Pizza Lover

Group members
Yuechen Tang (tang.1394)
Jiayi Zhou (zhou.2802)

Introduction

Every time when we want to order pieces of pizza, it's often hard to decide how much to order to satisfy everyone's flavor request. Think about the situations that people may order too less so that fight for the last piece of a certain flavor, or order too much so that there are many leftover pieces of another flavor. Both situations will result in unhappiness.

Our original thought was that ordering from one store with regular price and flavors, and another store with pizza of mix flavors. When we collected the price and flavor information for pizzas of mixed flavors, we found that there was only one store that contained mixed of two-flavor pizza and served with only medium size. Since its price for one mixed flavor pizza was more expansive than two of the regular pizzas, we decided not to optimize with this situation.

We then choose two stores where we often order pizzas from: the Domino's Pizza and the Happy's Pizza.

Domino's Pizza	F1-Chicken Taco	F2-Cheeseburger	F3-Ultimate Pepperoni	F4-Hawaiian	F5-Spinach&Feta	F6-Buffalo Chicken	Delivery fee
Small(10")	$7.99	$7.99	$7.99	$7.99	$7.99	$7.99	$3.99
Medium(12")	$13.99	$13.99	$13.99	$13.99	$13.99	$13.99	
Large(14")	$17.99	$17.99	$17.99	$17.99	$17.99	$17.99	
X-Large(16")	$19.99	$19.99	$19.99	$19.99	$19.99	$19.99	
Happy's Pizza							
Small(10")	$8.99	$8.99	$8.99	$10.99	$10.99	$8.99	$6.99
Medium(12")	$10.99	$10.99	$10.99	$12.99	$12.99	$10.99	
Large(14")	$12.99	$12.99	$12.99	$14.99	$14.99	$12.99	
X-Large(16")	$15.25	$15.25	$15.25	$17.25	$17.25	$15.25	

Figure 1. The prices of Domino's Pizza and Happy's Pizza

For these two pizza stores, we have found some tricky aspects:

- Domino's Pizza is cheaper in small and medium size.
- Happy's Pizza is cheaper in large and X-large size.
- Hawaiian pizza and Spinach & Feta pizza in Happy's Pizza have higher prices than other flavors in Happy's Pizza.
- Domino's Pizza has cheaper delivery fee.

When facing with various sizes and flavors of pizzas on the menu, we often have to

think about how to order to achieve the minimization of the total price.

For this project, we want to develop a pizza calculator to help people exactly know how much to order for each flavor, and how to order in two stores to reach the minimum price.

Problem setting

Suppose one small size pizza can server for two people:

$$\text{Area} = \pi*(10/2)^2 \approx 79.$$

So setting 39.5 inch^2 of pizza as a suitable amount for one person.

- One small pizza for 79/39.5 = 2 people.
- One medium pizza for 113/39.5 = 2.86 ≈ 3 people.
- One large pizza for 154/39.5 = 3.90 ≈ 4 people.
- One X-large pizza for 201/39.5 = 5.09 ≈ 5 people.

Decision Variables

P_i = # of people who want flavor i

$N_{j,\,n,\,i}$ = # of pizza with size n and flavor i in store j

$$D_j = \begin{cases} 1, & pizza\ is\ ordered\ in\ store\ j \\ 0, & otherwise \end{cases}$$

Flavor i = 1 for Chicken Taco flavor, i = 2 for Cheeseburger flavor, i = 3 for Ultimate Pepperoni flavor, i = 4 for Hawaiian flavor, i = 5 for Spinach & Feta flavor, and i = 6 for Buffalo Chicken flavor.

Store j = 1 for Domino‘s Pizza, and j = 2 for Happy’s Pizza.

Size n = 1 for small size, n = 2 for medium size, n = 3 for large size, n = 4 for X-large size.

Since Python does not support expressions of dimension greater than 2, we adjusted the variable $N_{j,\,n,\,i}$ into the following two:

- $N_{n,\,i}$ = # of pizza with size n and flavor i in Domino’s Pizza
- $M_{n,\,i}$ = # of pizza with size n and flavor i in Happy’s Pizza

```
N = cp.Variable((4, 6), integer  = True) # store1
M = cp.Variable((4, 6), integer  = True) # store2
P = cp.Variable(6, integer  = True)
D = cp.Variable(2, boolean = True)
```

Figure 2. Python code for decision variables

Objective function

Setting Z as the price of ordering pizza, and we want to minimize the value of Z. So Z is the price of pizza of each size and flavor multiplies the number of pizza ordered for each size and flavor, and then adds the delivery fee for two stores.

