Cargo bike delivery Simulation for Shipping packages.

Sarthak Bhatnagar
MSc. In Data Analytics
(School of Computing)
National College of Ireland.
Dublin, Ireland
X21185352@student.ncirl.ie

***Abstract*—In today's digital era, Retail businesses are facing cutthroat competition in the market for being best retailer of choice for its customers. The prime focus for business leaders is to sell products and services to maximize profits with minimal expenditure costs incurred. Retail businesses know that cost-effective techniques are important and can be attained by proper planning their logistics and transportation management. So, this project is designed an imitation of simulation model for "delivery-service" company "We-Doo". The objective is to carry out simulation scenarios of deliveries by cargo agents keeping business constraints in mind. The designed simulation is used to scrutinize the business model for optimal solution and cost effectiveness.**

Keywords— TSP, Simulation, Modelling, Optimal Routes, Shortest path algorithms.

I. INTRODUCTION (*HEADING 1*)

In the era of technological developments, item delivery has become one of the important sectors for people as their purchasing behavior has changed. The business companies are quite focused on the delivery system as it is an essential part of their management to keep the logistics department smooth and cost effective. To attain cost effectiveness, companies which contribute to delivery-based systems face a lot of problems in the process of item delivery. One of the common examples is time and distance that are not optimal with the priority of an item delivery. The industry is now looking for new ways to innovate the delivery process to increase efficiency and manage constraints like time, distance and priority in more business-oriented way.[10]

The research project is designed to address the delivery industry's bottlenecks by integrating cargo bikes and micro-depots into the parcel delivery process for "We-Doo" company. A creative plan for distributing city parcels using single delivery method has been presented and is based on the simulation models. The task has been assigned to simulate the delivery runs process over the period of 22 days (five days per week) in the evening time. The cargo rider here delivers the parcel from micro-depot to customer locations to maintain all constraints and business orders.

The company operates on several constraints which is a part of their business model. They shared a map of town "M" with the micro-depot "D" showing the location of the local distribution point. The customers acted in the scheme are denoted as "C" with their delivery addresses. The distribution of average daily delivery is denoted "N" extracted by the total sum of parcels a customer is expected. The total number of small parcels delivered per day (average daily delivery N=10) follows given Poisson distribution curve given. The Poisson process plays a very crucial role in the drop-off of each parcel such that the monthly distribution of the number of parcels generated for each customer and the average number of parcels arriving per working day are both equal. The company is focused on distribution of the parcels and should be fast paced to the final customer once it has arrived at their micro-depot. However, this goal depends upon the cargo bike riders which could be negotiable as per the conditions. Moreover, multiple parcels of a single customer should be delivered in one go, if arrived at the distribution centre.

To design the simulation model, we first created synthetic for testing and evaluating the performance of algorithm using generatedata() function. The generated data used seed.random command with the attribute value of "5352" to make sure the results can be reproducible for the person who re-runs the code in future with the same value. The synthetic data generated use several parameters and generates a map with random seed value "5352" which consist of 50 nodes, 100 as number of customer locations, and number of total deliveries in this case 423 over the period of 22 days (about 3 weeks).

The research report is structured in the following manner. Section I includes an introduction of the relevant topic. In Section II, the related work provides relevant studies conducted in the research. Section III discusses methodology opted to present the work and its stepwise significance. Section IV describes the result and the process of verification and validation used in simulation and Section V, which is about the improvements and problems which can be solved in future.

II. RELATED WORK

(Wei, Shun'an,Shen) proposed a study on optimized multi travel salesman problem which is based on improved simulated annealing problem, works on the constraints of minimum travel quotient, the path balance among travel agents, and the time window traversing the city. It is an extended version of typical NP hard problem, used to solve the MTSB problem rely on constraint of the minimum sum of all travel salesman paths. The author is successfully carried out results to not only get the shortest mileage, but also to get salesman to his designated position at right time with the help

of annealing model which has the capability to provide fast convergence speed and optimal solutions.[1]

