Simulation For Low-Cost Best Warehouse Location

Bolormaa Mendbayar- x23176725

National College of Ireland

MSc in Data Analytics

Modelling Simulation and Optimization

CA 1- Semester 2023/2024

Abstract*— This study investigates the optimal location for delivery centers for We-Doo, a startup aiming to enhance last-mile delivery in commuter areas. The objective is to identify the most cost-effective delivery center location and assess its significance. Simulation techniques are utilized to model the day-to-day operations of the delivery company, including routes, parcel prioritization, and costs. The results reveal significant cost differences among candidate warehouse locations, with implications for profitability. Statistical analysis, including ANOVA and F-statistics, confirms the significance of these findings. This study underscores the importance of selecting the optimal delivery center location and suggests potential areas for future research.

Keywords: Delivery optimization, Simulation, Last-mile logistics, Statistical analysis.

I. Introduction

Last-mile delivery, the final stage of the delivery process from distribution centers to customers' doorsteps, is crucial in the logistics industry. It represents a significant challenge due to its complexity and cost implications. In response to this challenge, startups like We-Doo are exploring innovative solutions to optimize last-mile delivery services in commuter townships.

This project aims to address the specific challenge faced by We-Doo in selecting the optimal location for delivery centers to facilitate efficient last-mile delivery services. The objective is to recommend the most cost-effective delivery center location and assess its statistical significance.

Our requirement for generating customer and warehouse locations utilizes the last four digits of the student ID, which in this case are 6725. Below, within those warehouse locations, we will find the optimal solution.

Generated 10 warehouse locations
(5120, 3440), (2880, 2320), (5680, 3440),(3440, 2880), (5680, 4560),(2880, 1200),(5680, 7360),(4000, 7360), (2320, 6240),(5680, 640)

Table 1: Warehouse locations

The applied constraints in the simulation are as follows:

Constraints for Simulation Study:

1. Cargo bike maximum range: 40 km
2. Driving speed: 15 km/h
3. Parcel handover time: Call mean time of 40 seconds. Additional handover time of 10 seconds.
4. Cumulative preparation time: 50 seconds per parcel.
5. Cargo bike charging: Day-End procedure takes 10 minutes.
6. Operational cost: Covering electricity and maintenance, is 8 cents per kilometer.
7. Driver's payment: Per hour rate of 30€ per hour, minimum payment of 60€ per working day.

II. Methodology

Overall, the simulation contains 7 steps.

Step 1: Generating Customer and Warehouse locations.

In the first step, the last 4 digits of the student ID were utilized to generate the customer and warehouse locations dataset and its randomly created on the map. Candidate warehouse locations(blue) in the town below shown in Figure 1. As well as green dots are customer locations which in this case created with 150 customers and 100 nodes.

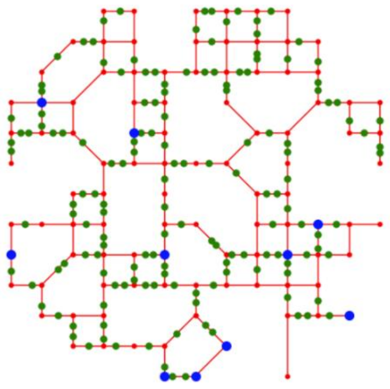


Figure 1. 10 Warehouses and Customers locations

Step 2: Generating Delivery Data

The dataset generated from step 1 was loaded to observe how delivery data is generated. Note: The average number of parcels expected by a customer per day is denoted by p = 0.15. For example, with n = 7, D is represented as:
D= [[3, 3, 7], [0, 1], [0, 4, 9, 9], [3, 7], [], [], [0, 0, 2, 2, 7]]

Over 7 days, a total of 16 parcels are to be delivered:
On the first day deliver one parcel to customer 7 and two parcels to customer 3.
On the second day deliver one parcel each to customers 0 and 1.
On the third day deliver one parcel each to customers 0, and 4, and two parcels to customer 9.
On the fourth day deliver one parcel to customers 3 and 7.
On the fifth and sixth day there are no parcels to deliver.
And on the seventh day two parcels to customers 0 and 2, and one parcel to customer 7.