- $\min Z = 7.99 * \sum_{i=1}^{6} N1,\ i + 13.99 * \sum_{i=1}^{6} N2,\ i + 17.99 * \sum_{i=1}^{6} N3,\ i + 19.99 * \sum_{i=1}^{6} N4,\ i + 8.99 * \sum_{i=1}^{6} M1,\ i + 10.99 * \sum_{i=1}^{6} M2,\ i + 12.99 * \sum_{i=1}^{6} M3,\ i + 15.25 * \sum_{i=1}^{6} M4,\ i + 2*(\sum_{n=1}^{4} Mn,\ 4 + \sum_{n=1}^{4} Mn,\ 5\ ) + 3.99*D_1 + 6.99*D_2$

```
obj_func = 7.99 * (N[0,0]+N[0,1]+N[0,2]+N[0,3]+N[0,4]+N[0,5]) + 13.99 * (N[1,0]+N[1,1]+N[1,2]+N[1,3]+N[1,4]+N[1,5]) +
           17.99 * (N[2,0]+N[2,1]+N[2,2]+N[2,3]+N[2,4]+N[2,5]) + 19.99 * (N[3,0]+N[3,1]+N[3,2]+N[3,3]+N[3,4]+N[3,5]) +
           8.99 * (M[0,0]+M[0,1]+M[0,2]+M[0,3]+M[0,4]+M[0,5]) + 10.99 * (M[1,0]+M[1,1]+M[1,2]+M[1,3]+M[1,4]+M[1,5]) +
           12.99 * (M[2,0]+M[2,1]+M[2,2]+M[2,3]+M[2,4]+M[2,5]) + 15.25 * (M[3,0]+M[3,1]+M[3,2]+M[3,3]+M[3,4]+M[3,5]) +
           2 * (M[0,3] + M[1,3] + M[2,3] + M[3,3] + M[0,4] + M[1,4] + M[2,4] + M[3,4]) +
           3.99 * D[0] + 6.99 * D[1]
```

Figure 3. Python code for objective function

Constraints

Every time we use this model, customers will give the value of P_i, which is the number of people who want flavor i. We will set the value of P_i in constraints as the given values from customers

```
constraints = []

constraints.append(P[0]== 2)
constraints.append(P[1]== 0)
constraints.append(P[2]== 4)
constraints.append(P[3]== 1)
constraints.append(P[4]== 0)
constraints.append(P[5]== 0)
```

Figure 4. Python code for P_i for example 1

Within the given values of P_i, we need to ensure that the number of pizzas satisfy the customers' requirements. In the problem setting part, we have already set the number of people that each size of pizza can serve.

For each of the six flavors, the number of pizzas ordered in one flavor in all four sizes and both two stores ($N_{n,\,i}$ and $M_{n,\,i}$) multiply with the number of people that each size of pizza can serve respectively must be greater than or equal to the number of people who want this flavor (P_i).

- $2 * N_{1,\,1} + 3 * N_{2,\,1} + 4 * N_{3,\,1} + 5 * N_{4,\,1} + 2 * M_{1,\,1} + 3 * M_{2,\,1} + 4 * M_{3,\,1} + 5 * M_{4,\,1} \geq P_1$

- $2 * N_{1,2} + 3 * N_{2,2} + 4 * N_{3,2} + 5 * N_{4,2} + 2 * M_{1,2} + 3 * M_{2,2} + 4 * M_{3,2} + 5 * M_{4,2} \geq P_2$
- $2 * N_{1,3} + 3 * N_{2,3} + 4 * N_{3,3} + 5 * N_{4,3} + 2 * M_{1,3} + 3 * M_{2,3} + 4 * M_{3,3} + 5 * M_{4,3} \geq P_3$
- $2 * N_{1,4} + 3 * N_{2,4} + 4 * N_{3,4} + 5 * N_{4,4} + 2 * M_{1,4} + 3 * M_{2,4} + 4 * M_{3,4} + 5 * M_{4,4} \geq P_4$
- $2 * N_{1,5} + 3 * N_{2,5} + 4 * N_{3,5} + 5 * N_{4,5} + 2 * M_{1,5} + 3 * M_{2,5} + 4 * M_{3,5} + 5 * M_{4,5} \geq P_5$
- $2 * N_{1,6} + 3 * N_{2,6} + 4 * N_{3,6} + 5 * N_{4,6} + 2 * M_{1,6} + 3 * M_{2,6} + 4 * M_{3,6} + 5 * M_{4,6} \geq P_6$

For delivery fee, if the total number of pizzas ordered in Domino's Pizza is greater than 0, then D_1 will be greater than 0. If the total number of pizzas ordered in Happy's Pizza is greater than 0, then D_2 will be greater than 0.

