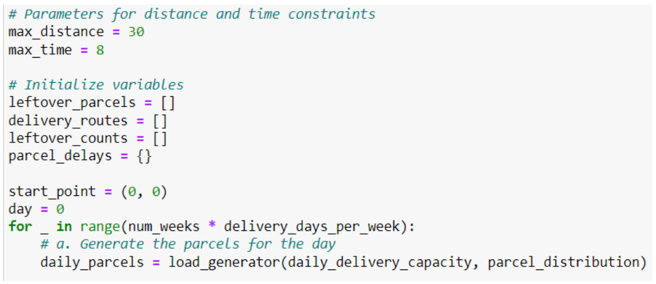


```
def calculate_time(distance, n_parcels):
    avg_speed = 15 / 60  # in km/min
    driving_time = distance / avg_speed

    call_time = np.random.exponential(40 / 60, n_parcels).sum()  # in sec
    handover_time = np.random.exponential(10 / 60, n_parcels).sum()  # in sec
    customer_time = (call_time + handover_time) / 60  # in min

    preparation_time = 50 * n_parcels / 60  # in min
    day_end_procedure = 10  # in min

    total_time = driving_time + customer_time + preparation_time + day_end_procedure
    return total_time
```

Figure 7

Then this code will do looping or iteration through the whole month (over 4 weeks of 5 delivery days per week)

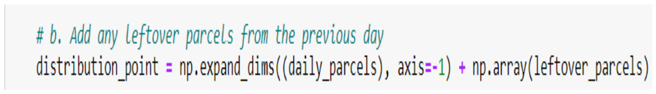
a. Generate the parcels for the day.



```
# Parameters for distance and time constraints
max_distance = 30
max_time = 8

# Initialize variables
leftover_parcels = []
delivery_routes = []
leftover_counts = []
parcel_delays = {}

start_point = (0, 0)
day = 0
for _ in range(num_weeks * delivery_days_per_week):
    # a. Generate the parcels for the day
    daily_parcels = load_generator(daily_delivery_capacity, parcel_distribution)
```

Figure 8

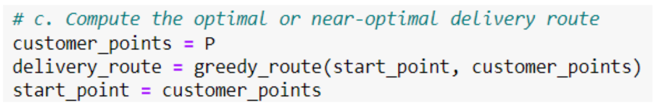
b. Add any leftover parcels from the previous day.



```
# b. Add any leftover parcels from the previous day
distribution_point = np.expand_dims((daily_parcels), axis=-1) + np.array(leftover_parcels)
```

Figure 9

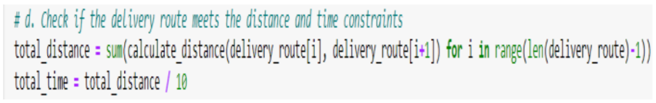
c. Compute the optimal or near-optimal delivery route.



```
# c. Compute the optimal or near-optimal delivery route
customer_points = P
delivery_route = greedy_route(start_point, customer_points)
start_point = customer_points
```

Figure 10

d. Check if the delivery route meets the distance and time constraints.

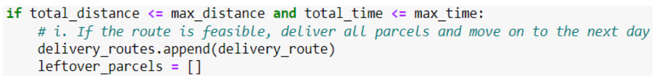


```
# d. Check if the delivery route meets the distance and time constraints
total_distance = sum(calculate_distance(delivery_route[i], delivery_route[i+1]) for i in range(len(delivery_route)-1))
total_time = total_distance / 10
```

Figure 11

e. Store the final delivery route and any leftover parcels for the next day.