Step 3: Implementing and Testing the A* Algorithm for Finding the Shortest Path

Implemented the A* algorithm to find the shortest path between two points in each map. The algorithm operates on a graph represented by vertices and edges, where each vertex denotes a location, and each edge represents a connection between two locations. This algorithm efficiently solves the shortest path problem, which is crucial for optimizing navigation in various applications. As we did in step 2, loaded the dataset to evaluate on our map to find the shortest path between the first and last targets. The total length of the path is 9506 m.

Step 4: Monte-Carlo Optimization for Fixed Number of Customers

Employed Monte-Carlo optimization to efficiently find an optimal solution for serving a fixed number of customers in a single loop. Monte Carlo optimization is a heuristic search algorithm that iteratively samples practical solutions from a given set of candidates and evaluates them to find the best solution. In the end of step 4 assessed the optimization approach on the dataset to find an optimal delivery route for a fixed 150 customers.

Step 5: Events

Established the connection between events across different entities in the event graphs.

Linking Events:

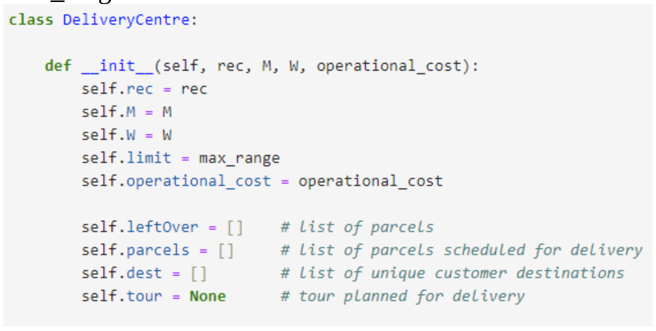
1. Delivery Center – Accept Parcel
Parcel – Arrive in Distribution Centre
2. Delivery Center – Prepare for Delivery
Driver – Leave for Delivery
Parcel* – Collect for Delivery
3. Driver – Ring at Customer
Customer – Call at door
4. Parcel and Delivery Centre
Parcel – Arrive in Distribution Centre
Parcel – Collect for Delivery
Driver – Leave for Delivery
Driver – Return to Delivery Centre

Step 6: Model Verification

In this step 6, employed constraints as mentioned in Introduction [1].

1. Cargo bike maximum range: 40 km

Included this limit in section 9, Class Delivery Centre, within def _init_ function where it is instantiated as self.limit = max_range, and in section 10.1, the parameter defined as max_range = 40000.

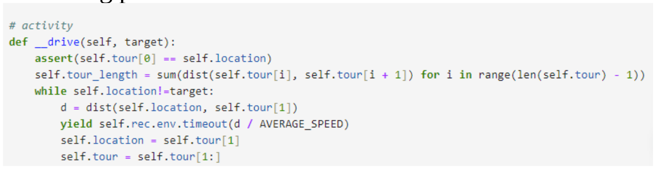


```
class DeliveryCentre:

    def __init__(self, rec, M, W, operational_cost):
        self.rec = rec
        self.M = M
        self.W = W
        self.limit = max_range
        self.operational_cost = operational_cost

        self.leftOver = []    # list of parcels
        self.parcels = []     # list of parcels scheduled for delivery
        self.dest = []        # list of unique customer destinations
        self.tour = None      # tour planned for delivery
```

Figure 2. Relevant section of code

2. Driving speed: 15 km/h

In section 8, Driver class, particularly within the _drive function, already established a key parameter that significantly influences the delivery operations: the average speed at which drivers travel. In section 10.1 of documentation, explicitly defined this parameter as average_speed = 15/3.6. This value, expressed in meters per second, represents the average speed drivers maintain while delivering parcels.

