MATH 406 FINAL PROJECT REPORT: PROBLEM B

Chenglin Zhang
cz155@duke.edu

Yijia Xue
yx179@duke.edu

Mengfan Gong
mg442@duke.edu

Keywords Pick-up and Delivery · Routing Problem · Optimization · Graph Theory · Dynamic Programming · Floyd-Warshall Algorithm · Monte Carlo Simulation

Abstract

It is crucial to investigate the optimal pick-up and delivery strategy for the delivery-food riders given the boom of the delivery-food market. Higher delivery efficiency will not only benefit the food delivery company but also the customers. For the companies, they can achieve higher revenue with the same amount of resources if they utilize a better taking and delivery strategy. On the customers' side, they can get their food faster, which is also an improvement of their overall well-being.

In this problem, we investigate the food delivery market in a specific region of Yushan Town, Kunshan City. Particularly, we are asked to construct an optimal pick-up and delivery strategy model for the 100 delivery-food riders $P_1, P_2, \cdots, P_{100}$ in the closed region in the given time interval from 11:00 AM to 2:00 PM so that the revenue is maximized. It is a typical **routing problem** that involves **the Graph Theory** and **Dynamic Programming algorithms** related to **the Shortest Path Problem**.

More precisely, we first construct a **weighted undirected cyclic graph** V, E with 120 vertices V_i $(1 = 1, 2, \cdots, 120)$ and a bunch of edges E connecting the vertices as a representation of Yushan Town. 40 vertices are randomly selected as restaurants that can generate orders $R_1, R_2, \cdots, R_n$, in which n is the total number of orders that are generated in the given time interval. 40 other vertices are also randomly selected as communities that can generate delivery destinations where customers can get their food. The destinations $C_1, C_2, \cdots, C_n$ are matched with the orders R_i. For example, order R_{35} generated in the vertex number 12 should be delivered to destination C_{35} generated in the vertex number 48.

We first used the **Enumerate Method** to list the possible paths and find the optimal of it, while also used **prune method** to decrease the computation complexity. We then used the **Floyd-Warshall Algorithm to compute the All-Pair-Shortest-Path (APSP)** [1] and then used it to get the distance of different choices of paths. Then, we introduced the **Permutation of paths** so that the rider can pick one of them which distance is the shortest and traverse it. We also introduced the credit score s_i $(i = 1, 2, \cdots, 10)$ ranging from 0 to 10 for each delivery food rider, that failure to deliver the order in 45 minutes will cause a 1-point-deduction in the credit score and will have negative impact on the riders.

Then, we employed the **Monte Carlo simulation** to test the performance of the last two different pick-up and delivery strategies, as well as the situation of weather changes and get the statistics. We find that for strategy II, the mean number of finished orders in the three hours from 11:00 am to 2:00 pm is about **1315**, while for strategy III, the mean number is about **1380**. For strategy III, the number of canceled orders and average order delivery time also decreases.

Further, we proposed that our model can be optimized in the assigning algorithms of the orders, and also by taking more factors into account.

Contents

1 Problem Introduction

1.1 Importance, Motivation & Main Questions

With the rapid development of the Internet, the takeaway market has become one of the most focused markets in China's catering industry, especially the takeaway platform.In the past two years, driven by the digital transformation of the service industry and online shopping consumption, the employment scale of online delivery personnel, including delivery riders, has grown rapidly.According to the "2020 China Instant Delivery Industry Development Report" released by Meituan in 2021, the scale of instant delivery consumers in China in 2020 reached 506 million,which is an increase of 85 million compared with 2019, with a year-on-year growth of 20.19.The report also showed that more than 3.7 million takeaway riders earned income from Meituan delivery in 2020. However, the topic of "high-risk occupation" of takeaway riders has been on the news for many times, and the number of traffic violations of takeaway riders released by traffic law enforcement departments is rising rapidly in these years. Meanwhile, the delivery platform is rational, they want to maximize their profits, which means that they hope the riders can deliver more food takeaways. Motivated by the gradually expanding takeaway market scale and the balance of interests of the platform, riders, customers and merchants, the delivery problem is of great importance to investigate.

Figure 1: Image of takeaway riders

The takeaway delivery orders have two demand points: food to be picked up and food to be delivered, which appear in pairs. Each demand point has strict time window requirements; The nodes to be picked up and delivered have a strict access order, that is, they must first go to the merchant to pick the meal, and then they can deliver the meal to the corresponding customer. If one rider is assigned too many orders at once, then he will be likely to being late in some orders, which harms the reputation of both himself and the platform. Meanwhile, if the platform want to maximize their profits, they want the riders deliver orders as much as possible and without time-out in each order. Then, how to achieve the balance in the above situation and efficiently determine the taking order and delivery schemes of the riders is the main question this paper addresses.

1.2 Summary of the Methods In the Study

In this paper, we develop the model to optimize the food distribution strategy and simulate the results. we abstract the positions of restaurants and customers into an undirected map. Then, the optimal food allocation problem is turned into a shortest path problem. We implemented the **Floyd-Warshall algorithm** to solve the shortest path problem and permutation method to find the optimal solutions.

Finally, we use the **Monte Carlo Simulation** to validate our proposed model. The detailed introduction for each algorithm can been seen in the Methodology section.

2 Literature Review

The "food distribution" problem is a class of **pickup and delivery problems (PDPs)**. It is about optimal path planning, in which goods must be transported from different starting points to different destinations [2]. Depending on the type of demand and route structure, PDP problems can be classified into three main categories, "many-to-many" problem, "many-to-one" problem, and "one-to-one" problem.

Current research related to the "food delivery" problem focuses on the PDPTW problem (pickup and delivery problem with time-windows), which considers multiple riders and multiple customers, where each customer's order contains a pickup point and a delivery point, and the riders visit each node in a prescribed order to complete the service of the order so as to achieve the optimality of some objective function.

Liu[3] developed a dynamic demand-based VRP (vehicle routing problem) model to solve such problems. For the demand of real-time arrival, they decompose the dynamic problem into multiple static problems. The initial solution is obtained with an initial insertion algorithm for each solution, and an improved variable neighborhood search algorithm is designed to improve the initial solution. However, the overall consideration of the equilibrium of the riders' task is lacking.

