Bob Lin
Math381
Homework4

The problem is to find out the shortest path of m cities among n cities that I want to visit, the shorter the path, the more likely I will choose to visit. I will list several cities that I would like to visit and choose fixed amount of them that take least time to visit. To better make the plan, I will also make a time bound to limit the total time for me to travel.

First, I choose 10 countries with direct distance between them, and here is some variables related to the graph:

Let $j = 1, \dots, n$ to denote each city a.

Let t_{ab} be the time in hour it takes to travel between city a and city b.

Let r_j be the rating for city j

This table shows the number representation for each city:

j	City name	rating
1	Beijing	11
2	Shenzhen	11
3	Shanghai	9
4	Seoul	7
5	Paris	8
6	Frankfurt	7
7	Montreal	3
8	Urumqi	9
9	Dubai	8
10	Bangkok	5

Here is the list of these cities with traveling time between each city, and city was written in decreasing order of their rating (ex: first city rating is the highest, and last city rating

lowest, and so on):

C2 \ C1	BeiJing	ShenZhen	ShangHai	Seoul	Paris	Frankfurt	Montreal	Urumqi	Dubai	Bangkok
BeiJing	0	3.17	2.08	2.08	11.8	16.5	17.25	3.92	14.67	5.08
Shenzhen	3.17	0	2.25	8.42	18.42	18.75	22.75	5.33	20.08	7.58
ShangHai	2.08	2.25	0	1.75	12.33	11.75	19.25	5.08	21	4.25
Seoul	2.08	8.42	1.75	0	12.17	11.5	17.92	9.25	14.75	5.67
Paris	11.08	18.42	12.33	12.17	0	3.72	7.42	19.67	10.25	15.42
Frankfurt	16.5	18.75	11.75	11.5	3.72	0	7.67	22.58	6.33	2.08
Montreal	17.25	22.75	19.25	17.92	7.42	7.67	0	29	19.42	20.33
Urumqi	3.92	5.33	5.08	9.25	19.67	22.58	29	0	19.33	10.33
Dubai	14.67	20.08	21	14.75	10.25	6.33	19.42	19.33	0	6.33
Bangkok	5.08	7.58	4.25	5.67	15.42	2.08	20.33	10.33	6.33	0

All data comes from google map, time is calculated in time of plane except for Pairs to Frankfurt which has train only

Some time may be longer than before due to coronavirus

To choose places to visit, I will define an LP to help me find out the path that can maximize the rating of my travel:

First, we build a complete graph G = (V, E) with n vertices which represent cities in the list above

Let V = {1,…,n}, and edge set E defined as:

$$E = \{(a, b): a, b \in V,\ a \neq b\}$$

To solve the problem, first I will define several helpful variables for the LP:

Define i = 1,…,m to be the steps I took in my travel

Define binary variable $x_{ij} \in \{0,\ 1\},\ and\ let\ x_{ij} = 1$ iff we are at city j at step i, 0 otherwise

Define binary variable $e_{ab} \in \{0,\ 1\},\ and\ let\ e_{ab} = 1$ iff we travel from city a to city b in one step, 0 otherwise

Define S to be my starting point and ending point

Define S_j to be the time used to travel between S and city j

Define t_{bound} be the maximum time we plan to take

In this question, I choose 10 cities in my favorite list, so n = 10. Then, I want to choose 5 of them to visit, which is m = 5. Let S to be Seattle, then the value from S_1 to S_j is:

{18.08, 23.58, 18, 11.42, 12.25, 12.17, 9.92, 25, 22, 17.92}

Also, the rating array is:

{11, 11, 9, 7, 8, 7, 3, 9, 8, 5}

Then, our objective function is:

$$\sum_{j=1}^{n} r_j x_{mj}$$

Our goal is to maximize this function in order to get the greatest rating of the 5 cities we choice.