The research paper studies to optimize the travel salesman problem and efforts are made to refashion the old Genetic algorithm with parent selection in randomized bias which has the capability to identify optimal relations in efficient CPU running time. GA is s metaheuristic technique which is inspired by the principle of natural selection and based on 3 components: selection, crossover, and mutation which is responsible for selection of best individuals out of the pool. The performance of both algorithms is done on 30 TSP problems, and it is proved Random Bias genetic algorithm used bias selection of parent mating to generate an individual performed far better than a genetic algorithm.[2]

The research paper studies Ant colony algorithm for TSP problems and provides analysis on premature stagnation phenomenon of standard ACO. The author is optimistic that if the strategy on information hormone is modified, and local optimal search is used in combination, it will produce an effective premature stagnation phenomenon in updating velocity of converging process. Thus, it acts as an improved algorithm for optimal TSP problems. In the conclusion, (Xue and Jie-sheng) proved that manipulating the aforementioned factors can improve the algorithm and accelerate the convergence rate with global optimal results.[3]

(Li, Yu, and Fang) proposed an ant colony optimization for generalized TSP problem which combines 2-opt algorithm, to minimize the total path, the author assigned the balanced task for different travelers. The focus in the research paper is to improve the accuracy and stability for GTSP problem by modified ant colony algorithm. It is an effective algorithm to identify the optimal path on the premise of bisectional routes.[4]

(Guangqiang et al.) proposed an improved imperialist competitive algorithm (IICA) which modifies the shortcomings of traditional continuous ICA algorithm suitable for solving combinatorial optimization problems. They proposed four performance boosting parameters which make it a suitable intelligent algorithm to solve TSP problem. The original algorithm is first discretized to ensure it will work in TSP problems, employed immune fitness-based mechanisms, parallel computational strategy and adaptive mechanism are used to promote convergence and increase its rate. The concept of simulated annealing is employed to ensure algorithm can accept the worst solution of probability, fall out of local minima, and carry out global optimum with more probability. In conclusion, it is supposed to be intelligent IICA algorithm which produce faster convergence speed and accuracy to complex travelling salesman problem.[5]

In this research paper, an artificial immune algorithm is introduced along with a hybrid swarm intelligence method that combines a particle swarm algorithm and an artificial bee colony. It has strong capabilities for global searching and superior search convergence, which help it overcome the issue of large-scale TSP apt to become locked in local optima. The simulation experiment data comparison demonstrates that the modified hybrid algorithm searches produce better results than similar algorithms, and it has acceptable outcomes when applied to large-scale TSP. [6]

The research proposed by (Changyou et al.) studies the shortcomings of Sparrow search algorithm in TSP problem due to insufficient stagnation and propose a new greedy genetic sparrow search algorithm hinged on sine and cosine search strategy and named it as "GGSC-SSA". At first, the greedy algorithm is employed to initialize and increase the diversity of the population, genetic operators are utilized to balance the capacities of local development and global search for the population and finally, the producer update includes the adaptive weight to improve the algorithm's adaptability and the quality of the answer. To increase the effectiveness of the above algorithms, sine and cosine functions were employed to act as moochers. This technique is compared with every other algorithm used in TSP problems, but its result significantly enhances the simulation's precision, optimization, and robustness.[7]

III. METHODOLOGY

The defined parameters and their notations are mentioned below which is used in methodology.

“M” - Map of the town.
“W” - Warehouse/Micro-depot.
“p” - Avg. Number of parcel generated for customers.
“C” - Customer participated in the scheme.
“V” - List of vertices.
“E” - List of edges.
“D” - List of delivery data.
“days” - Number of days for data to be generated.
“N” – Avg. Daily Delivery capacity

.

A. Generate Synthetic Data for Simulation training-

This is an initial step for the simulation process where we imported several but crucial libraries for simulation such as matplotlib, Pulp, math, random, etc. Pulp is one of the important libraries imported in this module. Pulp is popularly known and used for solving optimization problems which are based on specifically linear and mixed integer programming. It provides a prominent level of interface with intuitive syntax. Pulp has wide variety of applications such as supply chain optimization, production scheduling, resource allocation, etc. Another library which is helpful in dealing complex mathematical functions is "Math" library, particularly useful in solving complicated math problems such as log, sine, and cosine, and so on.