Mathematician R.E. Bellman [4] first proposed using dynamic programming method to solve the decision process in the optimization problem, along with the famous optimization principle, by dividing the large problem into a number of small stages, and then make each small stage optimal by different decisions. In this case, the whole decision-making process is made optimal as a whole.

Moreover, with the booming development of e-commerce and the increasing demand for efficient distribution, the study of dynamic vehicle path problems for distribution services has received increasing attention. Unlike the static vehicle path planning problem, the information in dynamic planning is uniquely updated over time. Ting et al. [5] provided a review of dynamic vehicle path problems and proposed the selective pickup and delivery problems (SPDPs), where vehicles do not collect goods from all pickup points and therefore do not have to visit all nodes in the PDP. Subsequently Ting et al. used multiple vehicles to improve the distribution efficiency and solve the multi-vehicle selective distribution problem (MVSPDP). Zhou [6] developed a multi-objective dynamic mathematical planning model considering dynamic demand, vehicle sharing, time window, and customer satisfaction, and proposed a two-stage solution strategy. In the first stage, the multi-objective hybrid particle swarm optimization algorithm is used to obtain the Pareto optimal solution in the pre-optimization stage, and the adaptive lattice technique is used to maintain the distribution of the solution. In the second stage, greedy interpolation and variable neighborhood search are used to adjust the path in real time for customer demand changes. However, the above algorithms cannot be applied to the problems in the distribution scenario because of the extremely high requirements for algorithm timeliness in the real distribution scenario.

Wang et al. [7] studied a food delivery route planning problem (FDRPP), in which a driver delivers multiple orders from a restaurant to a customer. The immediate nature of the delivery task provides very limited computation time for generating a satisfactory solution. To ensure the proper operation of the delivery system, the driver needs to consider order assignment and delivery route planning in his decision making. The system requires order assignment, which assigns orders to the appropriate drivers, and a route planning system that organizes a feasible route for each driver. They proposed an Extreme Gradient Boost-enhanced (XGBoost-enhanced) Fast Structured Algorithm (XGB-FCA) to solve the problem. To construct the complete path, an insertion-based heuristic algorithm and different ranking rules are used, and a geographic information-based acceleration strategy is employed to speed up the insertion process. Experimental results on the "Meituan take-out platform" dataset show that XGB-FCA saves a large amount of computation time with guaranteed solution quality. However, since the method requires a large amount of data for the machine learning process, this makes the method impractical sometimes due to the lack of data.

Considering the limitations of previous models in terms of algorithms and practical scenario applications, we will combine a static model with a dynamic model based on the actual number of merchants and communities in Yushan Town to develop an algorithm that increases the total merchant's revenue by maximizing the number of orders delivered by each rider within a specified time. We focus on achieving dynamic selection of the optimal path, which means the rider who currently has the least number of orders are assigned to pick up within a certain distance, and continuously refresh the decision process to achieve the overall optimal decision to pickup and deliver food. We introduce three different strategies to plan the shortest path for each rider. The simulation results verify the effectiveness of the strategies in Yushan town. Thus, our model is meaningful, but still has some limitations that may require further study.

3 Methodology

3.1 Problem analysis

We will develop a mathematical model to implement an optimal food distribution strategy for a company and simulate the results. Therefore, it is important to refine the problem into a mathematical optimization problem.

Our main objective is to maximize the total revenue for the food distribution company. Since we assume that the food company receives the same revenue from each order, we denote this value by o. Let R be the total number of orders received and delivered by riders to customers during the peak hours of 11 a.m. to 2 p.m. Then, our problem can be transformed into a constrained optimization problem:

Max oR subject to a specified time (45 min).

Since o is a constant, our main objective becomes to find a way to maximize R (i.e., the total number of orders delivered by a rider per unit time). To facilitate this problem, we assume that the average speed of riders on the road is always the same, and max$R$ is only related to the delivery path chosen by each rider.

3.2 Graph Theory

By observing the satellite map of Yushan town, since the locations of each merchant and customer could be connected in both directions, we try to abstract their relative positions into a weighted undirected map.

An undirected graph can be represented by $G = \{V, E\}$, where V stands for vertex, which is any reachable node, and E stands for edge, which is the path between nodes. In a graph, a path is a series of vertices connected sequentially by edges. A vertex is said to be connected to another vertex when there is a path between two vertices that connects both sides.

According to the actual situation of the distribution range of take-out merchants and customers in Yushan Town, we regard the locations of merchants and customers as nodes on an undirected graph, and the paths that riders can choose as edges on the undirected graph. According to the assumptions of the model, the locations of merchants and customers can be regarded as two interconnected undirected graphs $G_{merchant} = \{V_{merchant}, E_{merchant}\}$ and $G_{customer} = \{V_{customer}, E_{customer}\}$. When delivering, the two graphs have some logical correspondence because the rider needs to get the take-out from the corresponding merchant. In the customer location distribution graph, the locations of customers are considered as numbered nodes, while the delivery paths available to riders are bi-directionally connected edges. And based on the model assumption that the number of order generated by customers is greater than or equal to the number of orders that can be delivered by riders within the specified time, we can consider the locations of merchants and customers in the whole Yushan town as one undirected connected graph $G_{mc} = V_{mc}, E_{mc}$, and the locations of all merchants and customers are nodes on this graph, we denote the merchant's location by P and the customer's location by p, thus,

$$V_{mc} = P_1, P_2, P_3, ..., p_1, p_2, p_3, ...$$
$$E_{mc} = V_{mc} * V_{mc}$$

In this way the food distribution problem occurring in Yushan town can be regarded as a path selection problem on such undirected graph.

3.3 Shortest Path Problem

Based on the establishment of the above undirected graph and its properties, we can plan the optimal path for each rider to pick up and deliver the food by finding the shortest path between two nodes in the graph. We use node B_i to denote the i-th order generated, which also has its corresponding pickup location P_i, and node C_i to denote the i-th order delivered to the customer, while p_i is the location of the customer for that order. It is worth noting that the rider must pick up the food before it can be delivered, so node R_i must be traversed before node C_i. So, we need to solve the shortest path problem with the determined starting point and ending point. (i.e., we know the starting point and ending point and we need to find the shortest path between the two nodes.)