- If $\sum_{i=1}^{6} \sum_{n=1}^{4} Nn,\ i > 0$, then $D_1 > 0$.
- If $\sum_{i=1}^{6} \sum_{n=1}^{4} Mn,\ i > 0$, then $D_2 > 0$.

P_i, $M_{n,i}$, $N_{n,i}$ are non-negative integers, i = 1, 2, ⋯, 6, n = 1, 2, 3, 4.
D_1, D_2 are binary variables.

```
for i in [0,1,2,3,4,5] :
    constraints.append(2 * (N[0,i] + M[0,i]) + 3 * (N[1,i] + M[1,i]) + 4 * (N[2,i] + M[2,i]) + 5 * (N[3,i] + M[3,i]) >= P[i])

constraints.append(D[0] <= N[0,0]+N[0,1]+N[0,2]+N[0,3]+N[0,4]+N[0,5] + N[1,0]+N[1,1]+N[1,2]+N[1,3]+N[1,4]+N[1,5]
                  + N[2,0]+N[2,1]+N[2,2]+N[2,3]+N[2,4]+N[2,5] + N[3,0]+ N[3,1]+ N[3,2]+ N[3,3]+ N[3,4]+ N[3,5])
constraints.append(D[1] <= M[0,0]+M[0,1]+M[0,2]+M[0,3]+M[0,4]+M[0,5] + M[1,0]+M[1,1]+M[1,2]+M[1,3]+M[1,4]+M[1,5]
                  + M[2,0]+M[2,1]+M[2,2]+M[2,3]+M[2,4]+M[2,5] + M[3,0]+ M[3,1]+ M[3,2]+ M[3,3]+ M[3,4]+ M[3,5])
constraints.append(D[0] >= (N[0,0]+N[0,1]+N[0,2]+N[0,3]+N[0,4]+N[0,5] + N[1,0]+N[1,1]+N[1,2]+N[1,3]+N[1,4]+N[1,5]
                   + N[2,0]+N[2,1]+N[2,2]+N[2,3]+N[2,4]+N[2,5] + N[3,0]+ N[3,1]+ N[3,2]+ N[3,3]+ N[3,4]+ N[3,5])/1000)
constraints.append(D[1] >= (M[0,0]+M[0,1]+M[0,2]+M[0,3]+M[0,4]+M[0,5] + M[1,0]+M[1,1]+M[1,2]+M[1,3]+M[1,4]+M[1,5]
                    + M[2,0]+M[2,1]+M[2,2]+M[2,3]+M[2,4]+M[2,5] + M[3,0]+ M[3,1]+ M[3,2]+ M[3,3]+ M[3,4]+ M[3,5])/1000)

for n in [0,1,2,3]:
    for i in [0,1,2,3,4,5]:
        constraints.append(N[n,i]>=0)
        constraints.append(M[n,i]>=0)
```

Figure 5. Python code for constraints

Results

Example 1:

Two people want Chicken Taco pizza, four people want Ultimate Pepperoni pizza, and one person wants Hawaiian pizza.

First input the values of P_i according to the customer's requirements:

- $P_1 = 2$, $P_2 = 0$, $P_3 = 4$, $P_4 = 1$, $P_5 = 0$, $P_6 = 0$.

```
constraints = []

constraints.append(P[0]== 2)
constraints.append(P[1]== 0)
constraints.append(P[2]== 4)
constraints.append(P[3]== 1)
constraints.append(P[4]== 0)
constraints.append(P[5]== 0)
```

Figure 6. Python code for P_i for example 1

Then run the program, the optimal model will give out the minimum price of $35.95 for purchasing one small size Chicken Taco pizza, two small size Ultimate Pepperoni pizzas, and one small size Hawaiian pizza from Domino's Pizza.

```
obj_func =
35.95
N =
[[ 1. -0.  2.  1. -0. -0.]
 [-0. -0. -0. -0. -0. -0.]
 [-0. -0.  0. -0. -0. -0.]
 [-0. -0. -0. -0. -0. -0.]]
M =
[[ 0. -0. -0.  0. -0. -0.]
 [-0. -0. -0. -0. -0. -0.]
 [-0. -0.  0. -0. -0. -0.]
 [-0. -0. -0. -0. -0. -0.]]
P =
[ 2. -0.  4.  1. -0. -0.]
D =
[1. 0.]
```

Figure 7. Optimal result for example 1

Example 2:

Three people want Chicken Taco pizza, three people want Cheeseburger pizza, five people want Ultimate Pepperoni pizza, three people want Hawaiian pizza, six people want Spinach & Feta pizza, and three people want Buffalo Chicken pizza.

