Optimization Application Project – Customer Scheduling / Assignment

GEN BUS 730 – Fall 2020

Shih-Yuan Wang

What is the problem?

Family Customer Scheduling/Assignment

To schedule family customers to Santa's Workshop in a way that minimizes the cost of the Workshop:

- An Amusement park is launching a new activity, Santa's Workshop tours, to increase customers' willingness to come during the 100 days before Christmas.
- Because demand was so strong, and the park wanted to make things as fair as possible, they let each of the 5,000 families that will visit the workshop choose top 10 preferences for the dates they'd like to attend the workshop.
- Now that the park have received their preferences, they've realized it's impossible for every family to get their top picks, so they've decided to provide extra perks for families that don't get their preferences.
- In addition, the park 's accounting department has evaluated that, depending on how families are scheduled, there may be some unexpected costs incurred.

Reference: https://www.kaggle.com/c/santa-workshop-tour-2019/overview

https://www.iconparkorlando.com/entertainment/santa-experience/

Why is it important and interesting?

- It requires some strategies to linearize nonlinear programming problem, and it's common in real business settings.
- We need to solve large number of variables and might have some issues that we haven't encountered in the course.
- Business/Workshop might save a huge amount of costs if they can consider and estimate "unexpected costs" and optimize their decisions.

Who would care?

- Customers: can ensure everyone has an opportunity to attend and has good customer experience by limiting the number of people attending.
- Business/Workshop: find an efficient way to both minimize total costs and satisfy customer needs.

Data Description and Problem Setting

Data Description & Problem Setting

- **Data set and problem setting**: Kaggle Competition - Santa's Workshop Tour 2019 data set https://www.kaggle.com/c/santa-workshop-tour-2019/data
- **5,000 families** have listed their top **10 preferences** (choice 0 - 9) for the dates they'd like to attend Santa's workshop tour.
- Dates (**1-100**) are integer values representing the **days *before* Christmas**, e.g., the value 1 represents Dec 24, the value 2 represents Dec 23, etc.
- Each family also has a number of people attending (n_people).
- Every family must be scheduled for one and **only one assigned day**.
- The **total number of people** attending the workshop each day must be **between 125 – 300**.
- **Objective**: minimize total costs of workshop (Preference Cost and Accounting Penalty – described later)

Reference: https://www.kaggle.com/c/santa-workshop-tour-2019/data

Data Overview

Top 10 Preferred Days before Christmas

Number of People Attending

family_id	choice_0	choice_1	choice_2	choice_3	choice_4	choice_5	choice_6	choice_7	choice_8	choice_9	n_people
0	52	38	12	82	33	75	64	76	10	28	4
1	26	4	82	5	11	47	38	6	66	61	4
2	100	54	25	12	27	82	10	89	80	33	3
3	2	95	1	96	32	6	40	31	9	59	2
4	53	1	47	93	26	3	46	16	42	39	4
5	32	59	12	3	60	26	35	50	5	2	4
6	88	4	1	3	91	32	39	57	28	99	2
7	25	11	52	48	10	17	88	50	95	66	5
8	18	60	1	12	89	33	16	10	53	67	4
9	1	88	39	50	26	18	96	47	46	28	7
10	96	92	8	5	67	12	57	34	80	46	7

Model Design and Building

Optimization Tools: Python – Pyomo Package

- Efficiently solving very large-scale problems.
- Easy to make modifications if the situation changes.
- The code can be re-runnable, reusable, and reproducible.
- Open-source software package for optimization models: Pyomo
- Pyomo supports dozens of solvers, both open source and commercial, including many solvers supported by AMPL, PICO, CBC, CPLEX, IPOPT, Gurobi and GLPK.
- Using linear programming solver can guarantee to find the global optimal solution.

