Finding shortest path connecting m cities

Jingyi Cui

Due: 5/1

Contents

Background Information

Even though I live in China for more than 20 years, there are a lot places that I have not been to. This time I want to list 10 cities in China and choose a few of them that are connected in the shortest distances. I want to follow this path to visit them in this summer break. The top ten cities I want to visit is Beijing, Lhasa, Shenzhen, Hong Kong, Ürümqi, Nanjing, Suzhou, Guilin, Chongqing, and Wuhan. Below is the mileage chart between these ten cities(in kilometers). Since I will travel by plane, the distances between two cities I collect below are using the air mileage.

cities / cities	Beijing	Lhasa	Shenzhen	Hong kong	Ürümqi	Nanjing	Suzhou	Guilin	Chongqing	Wuhan
Beijing		2,562.82	1,943.43	1,971.89	2,413.07	897.96	1,028.52	1,723.40	1,459.04	1,054.80
Lhasa	2,562.82		2,421.32	2,431.94	1,604.21	2,649.19	2,825.08	1,952.25	1,493.32	2,227.47
Shenzhen	1,943.43	2,421.32		17.12	3,386.53	1,155.91	1,169.00	490.37	1,082.52	894.26
Hong Kong	1,971.89	2,431.94	17.12		3,402.04	1,169.62	1,180.84	504.23	1,097.67	910.76
Ürümqi	2,413.07	1,604.21	3,386.53	3,402.04		3,006.94	3,196.23	2,907.45	2,305.98	2,763.93
Nanjing	897.96	2,649.19	1,155.91	1,169.62	3,006.94		191.57	1,119.64	1,199.72	459.05
Suzhou	1,028.52	2,825.08	1,169.00	1,180.84	3,196.23	191.57		1,210.69	1,359.64	608.96
Guilin	1,723.40	1,952.25	490.37	504.23	2,907.45	1,119.64	1,210.69		602.40	707.23
Chongqing	1,459.04	1,493.32	1,082.52	1,097.67	2,305.98	1,199.72	1,359.64	602.40		750.98
Wuhan	1,054.80	2,227.47	894.26	910.76	2,763.93	459.05	608.96	707.23	750.98	

Table 1: Mileage Chart

Objective function and constraints

Now I have a list of 10 cities and I want to visit m of them. I want to know what is the shortest path that will connect m of the cities. I will have the objective function and a few constraints to make that function work.

Let the binary variable $x_{i,j} = 1$ if and only if we are at city j at step i(with step 1 as the start point). Let $e_{a,b} = 1$ if and only if we travel from city a to city b in one step. Let $d_{a,b}$ be the distance from city a to city b. Let the binary variable $f_{i,a,b}$ be 1 if and only if we visit city a at step i, and then city b in $i+1$ step(i.e., if and only if $(x_{i,a} + x_{i+1,b} = 2)$.

Then we want to minimize the sum of the distances among these 5 cities. We have a few constraints for this LP, as shown below. First, on each of m steps, we need to visit exactly one city. Secondly, we can only visit the city at most once. Next, we want to ensure that if we go from city a to city b $(x_{i,a} + x_{i+1,b} = 2)$, $e_{a,b}$ will be recorded as 1. That is $e_{a,b} = 1$ if and only if $x_{i,a} + x_{i+1,b} = 2$. We require $e_{a,b} \geq x_{i,a} + x_{i+1,b} - 1$, for $i = 1, \cdots, m-1$. In addition, we want to ensure that if we never move from city a to city b, $e_{a,b}$ will be 0. We use $f_{i,a,b}$ to satisfy this constraint. When we go from city a to b (i.e., $x_{i,a} + x_{i+1,b} = 2$), $f_{i,a,b}$ will be 1. If we do not move in these two cities, $f_{i,a,b}$ will be 0. The forth and fifth constraint are better implemented that we multiple each side by 3. The sixth constraint means that if we never travel from city a to city b, sum of 0 will be 0; then, $e_{a,b}$ is 0.

