MGMTMSA 408 – Operations Analytics

Homework 4 – Revenue Management and Assortment Optimization (Answer Sheet)

Due June 3, 2020 at 1pm PST

Name: Tigran Margarian **UID:** 105451572

Note: Please submit your Python code on CCLE in order to receive full credit.

1 Cloud Computing

Part 1: Capacity control formulation

a) In this setting our goal is basically to decide on x_i to increase the overall revenue. Let's note the price of each instance as r_i. Then our objective function which we have to maximize would look like:

$$\sum_{i=1}^{9} r_i x_i$$

b) Overall there is 1024 GB of memory available. In that case the memory constraint would look like:

$$8x_1 + 16x_2 + 32x_3 + 32x_4 + 64x_5 + 128x_6 + 16x_7 + 32x_8 + 64x_9 \leq 1024$$

c) Assuming that instance requests follow the Poisson distribution with λ being represented by Rate (# / day) the expected number of requests for instance 5 over 5 day interval would be equal to $5\lambda_5 = 5 \times 2.6 = 13$.

The corresponding constraint on the request numbers would look like:

$$x_5 \leq 13$$

d) The LP formulation for T = 5 capacity control problem:

$$maximize \sum_{i=1}^{9} r_i x_i$$

subject to:

$8x_1 + 16x_2 + 32x_3 + 32x_4 + 64x_5 + 128x_6 + 16x_7 + 32x_8 + 64x_9 \leq 1024$ (memory constraint)

$16x_1 + 32x_2 + 64x_3 + 8x_4 + 16x_5 + 32x_6 + 16x_7 + 32x_8 + 64x_9 \leq 512$ (CPU constraint)

$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + 2x_7 + 6x_8 + 8x_9 \leq 64$ (GPU constraint)

$x_1 \leq 25$; $x_2 \leq 25$; $x_3 \leq 9$; $x_4 \leq 15$; $x_5 \leq 13$; $x_6 \leq 5$; $x_7 \leq 4$; $x_8 \leq 2$; $x_9 \leq 1.5$
(demand constraints)

$x_i \geq 0,\ \forall i \in \{1, ..., 9\}$ (non-negativity constraints)

Part 2: Solving the capacity control problem in Python/Gurobi

a) Optimal objective value (revenue) equals \$1039.43

b) There according to the optimizer output there are 6.29 C1 instance requests accepted.

c) It doesn't make sense to add more GPU capacity because according to the optimized solution we still have 17.29 GBs of GPU memory unused.

d) Using the shadow prices of CPU and memory we can calculate that the considered server would bring us \$9 of additional revenue

Part 3: Simulating current practice

a) The average number of arrivals of type C1 instance in the set of simulated sequences is 26.63

b) The average number of arrivals of all types of instances over the set of simulated sequences is 101.

We know that for random variables obeying the Poisson distribution X and Y the random variable Z = X + Y would have a Poisson distribution as well with the expected value E(Z) = E(X) + E(Y). Therefore the obtained number makes perfect sense because it's almost the same as the sum for forecasts for all the instance types (99.5) and that is expected by theory.

c) The average revenue generated by the myopic strategy is \$528.28

d) The average remaining capacities are: 0.24 CPUs, 371.52 GBs of memory and 37.42 GBs of GPU memory. It seems like this following this strategy we quickly run out of CPUs and are getting left with plenty of memory and GPU capacity.

Part 4: A bid-price control policy

a) For an instance of type 5 (M2) requiring 16 CPUs, 64 GBs of memory and 1 GB of GPU capacity the opportunity cost in terms of shadow prices would look like:

$$Cost\ of\ M2\ instance\ acceptance = 16\pi_1 + 64\pi_2 + \pi_3$$

b) The expected number of requests of instance i with Poisson arrival rate of λ_i from time t to T is $(T - t + 1) \times \lambda_i$

c) The average revenue generated by the bid-price control strategy is \$925.59 which is much closer to the *ideal* result that the myopic strategy revenue.

d) The average remaining capacities are: 27.2 CPUs, 4.88 GBs of memory and 20.62 GBs of GPU memory. The distribution of capacity is much more even with bid-price control policy.

2 Designing a Sushi Menu

Part 1: Understanding the data

a) The top five sushis preferred by customer 1 are:

- negi-toro *(category 1)* - utility = 2.911
- ika *(category 5)* - utility = 2.808
- maguro *(category 1)* - utility = 2.806
- samon *(category 1)* - utility = 2.772
- inari *(category 11)* - utility = 2.751

Seems like customer 1 is a big fan of sushis belonging to the category 1 (tuna/salmon sushis).

b) The worst five sushis for customer 2 are:

- karasumi *(category 7)* - utility = -0.404
- komochi-konbu *(category 7)* - utility = -0.317
- kyabia *(category 7)* - utility = -0.258
- awabi *(category 4)* - utility = -0.257
- kazunoko *(category 7)* - utility = -0.231

We can clearly see that customer 2 hates category 7 sushis (almost all of them include roe or caviar).

c) The top five sushis by average ranking (with rank = 1 as best and rank = 100 as worst) are:

- negi-toro *(category 1)* - avg. ranking = 7.876
- maguro *(category 1)* - avg. ranking = 10.050
- kurumaebi *(category 6)* - avg. ranking = 10.084
- negi-toro-maki *(category 1)* - avg. ranking = 10.214
- ebi *(category 6)* - avg. ranking = 11.932

We can see that the top-5 sushis for all the customers surveyed are belonging either to category 1 or 6. Category 1 us characterized by including tuna and category 6 sushis mostly include shrimp.

d) The sushi with the worst average ranking is **kyabia** with average ranking value of 90.648.

e) The most controversial sushi based on ranking standard deviation is **awabi** with a ranking SD of 27.439 and average ranking value of 62.592.

