OPTIMIZE THE PATH OF FOOD DELIVERY

Han-Wen, Liu Han-Lin, Chen Tien-Yun, Kuo

[1]Department of Mechanical Engineering, National Taiwan University, Taipei 10617, Taiwan

Food delivery is a popular culture in Taiwan recently. It is a great part time job for everyone who has a two-wheel transportation, for instance, scooter, bicycle, etc., or you can even deliver the food on feet. The money everyone earned is quite different, this report provides you an idea about how to be the most outstanding delivery man among the rest.

First of all, we have built a model to better study the delivery process. Second, we put it into the optimization problem which constructed by an objective function and several constraints. Third, we use genetic algorithm as our optimization solver. In the end, we get the optimal solution, i.e., in the same map, we earn more money and satisfy the constraints at the same time.

The uncertainty is also considered in this report. The uncertainty in this model is the difference between different maps. We put our optimal result into 1000 different maps to check if the results are viable. The results show that we may still encounter cases that constraints won't be fulfill, but overall the optimization result should cover most cases.

***Index Terms*—food delivery; optimization; genetic algorithm; uncertainty analysis**

I. INTRODUCTION

First food delivery service is introduced into Taiwan at 2012. It's a service which provided by third party, for example, foodpanda, Uber Eats, etc., and it hires people to deliver the food to the customers. Customers only have to press a few buttons and wait. Because it's convenience and the extra is reasonable, more and more people start to use it, and that accompany with more delivery man.

The way delivery man earn money is different in each company. In this report, we choose Uber Eats as our model's background. Delivery fees of Uber Eats is calculated by distance and completed orders, and they are listed in **Table 1**.

Table 1

Delivery fees (Uber Eats)	
Pickup	42.5 NT / pickup
delivery	17 NT / deliver
Distance	10.5 NT / km
bonus	150 NT / 7 deliver

Although more pickups and distance are seen pretty good, there are time limit for each orders (around 20 minutes) and bonus. The delivery man can decide to ride more distance or complete more orders. With those decisions come many strategies. The purpose of the report is to investigate the best strategy with the optimization knowledge we learned in the course.

II. Simulation

A. *Regular and Intermag Papers Map randomized*

In the simulation process, we set up a 2-dimension grid map, with 100km length and width, and we make four assumptions to better study the process of food delivery.

(i) One delivery man
(ii) Checkerboard street system
(iii) Constant velocity
(iv) Manhattan distance

In order to make sure our optimization result can fit in most general cases, we randomly generated 20 costumer locations, 10 restaurant locations, and 1 delivery man's initial position on our grid map as **Figure 2.1** show.

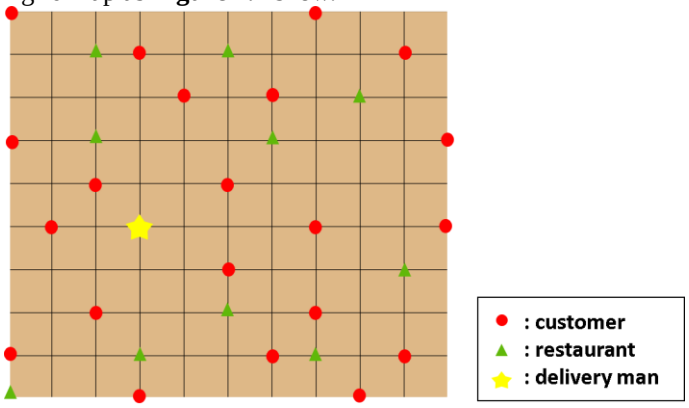

Figure 2.1 Grid map example

In this simulation, we will simplify the optimization problem by:

(i) Assume the vehicle speed remains the same throughout the sequence (Means no traffic light, traffic jam……etc.).
(ii) All food orders will be initial at the beginning of the simulation process.
(iii) Orders do not expire.
(iv) Only one delivery man is doing the order (Means no competition).

B. *Manhattan distance*

For distance calculation, we use Manhattan distance to calculate the path length between two points. Manhattan distance is the distance between two points measured along axes at right angles. For example, if we have two point (x_1, y_1) and (x_2, y_2), the Manhattan distance will be:

$$D = |x_1 - x_2| + |y_1 - y_2|.$$

C. *Priority function*

In the food delivery sequence, the delivery man will have two different action: pick up order and deliver food. In order to determine which action will be taken in the next step, we developed two priority function, F_1 which is pick up priority, and F_2 which is delivery priority.