Also combined with the real situation of food delivery, we can know that the riders do not know the location of all the merchants and customers being delivered by them at the beginning. Instead, they are assigned new orders by the

platform from time to time during the delivery process, which means we also need to consider the global shortest path problem, (i.e. the shortest path for the riders to deliver the take-out after each new order is obtained).

We can build the adjacency matrix with the shortest path between two nodes by the **Floyd-Warshall algorithm**. Let $d_{ij}^{(k)}$ be the weight of all intermediate nodes from node i to node j, all taken from the set V_{mc} of a shortest path, where k denotes an intermediate node within the set V_{mc}, known to be $k > 1$.$d_{ij}^{(k)}$ as follows

$$d_{ij}^{(k)} = min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$$

According to the shortest path distance above, adjacency matrix can be established as

$$\begin{bmatrix} d_{B_1,C_1} & \cdots & d_{B_1,C_j} \\ \vdots & \ddots & \vdots \\ d_{B_i,C_1} & \cdots & d_{B_i,C_j} \end{bmatrix}$$

where the d_{B_i,C_j} denotes the shortest path from vertex B_i to the vertex C_j

The relative location relationship between merchants and customers in Yushan Town is shown in Figure 4. We observe the satellite map of Yushan Town and abstract the position of some residential areas and commercial areas into vertex and some main roads into edge.

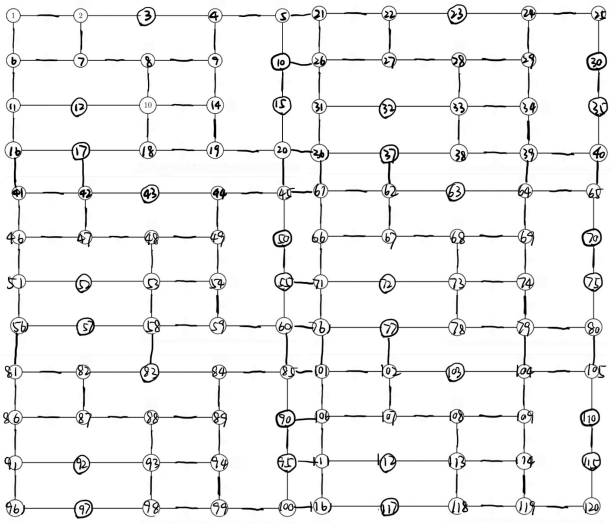


Figure 2: Image of the abstracted graph from Yushan Town

A more intuitive understanding of the algorithm is shown below in the examples below [8], which takes $O(n^3)$ time to compute. In this case of the abstracted graph of Yushan Town, $n = 120$.

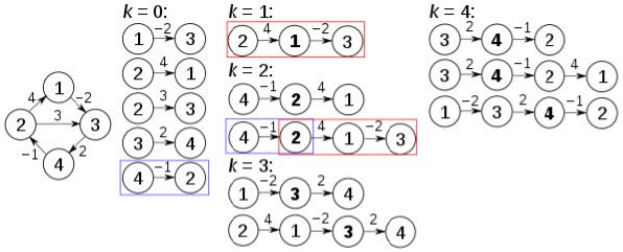


Figure 3: Floyd-Warshall Algorithm

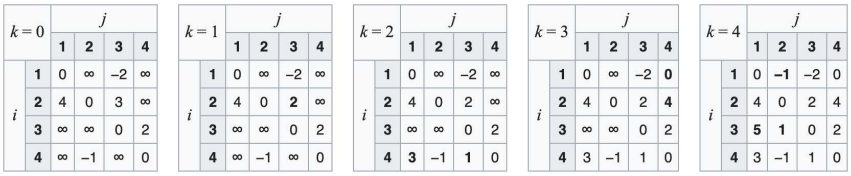


Figure 4: Floyd-Warshall Algorithm

So far we have figured out the shortest path between any two nodes in the graph which can be directly used in subsequent calculations

3.4 Dynamic Programming

Since we want to get the optimal path for each rider after constantly being reassigned orders from a global perspective, we achieve it with the help of **dynamic programming algorithm**. Because the delivery distance of each rider is in line with the accumulative optimal solution, on the basis of determining the specific location of the merchants and the customers, assuming that there is no time and sequence restriction, the instantaneous location and moving direction of a certain rider can be regarded as the common sub-problem of similar contents for all delivery routes. Each sub-problem can be solved by recursion and top-down combination to obtain the global optimal solution. Using the idea of dynamic programming, for each rider's position and next moving direction at any point, the sum of its path distances to complete each task should be minimized so as to ensure that each rider completes as many orders as possible in the specified time with a constant relative speed.

Since we have used the Floyd-Warshall algorithm to establish the adjacency matrix, each element of the matrix can record the distance between any two nodes. Since we assume that the rider needs to deliver the order received first, (i.e. we take the customer waiting time into account), the corresponding positions of the two non-adjacent nodes in the adjacency matrix should be filled with positive infinity. We update this matrix continuously by increasing the range of "transit points". When the set of "transit points" is equal to the full set of all points, we get the shortest path length from each node to each node of the record.

In addition, dynamic programming can also be applied to a simpler version of the optimal path selection problem, where the remaining time is not considered and the order is not re-updated in real time until the next order is delivered. In this case, for any rider, given the remaining tasks, the use of the dynamic application algorithm can give the unique optimal path better and faster. However, once the update order strategy is considered, the path may change greatly as the calculation progresses, so it may take a long time to save a new shortest path separately every time it is found.

4 Mathematics Model

4.1 Assumptions

To simplify the problems, some important assumptions are listed below. More assumptions are made in the process of modeling.

- The indoor time like the food preparing time and delivery time is neglected.
- The location of housing estates(customers) and restaurants will never overlap, which means that one geographical node can only represent one housing estate or one restaurant.
- A reputation rating mechanism is built based on the number of order delivered within 45 minutes. If a rider deliver one order over 45 minutes, then he will lose one reputation credit.
- The takeaway riders can not claim the order. Instead, the platform will automatically allocate the order to the rider based on the reputation rating mechanism.
- Only two orders can be fitted in one rider's takeaway storage box.
- If the food isn't delivered in 45 minutes, the reputation of delivery staffs could be damaged, which will decrease his reputation credit and thus decrease the possibility that he could get the order from the customer. However, the over time orders delivered before the ending time are still counted into the total orders that delivered within the lunch rush hour period.