Reference: https://en.wikipedia.org/wiki/Pyomo

Optimization Objective 1 – Minimize Preference Cost

- **Preference Cost**: Santa Workshop provides consolation gifts (of varying value) to families according to their assigned day relative to their preferences. These sum up per family, and the total represents the preference cost.
  - ❑ choice_0: no consolation gifts
  - ❑ choice_1: one $50 gift card to Santa's Gift Shop
  - ❑ choice_2: one $50 gift card, and 25% off Santa's Buffet (value $9) for each family member
  - ❑ choice_3: one $100 gift card, and 25% off Santa's Buffet (value $9) for each family member
  - ❑ choice_4: one $200 gift card, and 25% off Santa's Buffet (value $9) for each family member
  - ❑ choice_5: one $200 gift card, and 50% off Santa's Buffet (value $18) for each family member
  - ❑ choice_6: one $300 gift card, and 50% off Santa's Buffet (value $18) for each family member
  - ❑ choice_7: one $300 gift card, and free Santa's Buffet (value $36) for each family member
  - ❑ choice_8: one $400 gift card, and free Santa's Buffet (value $36) for each family member
  - ❑ choice_9: one $500 gift card, and free Santa's Buffet (value $36) for each family member, and 50% off Simulated North Pole Helicopter Ride tickets (value $199) for each family member
  - ❑ otherwise: one $500 gift card, and free Santa's Buffet (value $36) for each family member, and free Simulated North Pole Helicopter Ride tickets (value $398) for each family member

Reference: https://www.kaggle.com/c/santa-workshop-tour-2019/overview/evaluation

Optimization Objective 2 – Minimize Both Preference Cost and Accounting Penalty

- **Accounting Penalty**: The park's accountants have also developed an empirical equation for cost that arise from many different effects such as reduced shopping in the Gift Shop when it gets too crowded, extra cleaning costs, etc. This cost is in addition to the consolation gifts (**Preference Cost**), and is defined as:

$$accounting\ penalty = \sum_{d=100}^{1} \frac{(N_d - 125)}{400} N_d \left(\frac{1}{2} + \frac{|N_d - N_{d+1}|}{50}\right)$$

Absolute Occupancy Difference between Current and Previous Day

- **Nd** is the occupancy of the current day, and **Nd+1** is the occupancy of the *previous* day (we're counting backwards from Christmas). For the initial condition of d=100, N101=N100. (It starts on the date 100 days before Christmas and ends on Christmas Eve.)

Ultimate Objective:

Total Costs = Preference Cost + Accounting Penalty

Reference: https://www.kaggle.com/c/santa-workshop-tour-2019/overview/evaluation

Model Parameters Overview

Preference Cost Matrix by Assigned Days per Family:

Days before Christmas

family_id	1	2	3	4	5	6	7	8	9	10	...	91	92	93	94	95	96	97	98	99	100
0	2236	2236	2236	2236	2236	2236	2236	2236	2236	544	...	2236	2236	2236	2236	2236	2236	2236	2236	2236	2236
1	2236	2236	2236	50	136	444	2236	2236	2236	2236	...	2236	2236	2236	2236	2236	2236	2236	2236	2236	2236
2	1802	1802	1802	1802	1802	1802	1802	1802	1802	354	...	1802	1802	1802	1802	1802	1802	1802	1802	1802	0
3	68	0	1368	1368	1368	236	1368	1368	472	1368	...	1368	1368	1368	1368	50	118	1368	1368	1368	1368
4	50	2236	272	2236	2236	2236	2236	2236	2236	2236	...	2236	2236	136	2236	2236	2236	2236	2236	2236	2236

Accounting Penalty Equation:

Model 2: Minimize Sum of Absolute Occupancy Difference over 100 Days

$$accounting\ penalty = \sum_{d=100}^{1} \frac{(N_d - 125)}{400} N_d \left(\frac{1}{2} + \frac{|N_d - N_{d+1}|}{50}\right)$$

Model 3: Absolute Occupancy Difference for Each Day <= 25

Decision Variables and Constraints

- **DVs**: Whether family assigned to that day? (**Binary** value; Total: **5,000 families X 100 assigned days**)

	Assigned Day														
family_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

- **Constraint 1**: Every family must be scheduled for one and only one assigned day.
- **Constraint 2**: The total number of people attending the workshop (occupancy) each day must be between 125 – 300.