In this phase, we have given a map "M" with vertices "V "and edges "E" of the town with the radius of 10km. The generated map uses graph like structure where line function is created so that vertices and edges connected to each other which act as connected route or path. Another function is created to incorporate Warehouse/depot "W" point in the map from delivery is executed in the evening time. The first step is to generate synthetic data using generatedata() command and it is often implemented for simulation purposes. It generates data to simulate the behavior of a system used for testing or training. One of key functions we used to generate the random data is "Random-seed" because it allows the generation of reproducible code with same seed value.

The random seed value used in this scenario is “5352” for the generation of map which will provide 50 nodes, 100 customer locations and 423 deliveries in the town over 22 days. The customer location is marked with "Green" and micro-depot/warehouse local location with "Blue".

```
Generated map with 50 nodes, 100 customer locations, and
delivery data for 423 deliveries over 22 days.
```

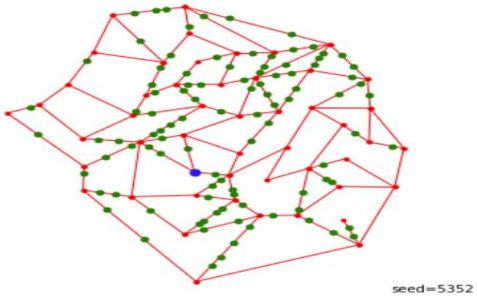

Fig 1. Town map generated with random seed “5352”.

B. Finding the shortest path -

In this step, we intend to find the shortest path between the micro-depot to the customer location for item delivery within the range of 30km. The cargo-bike rider must follow the optimized route and deliver items along the way. To identify the best feasible path as per constraints are calculated using A* search algorithm which is a variant of Dijkstra's algorithm that uses heuristics to prioritize exploration of paths that are more likely to be optimal. The map function is defined to load the map in the form of a graph where each node has a list of its neighboring nodes and the cost to traverse each edge. The seed value used is "5352" to generate graph and print the coordinates of warehouse, customer locations, respectively.

```
def shortestPath(Map, A, B):

    def h(p):
        return pathLength(p)+dist(p[-1],B) #heuristic function with equal value for all nodes.

    # candidates C are pairs of the path so far and
    # the heuristic function of that path,
    # sorted by the heuristic function, as maintained by
    # insert function
    def insert(consumer, p):
        hp = h(p)
        C = (p, hp)
        for i in range(len(c)):
            if C[i][1]>hp:
                return consumer[:i]+[c]+consumer[i:]
        return consumer+[c]

    V, E = Map
    assert(A in V and B in V)
    consumer = insert([], [A])

    while len(consumer)>0:
        # take the first candidate out of the list of candidates
        path, _ = consumer[0]
        consumer = consumer[1:]
        if path[-1]==B:
            return path
        else:
            for (x, y) in E:
                if path[-1]==x and y not in path:
                    consumer = insert(consumer, path+[y])
                elif path[-1]==y and x not in path:
                    consumer = insert(consumer, path+[x])
    return None
```

Fig. 3 A* Search algorithm.

The A* algorithm is also known as informed search algorithm because it keeps track of the actual cost from the start node and the previous node in the shortest path using dictionaries. In order to make the method function, a priority queue of nodes to be examined is kept, with the node with the lowest estimated cost being searched first.[11]

The start node, the goal node, and the graph are the inputs for this function. It adds the start node and its projected cost to the priority queue as initial values. To maintain track of the real cost from the start node to the previous node in the shortest path, it additionally initializes the dictionaries. Pop the node with the lowest predicted cost and investigate its neighbours while the priority queue is still full. Reconstruct the path from the start node to the goal node using the dictionaries if the goal node is located. If not, update the dictionaries and add the nearby nodes with their projected costs to the priority queue.

```
random.seed(5352)
for i in range(5):
    [A, B] = random.sample(V, k=2)
    P = shortestPath(Map, A, B)
    plotMap(Map, T=[A, B], P=P)
```

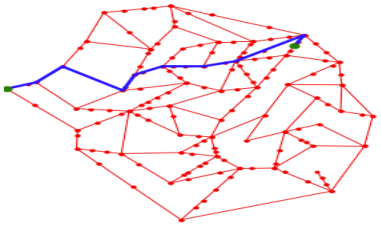

Fig 4. Distance between two point using shortest path.