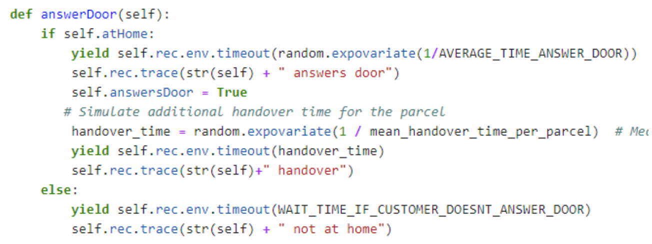


```
# activity
def __drive(self, target):
    assert(self.tour[0] == self.location)
    self.tour_length = sum(dist(self.tour[i], self.tour[i + 1]) for i in range(len(self.tour) - 1))
    while self.location!=target:
        d = dist(self.location, self.tour[1])
        yield self.rec.env.timeout(d / AVERAGE_SPEED)
        self.location = self.tour[1]
        self.tour = self.tour[1:]
```

Figure 3. Relevant section of code

3. Parcel handover time: Call mean time of 40 seconds. Additional handover time of 10 seconds.

In the section 7, Customer class, within the def answerDoor function, already established average time to answer door parameter. As well as additional handover time parameter is added in same function. In section 10.1 of documentation, defined those parameters as AVERAGE_TIME_ANSWER_DOOR = 40 and mean_handover_time_per_parcel = 10, those values expressed in seconds.

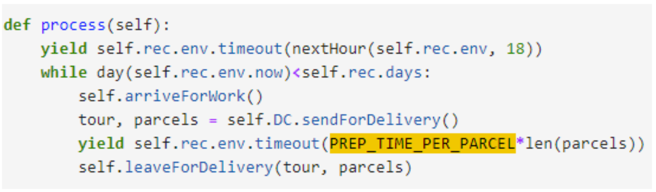


```
def answerDoor(self):
    if self.atHome:
        yield self.rec.env.timeout(random.expovariate(1/AVERAGE_TIME_ANSWER_DOOR))
        self.rec.trace(str(self) + " answers door")
        self.answersDoor = True
       # Simulate additional handover time for the parcel
        handover_time = random.expovariate(1 / mean_handover_time_per_parcel)  # Mec
        yield self.rec.env.timeout(handover_time)
        self.rec.trace(str(self)+" handover")
    else:
        yield self.rec.env.timeout(WAIT_TIME_IF_CUSTOMER_DOESNT_ANSWER_DOOR)
        self.rec.trace(str(self) + " not at home")
```

Figure 4. Relevant section of code

4. Cumulative preparation time: 50 seconds per parcel

In section 8, class Driver, within def _process function, already established prep_time_per_parcel parameter. In section 10.1 of documentation, defined parameter as PREP_TIME_PER_PARCEL = 50 and value expressed in seconds. This parameter represents the amount of time drivers spend preparing each parcel for delivery.

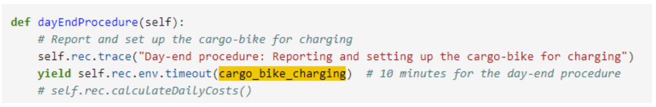


```
def process(self):
    yield self.rec.env.timeout(nextHour(self.rec.env, 18))
    while day(self.rec.env.now)<self.rec.days:
        self.arriveForWork()
        tour, parcels = self.DC.sendForDelivery()
        yield self.rec.env.timeout(PREP_TIME_PER_PARCEL*len(parcels))
        self.leaveForDelivery(tour, parcels)
```

Figure 5. Relevant section of code

5. Cargo bike charging: Day-End procedure takes 10 minutes.

In section 8, class Driver added new function which is def dayEndProcedure, end of the day spending extra 10 minutes for charging bike, the driver after arriving at Delivery Centre, before the driver goes home. Following this added this function within process function, ensuring that it is executed at the appropriate time within the driver's daily workflow. In section 10.3, Simulation routine, incorporated this function within generatorProcess function. In section 10.1 of documentation , defined parameter as cargo_bike_charging = 10*60, the value expressed in seconds. This step ensures that our drivers start each day with a fully charged bike, ready for optimal performance during deliveries.



```
def dayEndProcedure(self):
    # Report and set up the cargo-bike for charging
    self.rec.trace("Day-end procedure: Reporting and setting up the cargo-bike for charging")
    yield self.rec.env.timeout(cargo_bike_charging)  # 10 minutes for the day-end procedure
    # self.rec.calculateDailyCosts()
```