Model 1

Objective: Minimize "Preference Cost"

- Firstly, we tried to ignore the Accounting Penalty and minimize only "Preference Cost“.
- Declare the **decision variables**: Whether family assigned to each of 100 days? (binary)
- Specify the **objective**: Minimize preference cost.
- Specify the **constraint 1 and 2**.
- Use Pyomo '**glpk**' solver to solve, and specify time limit 10,000 seconds as it takes so long time to find the optimal solution. (could allow more time to get better solution)
- Leverage optimization algorithm(s): **Linear** and **mixed integer** programming
  - ❑ Binary decision variables
  - ❑ Linear constraint and objective functions

Model 1: Solution

Objective: Minimize "Preference Cost"

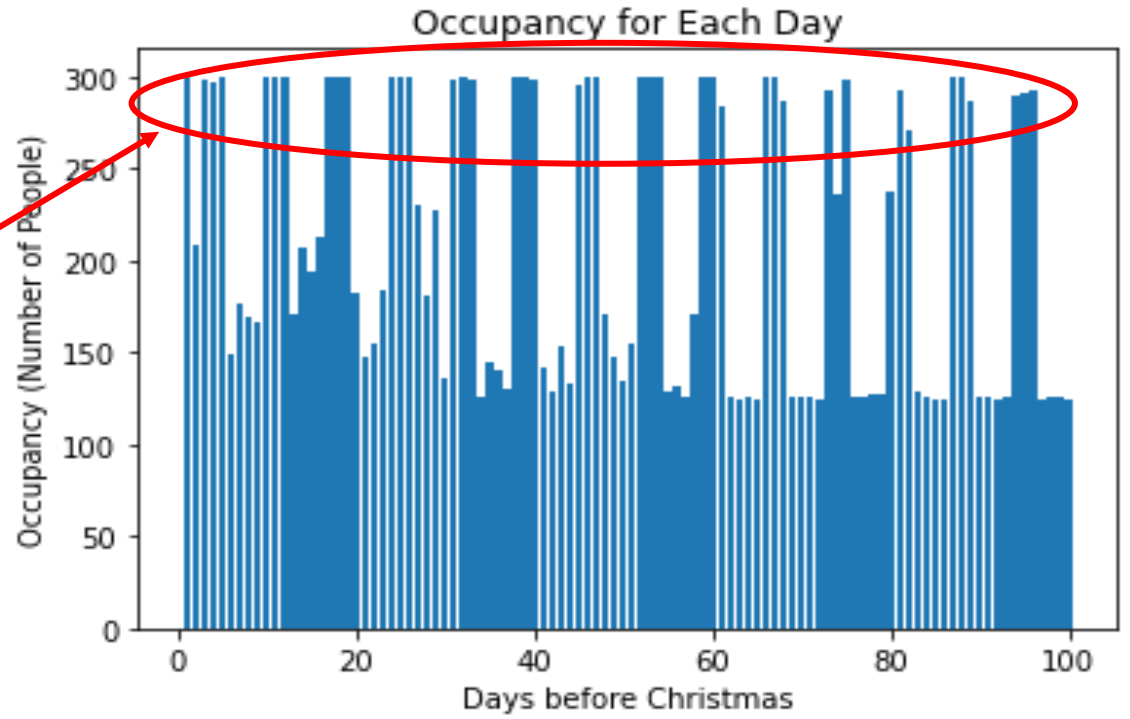
- Total preference cost: $48,918
- Total accounting penalty: $10,677,776,967
- Total costs (preference cost + accounting penalty): $10,677,825,885

The accounting penalty is large, just minimizing the preference cost is not enough.

The assigned day for each family (extract)

family_id	assigned_day
0	52
1	26
2	100
3	2
4	53
...	...
20	3
35	31
42	89
47	45
128	56

There is strong preference for certain days, and the demand variability is high.



Add Decision Variables and Constraints

- **Add DVs**: Positive and negative part of (current day occupancy - previous day occupancy) (**Non-negative integer**; Total: **2** (positive and negative parts) **X 100 assigned days**)

→ **Model 2**: To minimize sum of |Nd-Nd+1| (total absolute occupancy difference over 100 days)
→ **Model 3**: To limit |Nd-Nd+1| for each day <= Maximum Occupancy Difference 25 (selected by trials)

	Assigned Day														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Occupancy difference (Nd-Nd+1)	-6	2	4	0	-4	4	0	0	-3	3	0	-4	4	0	0
Positive part	0	2	4	0	0	4	0	0	0	3	0	0	4	0	0
Negative part	6	0	0	0	4	0	0	0	3	0	0	4	0	0	0

- **Constraint 3:** Positive - Negative part of (Nd-Nd+1) = Current day occupancy - Previous day occupancy
- **Constraint 4:** Positive part of (Nd-Nd+1) <= Maximum Occupancy Difference 25;
  Negative part of (Nd-Nd+1) <= Maximum Occupancy Difference 25