Then, our LP is

Minimize $\sum_{1 \le a,b \le n} e_{a,b} d_{a,b}$ subject to:

$$\sum_{j=1}^{10} x_{i,j} = 1, i = 1, \cdots, m \tag{1}$$

$$\sum_{i=1}^{m} x_{i,j} \le 1, j = 1, \cdots, 10 \tag{2}$$

$$e_{a,b} \ge x_{i,a} + x_{i+1,b} - 1, i = 1, \cdots, m-1 \tag{3}$$

$$f_{i,a,b} \le \frac{1}{3} + \frac{1}{3}(x_{i,a} + x_{i+1}, b), i = 1, \cdots, m-1 \tag{4}$$

$$f_{i,a,b} \ge \frac{-1}{3} + \frac{1}{3}(x_{i,a} + x_{i+1}, b), i = 1, \cdots, m-1 \tag{5}$$

$$e_{a,b} \le \sum_{i=1}^{m-1} f_{i,a,b}, a, b = 1, \cdots, 10 \tag{6}$$

Code generaing the LPSolve input

I use Java to generate the input file for LPSolve IDE:

```
import java.io.PrintStream;
import java.io.IOException;

public class hw4 {
    public static void main(String args[]) throws IOException {
        // creates a file named hw4.txt for saving our output
        PrintStream output = new PrintStream("hw4.txt");
        System.setOut(output);

        int total_city = 10;
        int step = 5;

        double[][] distance = new double[total_city+1][total_city+1];
        distance[1][2] = distance[2][1] = 2562.82;
        distance[1][3] = distance[3][1] = 1943.43;
        distance[1][4] = distance[4][1] = 1971.80;
        distance[1][5] = distance[5][1] = 2413.07;
        distance[1][6] = distance[6][1] = 807.96;
        distance[1][7] = distance[7][1] = 1028.52;
        distance[1][8] = distance[8][1] = 1723.40;
        distance[1][9] = distance[9][1] = 1459.04;
        distance[1][10] = distance[10][1] = 1054.80;
        distance[2][3] = distance[3][2] = 2421.32;
        distance[2][4] = distance[4][2] = 2431.94;
        distance[2][5] = distance[5][2] = 1604.21;
        distance[2][6] = distance[6][2] = 2649.19;
        distance[2][7] = distance[7][2] = 2825.08;
        distance[2][8] = distance[8][2] = 1952.25;
        distance[2][9] = distance[9][2] = 1493.32;
```

```
distance[2][10] = distance[10][2] = 2227.47;
distance[3][4] = distance[4][3] = 17.12;
distance[3][5] = distance[5][3] = 3386.53;
distance[3][6] = distance[6][3] = 1155.91;
distance[3][7] = distance[7][3] = 1169.00;
distance[3][8] = distance[8][3] = 490.37;
distance[3][9] = distance[9][3] = 1082.52;
distance[3][10] = distance[10][3] = 894.26;
distance[4][5] = distance[5][4] = 3402.04;
distance[4][6] = distance[6][4] = 1169.92;
distance[4][7] = distance[7][4] = 1180.84;
distance[4][8] = distance[8][4] = 504.23;
distance[4][9] = distance[9][4] = 1097.67;
distance[4][10] = distance[10][4] = 910.76;
distance[5][6] = distance[6][5] = 3006.94;
distance[5][7] = distance[7][5] = 3196.23;
distance[5][8] = distance[8][5] = 2907.45;
distance[5][9] = distance[9][5] = 2305.98;
distance[5][10] = distance[10][5] = 2763.93;
distance[6][7] = distance[7][6] = 191.57;
distance[6][8] = distance[8][6] = 1119.64;
distance[6][9] = distance[7][6] = 1199.72;
distance[6][10] = distance[10][6] = 459.05;
distance[7][8] = distance[8][7] = 1210.60;
distance[7][9] = distance[9][7] = 1359.64;
distance[7][10] = distance[10][7] = 608.96;
distance[8][9] = distance[9][8] = 602.4;
distance[8][10] = distance[10][8] = 707.23;
distance[9][10] = distance[10][9] = 750.98;

// gets objective function
System.setOut(output);
System.out.print("min: ");
for (int a = 1; a <= total_city; a++) {
    for (int b = 1 + a; b <= total_city; b++) {
        System.out.print("+" + distance[a][b] + "e_" + a + "_" +
        b + "+" + distance[a][b] + "e_" + b + "_" + a);
    }
}
System.out.println(";");
System.out.println();

// first constraint
System.setOut(output);