- If the order is not taken by any staff in 10 minutes, it will cancelled. However, the cancelled order number will decreases the number of new orders. If more than 15 cancellations were made in the previous minute due to long waiting times, the number of orders generated in the next minute will decrease by 10.
- No matter what road condition the rider is inthe speed of motorcycle is $30km/hr$. Riders will not encounter traffic lights and jams on the road.
- The revenue for each order is same.
- 100 riders deliver food at the same time, and each rider has his own number (from No.1 to No.100). Each rider can have as much as two orders at the same time, and the platform will distribute new orders per minute based on the number of orders the rider has at the time and his reputation. When two riders can undertake the order at the same time, the platform will give the order priority to the rider with higher credibility. If riders have the same number of orders and the same credibility, the platform will preferentially allocate orders to riders with smaller numbers.

4.2 Notations

The table below includes the notations mentioned in the model:

Table 1: Notations for the model

Symbol	Description	Symbol	Description
G	Graph of the abstract Yushan town	V	Vertex in the graph
v	One of the vertex in the graph v V	E	Edge in the graph
B	Geographical location for all the restaurants	C	Geographical location for all the customers
b	Geographical location for one of the restaurant	c	Geographical location for one of the customer
P	The restaurant location for the order	Q	The customer location for the order
p	The restaurant location for one of the order	q	The customer location for one of the order
S	The path record	D	The distance of the route

4.3 Model Construction

The establishment of the model considers the delivery problem in the specific region: Yushan town. The total administrative area of Yushan Town is 191.11 square kilometers. Yushan town has lots of commercial blocks, restaurants, residential areas. Besides, the road information of Yushan town is complex. Thus, for further model construction, we abstract the geographical location of Yushan town's restaurants and residents into an undirected graph. The abstracted undirected graph is shown below:

In the graph, each vertex can represent a restaurant, a residential area or represent nothing which means that there is no restaurant or residential area on that vertex. Each edge in the graph is bidirectionally connected. That is:

$$V = \{B, C, v_{empty}\}$$
$$E = V \times V$$

According to our assumptions, the rider can not take the order. Instead, the platform will allocate the order to the rider with highest reputation credit. The initial reputation credits for each rider is 10 and there is no upper limit for the rider's reputation score. If one of the rider does not deliver the order in 45 minutes, the he will lose 1 reputation credit,else, he will get 1 reputation credit. If the storage boxes of all the riders with the highest score are full then the platform will allocate the order to the rider with second highest score. Among the riders with the same score, the platform will give priority to the riders with shorter distance to the target restaurant. To conclude, in our model, which rider will get the order is depend on the reputation credit rating mechanism.

We assume the objective of this delivery resource allocation problem is to maximize the number of orders that delivered within the lunch rush hour period from 11:00am- 2:00pm. We proposed three possible strategies to quantify the objective.

4.3.1 Strategy I: Enumerate

The shortest global path strategy wants to minimize the path of each rider to achieve the global shortest path. Suppose a rider receives n deliver orders at one time, he needs to deliver $order_1$,$order_2$,$\cdots$ order$_n$ from p_1,p_2,$\cdots$ p$_n$ to q_1,q_2,$\cdots$,q$_n$. We applied the dynamic programming and Floyd-Warshall algorithm to construct our algorithm for strategy I. Then, the following algorithm will be the strategy for this rider to decide the delivery route.

Algorithm 1 One rider delivery Algorithm for strategy I

input : Total restaurant destinations $\{$ P$_1$, P_2, $\cdots$ $P_n\}$; $Totalcustomerdestinationsoftheorders$ $\{Q_1, Q_2, \cdots Q_n\}$
output : Shortest path record S_s

Initialize values: distance D = 0; shortest distance D_s = ∞;Path record S = [],
shortest path record S_s = [1,2,3,4,$\cdots$,∞], $Cargo = \{\}$
do
  do
    choose an adjacent vertex
    if *the vertex contains a destination restaurant* **then**
      S.append(current vertex v),D += current edge length,
      if *the rider picks the food* **then**
        cargo.append(current restaurant destination P_j)
    if *the vertex contains a destination customers j* **then**
      cargo.remove(corresponding restaurant location P_j),
      cargo.append(current customer location of the order Q_j)
    if D> D_s **then**
      break
  while *Cargo* $\neq \{Q_1, Q_2, \cdots Q_n, \}$;
  Shortest distance D_s = min(D_s,*D, Shortest path S_s = min(S_s, S)*
while *Not all the possible path have been tried;*

To more intuitively understand the algorithm, a flow chart for this algorithm is given in Figure. 5.

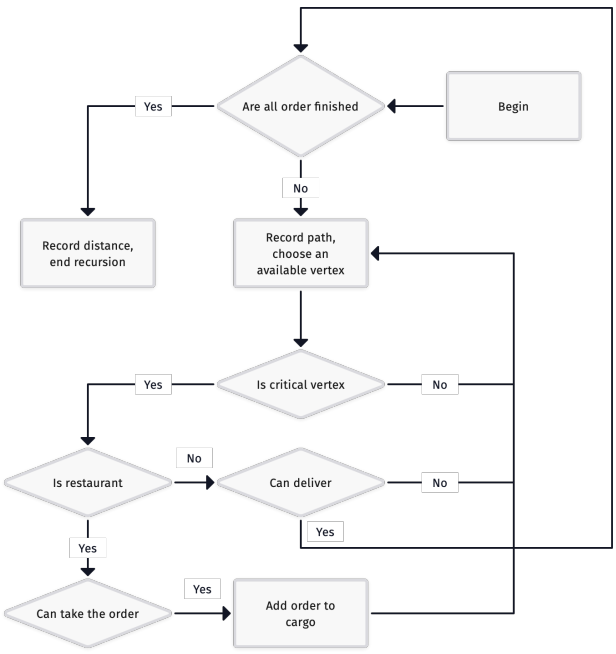


Figure 5: Flow chart of the one rider delivery algorithm for strategy 1

However, the order for the riders is updated as the time passing by. Then, we will re-run the model whenever order is updated to give the current shortest path for the rider. To get the global minimum distance, we add the shortest distance of each rider.