The pickup priority functions F_1's detail:

$$F_1 = -w_1 t_{m2r} + w_2 t_{r2c}$$

Where w_1 and w_2 are two positive weight value, t_{m2r} means the time from delivery man to restaurant, and t_{r2c} means the time from restaurant to costumer. Because the delivery man won't get paid during t_{m2r}, there is a minus in front of it. It's worth noting that if the stacked order situation happens, F_1 would divide by another positive weight value w_3 because the pickup fee counts only once.

The delivery priority functions' detail:

$$F_2 = -w_4 t_{m2c} + w_5 t_{r2c}$$

Where w_4 and w_5 are two positive weight value, t_{m2r} means the time from delivery man to restaurant, and t_{r2c} means the time from restaurant to costumer. However, if completed orders are close to multiples of 7, F_2 would go extremely big because of bonus. Both priority functions have four relevant parameters in the priority function:

(i) Distance
(ii) Completed orders
(iii) Stacked orders
(iv) Elapsed time

D. Priority decision

In previous chapter we discuss the priority function we set up for the optimization sequence, and in this chapter we are going to discuss our priority decision process.

There are three different scenarios where the priority functions will be used differently, depend on the backpack carry number N:

(i) $N = 0$: It means backpack is empty. In this scenario, the only action the delivery man can take is picking up order. Therefore, we consider only F_1:

$$F_1 = \{F_{11}, F_{12}, \dots \dots F_{1n}\}$$

Where n is the remain restaurant numbers that contain orders from customers. We will use $\max(F_1)$ to find which restaurant to pick up order in next step.

(ii) $N = max$: Means backpack is full. In this scenario, the only action we can take is delivering order. Therefore, we consider only F_2:

$$F_2 = \{F_{21}, F_{22}, \dots \dots F_{2m}\}$$

Where m is the remain order numbers that still not complete. We will use $\max(F_2)$ to find the next costumer to deliver order.

(iii) $0 < N < max$: Means backpack is neither full nor empty. In this scenario, we can choose either pick up or deliver order. Therefore, the decision will be made by the following process:

$$F_1 = \{F_{11}, F_{12}, \dots \dots F_{1n}\}$$
$$F_2 = \{F_{21}, F_{22}, \dots \dots F_{2m}\}$$

We will compare $\max(F_1)$ and $\max(F_2)$ to decide whether to picking up order or delivering food in next step, and which restaurant to pick up or which costumer to deliver order.

E. Objective function and constraints

We set up our decision-making model in previous chapter, now we will set up our objective function and constraint set for our optimization problem.

Our goal for the food delivery process is to maximize the earned money within the certain time period, therefore we will set up our objective function as total earned fee. Here're the parameters for Uber Eats:

(i) Pick up fee: $P_{pickup} = 42.5 * n_{pick}$ (NTD)
(ii) Deliver fee: $P_{delivery} = 17 * n_{order}$ (NTD)
(iii) Distance fee: $P_{distance} = 10.5 * D_{order}$ (NTD)
(iv) Bonus fee: $P_{bonus} = 150 * mod(n_{order}, 7)$ (NTD)

Where:

n_{pick} : orders picked up from restaurants
n_{order} : Completed orders
D_{order} : Distance between order and restaurant
mod : Quotient , ex: mod(8,7) = 1

So we can set up the objective function:

$$\max \ P_{total} = P_{pickup} + P_{delivery} + P_{distance} + P_{bonus}$$

Then, for our constraint set, there are two constraints we want to fulfill:

(i) Entire food delivery sequence must be within a certain time limit.
(ii) Food cannot stay on vehicle for too long to provide spoiled.

Therefore, we set up the constraint set:

$$g1 : T_{elapse} < T_{route}$$
$$g2 : T_{stay} < T_{limit}$$

Where:

T_{route} : Time for whole route
T_{stay} : Time of food stay on car
T_{limit} : 20 mins

Combine with previous components in the simulation model, we can construct the model's flow chart to represent our simulation model as **Figure2.2** shown.

We assume that deliveryman won't make any decision during his movement, so all parameters are updated and recorded after the next position is arrived. We set the speed of the delivery man (set to 55km / hr.) and multiply it with the movement distance to measure time passes. When deliveryman complete the conditions described in this chapter, he earns the income. Other details in **Figure2.2** are made to avoid any unexpected situation to ensure the correctness of the entire process.

Since the entire process takes the order distribution as input, the income and stack time as output. For optimization the whole process is our objective function. Since this function is discrete and composed of logical judgments, it's difficult to use gradient method to optimize, so we use heuristic algorithms to solve this problem such as Genetic Algorithm (GA).

