ME 308 Project : Group 17

Food Cart Optimization

Aruja Khanna 190100024

Ritwik Gupta 190110075

Anagha Savit 190100014

Shiv Modi 19D100011

Abstract

The paradox of choice is quite a prevalent occurrence in today's time, wherein being presented with multiple options does not always increase consumer satisfaction, but the abundance of options requires more effort to make a decision and leaves people with a feeling of anxiety as well. One of the most relevant examples of this in our daily lives is ordering food via an online delivery platform- A dilemma we have all faced.

This project aims at aiding the decision-making process of ordering food from an online food delivery platform. This project will optimize the user's food cart such that it provides maximal utility, keeping in mind the satiability, cost, quality and time taken for delivery. We would have to account for the choice of platform (Zomato, Swiggy, etc), the user's interest in a particular type of cuisine, ratings and reviews of the restaurant, average processing and preparation time for a given outlet, extra charges levied by the restaurant, and most importantly, how far away the outlet is from the given location.

Problem Formulation

We first modeled the world by defining the constants and parameters, then made a provision for user input to be given to a few variables. The parameters that were used were the following:

1. C_{ij} = Total cost of an item j in an i^{th} shop. Assigned for each food item in every outlet. If a food item doesn't exist, put 10M
2. D_i = Distance of the shop i from the user
3. L_{ij} = Rating of food item j in an i^{th} shop; zero if the food item isn't served in a particular outlet
4. E_{ij} =: Time efficiency of the i^{th} shop in preparing and serving a particular food item j
5. S_{ij} =: Stores the calories of each food item j, changes from outlet to outlet depending on the preparation
6. P_j = Standard preparation time of a particular food item j; represents perfect efficiency of preparation of the food
7. v = Average vehicular speed in Mumbai
8. R_{ij} = Desirability of j^{th} food item in shop i
9. T_{ij} = Total delivery time of j^{th} food item from i^{th} shop

   Now, we introduce the user's input parameters:
10. Y_j = User's cuisine preference input, it can be implemented as a hard filter or a soft preference – will be dwelled upon in the later sections.
11. Maximum wait time = t
12. Budget = b
13. Calorie Requirement = s

Calculations

Total time taken: $T_{ij} = P_j \cdot E^{-1}_{ij} + D_i \cdot v^{-1}$

Desirability: $R_{ij} = 0.6 \cdot L_{ij} + 2 \cdot Y_j$

Here, we have assigned weights to the user preference of cuisine of food j and the rating of the j^{th} food item in i^{th} shop to make a combination of this that results in desirability of a food item. Hence, we have here implemented a soft preference for the user's cuisine input.

Another method of inculcating cuisine input would be a hard filter- having Cij be a Boolean valued matrix and accept only 0's and 1's. Only the cuisines which are marked 1's food items will be considered in the problem from thereon, rest of the food items belonging to other cuisine types will be removed from consideration by either being given a 0 rating or a cost of 10M.

Though initially having implemented both, we chose the soft filter method as it worked better for us and we thought that would be a more realistic way of having desirability of a food item- as there are many food items that are a blend of cuisines and if an item has a very high rating, then it prods the user to try it out, though he might not like other food items of that cuisine.

The Problem

Decision Variable:

Decision = X

Objective Function:

Utility = $\Sigma^{n}_{i=1,j=1}\ R_{ij} \cdot X_{ij}$

Constraints:

Cost: $\Sigma^{n}_{i=1,j=1}\ C_{ij} \cdot X_{ij} \leq b$

Time: $\forall_{i,\,j}\ T_{ij} \cdot X_{ij} \leq t$

Satiability: $(1.2 \cdot s + 200) \geq \Sigma^{n}_{i=1,j=1}\ S_{ij} \cdot X_{ij} \geq s$

Though initially, our plan was to consider separate platforms separately, we realized that each of the platforms is essentially independent of each other and their margins, delivery time, the existence of a particular outlet on each platform etc., can all be adjusted in other matrix parameters that were defined. Hence, this simplified our problem statement quite a lot. The previous formulation of complicated constraints, parameters and decision variables then transformed into simple, concise, and easy to understand constraints with clean parameters as well. Hence, this formulation of the problem is now ready to implement on AMPL and python.

AMPL Code

- **Model file (.mod):**

```
set shop;
set foodItem;

param cost{i in shop, j in foodItem} >=0;
param time{i in shop, j in foodItem} >=0;
param rating{i in shop, j in foodItem} >=0;
param satiability{i in shop, j in foodItem} > 0;

param t;
param b;
param s;

var decision{i in shop, j in foodItem} >=0 binary;

maximize utility: sum{i in shop, j in foodItem} rating[i,j]*decision[i,j]  ;

subject to Cost: sum{i in shop, j in foodItem} cost[i,j]*decision[i,j] <= b;
subject to Time {i in shop, j in foodItem}: time[i,j]*decision[i,j] <= t;
subject to Satiability: sum{i in shop, j in foodItem} decision[i,j]*satiability[i,j] >= s;
#subject to diffShop {j in foodItem}: sum{i in shop} decision[i,j] <= 1;
```