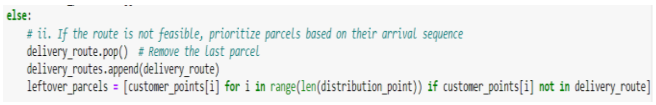
i). If the route is feasible, deliver all parcels and move on to the next day



```
if total_distance <= max_distance and total_time <= max_time:
    # i. If the route is feasible, deliver all parcels and move on to the next day
    delivery_routes.append(delivery_route)
    leftover_parcels = []
```

Figure 12

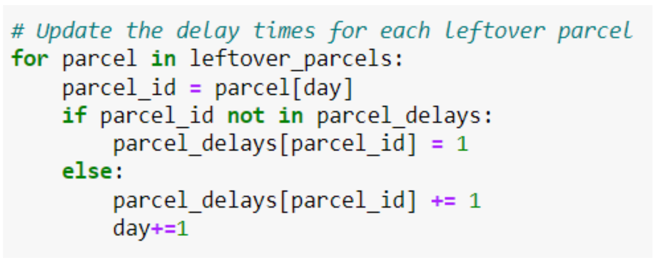
ii). If the route is not feasible, prioritize parcels based on their arrival sequence.



```
else:
    # ii. If the route is not feasible, prioritize parcels based on their arrival sequence
    delivery_route.pop()  # Remove the last parcel
    delivery_routes.append(delivery_route)
    leftover_parcels = [customer_points[i] for i in range(len(distribution_point)) if customer_points[i] not in delivery_route]
```

Figure 13

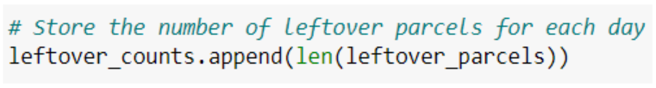
After finishing the loop, this code will update the delay times for each leftover parcel



```
# Update the delay times for each leftover parcel
for parcel in leftover_parcels:
    parcel_id = parcel[day]
    if parcel_id not in parcel_delays:
        parcel_delays[parcel_id] = 1
    else:
        parcel_delays[parcel_id] += 1
        day+=1
```

Figure 14

At the end of the day, it will store the number of leftover parcels for each day.

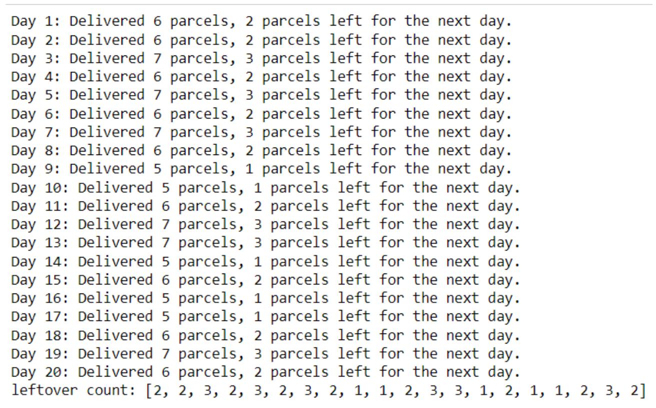


```
# Store the number of leftover parcels for each day
leftover_counts.append(len(leftover_parcels))
```

Figure 15

RESULTS

Based on generated data, this study will analyze and visualizing data using Python



```
Day 1: Delivered 6 parcels, 2 parcels left for the next day.
Day 2: Delivered 6 parcels, 2 parcels left for the next day.
Day 3: Delivered 7 parcels, 3 parcels left for the next day.
Day 4: Delivered 6 parcels, 2 parcels left for the next day.
Day 5: Delivered 7 parcels, 3 parcels left for the next day.
Day 6: Delivered 6 parcels, 2 parcels left for the next day.
Day 7: Delivered 7 parcels, 3 parcels left for the next day.
Day 8: Delivered 6 parcels, 2 parcels left for the next day.
Day 9: Delivered 5 parcels, 1 parcels left for the next day.
Day 10: Delivered 5 parcels, 1 parcels left for the next day.
Day 11: Delivered 6 parcels, 2 parcels left for the next day.
Day 12: Delivered 7 parcels, 3 parcels left for the next day.
Day 13: Delivered 7 parcels, 3 parcels left for the next day.
Day 14: Delivered 5 parcels, 1 parcels left for the next day.
Day 15: Delivered 6 parcels, 2 parcels left for the next day.
Day 16: Delivered 5 parcels, 1 parcels left for the next day.
Day 17: Delivered 5 parcels, 1 parcels left for the next day.
Day 18: Delivered 6 parcels, 2 parcels left for the next day.
Day 19: Delivered 7 parcels, 3 parcels left for the next day.
Day 20: Delivered 6 parcels, 2 parcels left for the next day.
leftover count: [2, 2, 3, 2, 3, 2, 3, 2, 1, 1, 2, 3, 3, 1, 2, 1, 1, 2, 3, 2]
```