→ **Model 2:** add **Constraint 3**
→ **Model 3 :** add **Constraint 3 & 4**

Model 2

Objective: Minimize “Preference Cost” & "Partial Accounting Penalty"

Sum of Absolute Occupancy Difference

- To try to also minimize Accounting Penalty but **keep it linear**, let's consider minimizing "total |Nd-Nd+1| over the 100 assigned days" (sum of |current day occupancy - previous day occupancy|) first.
- Declare **decision variables**:
  - ❑ Whether family assigned to each of 100 days? (binary)
  - ❑ Positive and negative parts of occupancy difference. (non-negative integer)
- Specify the **objective**: Minimize preference cost and total absolute occupancy difference.
- Specify the **constraint 1, 2, and 3**.
- Use Pyomo '**glpk**' solver to solve, and specify time limit 10,000 seconds as it takes so long time to find the optimal solution. (could allow more time to get better solution)
- Leverage optimization algorithm(s): **Linear** and **mixed integer** programming
  - ❑ Binary and non-negative integer decision variables
  - ❑ Separate positive and negative parts of occupancy difference to get the absolute value
  - ❑ Linear constraint and objective functions

Model 2: Solution

Objective: Minimize "Preference Cost“ & "Partial Accounting Penalty"

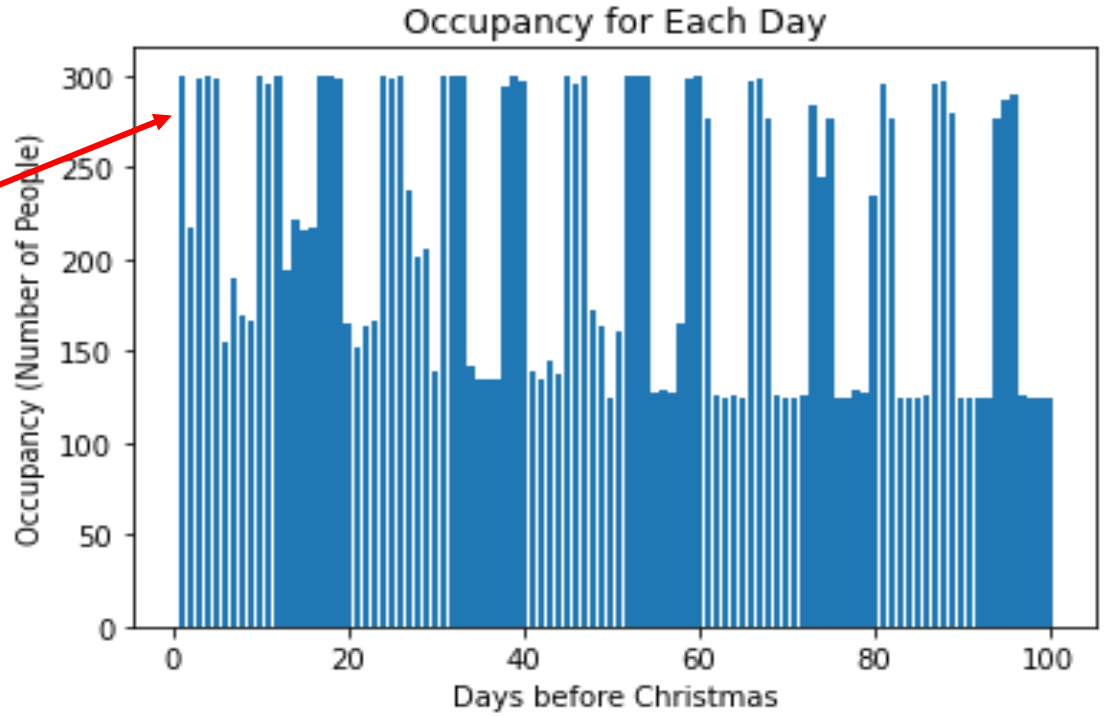
- Total preference cost: $48,517
- Total accounting penalty: $5,718,658,622
- Total costs (preference cost + accounting penalty): $5,718,707,139

The accounting penalty decreases a lot, but it is still quite large.