4.3.2 Strategy II: Fixed Paths

The two orders per time strategy will ask the platform allocate two new orders to the rider only when the rider has finished the delivery for the last two orders, which means that the rider must finished order 1,2 to get next 2 orders. This strategy minimize the possibility that the riders deliver one order over 45 minutes. And thus maximize the customers satisfaction degree.

To more intuitively understand the algorithm, a flow chart for this algorithm is given in Figure. 6.

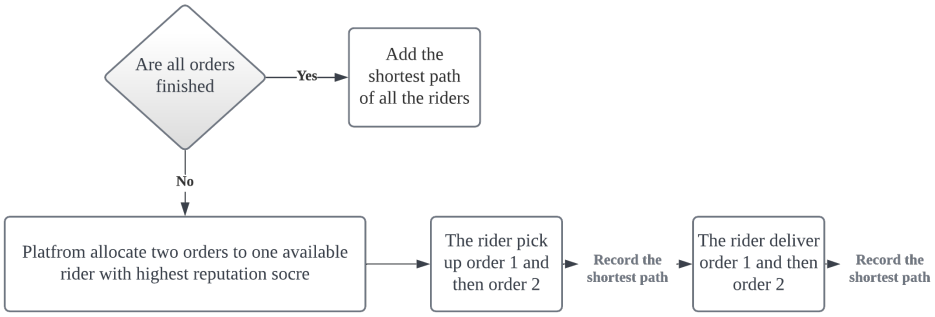


Figure 6: Flow chart of the one rider delivery algorithm for strategy 2

4.3.3 Strategy III: Permutation of Paths

Suppose a rider wants to pick up food from restaurants R1 and R2 and deliver it to the customer's community C1 and C2, and customer 1 orders before customer 2. Under strategyII, we stipulate that the rider must deliver the order to customer 1 before delivering the order to customer 2. However, under Strategy 3 we don't have this limitation, which means we have several path options:

$$R_1 \longrightarrow R_2 \longrightarrow C_1 \longrightarrow C_2$$
$$R_1 \longrightarrow C_1 \longrightarrow R_2 \longrightarrow C_2$$
$$R_1 \longrightarrow R_2 \longrightarrow C_2 \longrightarrow C_1$$

The flow chart of strategy III is shown in 7

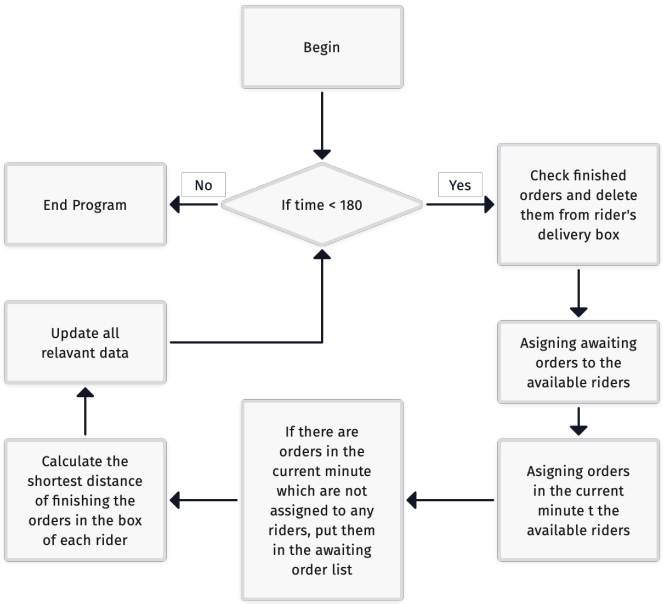


Figure 7: Flow chart of the algorithm for strategy III

5 Results Summary and Model Validation

We have implemented all three strategies and obtained the results by Monte Carlo simulations. For strategy I, due to the poor performance of dynamic programming, even if we reduce the branch of abstract graph, the complexity of calculation for the whole graph is still high, so we do not show relevant results here

For Strategy II, we use random numbers to generate orders for different restaurants, 20-40 orders per minute. Riders always start in a neighborhood, pick up their orders at restaurants and deliver them to customers in order. Based on the order generation, we can simulate the orders that are cancelled due to they have not been received by riders for more than 10 minutes and also simulate the orders that the rider has completed. We can also track the delivery time and customer waiting time for each rider. The order is checked to see if it takes more than 45 minutes to deliver, thereby increasing or decreasing each rider's credit points. The simulation results are as follows:

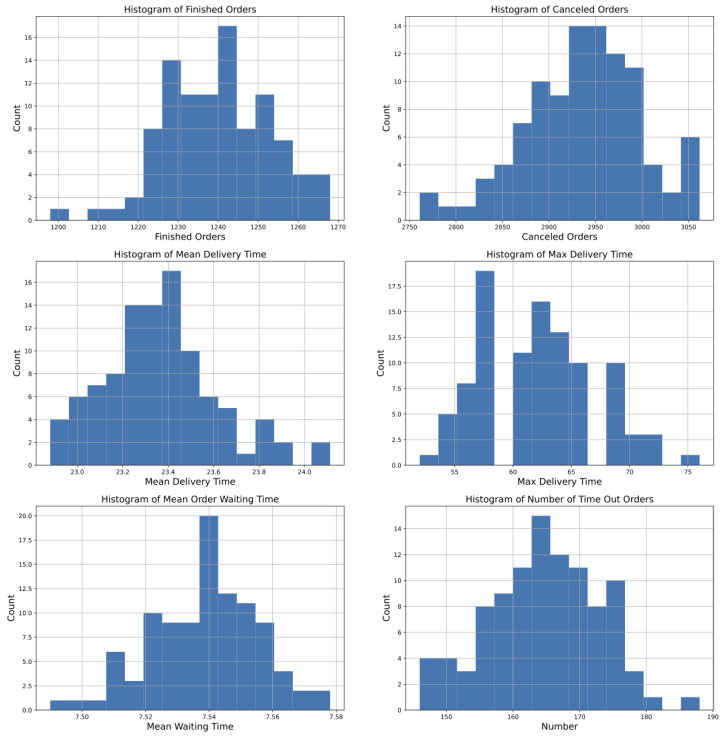


Figure 8: Statistics distribution for Strategy II with speed of 30km/h

To more intuitively understand our results, we present the expectation, maximum, minimum and standard deviation corresponding to the distribution in the table.