Figure 6. Relevant section of code

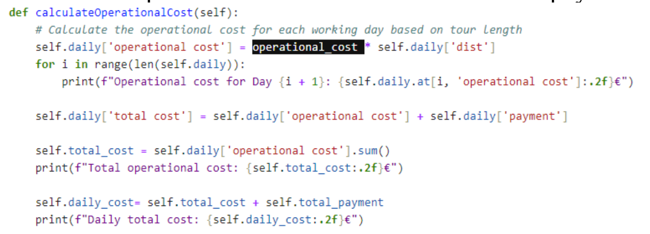
At the end of Step 6, a crucial component was the Model Verification phase, which aimed to validate the effectiveness and functionality of the newly introduced parameters within the simulation framework. This verification step is essential to ensure that the simulation accurately reflects real-world scenarios and behaves as intended based on the specified parameters. In the Simulation modified the average number of parcels expected by a customer per day: p = 0.15. Run the verification part with 4 days, 10 days, and 20 days.

Step 7A and 7B: Working time and Route Length

Since 7B subsumes 7A, did not run both. Only ran 7B at this time.

6. Operational cost: Covering electricity and maintenance is 8 cents per kilometer.

In section 5, within the Recorder class, a new function called calculateOperationalCost has been created. This function is responsible for calculating the operational cost for each working day based on the tour length and then computing the total cost for each day, which includes both the operational cost and the driver payment. Additionally, a function called finish within the same class has been modified to call the calculateOperationalCost function at the end to compute the costs. In section 10.1 of documentation, defined this parameter as operational_cost = 8*0.01/1000, which represents the operational cost per meter in Euros. And Daily total cost is sum of operational cost and driver's payment.

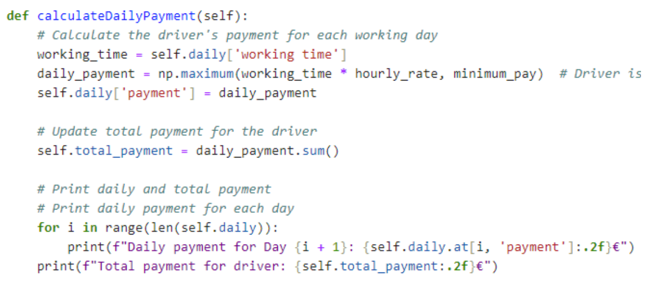


```
def calculateOperationalCost(self):
    # Calculate the operational cost for each working day based on tour length
    self.daily['operational cost'] = operational_cost * self.daily['dist']
    for i in range(len(self.daily)):
        print(f"Operational cost for Day {i + 1}: {self.daily.at[i, 'operational cost']:.2f}€")

    self.daily['total cost'] = self.daily['operational cost'] + self.daily['payment']

    self.total_cost = self.daily['operational cost'].sum()
    print(f"Total operational cost: {self.total_cost:.2f}€")

    self.daily_cost= self.total_cost + self.total_payment
    print(f"Daily total cost: {self.daily_cost:.2f}€")
```