Then, I need constraints to limit all variables

First, for each of 5 steps, I only want to visit one city, so we need:

$$\sum_{j=1}^{n} x_{ij} = 1 \ for\ all\ i$$

Second, each city can only be visited at most one time, so we need:

$$\sum_{i=1}^{m} x_{ij} \leq 1 \ for\ all\ j$$

Third, we want e_{ab} = 1 iff we visit city a in a step i and we visit b in the step i + 1, this can be achieve by:

$$e_{ab} \geq x_{ia} + x_{i+1,b} - 1, i = 1, \dots, m-1 \ for\ all\ a\ and\ b\ with\ a \neq b$$

Plus, in this constraint, because e_{ab} can equal one even when we don't satisfy the condition. However, because there is a bound in next constraint, we want e_{ab} the least the better, so LP solver will take it zero when possible.

Finally, in order to limit the time use in traveling, I will set a time bound by:

$$\sum_{1 \leq a,b \leq n}^{n} e_{ab} t_{ab} + \sum_{j=1}^{n} (x_{mj} + x_{1j}) S_j \leq t_{bound}\ with\ a \neq b$$

By combing all of variables and constraints above, I use python to create LP solver shown below:

```
n = 10
m = 5
time_limit = 50;

bj = [0, 3.17, 2.08, 2.08, 11.8, 16.5, 17.25, 3.92, 14.67, 5.08]
sz = [3.17, 0, 2.25, 8.42, 18.42, 18.75, 22.75, 5.33, 20.08, 7.58]
sh = [2.08, 2.25, 0, 1.75, 12.33, 11.75, 19.25, 5.08, 21, 4.25]
se = [2.08, 8.42, 1.75, 0, 12.17, 11.5, 17.92, 9.25, 14.75, 5.67]
pa = [11.08, 18.42, 12.33, 12.17, 0, 3.72, 7.42, 19.67, 10.25, 15.42]
fr = [16.5, 18.75, 11.75, 11.5, 3.72, 0, 7.67, 22.58, 6.33, 2.08]
mt = [17.25, 22.75, 19.25, 17.92, 7.42, 7.67, 0, 29, 19.42, 20.33]
ur = [3.92, 5.33, 5.08, 9.25, 19.67, 22.58, 29, 0, 19.33, 10.33]
db = [14.67, 20.08, 21, 14.75, 10.25, 6.33, 19.42, 19.33, 0, 6.33]
bk = [5.08, 7.58, 4.25, 5.67, 15.42, 2.08, 20.33, 10.33, 6.33, 0]

rating = [11, 11, 9, 7, 8, 7, 3, 9, 8, 5]
'''start and ending city'''
sj = [18.08, 23.58, 18, 11.42, 12.25, 12.17, 9.92, 25, 22, 17.92]

city = [bj, sz, sh, se, pa, fr, mt, ur, db, bk]

'''objective function'''
output = "max: "
for i in range(1, m + 1):
    for j in range(1, n + 1):
        output = output + "+" + str(rating[j - 1]) + "x_" + str(i) + "_" + str(j)
print(output + ";")

'''constraints 1'''
for i in range(1, m + 1):
    c1 = ""
    for j in range(1, n + 1):
        c1 = c1 + "+" + "x_" + str(i) + "_" + str(j)
    print(c1 + "=1;")

'''constraints 2'''
for j in range(1, n + 1):
    c2 = ""
    for i in range(1, m + 1):
        c2 = c2 + "+" + "x_" + str(i) + "_" + str(j)
```

```
        print(c2 + "<=1;")


'''constraints 3'''
for i in range(1, m):
    for a in range(1, n + 1):
        for b in range(1, n + 1):
            if a != b:
                c3 = ""
                c3 = c3 + "+" + "e_" + str(a) + "_" + str(b)
                c3 = c3 + ">="
                c3 = c3 + "+" + "x_" + str(i) + "_" + str(a)
                c3 = c3 + "+" + "x_" + str(i + 1) + "_" + str(b)
                print(c3 + "-1;")


'''distance bound'''
bound = ""
for a in range(1, n + 1):
    if sj[a - 1] != 0:
        bound = bound + "+" + str(sj[a - 1]) + "x_" + str(m) + "_" + str(a)
        bound = bound + "+" + str(sj[a - 1]) + "x_1" + "_" + str(a)
    for b in range(1, n + 1):
        if a != b:
            bound = bound + "+" + str(city[a - 1][b - 1]) + "e_" + str(a) + "_" + str(b)
print(bound + "<=" + str(time_limit) + ";")


'''variables'''
var = "bin "
for a in range(1, n + 1):
    for b in range(1, n + 1):
        if a != b:
            var = var + "e_" + str(a) + "_" + str(b) + ","
for i in range(1, m + 1):
    for j in range(1, n + 1):
        var = var + "x_" + str(i) + "_" + str(j) + ","
var = var + "x_" + str(m) + "_" + str(n)
print(var + ";")
```