for (int i = 1; i <= step; i++) {
    for (int j = 1; j <= total_city; j++) {
        System.out.print("+x_" + i + "_" + j);
    }
    System.out.println("= 1;");
}
System.out.println();

// second constraint
```

```
System.setOut(output);
for (int i = 1; i <= total_city; i++) {
    for (int j = 1; j <= step; j++) {
        System.out.print("+x_" + j + "_" + i);
    }
    System.out.println(" <= 1;");
}
System.out.println();

// third constraint
for (int a = 1; a <= total_city; a++) {
    for (int b = a + 1; b <= total_city; b++) {
        for (int i = 1; i <= step - 1; i++) {
            System.out.println("+e_" + a + "_" + b + " - x_"
            + i + "_" + a + " - x_" + (i + 1) + "_" + b + " + 1 >= 0;");
            System.out.println("+e_" + b + "_" + a + " - x_"
            + i + "_" + b + " - x_" + (i + 1) + "_" + a + " + 1 >= 0;");
        }
    }
}
System.out.println();

// forth constraint
for (int i = 1; i <= step - 1; i++) {
    for (int a = 1; a <= total_city; a++) {
        for (int b = 1 + a; b <= total_city; b++) {
            System.out.println("3f_" + i + "_" + a + "_" + b
            + " - 1 - x_" + i + "_" + a + " - x_" + (i + 1)
            + "_" + b + "<= 0;");
            System.out.println("3f_" + i + "_" + b + "_" + a
            + " - 1 - x_" + i + "_" + b + " - x_" + (i + 1)
            + "_" + a + "<= 0;");
        }
    }
}
System.out.println();

// fifth constraint
for (int i = 1; i <= step - 1; i++) {
    for (int a = 1; a <= total_city; a++) {
        for (int b = 1 + a; b <= total_city; b++) {
            System.out.println("3f_" + i + "_" + a + "_" + b + "
            + 1 - x_" + i + "_" + a + " - x_" + (i + 1) + "_" +
            b + ">= 0;");
            System.out.println("3f_" + i + "_" + b + "_" + a + "
            + 1 - x_" + i + "_" + b + " - x_" + (i + 1) + "_" +
            a + ">= 0;");
        }
    }
}
System.out.println();

// sixth constraint
```

```
        for (int a = 1; a <= total_city; a++) {
            for (int b = 1 + a; b <= total_city; b++) {
                for (int i = 1; i <= step - 1; i++) {
                    System.out.print(" +f_" + i + "_" + a + "_" + b);
                }
                System.out.println(" - e_" + a + "_" + b + ">= 0;");
            }
        }
        System.out.println();

        for (int b = 1; b <= total_city; b++) {
            for (int a = 1 + b; a <= total_city; a++) {
                for (int i = 1; i <= step - 1; i++) {
                    System.out.print(" +f_" + i + "_" + a + "_" + b);
                }
                System.out.println(" - e_" + a + "_" + b + ">= 0;");
            }
        }
        System.out.println();

        // seventh constraint
        System.out.print("bin ");
        for (int i = 1; i <= step; i++) {
            for (int j = 1; j <= total_city; j++) {
                System.out.print("x_" + i + "_" + j + ",");
            }
        }
        for (int a = 1; a <= total_city; a++){
            for (int b = 1 + a; b <= total_city; b++){
                System.out.print("e_"+ a + "_" + b + ",");
            }
        }
        for (int i = 1; i <= step-1; i++){
            for(int a = 1; a <= total_city; a++){
                for (int b = 1 + a; b <= total_city; b++){
                        System.out.print("f_" + i + "_" + a + "_" + b + ",");
                        System.out.print("f_" + i + "_" + b + "_" + a +",");
                    }
                }
            }
        }
    }
```