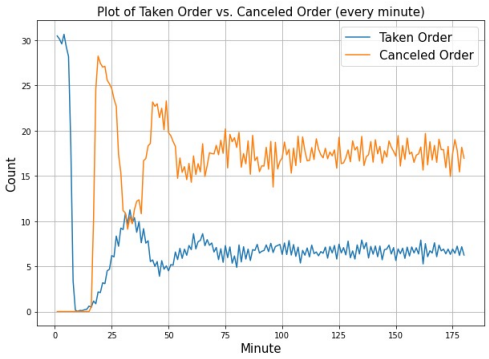
Table 2: Expectation, min, max value and standard deviation of the statistics given by Monte Carlo Simulation for Strategy II

Variable Types	**mean**	**maximum**	**minimum**	**standard deviation**
num_finished	1315.36	1350	1278	14.44
num_canceled	2885.91	3009	2732	56.19
mean_delivery_time	21.94	22.54	21.41	0.23
max_delivery_time	58.1	66.0	48.0	3.90
mean_waiting_time	7.62	7.66	7.58	0.02
time_out_orders	119.0	148	93	10.36

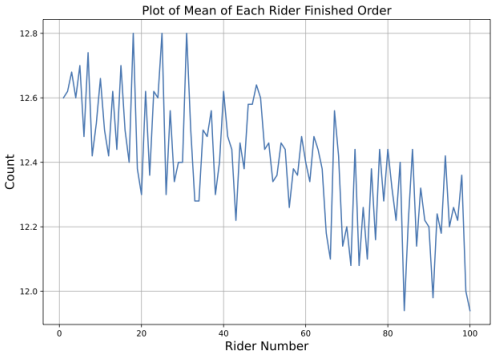
We can see from the table that 100 riders can complete about **1315** orders during the peak time period, and in our many simulations, riders can complete up to **1350** orders and at least **1278** orders. Due to the excessive number of generated orders, the average number of cancelled orders was about **2886**, with the most cancelled orders being **3009** and the least cancelled orders being **2732**. The average delivery time for each rider to deliver a single order was about **22** minutes, and the maximum delivery time was about **58** minutes. It takes about **7** minutes for a generated order to be assigned to the appropriate rider by the platform, and the number of orders that are not delivered within 45 minutes are about **119** orders. We can also obtain the confidence interval of our results from the table.

Moreover, one interesting phenomenon shown in Figure6 (a) is that if we visualize the number of cancelled orders and the number of orders successfully assigned orders, we can see that in the first 20 minutes, a large number of orders were successfully assigned by the platform to different riders because we had a hundred riders available at the beginning. But

as time goes on, the number of cancellations due to long waiting times increase dramatically as a large number of orders are generated and many riders do not complete their previous order tasks. By about 25 minutes, the number of cancellations will be reduced again as a group of riders have completed their orders and can continue to take orders. This oscillating process continues, with assigned orders at their peak and canceled orders at their trough, and vice versa.



(a) Taken Cancelled order in every minutes for Strategy II with speed of 30km/h



(b) Mean number of each rider finished order for Strategy II with speed of 30km/h

Figure 9: Statistics for Strategy II with speed of 30km/h

Figure6 (b) shows the number of orders completed by each rider. Since we assign orders to riders with smaller numbers in preference, given that all conditions are consistent, riders with smaller numbers will complete more orders overall than riders with larger numbers.

Also, we think the rider's delivery speed is one important factor that influence the allocation plan. Thus, we change the speed from $30km/h$ to $120km/h$ to see the influence of the speed to the model.It also applies to simulating some extreme weather requests. The simulation results are as follows:

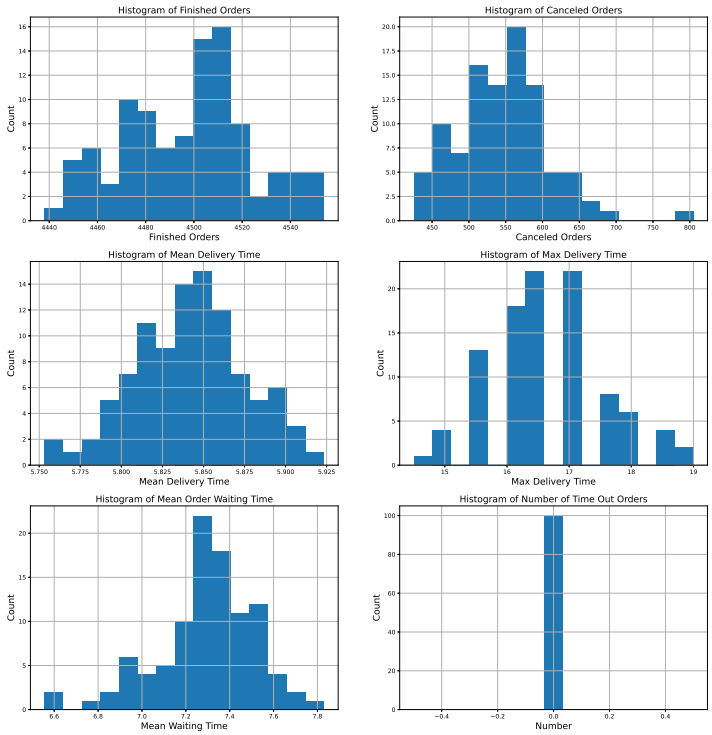
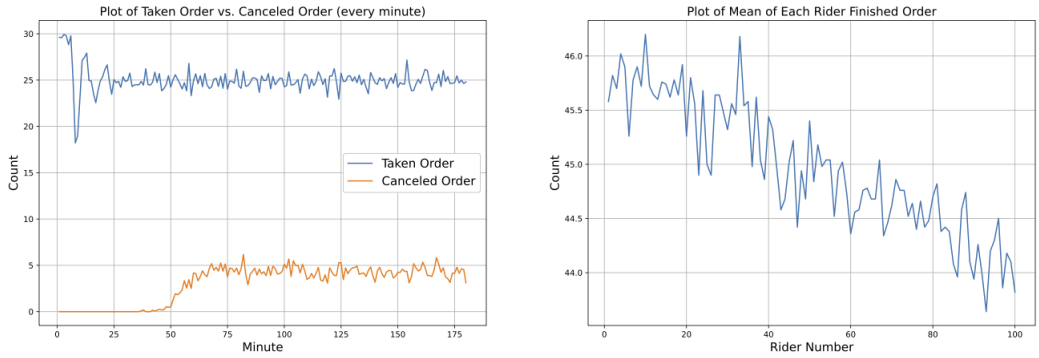


Figure 10: Statistics distribution for Strategy II with speed of $120km/h$

From the visualization results, we can find that when we set the speed 4 times as the original speed. The simulation results has a dramatic change. The finished order increases and the cancelled order decreases significantly. The mean and max delivery time decreases to around $\frac{1}{4}$ of the time with $30km/h$ which validates the effectiveness of our model. Besides, when the speed increases to $120km/h$, none of the riders will run out of time.