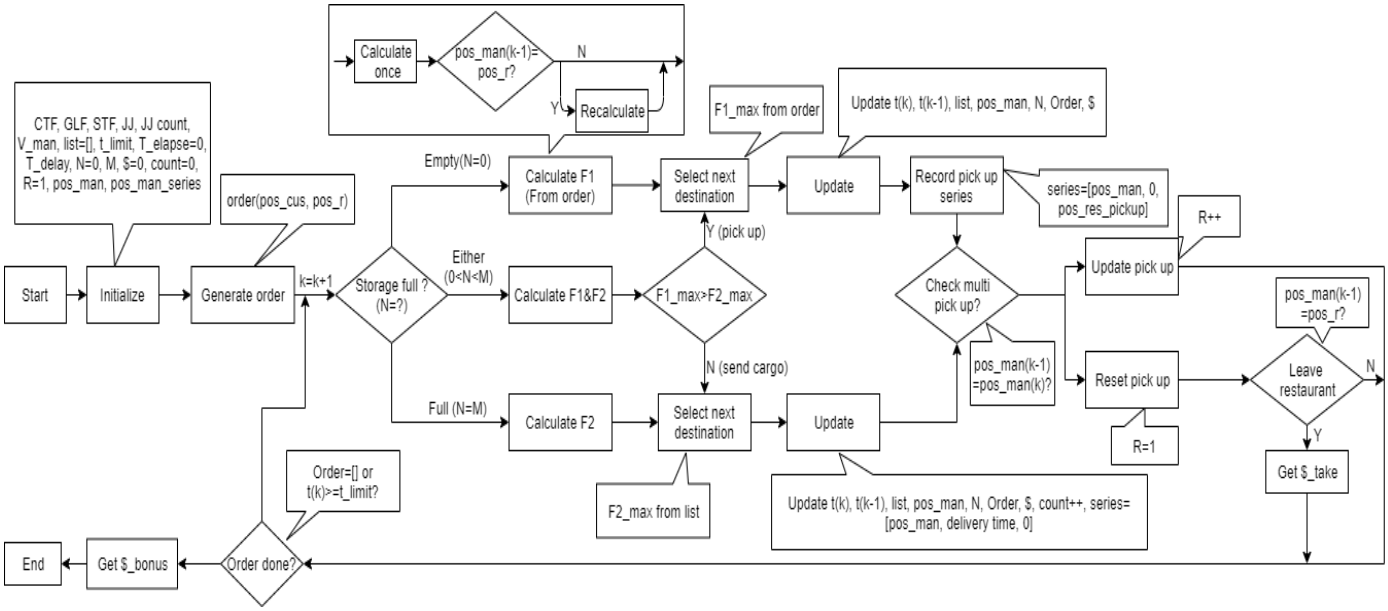

Figure 2.2 Food delivery model's flow chart

F. Genetic algorithm

The genetic algorithm is a model that simulates the reproduction, evolution, and elimination of living organisms in nature to find the algorithm that is most suitable for their living environment (in this case the optimization problem).

In the algorithm, the number of individual organisms, the length of the gene chromosome, and the number of iterations will be set in advance. The first-generation genes will be randomly generated within the constraints of the problem. In the optimization problem, the objective function is to simulate the environment that the organisms are adapting, and the optimization variables are decoded into a binary form to simulate the biological gene. At each iteration, the gene will be recombined through the following modes:

(i) Replication

(ii) Mating

(iii) Mutation

When the gene recombination ends, the algorithm will recombine the gene back to the decimal variable form and input them into the objective function. Algorithm will select a certain set of chromosomes that's nearest the best solution as the result of this iteration.

By using this algorithm that simulates the evolution of the nature, you can ensure that each generation will have better variables than the previous generation. It is expected that at the end of the algorithm, a set of optimal solutions for the objective function will be obtained.

III. RESULT

According to the model architecture and methods of establishing an optimized food delivery model proposed in the previous chapters, we have trained a set of parameters for the function as optima. To verify the generalization ability of the optimization model, that means it can be used at any time under the conditions of different number of restaurants and the customers, to achieve as much income as possible and the lowest T_{stay}. Our study first considers the situation of simple order distribution, so that we can compare the optimal function and artificial decision to verify whether the function determines the food delivery route corresponding to our expectations. Furthermore, we can check whether the function has obvious logical errors or making stupid decisions.