Figure 7. Relevant section of code

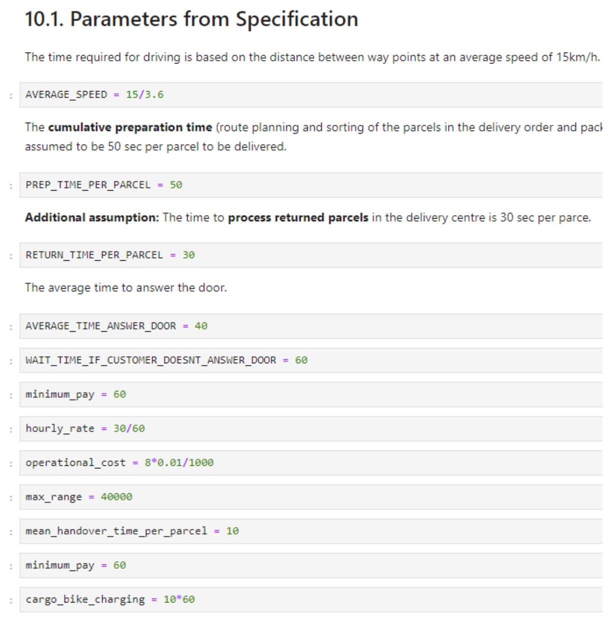
7. Driver's payment: Per hour rate of 30€ per hour, minimum payment of 60€ per working day.

In section 5, within the Recorder class, a new function called calculateDailyPayment has been added. This function is for calculating the driver's payment for each working day based on the working time. It retrieves the working time for each day, calculates the daily payment based on an hourly rate of 30€ with a minimum pay of 60€ per day, and updates the total payment for the driver. As well as, a function called finish within the same class has been modified to call the calculateDailyPayment function at the end. This ensures that the driver's costs, daily payment, are computed before finalizing the simulation. In section 10.1 of documentation, defined those parameters as minimum_pay = 60, hourly_rate = 30/60, which represents the per minutes in Euros.



```
def calculateDailyPayment(self):
    # Calculate the driver's payment for each working day
    working_time = self.daily['working time']
    daily_payment = np.maximum(working_time * hourly_rate, minimum_pay)  # Driver is
    self.daily['payment'] = daily_payment

    # Update total payment for the driver
    self.total_payment = daily_payment.sum()

    # Print daily and total payment
    # Print daily payment for each day
    for i in range(len(self.daily)):
        print(f"Daily payment for Day {i + 1}: {self.daily.at[i, 'payment']:.2f}€")
    print(f"Total payment for driver: {self.total_payment:.2f}€")
```

Figure 8. Relevant section of code



10.1. Parameters from Specification

The time required for driving is based on the distance between way points at an average speed of 15km/h.

```
AVERAGE_SPEED = 15/3.6
```

The **cumulative preparation time** (route planning and sorting of the parcels in the delivery order and pack
assumed to be 50 sec per parcel to be delivered.

```
PREP_TIME_PER_PARCEL = 50
```

Additional assumption: The time to **process returned parcels** in the delivery centre is 30 sec per parce.

```
RETURN_TIME_PER_PARCEL = 30
```

The average time to answer the door.

```
AVERAGE_TIME_ANSWER_DOOR = 40
```

```
WAIT_TIME_IF_CUSTOMER_DOESNT_ANSWER_DOOR = 60
```

```
minimum_pay = 60
```

```
hourly_rate = 30/60
```

```
operational_cost = 8*0.01/1000
```

```
max_range = 40000
```

```
mean_handover_time_per_parcel = 10
```

```
minimum_pay = 60
```

```
cargo_bike_charging = 10*60
```

Figure 9. Parameters section of code

As shown in Figure 9, All parameters are based on the specifications outlined in section 10.1.

In the end of Step 7B, I simulated 4, 20, and 40 days of simulation to see whether Daily working time and Daily tour length are normally distributed. As an illustration, a single warehouse location was randomly selected, and the plots showing the outcomes of 40 days of simulation are presented below.