Figure 16. Result of Simulation

1. The distribution of the length of the delivery route per day (in km)

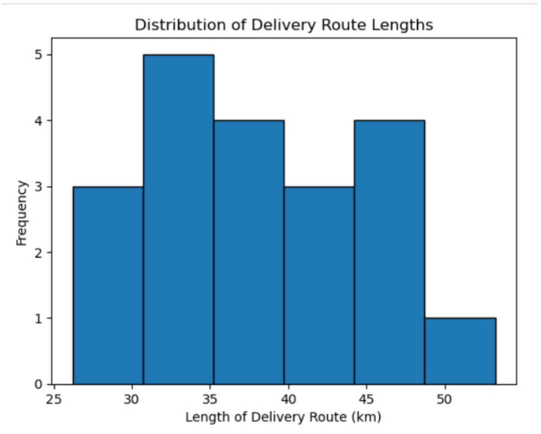


Figure 17

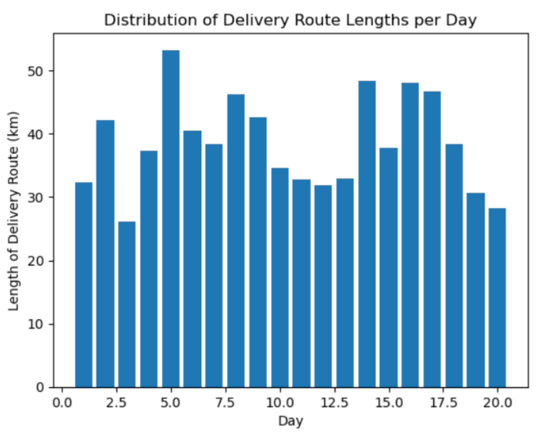


Figure 18

This histogram refers to the spread or variation in the total distances traveled by the delivery vehicle (e.g., cargo bike) for each day during the simulation period. It provides an overview of how long the delivery routes are on different days.

2. The distribution of the working time per day for the driver (in minutes)

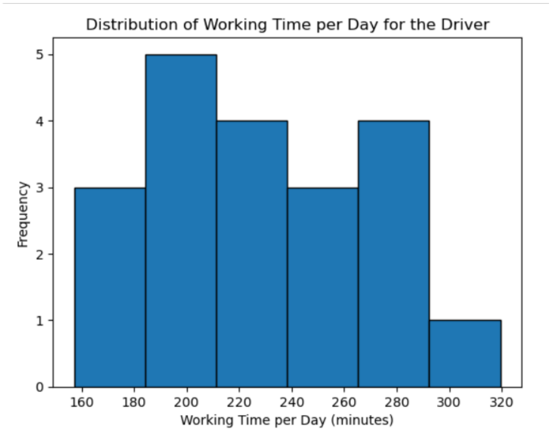


Figure 19

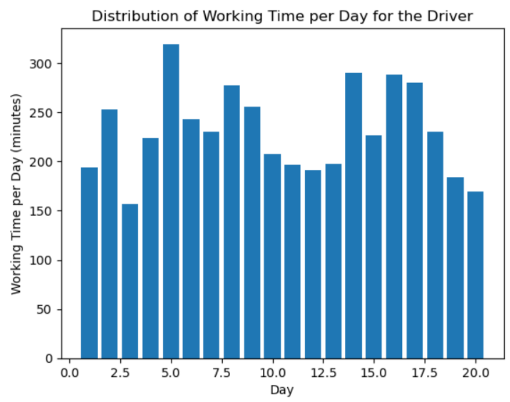


Figure 20

This histogram refers to the spread or variation in the total amount of time the driver spends on their daily delivery route during the simulation period.