C. Shortest path for Delivery routes-

A roundtrip function is created which generates a set of trips that cover all the customer locations in the graph. The main task is to create a binary decision variable parameter which indicates whether an edge exists between two nodes or not. This problem is solved using Linear programming in which it generates a loop by following the shortest path between each pair of consecutive customer locations in the solution. Moreover, the function creates the set of trips using a method referred to as greedy algorithm, where each trip begins at a node that has never been visited before and proceeds until it either returns to the beginning node or visits every last unvisited node. The trips are then returned after being sorted by length (in decreasing order).

```
In [33]: plotMap(Map, T=T, w=warehouse, P=P)
```

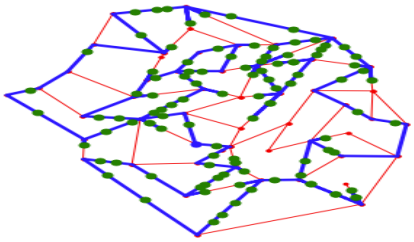

Fig. 5. Delivery routes covered using Greedy approach.

D. Entity classes and Scenarios for Simulation.

(a) Class recorder:

The class "Recorder" is defined in this function to record and store the data during simulation. It is a common method used to track and record various events and states of simulation at time of interval. In this process, we captured the events by passing the variables for Map, warehouse, customer, number of days, and current time interval.

```
class Recorder:

    def __init__(self, env, M, W, C, days,
                 log=False, plot=False):
        self.env = env
        self.M = M
        self.W = W
        self.C = C
        self.days = days
        self.log = log
        self.plot = plot
        Customer.REGISTER = []
        Parcel.REGISTER = []

    def trace(self, event):
        if self.log:
            print(timestamp(self.env.now), event)
```

Fig. 6 Class Recorder.

(b) Class Parcel:

The Parcel class is created to capture the delivery parameters such as "rec" for recorder, "cust" for customer class. This class is incorporated to capture the track and states of parcels, once it is out for delivery. The journey of Parcels from processing state to delivered state to the customer has been captured using this class object.

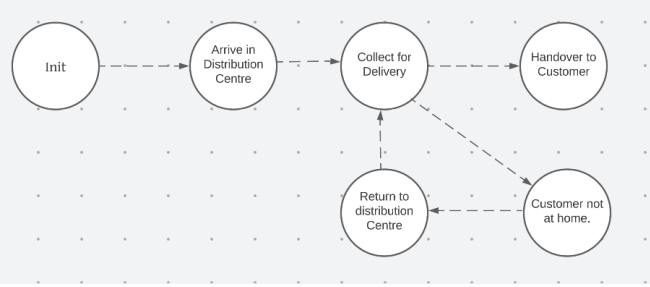

Fig 7 Process for Class Parcel.

Once the order is made by the customer, the parcel is sent to the processing state where companies pack the item and send the item to the town's micro-depot. Once it arrives at the Distribution centre, it is the job of the delivery guy to sort and prepare the parcel for delivery, which should be 50s. As per the planned and optimized route the delivery guy has responsibility to call and deliver the parcel to customer within 60s. Now, there are two scenarios based on availability of customer. If the customer is available, the parcel is handed over to the customer or if the customer is not available the parcel is returned to the micro-depot and delivery attempt is made some other time.

(c) Class Customer:

The customer class incorporated all the parameters associated with the customers required to run and record simulation process. The rec and location parameter are used in all the mentioned scenarios to capture the log of events at the specific time stamp.