(a) Taken Cancelled order in every minutes for Strategy II with speed of 120km/h

(b) Mean number of each rider finished order for Strategy II with speed of 120km/h

Figure 11: Statistics for Strategy II with speed of 120km/h

From the plot of taken cancelled order by time, we find that, if the speed increases from $30km/h$ to $120km/h$, then the initial oscillation situation will be relived. The difference between the peak and the valley value decreases. We think

the possible reason is that when speed was increases to 120km/h, the rider will finished the orders quickly. Thus, in almost every minute, there will be available riders to deliver the order and thus decreases the oscillation situation.

For Strategy III, in this case the rider is not necessarily deliver food in order, but can freely choose the first delivery order and the second delivery order to make the shortest path. This means that we have a lot of new paths to take, and the second customer doesn't have to be served after the first Customer, which increases the number of taken orders.

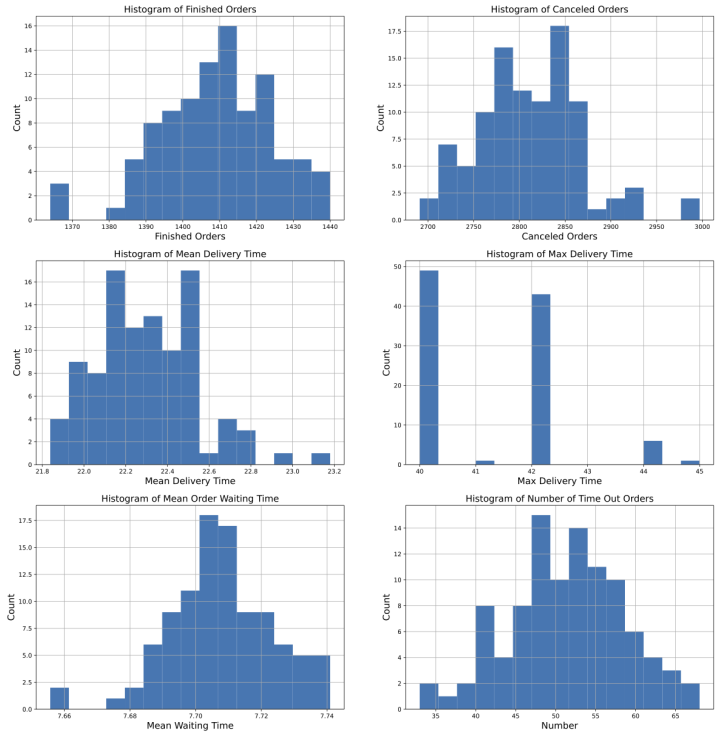
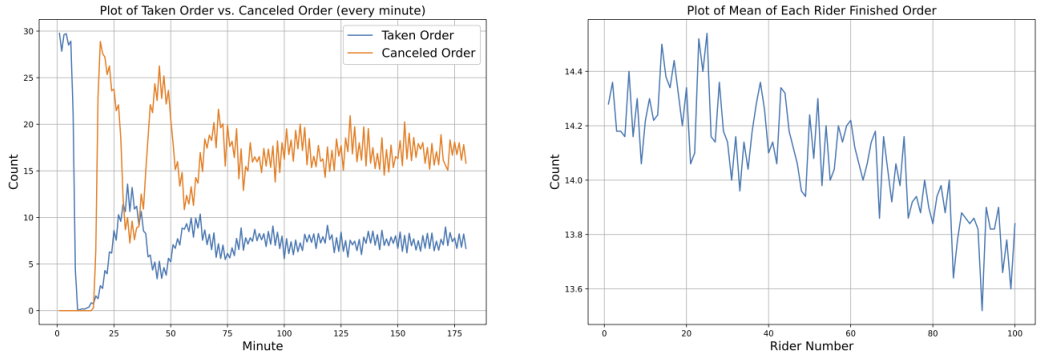


Figure 12: Statistics distribution for Strategy III with speed of $30km/h$

From the results, we can see that comparing with the results of strategy II, the number of finished order increases from around **1200** to around **1400** and the number of cancelled order decreases from **3000** to **2800**. Bedsides, the number of time-out orders decreases significantly from **150** to **50**. The table presenting the expectation, maximum, minimum and standard deviation is shown as well.

Table 3: Expectation, min, max value and standard deviation of the statistics given by Monte Carlo Simulation for Strategy III

Variable Types	**mean**	**maximum**	**minimum**	**standard deviation**
num_finished	1376.6	1406	1342	11.06
num_canceled	2843.17	3038	2744	59.69
mean_delivery_time	23.21	23.67	22.83	0.18
max_delivery_time	40.8	48.0	40.0	1.20
mean_waiting_time	7.68	7.71	7.64	0.01
time_out_orders	60.62	77	37	7.04



(a) Taken Cancelled order in every minutes for Strategy III with speed of 30km/h

(b) Mean number of each rider finished order for Strategy III with speed of 30km/h

Figure 13: Statistics for Strategy III with speed of 30km/h

We can see from Figure 13 (a), the second peak value increases which indicates when we implement strategy 3, there will be more orders delivered with same amount of riders. And the number of orders each rider can finish has increased as well.

To summarize, comparing the results between strategy II and strategy III, we find that implementing strategy III will let the riders deliver more orders in the given three hours and thus, the plat-from can get more revenues. Thus, we conclude that strategy III: Permutation of Paths is the best strategy among the two strategies which have been simulated. Also, we change the speed of riders from $30km/h$ to $120km/h$ and simulate the corresponding results to validate the model.

