Optimum Trip

Xinming Zhang

I picked ten buildings on our campus to go for my optimum trip problem. The buildings I picked are Odegaard Undergraduate Library(OUG), Mary gates hall(MGH), Gould hall(GLD), Paul G. Allen Center(CSE), Bagley Hall(BAG), Fluke hall(FLK), Denny hall(DEN), Alaska airlines arena(EDP), Husky stadium(STD), and Hitchcock(HCK). I list them in order from location 1 to 10. Those are the ones either I will go to or the ones I have been there. Then, I use google maps to check the distance between them manually. The distance could be shown as following table, the units of the distance is in 100 meters:

City number	1	2	3	4	5	6	7	8	9	10
1 OUG	0	4.5	4.5	8	5	8	3	11	12	7
2 MGH	4.5	0	5	3	2	6	5	7	7	6
3 GLD	4.5	5	0	8	6	10	7	12	10	3
4 CSE	8	3	8	0	3.5	4	8	3.5	4.5	6
5 BAG	5	2	6	3.5	0	6	7	7	7	4.5
6 FLK	8	6	10	4	6	0	7	5	7	10
7 DEN	3	5	7	8	7	7	0	11	12	9
8 EDP	11	7	12	3.5	7	5	11	0	9	9
9 STD	12	7	10	4.5	7	7	12	9	0	7
10 HCk	7	6	3	6	4.5	10	9	9	7	0

Distance table (Unit: 100m)

We are trying to get the shortest path between some of the locations, so we are minimizing the total traveled distance through the locations. I will use solving a LP (Linear Program) to get the optimum trip between the locations.

First, we are going to define our variables.

Set n as the total number of locations, in this case n is equal to 10.

Set m as the number of locations we want to visit (number of locations on the shortest path).

Define $x_{i,j}$ as a binary variable (either be 0 or 1) for all $1 \leq i \leq m, 1 \leq j \leq n.$

$x_{i,j} = 1$ represents we are in location j at step i, when it equals 0, that means we are not in location j at step i.

Define $e_{a,b}$ as a binary variable (either be 0 or 1) for all $1 \leq a, b \leq n.$

$e_{a,b} = 1$ if we traveled from location a to location b in one step, when it equals 0, that means we are not traveling from location a to location b.

Define $d_{a,b}$ as a the distance between location a and b, for all $1 \leq a, b \leq n.$

We are trying to minimize the total traveled though m out of n locations. Using the variable we defined, we can get the objective function as:

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

Then, we need to set out constraints.
First one will be for our each step we are only visiting one location (one step one location), then it will be as:

$$\sum_{j=1}^{n} x_{i,j} = 1 \ , i = i,\dots, m$$

Second one will be every location will be visited at most once, and the constraint will be as:

$$\sum_{j=1}^{n} x_{i,j} \le 1 \ , j = i,\dots, n$$

Third one be the definition of variable $e_{a,b}$, when it equals to 1 if and only if we travel from location a to location b. Then, $x_{i,a} + x_{i+1,b}$ must be 2 if $e_{a,b}$ equals 1. (where i stands for the step) If either or both of $x_{i,a}$ and $x_{i+1,b}$ is 0, then $e_{a,b}$ is zero. Sum that up to get our third constraint as:

$$e_{a,b} \ge x_{i,a} + x_{i+1,b} - 1 \ , i = 1,\dots, m-1, 1 \le a, b \le n$$

For last constraint, we will need to make sure both variable $e_{a,b}$, $x_{i,j}$ are binary variables (either 0 or 1):

$$e_{a,b}, x_{i,j} \in \{0,1\} \quad 1 \le i \le m, \ 1 \le a, b, j \le n$$

Above all, we can get our LP in compact form as:

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

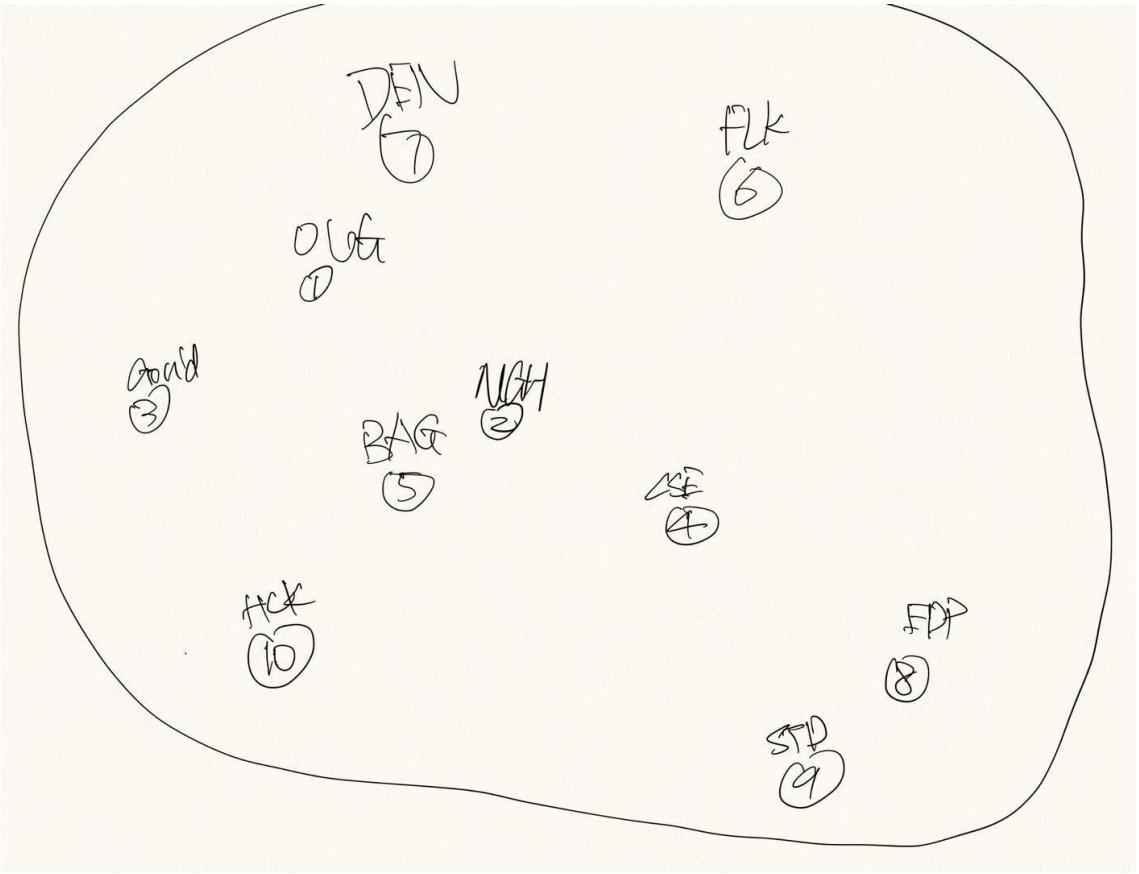
$$\text{Subject to: } \sum_{j=1}^{n} x_{i,j} = 1 \ , i = i,\dots, m$$

$$\sum_{j=1}^{n} x_{i,j} \le 1 \ , j = i,\dots, n$$

$$e_{a,b} \ge x_{i,a} + x_{i+1,b} - 1 \ , i = 1,\dots, m-1, 1 \le a, b \le n$$

$$e_{a,b}, x_{i,j} \in \{0,1\} \quad 1 \le i \le m, \ 1 \le a, b, j \le n$$

A hand-drawn map to visualize the locations (buildings):