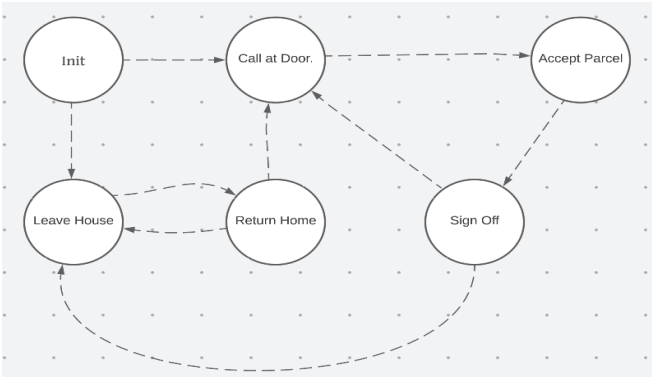

Fig. 8 Process for Class Customer.

(a) The delivery guy is out to deliver the package to the customer and calls the customer for delivery, customer is available at its location to receive the package and package got delivered.

(b) The delivery guy calls the customer for package delivery, but the customer is not available at home. So, he denied the parcel, the parcel sent back to the distribution centre.

(c) The delivery guy calls the customer and updates him to receive the parcel, the customer gives confirmation of his availability but does not accept the parcel and keeps his door closed. Thus, the parcel is sent back to the distribution centre.

(d) In this scenario, the customer is not available at its location, but when the delivery guy calls him to check his availability, the customer agrees and returns to his home to receive the parcel.

(d) Class Driver:

The process initially starts until the beginning of the workday, after which it begins a loop that lasts until the simulation is finished. The driver enters the delivery center for work, is given a tour and a list of packages to deliver, and then leaves for delivery in each iteration of the loop. The driver then makes a delivery to each customer in their tour, makes a house call, and drives to their location to pick up any packages. The driver waits for 1 min, if the customer is not home before going on to the next client. The driver leaves the delivery center once all the packages have been delivered and logs a note stating that they have returned from delivery. Overall, the Driver class is an important element of the delivery service simulation since it imitates all aspects of the interactions of a delivery driver with clients and the delivery centre.

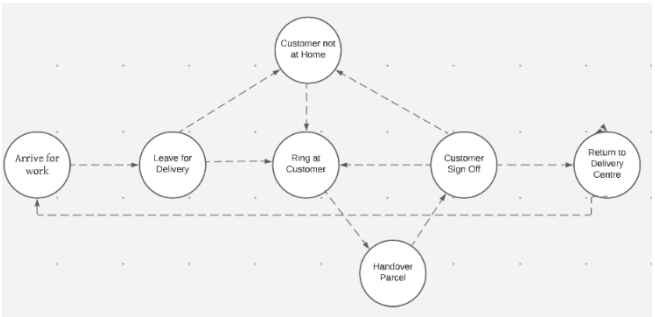

Fig. 9 Process for Class Driver.

(e) Class Delivery Centre:

The Delivery centre class and defined methods are briefly explained in this section. The classes are designed specifically to imitate the process followed by Distribution centres in the town. The "Init" method is created to initialize the delivery centre with attributes associated with it such as its location, list of packages, leftover packages, and customer destination. The "__accept" method is created to access the information related to parcel delivery/arrival with timestamp. But if the parcel is not accepted by the customer, it is added to the leftover parcel list. These methods are explicitly incorporated in this method which confirms the real time scenario. Another two most necessary methods are "sendForDelivery" which sends the parcel to delivery's centre list to plan the tour for delivery and other one is "returnFromDelivery" which is inculcated if the package is not received by the customer. So, it will send the package information back to "sendForDelivery" method to plan the tour for next consecutive day.

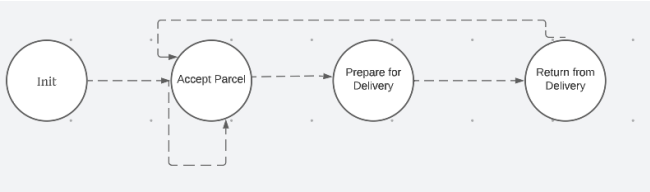

Fig. 10 Process for Class Delivery centre.