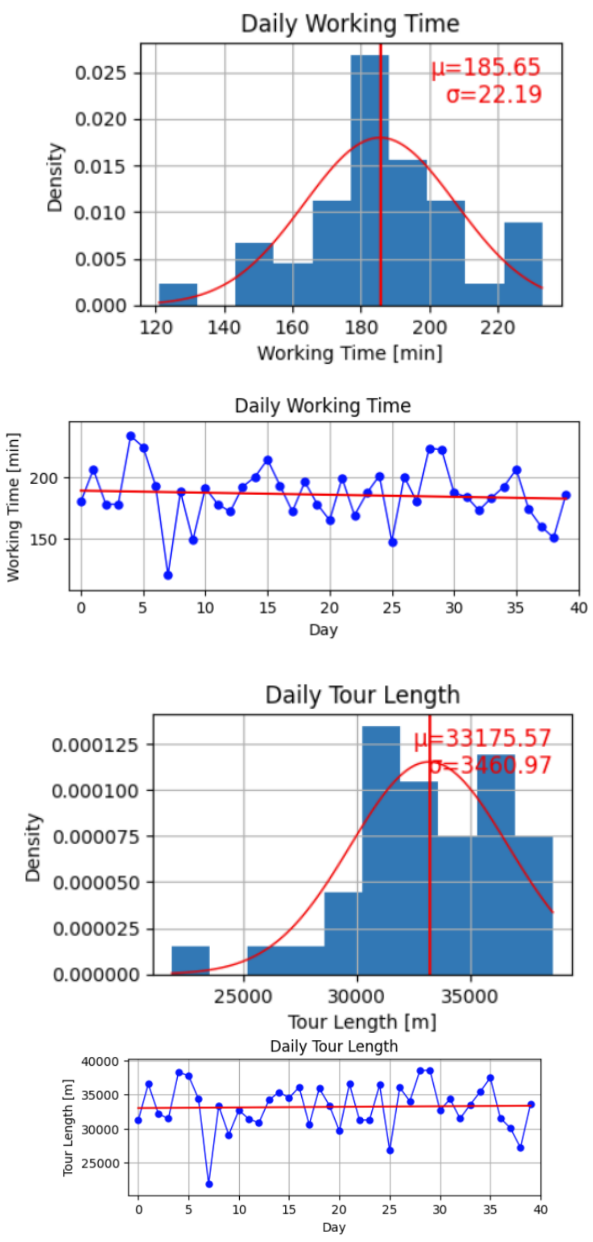


Figure 10. Plots for 40 days of Simulation

From Figure 10, As we can see that Daily working time is normally distributed. Notably, there is a significant decrease in working time observed from day 5 to day 8 compared to other days. As a consequence of the decrease in working time during days 5 to 8, it is expected that the tour length would also exhibit a corresponding decrease. Shorter working times typically result in shorter tour lengths due to reduced travel and operational activities.

From Step 1, I assessed all 10 warehouse locations through a simulation spanning 10 days to determine which location had the lowest cost base. Subsequently, I selected one warehouse representing efficient operations and another representing inefficient operations for further analysis. The aim was to determine the required sample size of simulation days to achieve statistical significance. Lowest cost based bad location was (3440, 2880), on the other hand highest cost based good location was (5680, 640).

Step 7C: Parcels Leftover

In this step, leftover parcels are measured each day in the Recorder class, as mentioned in Steps 7A and 7B, which were simulated on the same days.

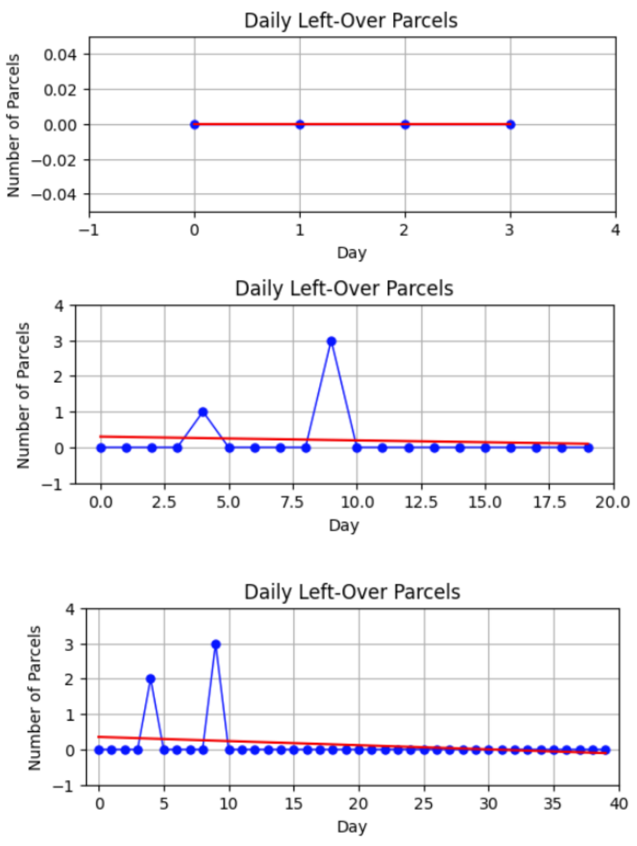
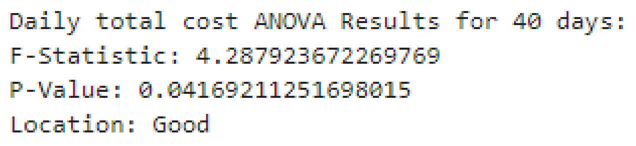