The assigned day for each family (extract)

family_id	assigned_day
0	52
1	26
2	100
3	2
4	53
20	88
35	98
42	81
47	19
128	72

Also quite high demand variability



Model 3

Objective: Minimize “Preference Cost” by limiting Accounting Penalty

Absolute Occupancy Difference for each day <= 25

- To try to minimize Accounting Penalty but **keep it linear**, this time consider limiting "|Nd-Nd+1|" for each day (|current day occupancy - previous day occupancy| <= Occupancy Difference Max 25).
- Declare **decision variables**:
  - ❑ Whether family assigned to each of 100 days? (binary)
  - ❑ Positive and negative part of occupancy difference. (non-negative integer)
- Specify the **objective**: Minimize preference cost.
- Specify the **constraint 1, 2, 3, and 4**.
- Use Pyomo '**glpk**' solver to solve, and specify time limit 22,000 seconds as it takes so long time to find the optimal solution. (could allow more time to get better solution)
- Leverage optimization algorithm(s): **Linear** and **mixed integer** programming
  - ❑ Binary and non-negative integer decision variables
  - ❑ Separate positive and negative parts of occupancy difference to get the absolute value
  - ❑ Linear constraint and objective functions

Model 3: Solution

Objective: Minimize "Preference Cost" by limiting Accounting Penalty

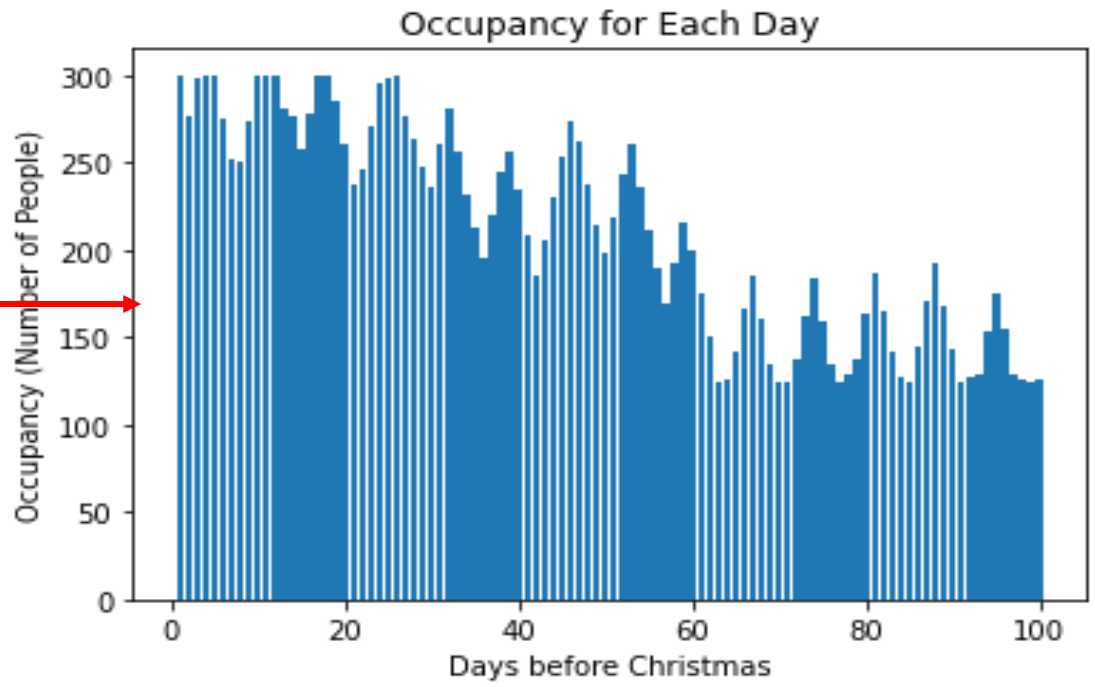
- Total preference cost: $106,186 ↑
- Total accounting penalty: $3,163 ↓
- Total costs (preference cost + accounting penalty): $109,349 ↓

A huge improvement on accounting penalty at the cost of small increase in preference cost

The assigned day for each family (extract)

	assigned_day	occupancy
0	1	300.0
1	2	276.0
2	3	298.0
3	4	300.0
4	5	300.0
95	96	154.0
96	97	129.0
97	98	126.0
98	99	125.0
99	100	126.0

More stable occupancy for each day, the low and high peaks are smoother than the Model 1 and 2.