Results

The result is:

min : $+2562.82e\_1\_2 + 2562.82e\_2\_1 + \cdots + +750.98e\_10\_9;$
(1) (4 lines of the following type)
$+x\_1\_1 + x\_1\_2 + x\_1\_3 + x\_1\_4 + x\_1\_5 + x\_1\_6 + x\_1\_7 + x\_1\_8 + x\_1\_9 + x\_1\_10 = 1;$
.
.

(2) (9 lines of the following type)
$+x\_1\_1 + x\_2\_1 + x\_3\_1 + x\_4\_1 + x\_5\_1 <= 1;$
.
.
(3) (359 lines of the following type)
$+e\_1\_2 - x\_1\_1 - x\_2\_2 + 1 >= 0;$.
.
(4) (359 lines of the following type)
$3f\_1\_1\_2 - 1 - x\_1\_1 - x\_2\_2 <= 0;$
.
.
(4) (359 lines of the following type)
$3f\_1\_1\_2 + 1 - x\_1\_1 - x\_2\_2 <= 0;$
.
.
(5) (89 lines of the following type)
$+f\_1\_1\_2 + f\_2\_1\_2 + f\_3\_1\_2 + f\_4\_1\_2 - e\_1\_2 >= 0;$
.
.
(6) bin $x\_1\_1, \cdots, f\_4\_10\_9;$
Using the above input, I solved the LP using LPSolve IDE. We get the result:

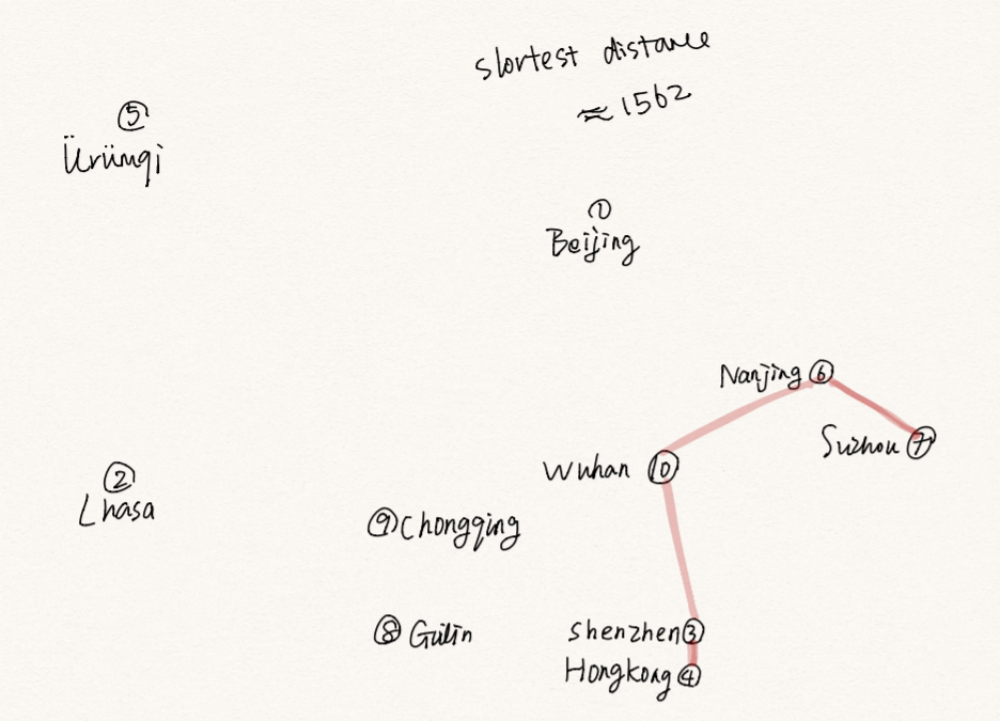
Value of objective function: 1562.00000000

Actual values of the variables:

e_3_4	1
e_10_3	1
e_7_6	1
e_6_10	1
x_1_7	1
x_2_6	1
x_3_10	1
x_4_3	1
x_5_4	1
f_1_7_6	1
f_2_6_10	1
f_3_10_3	1
f_4_3_4	1

The result shows that I can travel city 3, 4, 10, 6, 7 and the sum of distances will be the shortest: 1523.

I also draw the path for these five cities in below.



Figures and Tables

	cities	distances
m = 2	city 3, 4	17.12
m = 3	city 3, 4, 8	507.49
m = 4	city 3, 4, 8, 9	1109.89
m = 5	city 3, 4, 6, 7, 10	1562.00

No matter how large m is, the cities of shortest path will always contain the cities of case $m = 2$. I tried to run as m becomes larger than 5, but the running time is pretty long, even longer than 20 minutes and I cannot get the solution(the program is keeping running for a long time and I have to stop it by myself, maybe because the code I write is a little bit reluctant).

Before I run $m = 5$, I thought the cities may include 3, 4, 8, 9, but surprisingly, the solution only includes city 3 and city 4. I guess the distance from city 9 to the next city is too large, so that the path next to city 9 is given up. The program just found another path that is shorter than path including city 9: city 3, 4, 6, 7, 10.