- **Data file (.dat):**

  Attached at the end of the report

- **Run file (.run):**

```
reset;
option solver cplex;
model model1.mod;
data model1.dat;
solve;
display decision;
```

Algorithms

Greedy Algorithm

The metric that we used for the greedy algorithm was: *Rating / Cost.* The algorithm was implemented as follows:

1. Sort all items on the basis of this metric
2. Keep adding the items starting at the top, keeping in mind-
    - Adding an item shouldn't lead us to spend more than the user's budget
    - The time taken for the food to be delivered should be lesser than the limit imposed by the user
    - The item must exist, which means the rating of the item should be > 0

Ideally, we should end up with an order that maximises the average rating. However, the calorie requirement might not be met. In order to solve this problem, we consider another metric, that is: $(Calories \cdot Rating^2) / Cost$

1. Sort all items on the basis of this metric
2. From the initial order, remove the item that performs the worst on the basis of this metric
3. Search for the items that perform the best according to this metric which are not in the cart
4. Add these items, keeping in mind-
    - Adding an item shouldn't lead us to spend more than the user's budget
    - The time taken for the food to be delivered should be lesser than the limit imposed by the user
    - The item must exist, which means the rating of the item should be > 0

Keep replacing worse performing items in the cart with better performing ones subject to above conditions till we cross the minimum satiability threshold

Advantages of the Heuristic Approach

There are four major reasons behind the adoption of the heuristic approach:

Reliability: The heuristic approach provides an optimal solution over widely different operating conditions.

Flexibility: The heuristic approach makes it possible to try out various combinations of metrics for improving results.

Efficiency: It is efficient as it is of known time complexity: O(nlogn).

Easy to Use: It is easier to run diagnostics for someone who is not well-versed in LP optimisations.

Results

To test our program, we made use of an extensive dataset that consisted of 26 food items and 50 shops. We used datasets of the following parameters:

1. Cost: Consists of a 50x26 matrix of items ranging in cost from Rs. 150 to Rs. 650
2. Rating: Consists of a 50x26 matrix of ratings ranging from 1-5 with an average of 3 and a standard deviation of 1.28
3. Calories: Consists of a 50x26 matrix of items belonging to 5 different cuisines with an average of 511.39 calories per item and a standard deviation of 189.97
4. Processing time: Consists of a 26x1 matrix of standard processing times of various food items with an average of 28.85 minutes and a standard deviation of 9.47
5. Distance: Consists of a 50x1 matrix of distances of restaurants from IIT Bombay with an average distance of 11.09 Km and a standard deviation of 5.78 Km.
6. Shop efficiency: Consists of a 50x26 matrix of efficiency values ranging from 0-1 of a particular shop in preparing a certain food item.

Upon running our AMPL code as well as the greedy algorithm with user defined constraints of a budget of Rs. 1200, maximum waiting time of 41 minutes and a calorie requirement of 3200 calories the results obtained are displayed in the tables below:

Linear program:

Shop Number	Item Number	Cost	Rating	Calories
20	24	157	4.3	350
34	24	191	4.8	340
29	25	164	4.1	420
5	5	106	3.2	400
16	5	124	3.8	480
29	5	133	3.5	470
50	5	147	4.6	480
50	1	176	4.0	760

Optimal Order: Cost: Rs 1198 Average rating: 4.038

Calories: 3700

Greedy Algorithm:

Shop Number	Item Number	Cost	Rating	Calories
20	24	157	4.3	350
46	20	86	4.3	240
50	1	176	4.0	760
34	24	191	4.8	340
29	25	164	4.1	420
5	5	106	3.2	400
1	5	103	3.6	390
12	8	190	3.7	330

Optimal Order: Cost: Rs 1173 Average rating: 4.

Calories: 3230

It was found that there were 6 overlapping items in the cart results returned by the 2 algorithms.

Uncertainty Analysis

To take into account variability in food preparation time and uncertainties introduced by varying traffic conditions we performed an uncertainty analysis by carrying out an element wise multiplication of the processing time array and the distance array with random samples from a Gaussian distribution with mean 1 and standard deviation 0.12. This was done over 200 trials and we found the number of items delayed per order in each one of these trials. It was found that for the specified user constraints, in 28% of our orders 0 items were delayed, in 22% 1 item was delayed and in 50% of the orders 2 items were delayed. The results are shown in the histogram below:

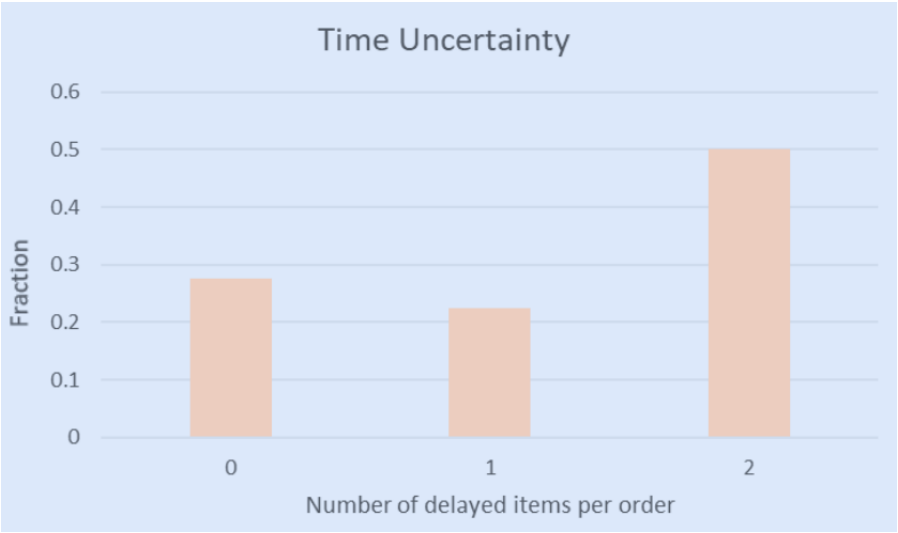

Figure 1. Histogram for time uncertainty analysis

To take into account variability introduced in calorie count during food preparation we performed an uncertainty analysis by carrying out an element wise multiplication of the processing time array and the distance array with random samples from a uniform distribution with lower bound 0.8 and upper bound 1.2. This was done over 200 trials and we found the number of items that go above or below the calorie specification, which was 2200 in our test. It was found that for the specified user constraints, 26% of our orders were below 2200 calories, 5% of our orders were between 2200 and 2300 calories and 69% of our orders were above 2300 calories. The results are shown in the histogram below:

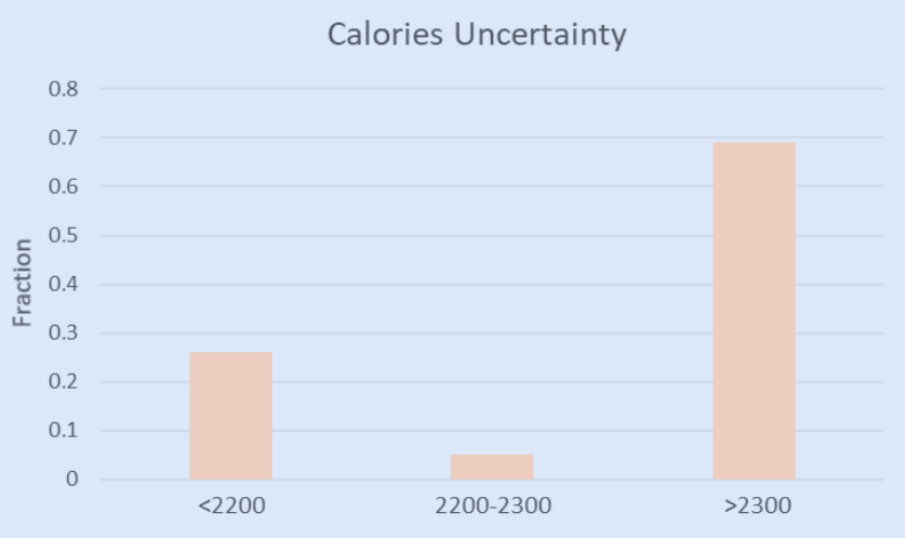

Figure 2. Histogram for calorie uncertainty analysis

Further Improvements and Additions

We plan on carrying out extensive tweaking of the metrics in the greedy algorithm to maximise accuracy and improve the performance. We also could use a larger, more realistic dataset for our analysis after web scraping from different information sites, location sites and food networks. Finally, we were thinking of integrating this model into a GUI to make it user-friendly. The user would input his budget and maximum time he is willing to wait, and give an (optional) cuisine preference – and he will get his optimal order(s) as an output. The insights from the uncertainty analysis could be made available to the different delivery platforms so that they could formulate offers and relook at their margins so that they can offer discounts without incurring losses (loosely can be modelled by a quadratic loss function)

We could then look at integrating the code into an app and a collaboration to develop a mobile application.

References & Codes

The link to our code:

https://drive.google.com/drive/folders/1s2KcgfnI_wh4F_0CpGK3VyEsgHmE22gt?usp=sharing

Greedy:

https://colab.research.google.com/drive/17sjFE1FIz77U03Xq0epIp4z4f3TSPlv5?usp=sharing

Uncertainty Analysis:

https://colab.research.google.com/drive/1K1CSnnW0lSSHeBicpEt1MEDuomk4ehKd?usp=sharing

Sites referred to for dataset creation:

https://en.wikipedia.org/wiki/Fast_food

https://www.sciencedirect.com/science/article/pii/S0749379714004000

https://www.google.com/maps/search/restaurants+near+me/@19.1457047,72.9106199,15z/data=!3m1!4b1

Zomato : for estimates of restaurant rating

Also made extensive use of the slides given by Prof. Avinash for AMPL learning