This creates the following lp input file:

max: +11x_1_1+11x_1_2+9x_1_3+7x_1_4+8x_1_5…;

(50 items about $r_j x_{mj}$, $for\ all\ m\ and\ j$ to calculate the sum of our rating)

…

(5 lines of following types:

ensure that only one place in each step is 1, and other is 0)

+x_1_1+x_1_2+x_1_3+x_1_4+x_1_5+x_1_6+x_1_7+x_1_8+x_1_9+x_1_10=1;

…

(10 lines of following types:

ensure that for a same place we only visit it once)

+x_1_1+x_2_1+x_3_1+x_4_1+x_5_1<=1;

…

(450 lines of following types:

ensure that e_{ab} = 1 iff we visit place a in a step i and place b in step i + 1)

+e_1_2>=+x_1_1+x_2_2-1;

…

(108 items of following types:

ensure that total distance is smaller than the bound given)

+18.08x_5_1+18.08x_1_1+3.17e_1_2+2.08e_1_3…;

This creates following solution:

Value of objective function: 23.34000000

Value of objective function: 47.00000000

Actual values of the variables:

x_1_1	1
x_2_2	1
x_3_3	1
x_4_8	1
x_5_4	1

e_1_2 1

e_2_3 1

e_3_8 1

e_8_4 1

(All other variables are 0)

Thus, the route of my trip is(the city number written in first page):

Seattle->1->2->3->8->4->Seattle, which is:

Seattle -> Beijing ->Shenzhen -> Shanghai -> Urumqi -> Seoul -> Seattle

This is actually a good trip, because the time is below the bound, it allows me to visit my top 4 rating cities and the last one is rating 7 which is also a good rating. So the trip must be interesting!

For different m will create different results, here I will use 15m + 10 for time bound:

m = 2: Seattle->5->6->Seattle

m = 3: Seattle->1->2->3->Seattle

m = 4: Seattle->1->3->8->2->Seattle

m = 5: Seattle->1->2->3->8->5->Seattle

(we see that the route change as time bound change)

m = 6: Seattle->1->8->3->2->5->9->Seattle

m = 7: Seattle->1->3->4->8->2->5->9->Seattle

m = 8: Seattle->1->5->6->8->4->3->2->9->Seattle

m = 9: Seattle->1->4->2->8->3->10->6->5->9->Seattle

m = 10: Seattle->5->6->9->4->1->7->8->3->2->10->Seattle

this shows that 1 is always the best choice, and when the distance is satisfied, it always gives me the city with greatest rating. Here, when m increases, the time-bound increases, and then the rating will increase as a result. Then, it we make the time-bound to infinity, we can always get the first m rating city as result. At first, I pick 50 as my initial time bound because I think 10 hours for each city is a good bound. But then I meet the

problem that 20 cannot find result for 2 city, so I decide to use 15m + 10, because some city is too far away. I see most of them help me find the city with high rating, and it always gives me the result to visit Beijing and Shenzhen which are the highest rating cities in my plan.

The relationship between number of cities I choice and the total rating seems to be linear.