Figure 11. Plots for Leftover Parcels over 4, 20 and 40 days

As shown in Figure 11 above, as the number of days increases, some days exhibit leftover parcels, while most days do not. This indicates that parcels do not remain in the distribution center for extended periods; instead, Delivery Centre prioritized based on the order of arrival.

Step 7D: Summary of Statistics

In Section 10.5, a small simulation was created to identify a significant difference in the daily total cost, which comprises operational cost and driver's payment. As indicated in Steps 7A and 7B, only the good and bad locations were utilized for this purpose.



```
Daily total cost ANOVA Results for 40 days:
F-Statistic: 4.287923672269769
P-Value: 0.04169211251698015
Location: Good
```

Figure 12. Statistically significant days

Now that we have determined that a 40-day simulation shows a statistically significant difference in daily total cost, in the end of Step 7D, we simulated the same number of days but found no significant differences. Consequently, I decided to increase the simulation duration from 50 days by 10 or 5 days.

III. RESULTS AND INTERPRETATION

Based on comprehensive of the models and simulations, the results indicate significant findings that warrant discussion and interpretation. The statistical analyses applied to the data ensured robustness and accuracy throughout the study. Below, the key findings along with their interpretations are presented:

The duration of the simulation significantly impacts the observed outcomes, particularly in terms of daily total cost. Extending the simulation from 50 days, 70 days, 80 days, and 85 days were deemed necessary due to the absence of significant differences in the earlier duration.

Simulating different warehouse locations over varying durations led to the identification of both advantageous and disadvantageous locations based on their impact on daily total cost. This identification was crucial for understanding the efficiency and cost-effectiveness of each location in the distribution network.

The statistical analyses, including ANOVA tests, provided robust evidence of the differences observed in daily total costs between various warehouse locations and simulation durations. Flawless application of statistical methods enhanced the reliability and validity of the findings.

In Table 2 shows that each simulation day's statistical result, and relative coding part is in Step 7D, section from 10.5.1-10.5.4.

Days	F-Statistics	P-Value	Best Warehouse location
50	1.153	0.3233	(5680, 3440)
70	1.646	0.0985	(5680, 3440)
80	1.839	0.0578	(5680, 3440)
85	2.328	0.0136	(5680, 3440)

Table 2: Statistical result on all warehouse locations daily total cost

From Table 2, it is evident that as the number of days increases, the P-Value decreases. All simulations took a considerable amount of time to complete. However, the resulting optimal warehouse location was found to be (5680, 3440), and this finding is statistically significant.