6 Extensions and Limitations

6.1 Advantages

6.1.1 Undirected weighted graph representing position

We abstract the Yushan town as an undirected weighted graph based on the relative locations of merchants and customers, where each merchant and customer can be considered as nodes and the paths available to riders can be regarded as edges. Since path planning only cares about connectivity, not specific locations, the undirected graph simplifies the problem nicely and serves to encode connectivity.

6.1.2 With the help of dynamic programming algorithms, various strategies are proposed

We propose three strategies to ensure that our model is more in line with the real food delivery situation. The first strategy is knowing all pickup locations and delivery locations and plans the optimal path without considering the customer waiting time; the second strategy is updating the number of orders for each rider in real time and plans the optimal path while satisfying the interests of the customer who orders first as much as possible; the third is updating the orders for each rider in real time and plans the optimal path without considering the delivery order (within 45 min). With the help of dynamic programming, we divide a complete task into multiple sub-tasks with the same content and find the unique optimal path by solving each sub-task.

6.1.3 Use of multiple suitable algorithms

In the construction our model, we utilize Ford-Warshall algorithm, dynamic programming algorithm, etc. Each of these algorithms is suitable for application in the problem of finding optimal paths, especially in our hypothetical scenario. These algorithms are easy to understand and have high accuracy.

6.1.4 Applying Monte Carlo simulations to verify the feasibility of our strategy

For the many different strategies we propose, we try to simulate the delivery of food in Yushan Town using Monte Carlo simulations and show the corresponding statistical results to make the feasibility of the strategies we provide more intuitive to the readers.

6.2 Limitations

6.2.1 Lack of optimization of model performance

When proposing strategies, we apply knowledge of dynamic programming to separately "save" each new shortest path being founded, which means in the real case of a large number of orders, our algorithm may take a long time to give us result. In our assumptions, the maximum number of orders a rider can have at the same time is two, which means that we can only handle the optimal path for a rider to pick and deliver two orders. However, in general situation, a rider can pick up as many orders as he/she can depending on his/her willingness. In this case, our model may have a poor performance.

6.2.2 Limitations of the optimization function

In constructing our model, we use the number of orders delivered per rider per unit of time to represent the maximum revenue that a merchant can earn, and solve this problem by finding the optimal path for riders. However, in real-world scenarios, this is not the case. Merchants focus more on the actual profit earned, which is revenue - cost. The determination of cost is related to many factors, such as employee's salary, rider's commission per order, and so on. Assuming that employee wages and rider commissions are two other important factors to consider, the revenue function F can be viewed as a function of employee wages M, rider commissions per order N, and the number of orders delivered R

$$F = f(N, M, R)$$

From the knowledge of multivariate calculus, F may not be optimal when N is the maximum. Therefore, sometimes we may need to drop some orders so that the total revenue function F is optimal

6.3 Further Study

Taking different factors into account: In our model we only consider the changes in our results when the weather changes, which means we only show the effectiveness of our strategy when the delivery speed of the riders' changes. We do not take into account more conditions, such as the presence of traffic lights on the road, the decrease in the number of orders generated due to a burst of orders, etc. We will subsequently change our assumptions to include these factors and consider them together to verify the feasibility of our strategy.

Implementation of the assignment algorithm: In addition to applying the Floyd-Warshall algorithm, we have many other options to generate the adjacency matrix. One of the important algorithms is the allocation algorithm. Using the allocation algorithm, we can obtain more strategies and further optimize our model. For example, we make customers closer to the target restaurant are served first, which is also a possible optimization strategy.

7 Conclusion

In this paper, we investigate the food delivery problem in a specific region: Yushan Town, Kunshan City. We introduce the reputation rating system to solve the order allocation problem. Then, we abstract the geographical conditions of Yushan Town into weighted undirected cyclic graph and then construct three strategies: **Enumerate Strategy**, **Fixed Path Strategy** and **Permutation of Paths Strategy** to finish the optimal pick-up and delivery task and determine the optimal delivery scheme for each rider. For strategy I, the platform will give the optimal delivery scheme according to the global shortest path. For strategy II, the order sequence of each rider is fixed, the platform will only determines the optimal path between restaurant and customer for each order. For strategy III, given two allocated orders for one rider, besides determining the optimal path between restaurant, the platform also needs to determine the best allocating sequence for these 2 orders. We implemented Monte Carlo Simulation to test the performance of the result and find that strategy III will maximize the number of finished orders. Besides, we consider the influence of the speed of the riders to the model. We find that if the speed of the riders increases, the total finished order will increases. Then, when the road condition is terrible, the total finished order will decrease, which decreases the total revenue.

8 Author contributions

Chenglin Zhang: design algorithms and strategies, and complete the model and simulation part.
Mengfan Gong: Report writing, including methodology part, result part and literature review part.

Yijia Xue: Report writing, including drawing flow chart,model part and introduction part.

9 Acknowledgement

We thank Prof. Shixin Xu for the generous and insightful guidance in this session's Mathematical Modeling course, that it will empower our future learning greatly.

References

[1] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[2] Gerardo Berbeglia, Jean-François Cordeau, and Gilbert Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010.

[3] Chengqing Liu, Dawei Hu, and Rong Cai. Research on two-echelon open location routing problem with simultaneous pickup and delivery base on perishable products. *CICTP 2019*, 2019.

[4] Richard E. Bellman. Dynamic programming. 2010.

[5] Yu-Hsuan Huang and Chuan-Kang Ting. Genetic algorithm with path relinking for the multi-vehicle selective pickup and delivery problem. *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011.

[6] Changfeng Zhou. Solving the vehicle routing problems with time window using hybrid genetic algorithm. *ICTIS 2013*, 2013.

[7] Xing Wang, Ling Wang, Shengyao Wang, Jing-fang Chen, and Chuge Wu. An xgboost-enhanced fast constructive algorithm for food delivery route planning problem. *Computers amp; Industrial Engineering*, 152:107029, 2021.

[8] Wikipedia contributors. Floyd–warshall algorithm — Wikipedia, the free encyclopedia, 2022. [Online; accessed 9-March-2022].