E. Simulation.

The Simulation test on package delivery distribution system with seed values od "5352" are shown in the figure below.

```
random.seed(5352)
simulation(M, W, C, D)
```

```
Simulating delivery of 398 parcels over 22 days to 100 customers
[ 0] 08:00:00.0 Customer:  0 (4645, 8813) leaves house
[ 0] 08:00:00.0 Customer:  1 (3431, 7640) leaves house
[ 0] 08:00:00.0 Customer:  2 (3099, 6435) leaves house
[ 0] 08:00:00.0 Customer:  3 (2720, 5820) leaves house
[ 0] 08:00:00.0 Customer:  4 (6776, 3541) leaves house
[ 0] 08:00:00.0 Customer:  5 (3931, 6210) leaves house
[ 0] 08:00:00.0 Customer:  6 (4354, 3805) leaves house
```

Fig. 11 Simulation for package delivery system.

IV. Result and Evaluation

The goal of this study was to recreate a study of the normal delivery runs that take place at night for a small township in order to evaluate their business plan. The recorder class's dataframe will be used to find the findings. The 'daily' variable keeps track of the delivery agent's beginning and ending times as well as the total daily mileage. Four criteria are used to evaluate the simulation model:

(a) Question 1: The distribution of the length of the delivery route per day: The simulation test depicts the distance covered by the delivery guy in the period of 22 days is 29.9km a day which is almost 30km in 1 day. This demonstrates the viability of the business strategy to use cargo bikes for deliveries in the small township in terms of daily miles is in sync with business model.

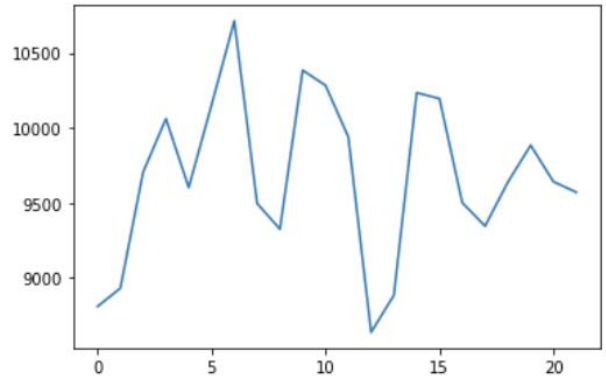

Fig 12 Distribution of the length of the delivery route per day.

(b) Question 2: The distribution of the working time per day for the driver (in minutes): In this graph, the x-axis depicts number of days and y-axis shows the time taken by the delivery agent to drop the package from warehouse/micro-depot to the respective customer locations. This graph also provides an understanding of the working hours of a delivery agent which is 3 hours a day.

```
min(rec.daily['time'])
```

```
8638
```

```
max(rec.daily['time'])
```

```
10714
```

```
np.mean(rec.daily['time'])
```

```
9679.681818181818
```

```
_ = plt.hist(rec.daily['time'])
```

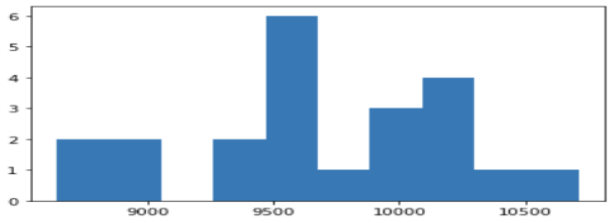

Fig 13 The distribution of the working time per day for the driver (in minutes)

(c) Question 3: The distribution of the number of parcels left over for the next day (per day): The histogram in the figure shows the number of parcels left for delivery on the next day. This graph provides insights of delayed deliveries, wait time from the customer side and helps us to optimize the package scheduling and allocation.

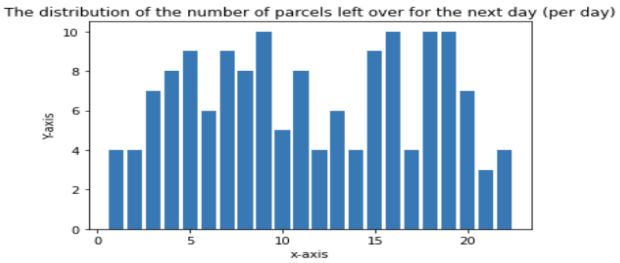

Fig 14 The distribution of the number of parcels left over for the next day (per day)

Question 4: The distribution of the delay time of the parcels (in days) as histogram: The delay time for the packages is shown in the chart below. It provides insight into the performance of the delivery agent and helps with delivery operations efficiency to reduce parcel delays and enhance customer satisfaction.