We simplified the order and randomly generated up to three or four restaurants and six homes for decision-making, as shown in **Figure 3.1** to **Figure 3.3**. The start point is the initial position of the delivery man, red points are the restaurant, and the blue points are the customers. **Figure 3.1** represents the situation where the restaurant is in the center of the customers. To obtain the maximum income, we expect the delivery person picks up the restaurant closest to deliveryman's current position, and after completing the first order delivery, he takes the restaurant order in the center of the map and sends it to the nearest customer from that restaurant, although it will lose some opportunity of taking larger orders, but it can speed up the completion of the order and get more money. According to the figure, the decision make from the optimal function is as same as the above delivery logic, which can explain the optimal function for this type of order distribution can reach the goal. In this case, T_{stay} has less influence of making the decision during process, so greedy going delivery of each order is also a better approach from this perspective.

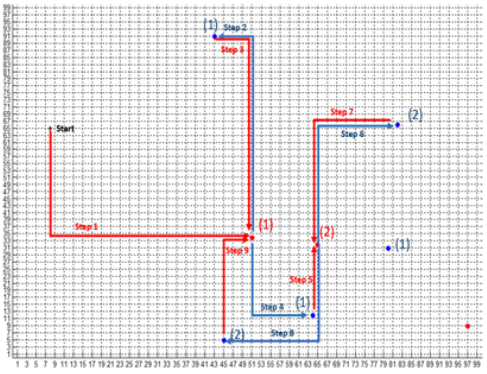

Figure 3.1 restaurant at center case

We consider the distribution of another order, shown in **Figure 3.2**. For this situation, the location of many designated restaurants is a long way from their home by the orders. If we only consider taking the restaurant closest to the deliveryman's current position, it will cause the T_{stay} increases significantly and violate the constraints. In addition, deliveryman is not suitable for taking two or more orders from the same restaurant at a time when the T_{stay} is short. Combining the above factors, the deliveryman is recommended to taking the largest distance fee and shipped one by one to maximize income and ensure T_{stay} won't exceed 20 mins. The decision function of the optimal function also indicates the same result as the above-mentioned delivery logic.

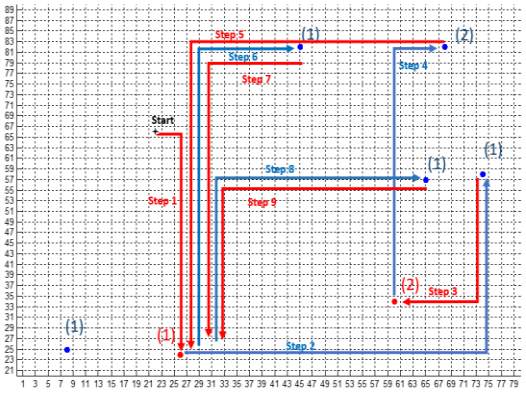

Figure 3.2 large distance needed case

The order distribution situation shown in **Figure 3.3** is more compromised than the previous two examples. The optimized function can still maintain the regulations and achieve a certain degree of income. We tried to use human decision-making for comparison and discover that the food delivery route planned by the function is a way to save distance of paths, while maintaining the enthusiasm for increasing income and maintaining the conservativeness of T_{stay}, which can represent the generalization ability of the optimized function.

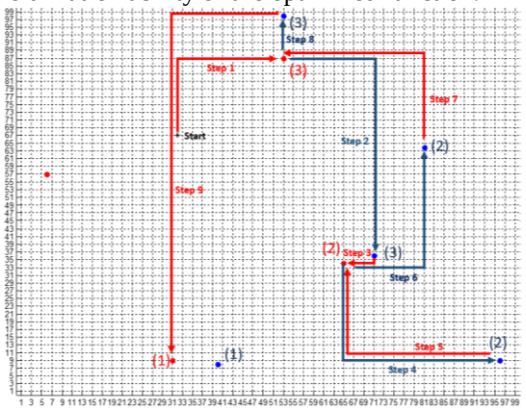

Figure 4-3 compromised case

IV. Uncertainty Analysis

So far, we have only used a few simple examples to illustrate the generalization ability of the optimized function. We have not simulated more complex order distribution situations yet. For the problem of food delivery, changes in the order will have a huge impact on decision-making and these differences can be used as the uncertainty of analysis. The difficulty we encountered is when the order is complicated, it is difficult to artificially determine and compare whether the function achieves the established task. This analysis uses pure Monte Carlo method, we randomly generate many complex order distributions under the same total time limit as input and output the results after same optimized function. According to the theory of probability distribution, if the function is stable, the output will be Gaussian distribution and have an acceptable standard deviation.

For our study, the output only includes income and T_{stay}. The foundation for deciding whether the income is good or not is based on the simple example proposed in the previous chapter. The total time and income of the previous chapter are compared with this chapter. The income is normalized as making how much money per unit of time. Because both simulations are set up that the delivery man cannot complete all the orders, and the previous chapter has obtained a certain degree of optimization. Therefore, if the normalized income of Monte Carlo method is like or larger than the simple ones, indicating that the optimized function has a good plan for complex order distributions. On the other hand, if the proportion of T_{stay} output which exceeds the constraint is below on a certain level, which explain that functions can maintain constraints even in complex situations.