Part 2: Common-sense solutions

a) The expected revenue from offering all the sushis is \$21.51

b) The ten most expensive sushis are: **toro, awabi, tarabagani, akagai, umi, kurumaebi, chu-toro, suzuki, hirame, botanebi**.

The expected revenue with this set is equal to \$25.64

c) The expected revenue from offering the customers' favorite sushis is \$21.51.

Interestingly that is the same result we obtained from offering all the sushis. I believe it happens because from the whole variety of sushis we offer in part a), customers choose one bringing the highest utility for them. In part c) we explicitly arrange the menu in such way to include the favorite sushi of each customer. Therefore, in both cases customers end up with the same choice delivering the same revenue to the company.

d) There are several reasons why offering all sushis would be suboptimal strategy. First of all, clearly there are some sushis that would never be ordered but their offering will have an impact on the budget of the restaurant. Second, by offering everything we provide our customers an opportunity to choose the highest utility option which may be actually not very profitable for the restaurant. If in turn we could find a set what would still provide an option with a utility higher than 3 (no option) and additionally boost the expected profit maximally - that would provide a more optimal solution.

e) The approach of offering the most expensive and revenue driving sushis may be suboptimal because usually there is a negative correlation between utility of a good and its price. Customers don't quite enjoy paying much and therefore leaving only expensive choices my force many customers to leave the restaurant without an order what will in turn decrease the potential revenue.

Part 3: An integer optimization model

a) In this model $y_{k,i}$ is a binary variable which equals to 1 if k-th customer chooses the i-th sushi and zero otherwise. x_i is binary variable indicating whether we include the i-th sushi in the menu. r_i is the price of i-th sushi and finally $u_{k,j}$ is the utility of j-th sushi for the k-th customer. Let's go over the considered constraints:

- The 1b) constraint means that each customer can choose only one option (order one type of sushi or walk away).
- The 1c) constraint means that the utility of the option chosen by each customer should be greater or equal than the utility of other options (that said each customer's choice has the highest utility available).
- The 1d) constraint means that the utility of the option chosen by each customer should be greater or equal then the utility of non-purchase option (that said each customer's choice shouldn't have a lower utility than the non-purchase option).
- The 1e) constraint means that it's impossible for the customer to choose an option not offered by the restaurant.

b) The optimized revenue from LP relaxation equals to \$30.59.

c) That is impossible because even the linear LP relaxation gave the optimal revenue of \$30.59 per customer. That means that the integer LP output cannot be greater and therefore a set of menu delivering \$32 of per-customer revenue cannot exist.

d) The optimized revenue from integer LP equals to \$26.24 what is \$4.73 more than the result of common-sense strategy of offering all sushis (or only customers' favorites) and \$0.60 more than the result of expensive sushi offering. We should also keep in mind that this is the per-customer level revenue which means that 60 additional cents from customer can convert into much higher number depending on how much clients the restaurant can serve.

e) The optimal set of sushis now includes 7 options: **toro, awabi, saba, tarabagani, mentaiko-maki, ika-nattou, tobiuo**.

Part 4: A constrained model

a) First we should reshape the category data into 2 dimensional matrix $c_{100,12}$ of binary variables where $c_{i,c} = 1$ if the i-th sushi belongs to the category c and zero otherwise. With that we can add the necessary constraint which will look like:

$$\sum_{i=1}^{n} x_i c_{i,c} \geq 1 \ \ \forall \ c \in \{1, 2, ..., 12\}$$

This condition will force the optimizer to keep at least one sushi of each category.

b) The optimized revenue from constrained integer LP equals to \$25.88.

c) The optimal set of sushis now includes 19 options: **tamago, toro, awabi, shako, saba, unagi, geso, tarabagani, ankimo, gyusashi, fugu, okura, ika-nattou, kaiware, kue, kamo, mamakari, kyabia, kaki.**

Part 5: Next steps

a) Some limitations of this model I can think about are:

- The fact that we assume that each customer is allowed to purchase only one single sushi. It may easily be the case in real conditions that some customers would choose more than one sushi.
- The equal probability of each customer. The created model assumes that there is an equal probability of each customer to show up what may turn out to be wrong.
- Finally, we assume that the survey conducted with 500 customers should be descriptive for all future customers and that also may turn out to be wrong.

b) One approach which may help to reduce the number of customers considered may be clustering. We can implement clustering algorithms like K-Means or Factor Analysis to group customers with similar preferences. That may in turn significantly simplify calculations for the optimizer.