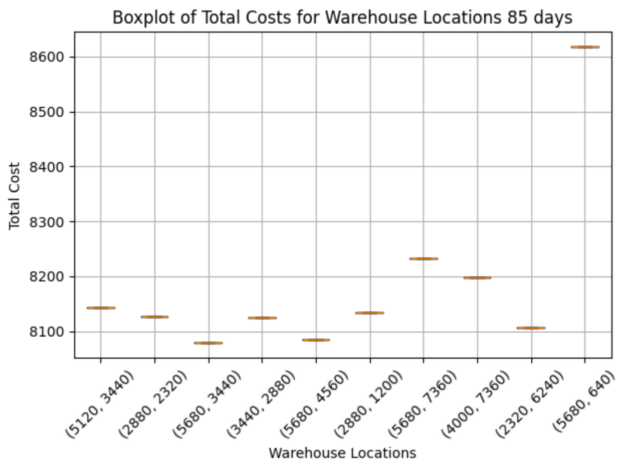


Figure 13. Total costs for Warehouse locations

From Figure 13, (5680, 3440) has lowest cost base which is 8078.22 euro, and average cost per day is 95.04 euro.

Throughout the project, several challenges were encountered, such as determining the optimal simulation duration and interpreting nuanced differences in daily total costs. These challenges prompted the implementation of parameter adjustments, to ensure the accuracy and relevance of the results.

IV. REFLECTIONS AND FUTURE WORK

Reflecting on the research conducted and the result obtained, there are several areas can be explored:

In future studies, optimizing the computational efficiency of simulations for each location is crucial, particularly as the duration of simulations grows, leading to longer running times. It is essential to investigate and apply appropriate optimization techniques to resolve this issue and enhance the simulation process. To optimize and save time in the simulation results, could be consider modifying the customer and node numbers. By adjusting these parameters, we can potentially achieve more efficient simulations without compromising the accuracy of the results. Potential strategies include leveraging heuristic algorithms, evolutionary algorithms, machine learning techniques, and metaheuristic algorithms.

Even though the models used in this study have undergone extensive verification and validation, further improvement and validation could enhance their accuracy and reliability. Future research could focus on conducting real-world validation experiments or utilizing historical data to validate model predictions against observed outcomes and explore methods for integrating real-time data on delivery routes, traffic conditions, and parcel volumes to enhance the accuracy of the simulations.

While the focus of this study was optimizing delivery center location, future research could explore alternative delivery strategies, such as car and drone delivery. Investigating these alternative strategies could provide valuable insights into their feasibility, effectiveness, and potential impact on last-mile delivery operations.

V. REFERENCES

[1] Y. Issaoui, A. Khiat, K. Haricha, A. Bahnasse, and O. Hassan, "An Advanced System to Enhance and Optimize Delivery Operations in a Smart Logistics Environment," IEEE Access, vol. PP, pp. 1-1, 01/07 2022, doi: 10.1109/ACCESS.2022.3141311.

[2] Y. Lin, A. Chen, S. Zhong, V. Giannikas, C. Lomas, and T. Worth, "Service supply chain resilience: a social-ecological perspective on last-mile delivery operations," International Journal of Operations & Production Management, vol. 43, no. 1, pp. 140-165, 2023.

[3] Y. H. Lin, Y. Wang, D. He, and L. H. Lee, "Last-mile delivery: Optimal locker location under multinomial logit choice model," Transportation Research Part E: Logistics and Transportation Review, vol. 142, p. 102059, 2020.

[4] L. G. Birta and G. Arbez, Modelling and Simulation: Exploring Dynamic System Behaviour, 2nd ed. New York, NY, USA: Oxford University Press, 2013.

[5] F. Glover, M. Laguna, and R. Martí, Handbook of Metaheuristics. New York, NY, USA: Springer, 2019.

[6] S. Luke, Essentials of Metaheuristics. Fairfax, VA, USA: George Mason University, 2013.

[7] S. Cagnoni, Evolutionary Optimization Algorithms: Biologically Inspired and Population-Based Approaches to Computer Intelligence. New York, NY, USA: Wiley, 2013

[8] X. Shi, W. Liu, and J. Zhang, "Present and future trends of supply chain management in the presence of COVID-19: a structured literature review," International Journal of Logistics Research and Applications, vol. 26, no. 7, pp. 813-842, 2023.

[9] S. Alfano, A. S. Brettone, M. C. Groccia, R. Sia, and A. F. Vita, "Optimizing the order picking process via simulation: a case study," Jan. 2022. doi: 10.46354/i3m.2022.mas.017