Comparison of Models

	Model 1	Model 2	Model 3
Programs	LP / MIP	LP / MIP	LP / MIP
Python Pyomo solver	GLPK	GLPK	GLPK
Decision Variables	5,000 families x 100 days	5,000 families x 100 days; positive and negative parts of (Nd-Nd+1) * 100 days	5,000 families x 100 days; positive and negative parts of (Nd-Nd+1) * 100 days
Objective: Minimize	Preference cost	Preference cost + sum of |Nd-Nd+1|	Preference cost
Constraint	one assigned day; occupancy 125-300	one assigned day; occupancy 125-300; pos (Nd-Nd+1) - neg (Nd-Nd+1) = (Nd-Nd+1)	one assigned day; occupancy 125-300; pos (Nd-Nd+1) - neg (Nd-Nd+1) = (Nd-Nd+1); pos (Nd-Nd+1) <= 25, neg (Nd-Nd+1) <= 25
Preference cost	**48,918**	**48,517**	**106,186**
Accounting penalty	**10,677,776,967**	**5,718,658,622**	**3,163**
Total costs	**10,677,825,885**	**5,718,707,139**	**109,349**

What does the model(s) look like in Python?

E.g. Model 3

```
# declare a concrete model
model = ConcreteModel()

# declare the decision variables:

# Whether family assigned to that day?
model.x = Var(range(n_family), range(n_assignedday), domain = Binary) # (n_family * n_assignedday)

# positive and negative part of (Nd-Nd+1) -> to limit |current day occupancy - previous day occupancy| <= Occupancy Difference Mc
model.y = Var(range(2), range(n_assignedday), domain = NonNegativeIntegers)   # 2 (pos and neg part) * n_assignedday

# f refers to the family_id
# d refers to the assigned day
# c refers to the choice

# specify the objective: Minimize Preference Cost
model.Objective = Objective(expr = sum(model.x[f,d]*prefCost_matrix[f][d] for f in range(n_family) for d in range(n_assignedday))

#-------------------------

# Constraint: Every family must be scheduled for one and only one assigned day
model.OneAssignedDay =  ConstraintList()
for f in range(n_family): # for each family
    model.OneAssignedDay.add(expr = sum(model.x[f,d] for d in range(n_assignedday)) == 1)

# Constraint: The total number of people attending the workshop each day must be between 125 - 300

# lower bound occupancy constraint
model.OccupancyMin =  ConstraintList()
for d in range(n_assignedday): # for each assigned day
    model.OccupancyMin.add(expr = sum(model.x[f,d]*n_people[f] for f in range(n_family)) >= min_occupancy)

# upper bound occupancy constraint
model.OccupancyMax =  ConstraintList()
for d in range(n_assignedday): # for each assigned day
    model.OccupancyMax.add(expr = sum(model.x[f,d]*n_people[f] for f in range(n_family)) <= max_occupancy)
```

What does the model(s) look like in Python?

E.g. Model 3 Outputs

```
# save OccupancyperDay_M3 to a dataframe

assigneddayList = list(range(1, n_assignedday+1))
occupancy_M3_dict = {"assigned_day": assigneddayList, "occupancy": OccupancyperDay_M3}
df_sol_occupancy_M3 = pd.DataFrame(occupancy_M3_dict)
df_sol_occupancy_M3
```

	assigned_day	occupancy
0	1	300.0
1	2	276.0
2	3	298.0
3	4	300.0
4	5	300.0
...	...	...
95	96	154.0
96	97	129.0
97	98	126.0
98	99	125.0
99	100	126.0

100 rows × 2 columns

```
# Total Cost Summary

# Total preference cost
print("Total preference cost:", totPrefCost_M3) # 106186.0

# Total accounting penalty
print("Total accounting penalty:", totAccPenalty_M3)  # 3162.9877796508863

# Total costs (preference cost + accounting penalty)
print("Total costs (preference cost + accounting penalty):", totPrefCost_M3+totAccPenalty_M3) # 109348.98777965088
```

```
Total preference cost: 106186.0
Total accounting penalty: 3162.9877796508863
Total costs (preference cost + accounting penalty): 109348.98777965088
```

Assumptions for the Models

- The actual other incurred costs, such as reduced shopping in the Gift Shop and extra cleaning costs, are equal to the cost calculated from Accounting Penalty equation.
- Families are still willing to attend Santa Workshop tour if they are assigned to the date that is not their top preference.
- Not considering the change in expected purchases in the Gift Shop from families if they are not assigned to their top choices.
- Assume that customers will indeed use consolation gifts (gift card, buffet, simulated North Pole helicopter ride) provided from Santa Workshop.

