ISYE 6501 Homework 7

Due: July 8, 2021

1 Optimization

1.1 Question 15.2

1.1.1 Part 1

Using a Python LP modeling library, PuLP, I could model the classic "diet problem". In the 30s and 40s the Army aimed to meet the nutritional requirements of its soldiers while minimizing costs. Using data in the *diet.xls* spreadsheet, I solved a linear program to produce the serving size amounts of each food in the spreadsheet that met the given constraints for each nutrient/nutritional attribute (i.e. calories, total fat, carbohydrates etc.) and minimized the objective function shown in Equation 1 where x_i is the amount of food i in the daily diet and c_i is the per-unit cost of food i.

$$Minimize \quad \sum_i c_i x_i \tag{1}$$

My Python code which followed a similar method to [1] can be found in Appendix A. The results of the optimization claims that each soldier should consume the following foods and amounts shown in Table 1 below. The total cost came out to be \$4.34 per soldier.

Type of food	# of servings	Serving size
Raw celery	52.6	1 Stalk
Frozen broccoli	0.260	10 Oz Pkg
Raw iceberg lettuce	64.0	1 Leaf
Oranges	2.29	1 Frt,2-5/8 Diam
Poached eggs	0.142	Lrg Egg
Air popped popcorn	13.9	1 Oz

Table 1: Optimization results using given constraints for all nutrients.

1.1.2 Part 2

Next, for comparison, I solved the exact same LP model with the additional constraint that if a given food item is selected, a minimum of 1/10 of a serving must be consumed. With this constraint comes adding a binary variable: whether or not food i is selected. Now, if it is selected, the optimizer needs to use at least 1/10 of a serving. Because of this addition, I needed to add an additional constraint to link the two variables. See Appendix B for this additional code block

Additionally, I used two more constraints before optimizing the new model. Because many people dislike celery and frozen broccoli, only one of the two items can be selected. Equation 2 below shows the mathematical constraint equation where b_i is the binary indicator whether food i is included in the diet or not.

$$b_{frozenbroccoli} + b_{celery} \leq 1 \tag{2}$$

Lastly, I added the constraint that a soldier needs to consume protein from at least three different sources/types of protein-rich food. Here, I consider protein-rich foods to be any meat, poultry, fish or eggs but something with meat as a topping such as bean and bacon soup is not considered. Equation 3 shows this mathematically where j is now the subset of protein-rich foods.

$$\sum_j b_j \geq 3 \tag{3}$$

After re-running the optimizer, as expected, I found that the price per soldier per day rose to $4.51 which isn't a notable increase given the three new constraints. The summary of the new foods in an optimized diet is shown in Table 2. As it can be seen, all three constraints are met and new foods are introduced to find the optimal solution (which inevitably increases cost with additional constraints).

Type of food	# of servings	Serving size
Raw celery	42.4	1 Stalk
Kielbasa Pork	0.100	1 Sl,6x3-3/4x1/16 In
Raw iceberg lettuce	82.8	1 Leaf
Oranges	3.08	1 Frt,2-5/8 Diam
Peanut butter	1.94	2 Tbsp
Poached eggs	0.100	Lrg Egg
Air popped popcorn	13.2	1 Oz
Scrambled eggs	0.100	1 Egg

Table 2: Optimization results using three new constraints.

References

[1] https://towardsdatascience.com/linear-programming-and-discrete-optimization-with-python-using-pulp-449f3c5f6e99