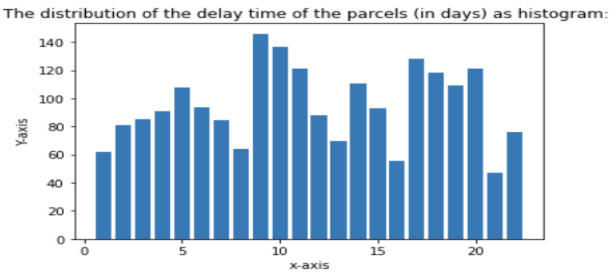

Fig 15 The distribution of the delay time of the parcels (in days) as histogram.

V. Reflections and Future Work.

(a) The project designed for simulation of package delivery system has shown significant results but the the delivery operations are considered under static conditions. There are several real time factors which can affect the simulation results are weather condition, traffic congestion and road closures which can be improved using dynamic factors like incorporating real time traffic tracking, implementing weather API in the system, etc.

(b) The "nearest neighbor" technique was employed in your simulation to determine the driver's shortest delivery path. There are, however, a variety of sophisticated algorithms that can optimize delivery routes and cut down on both delivery time and expense such ant colony optimization and Simulated annealing.

(c) Simulation can be used to evaluate and improve how new technologies like autonomous drones are used in package delivery systems as they become more common in the delivery sector and also helps to reduce carbon emissions.

References

[1] W. Xia, G. Li and Y. Zeng, "Research on multi-traveling salesman problem based on simulated annealing algorithm," 2022 International Conference on Computers, Information Processing and Advanced Education (CIPAE), Ottawa, ON, Canada, 2022, pp. 390-393, doi: 10.1109/CIPAE55637.2022.00088.

[2] I. K. Gupta, A. Choubey and S. Choubey, "Randomized bias genetic algorithm to solve traveling salesman problem," 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Delhi, India, 2017, pp. 1-6, doi: 10.1109/ICCCNT.2017.8204127.

[3] X. Yang and J. -s. Wang, "Application of improved ant colony optimization algorithm on traveling salesman problem," 2016 Chinese Control and Decision Conference (CCDC), Yinchuan, China, 2016, pp. 2156-2160, doi: 10.1109/CCDC.2016.7531342.

[4] L. Meng, Y. Lin, S. Qing and F. Wenjing, "Research on Generalized Traveling Salesman Problem based on Modified Ant Colony Optimization," 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 2019, pp. 4570-4574, doi: 10.1109/CCDC.2019.8833167.

[5] G. Li, C. Xu, Y. Wang, W. Dong and D. Zhu, "An Improved Imperialist Competitive Algorithm For Solving Traveling Salesman Problems," 2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE), Dalian, China, 2021, pp. 311-315, doi: 10.1109/CACRE52464.2021.9501295.

[6] Y. Wang, "Improving Artificial Bee Colony and Particle Swarm Optimization to Solve TSP Problem," 2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS), Hunan, China, 2018, pp. 179-182, doi: 10.1109/ICVRIS.2018.00051.

[7] C. Wu, X. Fu, J. Pei and Z. Dong, "A Novel Sparrow Search Algorithm for the Traveling Salesman Problem," in IEEE Access, vol. 9, pp. 153456-153471, 2021, doi: 10.1109/ACCESS.2021.3128433.

[8] https://smartroutes.io/blogs/the-travelling-salesman-problem/

[9] https://www.upperinc.com/guides/travelling-salesman-problem/

[10] H. N. Ginting, A. B. Osmond, and A. Aditsania, "Item Delivery Simulation Using Dijkstra Algorithm for Solving Traveling Salesman Problem," Journal of Physics, vol. 1201, no. 1, p. 012068, May 2019, doi: 10.1088/1742-6596/1201/1/012068.

[11] https://stackabuse.com/courses/graphs-in-python-theory-and-implementation/lessons/a-star-search-algorithm/

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.