Project Reflection

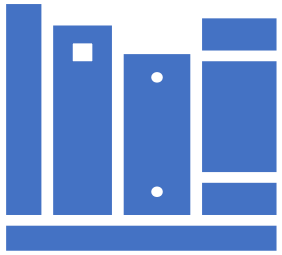
Challenges I faced

- Due to the curse of dimensionality, it took much time to find a solution and haven't gotten an optimal one yet (have tried to run more than 12 hours).
- Had difficulty finding a way to linearize Accounting Penalty.
- Tried to use OR-Tools to optimize the problem but encountered some difficulties.

Impressive thing I did

- Built several models to dig into the problem and compared their performance.
- Was able to convert partial non-linear term into linear form.
- Learned how to add more "solver options" to tackle the problem.

Project Applicability for Clients



Example we can overview: Disney Parks

- Disney parks may host special events to reward frequent visitors and attract more potential customers.
- Instead of saving spots for customers who register early, Disney wants to guarantee that some most frequent visitors can get their event tickets for their preferred days.
- They also offer special discounts or gift cards for frequent visitors who can’t get their preferences to increase their willing to attend as they found that these customers also have higher purchasing power.
- To both deliver great customer experience and maximize profits, Disney wants to schedule customers fairly but consider potential incurred costs and expected sales from this special event.

Limitations of the Project

- It takes a long time for the solver to find an optimal solution.
- There exists speed-accuracy tradeoff if we want to solve more variables.
- If the empirical equation for accounting penalty changes, we need to find another way to linearize it, add more constraints, or set different starting points to solve nonlinear problem.

Projected Benefits of the Project

- Solve a large-scale customer scheduling problem.
- Save a huge amount of cost but still ensure great customer experience.
- Increase work efficiency by using powerful optimization solver.

Next steps I would take…

- Try to use other faster solver software, such as OR-Tools or Gurobi.
- Try other approaches to “linearize” the Accounting Penalty and set different constraints to see whether there is an improvement.
- Spend more time until the real “optimal” solution found.
- Consider expected purchase sales to maximize total profits, not just minimizing costs.

Appendix. Supporting Files

Supporting Files

- Excel file: Simplified Model for Starters.xlsx
  - ❑ Sheet “Simplified”: A simplified version of the problem
  - ❑ Sheet “Comparison”: Comparison of models from Python
- Jupyter Notebook (coding scripts): Customer Scheduling Assignment.ipynb
  - ❑ Part A and B: Import libraries and data preparation
  - ❑ Part C: Model 1
  - ❑ Part D: Model 2
  - ❑ Part E: Model 3 (best in this setting)
  - ❑ Part F: Appendix (Trial and Error)
- CSV files:
  - ❑ Input data: family_data.csv
  - ❑ Solution outputs: Model 1, 2, and 3 family assignment, and Model 3 daily occupancy (solution_M1_prefonly.csv, solution_M2_prefPartacc.csv, solution_M3_prefLimitAcc.csv, solution_M3_occupancy.csv)

References

- Kaggle Competition: https://www.kaggle.com/c/santa-workshop-tour-2019/overview
- Code:
  - https://www.kaggle.com/inversion/santa-s-2019-starter-notebook
  - https://www.kaggle.com/c/santa-workshop-tour-2019/discussion/120764
- Python Pyomo:
  - https://pyomo.readthedocs.io/en/stable/index.html
  - https://en.wikipedia.org/wiki/Pyomo
- Ipopt Documentation: https://coin-or.github.io/Ipopt/OPTIONS.html
- Pictures:
  - https://www.iconparkorlando.com/entertainment/santa-experience/
  - https://www.hathaway-sycamores.org/2012/01/thank-you-for-brightening-the-holidays/paspolice-helicopter-santa/
  - https://www.alignable.com/port-saint-lucie-fl/santas-workshop-warehouse-experience/christmas-tour-of-santas-workshop
  - http://en.artandlogic.co.rs/how-to-hire-us
  - https://www.businessinsider.com/disney-plus

Presentation End
Thank You & Merry Christmas!