Appendix A Optimization code

```
from pulp import *
import pandas as pd

prob = LpProblem("diet_problem", LpMinimize)

df = pd.read_excel("diet.xls", nrows=64)

print(df.columns)
food_list = list(df['Foods'])

cost_dict = dict(zip(food_list, df['Price/ Serving']))
calories_dict = dict(zip(food_list, df['Calories']))
cholesterol_dict = dict(zip(food_list, df['Cholesterol mg']))
fat_dict = dict(zip(food_list, df['Total_Fat g']))
sodium_dict = dict(zip(food_list, df['Sodium mg']))
carbs_dict = dict(zip(food_list, df['Carbohydrates g']))
fiber_dict = dict(zip(food_list, df['Dietary_Fiber g']))
protein_dict = dict(zip(food_list, df['Protein g']))
vita_dict = dict(zip(food_list, df['Vit_A IU']))
vitc_dict = dict(zip(food_list, df['Vit_C IU']))
calcium_dict = dict(zip(food_list, df['Calcium mg']))
```

```
iron_dict = dict(zip(food_list, df['Iron mg']))

# Variable
food_vars = LpVariable.dicts("Food", food_list, lowBound=0, cat='Continuous')

# Objective function
prob += lpSum([cost_dict[i]*food_vars[i] for i in food_list])

# Constraints

# Calories
prob += lpSum([calories_dict[f] * food_vars[f] for f in food_list]) >= 1500.0, 'CalsMinimun
prob += lpSum([calories_dict[f] * food_vars[f] for f in food_list]) <= 2500.0, 'CalsMaximun

# Cholesterol
prob += lpSum([cholesterol_dict[f] * food_vars[f] for f in food_list]) >= 30.0, 'CholMinimu
prob += lpSum([cholesterol_dict[f] * food_vars[f] for f in food_list]) <= 240.0, 'CholMaxim

# Fat
prob += lpSum([fat_dict[f] * food_vars[f] for f in food_list]) >= 20.0, 'FatMinimum'
prob += lpSum([fat_dict[f] * food_vars[f] for f in food_list]) <= 70.0, 'FatMaximum'

# Sodium
prob += lpSum([sodium_dict[f] * food_vars[f] for f in food_list]) >= 800.0, 'SodiumMinimum
prob += lpSum([sodium_dict[f] * food_vars[f] for f in food_list]) <= 2000.0, 'SodiumMaximun

# Carbs
prob += lpSum([carbs_dict[f] * food_vars[f] for f in food_list]) >= 130.0, 'CarbsMinimum'
prob += lpSum([carbs_dict[f] * food_vars[f] for f in food_list]) <= 450.0, 'CarbsMaximum'

# Fiber
prob += lpSum([fiber_dict[f] * food_vars[f] for f in food_list]) >= 125.0, 'FiberMinimum'
prob += lpSum([fiber_dict[f] * food_vars[f] for f in food_list]) <= 250.0, 'FiberMaximum'

# Protein
prob += lpSum([protein_dict[f] * food_vars[f] for f in food_list]) >= 60.0, 'ProteinMinimun
prob += lpSum([protein_dict[f] * food_vars[f] for f in food_list]) <= 100.0, 'ProteinMaximu

# Vitamin A
prob += lpSum([vita_dict[f] * food_vars[f] for f in food_list]) >= 1000.0, 'VitaMinimum'
prob += lpSum([vita_dict[f] * food_vars[f] for f in food_list]) <= 10000.0, 'VitaMaximum'

# Vitamin C
prob += lpSum([vitc_dict[f] * food_vars[f] for f in food_list]) >= 400.0, 'VitcMinimum'
prob += lpSum([vitc_dict[f] * food_vars[f] for f in food_list]) <= 5000.0, 'VitcMaximum'

# Calcium
prob += lpSum([calcium_dict[f] * food_vars[f] for f in food_list]) >= 700.0, 'CalciumMinimu
prob += lpSum([calcium_dict[f] * food_vars[f] for f in food_list]) <= 1500.0, 'CalciumMaxim

# Iron
prob += lpSum([iron_dict[f] * food_vars[f] for f in food_list]) >= 10.0, 'IronMinimum'
prob += lpSum([iron_dict[f] * food_vars[f] for f in food_list]) <= 40.0, 'IronMaximum'
```

```python
prob.solve()
print("Status:", LpStatus[prob.status])

for v in prob.variables():
    if v.varValue>0:
        print(v.name, "=", v.varValue)

obj = value(prob.objective)
print("The total cost of this balanced diet is: ${}".format(round(obj,2)))
```

Appendix B Additional constraints code

```python
food_chosen = LpVariable.dicts("Chosen",food_list, 0, 1, cat='Integer')

# New constraint of serving size
for f in food_list:
    prob += food_vars[f]>= food_chosen[f]*0.1
    prob += food_vars[f]<= food_chosen[f]*1e6

# New either/or constraint
prob += food_chosen['Frozen Broccoli'] + food_chosen['Celery, Raw'] <= 1

# New constraint on protein variety
prob += food_chosen['Roasted Chicken'] + food_chosen['Poached Eggs'] + food_chosen['Scramb
        food_chosen['Bologna,Turkey'] + food_chosen['Frankfurter, Beef'] + food_chosen['Ham
        food_chosen['Kielbasa,Prk'] + food_chosen['Hamburger W/Toppings'] + food_chosen['H
        food_chosen['Pork'] + food_chosen['Sardines in Oil'] + food_chosen['White Tuna in 
```