3. The distribution of the number of parcels left over for the next day (per day)

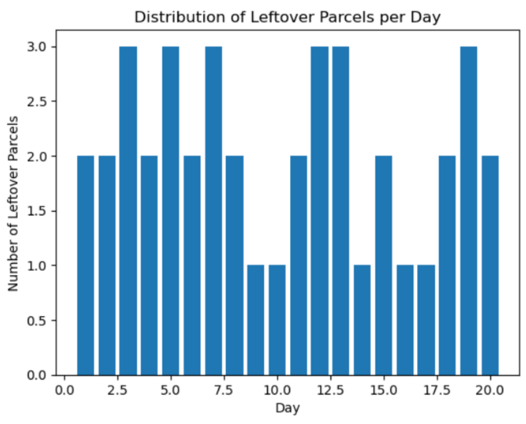


Figure 21

This histogram refers to an overview of how many parcels remain undelivered at the end of each day during a simulation period which can not be delivered due to constraints like distance, time, or capacity.

4. The distribution of the delay time of the parcels (in hours)?

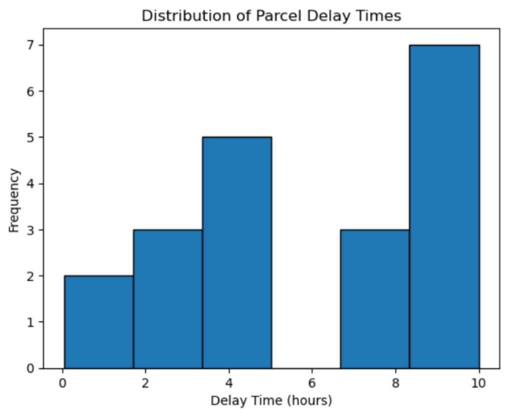


Figure 22

This histogram refers to the spread or variation in the amount of time parcels spend waiting at the distribution point before being delivered to the customers. Based on project descriptor, this study assume that 3 working hours equals to 1 day of delivery.

CONCLUSION AND FUTURE WORKS

Conclusion: Building a simulation on a vehicle routing problem provides valuable insights into the efficiency and effectiveness of the delivery system and routing algorithms. The simulation can help identify bottlenecks, inefficiencies, and areas for improvement in the delivery process. By analyzing various aspects, such as the length of delivery routes, driver working time, parcel delay times, and leftover parcel counts, stakeholders can make informed decisions to optimize delivery routes, capacity, and resource allocation. The simulation can also help validate new routing algorithms and compare different strategies to find the most suitable one for the specific delivery system.

Future Works:

- Enhancing the routing algorithm: Investigate and develop more advanced and efficient routing algorithms.
- Adapting to dynamic scenarios: Extend the simulation to handle real-time changes in demand, traffic conditions, and other factors that can impact the delivery process.
- Evaluating alternative delivery methods: Investigate the use of different types of vehicles.
- Integration with real-world data: Connect the simulation to actual data sources, such as GPS tracking, traffic data, or parcel tracking systems.

REFERENCES

[1] Toth, P. and Vigo, D. (2020) “The vehicle routing problem,” Society for Industrial and Applied Mathematics.

[2] Laporte, G. and Semet, F. (2020) “Classical heuristics for the capacitated VRP,” in The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, pp. 109–128.

[3] Dorigo, M. and Stützle, T. (2014) “Ant colony optimization algorithms for the traveling salesman problem,” pp. 65–119.