We randomly generated 1000 distribution maps, and presented the output distribution as histogram, as shown in **Figures 4.1** and **Figures 4.2**. Both graphs show that the optimized function can reduce the food stay time to 95% under the constraint. For income, under multiple total time the Monte Carlo's results have more multiple of the average income compared with the simple cases. By the way, we still consider only one map each time for optimizing the parameters. The resulting function is similar or even the same as the planning of the optimized function obtained through the Monte Carlo method, which can explain that the optimized function proposed in this study still maintains good robustness under high-variation input.

It is worth mentioning that we talked to delivery man in real life, and he said the average of fee he earned in an hour is around 200 (NTD), and our results are unbelievable. However, the assumptions of one delivery man and checkerboard street system are too ideal. There are too many competitors, and the traffic usually takes a lot of time, especially in Taipei.

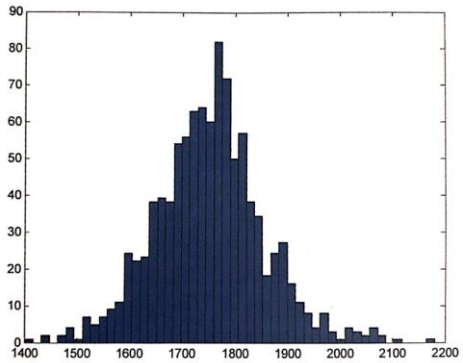

Figure 4.1a Income

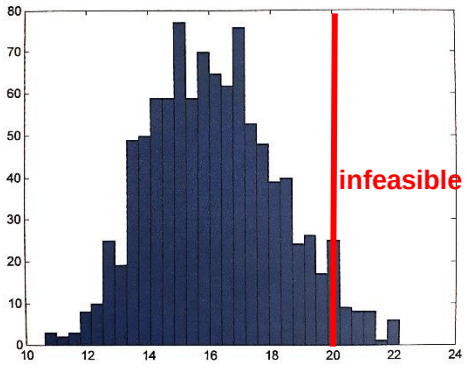

Figure 4.1b T_{stay}

Figure 4.1 Simulation of Monte Carlo method - 1

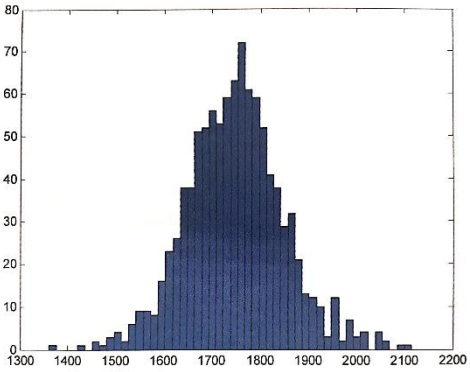

Figure 4.2a Income

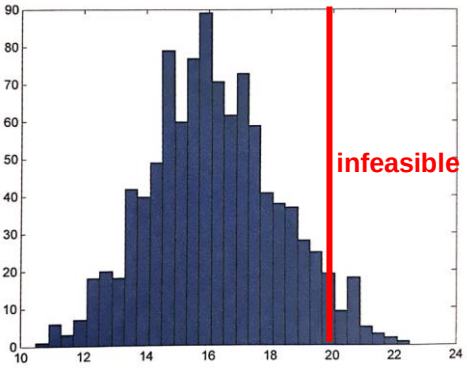

Figure 4.2b T_{stay}

Figure 4.2 Simulation of Monte Carlo method – 2

V. Summary

(1) Our goal is to maximize the earned fee for one delivery man, also try to prevent food from staying on car too long. So we put the total fee and time of food stay on car in our optimization.
(2) We optimize the objective function via GA because we're not sure our optimization problem's gradient.
(3) We put our optimization result into 1000 different cases to do uncertainty analysis, to check if the cases are normal distribution and also satisfy the constraints. The results show that depend on the map, we may still encounter cases that constraints won't be fulfill, but overall the optimization result should cover most cases.

Acknowledgement

The authors wish to acknowledge the teacher of this course. Help us overcome the difficulty of this project and give us the wealth of knowledge. It is definitely an eye-opening semester.

Reference

[1] 【心得】外送司機全攻略! UberEats 送餐之路心得分享
https://forum.gamer.com.tw/C.php?bsn=60561&snA=14326