First input the values of P_i according to the customer's requirements:

- $P_1 = 3$, $P_2 = 3$, $P_3 = 5$, $P_4 = 3$, $P_5 = 6$, $P_6 = 3$.

```
constraints = []

constraints.append(P[0]== 3)
constraints.append(P[1]== 3)
constraints.append(P[2]== 5)
constraints.append(P[3]== 3)
constraints.append(P[4]== 6)
constraints.append(P[5]== 3)
```

Figure 8. Python code for P_i for example 2

Then run the program, the optimal model will give out the minimum price of $94.18 for purchasing one medium size Chicken Taco pizza, one medium size Cheeseburger pizza, one X-large size Ultimate Pepperoni pizza, one medium size Hawaiian pizza, one small size Spinach & Feta pizza, one large size Spinach & Feta pizza, and one medium size Buffalo Chicken pizza from Happy's Pizza.

```
obj_func =
94.17999999999999
N =
[[ 0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.]
 [-0. -0. -0. -0.  0. -0.]
 [-0. -0.  0. -0. -0. -0.]]
M =
[[ 0.  0.  0.  0.  1.  0.]
 [ 1.  1.  0.  1.  0.  1.]
 [-0. -0. -0. -0.  1. -0.]
 [-0. -0.  1. -0. -0. -0.]]
P =
[3. 3. 5. 3. 6. 3.]
D =
[0. 1.]
```

Figure 9. Optimal result for example 2

Example 3

Four people want Chicken Taco pizza, seven people want Cheeseburger pizza, one person wants Ultimate Pepperoni pizza, six people want Hawaiian pizza, and two people want Spinach & Feta pizza.

First input the values of P_i according to the customer's requirements:

- $P_1 = 4$, $P_2 = 7$, $P_3 = 1$, $P_4 = 6$, $P_5 = 2$, $P_6 = 0$.

```
constraints = []

constraints.append(P[0]== 4)
constraints.append(P[1]== 7)
constraints.append(P[2]== 1)
constraints.append(P[3]== 6)
constraints.append(P[4]== 2)
constraints.append(P[5]== 0)
```

Figure 10. Python code for P_i for example 3

Then run the program, the optimal model will give out the minimum price of $86.17 for purchasing one small size Cheeseburger pizza, one small size Ultimate Pepperoni pizza, one small size Hawaiian pizza, and one small size Spinach & Feta pizza from Domino's Pizza, one large size Chicken Taco pizza, one X-large size Cheeseburger pizza, one large size Hawaiian pizza from Happy's pizza.

```
obj_func =
86.16999999999999
N =
[[ 0.  1.  1.  1.  1. -0.]
 [-0.  0. -0.  0. -0. -0.]
 [ 0.  0. -0.  0. -0. -0.]
 [-0.  0. -0. -0. -0. -0.]]
M =
[[ 0.  0.  0.  0.  0. -0.]
 [-0.  0. -0.  0. -0. -0.]
 [ 1.  0. -0.  1. -0. -0.]
 [-0.  1. -0. -0. -0. -0.]]
P =
[ 4.  7.  1.  6.  2. -0.]
D =
[1. 1.]
```

Figure 11. Optimal result for example 3

Conclusion

For this project, we take price, size, and flavors into consideration. While some customers may want to add toppings, try more flavors, or order pizza with other beverage sets, our model can improve to satisfy the customers' needs in the future.

Our first priority is always to help customers keep the cost at the minimum when ordering pizzas.

Appendix

Youtube video link: https://www.youtube.com/watch?v=jLlDSWQ3xDU&t=1s

*Mistake correction of presentation video: there's a typo in the Constraints part. It should be "non-negative integers" instead of "negative integers".

Github link: https://github.com/tristatyc/ise3230-project/tree/main

Tasks:

Yuechen Tang

- Project proposal writing
- Python code writing & discussion
- Presentation slides writing
- Presentation recording
- Project report writing
- Project review

Jiayi Zhou

- Project proposal discussion
- Data collection
- Python code writing & discussion
- Presentation recording
- Project report discussion
- Project review