I will be generating an lpsolve input file with the following code:
Coding:

```
public class Math381_4_1 {

    public static void main(String[] args) {
        //The 2d array d stand for the distance between two cities, di,j stands for
        //the distance between city i and j
        double[][] d = new double[][]{{0, 4, 4.5, 8, 5, 8, 3, 11, 12, 7},
        {4.5, 0, 5, 3.5, 2, 6, 5, 7, 7, 6}, {4.5, 5, 0, 8, 6, 10, 7, 12, 10, 3},
        {8,3.5,8,0,3.5,4,8,3.5,4.5,6}, {5,2,6,3.5,0,6,7,7,7,4.5},
        {8,6,10,4,6,0,7,5,7,10},{3,5,7,8,7,7,0,11,12,9},{11,7,12,3.5,7,5,11,0,9,9},
        {12,7,10,4.5,7,7,12,7,0,7},{7,6,3,6,4.5,10,9,9,7,0}};

        //total number of cities, always 10
        int n = 10;
        //number of cities we want to connect in the shortest path
        int m = 5;

        //object function minimize the connected length
        System.out.print("min: ");
        for (int i = 1; i <= n; i++) {
```

```
    for (int j = 1; j <= n; j++) {
        if(i != j){
        System.out.print("+ " + d[i - 1][j - 1] + "e_" + i + "_" + j);
        }
    }
}

System.out.println(";");


//e_a,b  iff x_i,a and 1+ x_i+1,b = 2
for (int i = 1; i <= m + 1; i++) {
    for (int a = 1; a <= n; a++) {
        for (int b = 1; b <= n; b++) {
            System.out.print("e_" + a + "_" + b);
            System.out.print(" >= ");
            int k = i + 1;
            System.out.println("x_" + i + "_" + a + "+ x_" + k + "_" + b + " -1;");
        }

    }
}


//Only one city each step constraint
for (int i = 1; i <= m; i++) {
    for (int j = 1; j <= n; j++) {
        if (j == n) {
            System.out.println(" + " + "x_" + i + "_" + j + " = 1;");
        } else {
            System.out.print(" + " + "x_" + i + "_" + j);
        }
    }
}

//Each city can not be visited twice (only once)
for (int j = 1; j <= n; j++) {
    for (int i = 1; i <= m; i++) {
        if (i == m) {
            System.out.println(" + " + "x_" + i + "_" + j + " <= 1;");
        } else {
            System.out.print(" + " + "x_" + i + "_" + j);
        }
    }
}
```

```
        //solutions are binary numbers (0 or 1)
        System.out.print("bin");
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                System.out.print(" x_" + i + "_" + j + ",");
            }
        }
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (j == n && i==n) {
                    System.out.println(" e_" + i + "_" + j + ";");
                } else {
                    System.out.print(" e_" + i + "_" + j + ",");
                }
            }
        }
    }

}
```

Running the code to get the input file as:

```
min: + 4.0e_1_2+ 4.5e_1_3 + … + 9.0e_10_7+ 9.0e_10_8+ 7.0e_10_9;
```

(400 lines of the following type:
ensure moving from one place to another in one step if the step is taken)

```
e_1_1 >= x_1_1+ x_2_1 -1;
.
.
```

(5 lines of the following type:
ensure that only one location is visited at each step)

```
 + x_1_1 + x_1_2 + x_1_3 + x_1_4 + … + x_1_9 + x_1_10 = 1;
.
.
```

(10 lines of the following type:
ensure that Each city can not visited more than once)

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

(ensure all variables are binary)

```
bin x_1_1, x_1_2, x_1_3, x_1_4, ... , e_10_9, e_10_10;
```

Using the input file above to get the solution for m = 5 (shortest path visiting 5 locations):
Value of objective function: 12.50000000

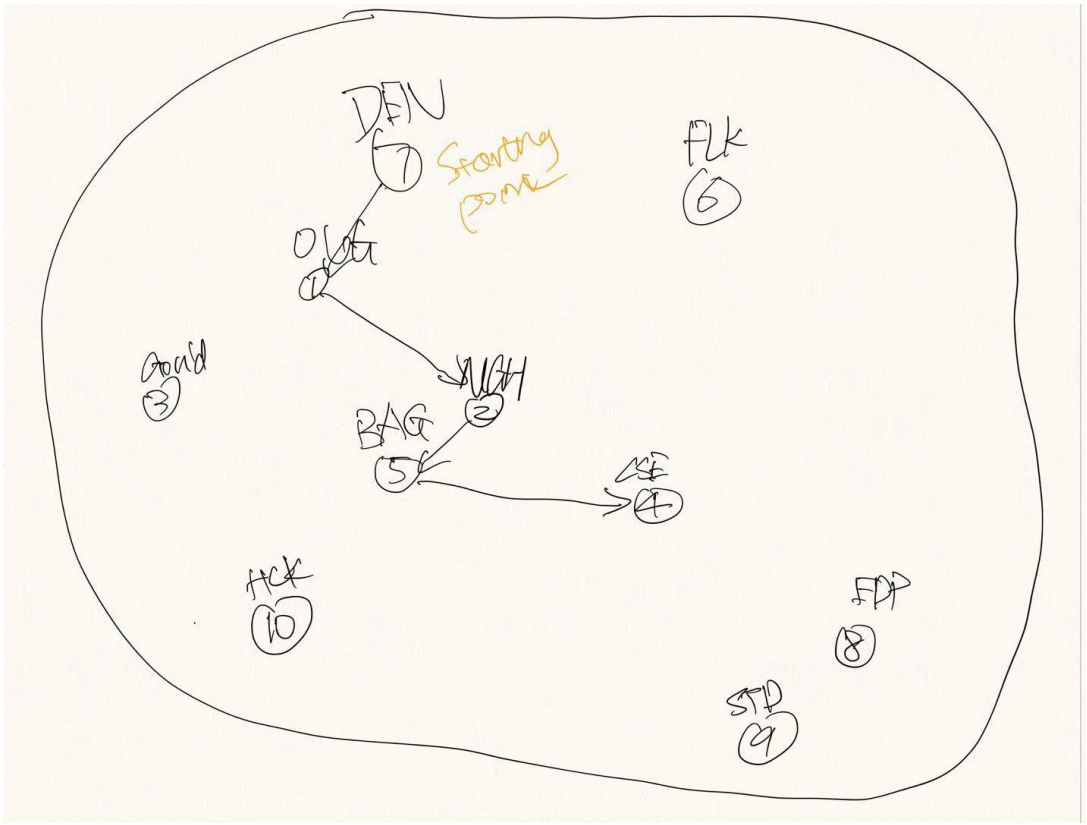
Actual values of the variables:

e_1_2	1
e_2_5	1
e_5_4	1
e_7_1	1
x_2_1	1
x_1_7	1
x_3_2	1
x_4_5	1
x_5_4	1

All other variables are zeros.
Used 1.137 seconds.

From the solution we can know, the length of the shortest path going through 5 locations out of 10 locations I picked will be 1250 meters. We will start at location 7 which is Denny hall(DEN). Then we will go through Odegaard Library(OUG), Mary Gates Hall (MGH), Bagley Hall (BAG), and Pual G. Allen Center (CSE) in order. We can express them in locations' numbers as 7-1-2-5-4, and the total traveled distance is 1250 meters.

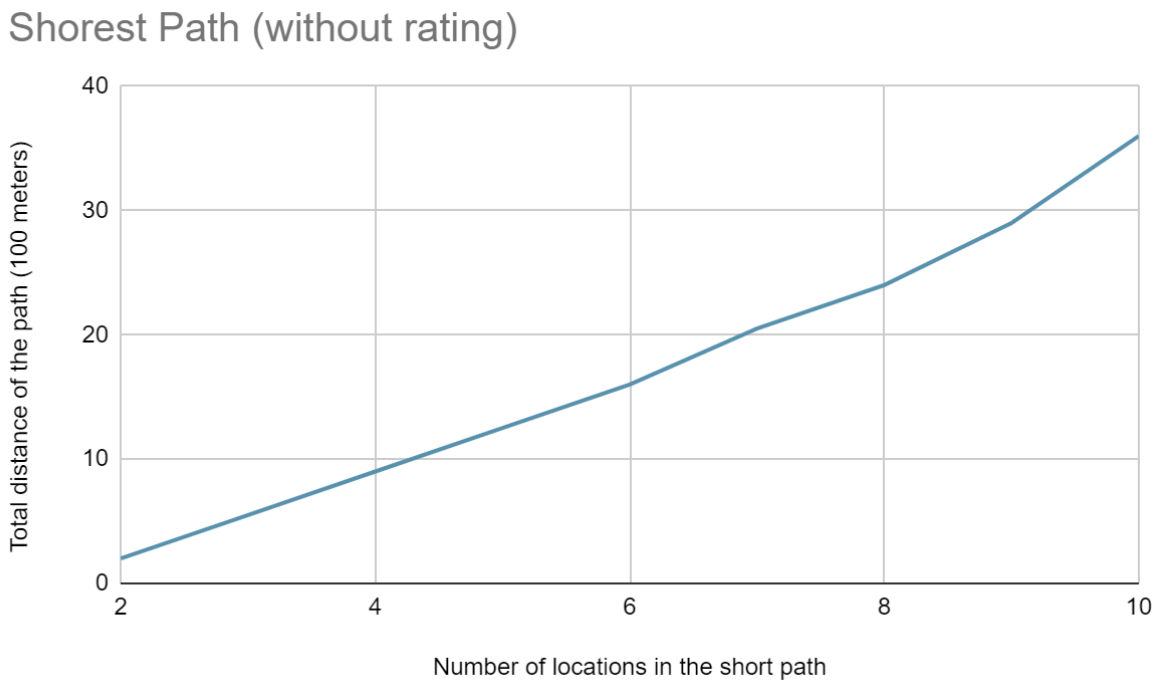
Visualization (m=5):



(Makes sense by just looking at the map)
We can generate the code and get the result for m = 2 to 10 (including 5) as the table next page:

Number of location traveled (variable m)	The length of shortest path(Units: 100 meters)	Locations' order in the path (in numbers form)	Runtime (Units: sec)
2	2	5-2	0.19
3	5.5	5-2-4	0.03
4	9	8-4-5-2	0.132
5	12.5	7-1-2-5-4	1.137
6	16	7-1-2-5-4-8	7.13
7	20.5	7-1-3-10-5-2-4	33.691
8	24	7-1-3-10-5-2-4-8	144.123
9	29	7-1-3-10-5-2-4-8-6	1436.023
10	36	9-2-8-6-2-5-10-3-1-7	4581

From the result table we can get the plot as :



There are some findings from the result table and the plot we got:

1. The runtime and the length of the short paths is increasing by increasing the variable m, which is the number of locations we want to travel. For m=10, the runtime is huge, and it is about 76 minutes.
2. From the path column in the result table, we can see that the short path 2-5 or 5-2 is always included in every path. That means no matter which buildings we are going to, the path from Mary Gates Hall to Bagley Hall is always the shortest path. That makes sense, because from the map we can see they are the closest ones. The paths 1-7 or 7-1 start to be included from m = 5. Location 1 is Odegaard Library, and Location 7 is Denny Hall. Looking at the distance matrix (table) at the beginning, they are be the second shortest pair. That is why they are always on the path.
3. From the plot (graph), the relationship between the number of locations traveled and total distance traveled are in a positive linear relationship. The plot is close to a linear line with a positive slope. Especially for the first five points, the line is almost straight. From m = 6, the line is not that straight, but we still can see the linear relationship.

Then we move on to the problem with a rating for each location.
First, here is my rating for each location:

Location number	Rating
1	7
2	6
3	7
4	5
5	8
6	5
7	9
8	8
9	10
10	5

We will change the object function a little bit, we will maximize the total rating of the locations we visited as:

$$\text{Maximize: } \sum_{1 \le i,j \le n} R_j x_{i,j}$$

Where R_j represent the rating the location j.

In this case, we will not set a number of locations we want to travel, that means we are setting m = n . Then, the second constraint for the previous part will change to:

$$\sum_{j=1}^{n} x_{i,j} \leq 1 \quad , j = 1,\dots, n$$

On the other hand, I am adding two new constraints to the LP.
The first one will be the upper bound of the total distance traveled.
We can modify it as:

$$\sum_{1 \leq i,j \leq n} e_{i,j} d_{i,j} \leq B$$

Where B is the upper bound (traveled distance) we want to input.
The second one will be added is we cannot skip steps, which means we must take (i-1)th step first to take ith step (i is between 2 to n-1 to make (i-1) not smaller than 1):

$$\sum_{j=1}^{n} x_{i,j} \leq \sum_{j=1}^{n} x_{i-1,j} \quad , i = 2,\dots, n-1$$

Above all, we can get our new LP in compact form as:

$$\text{Maximize: } \sum_{1 \leq i,j \leq n} R_j x_{i,j}$$

$$\text{Subject to: } \sum_{1 \leq i,j \leq n} e_{i,j} d_{i,j} \leq B$$

$$\sum_{j=1}^{n} x_{i,j} \leq 1 \quad , i = 1,\dots, n$$

$$\sum_{j=1}^{n} x_{i,j} \leq 1 \quad , j = 1,\dots, n$$

$$e_{a,b} \geq x_{i,a} + x_{i+1,b} - 1 \quad , i = i,\dots, n-1, 1 \leq a, b \leq n$$

$$\sum_{j=1}^{n} x_{i,j} \leq \sum_{j=1}^{n} x_{i-1,j} \quad , i = 2,\dots, n-1$$

$$e_{a,b}, x_{i,a} \in \{0, 1\} \quad , 1 \leq a, b, i, j \leq n$$

We have to change the code a little bit according to the LP:
We will change every variable m appeared in the code to variable n.
Then changing the objective function and adding the rating list as:

```
//rating array for each location
int[] rating = new int[]{7, 6, 7, 5, 8, 5, 9, 8, 10, 5};
//n equals to the number of locations
int n = rating.length;
//New objective function with rating (maxing total rate)
```

```
System.out.print("max: ");
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        System.out.print("+ " + rating[i - 1] + "x_" + i + "_" + j);
    }
}
System.out.println(";");
```

We can use the objective function code from the last part to get our first new constraint.

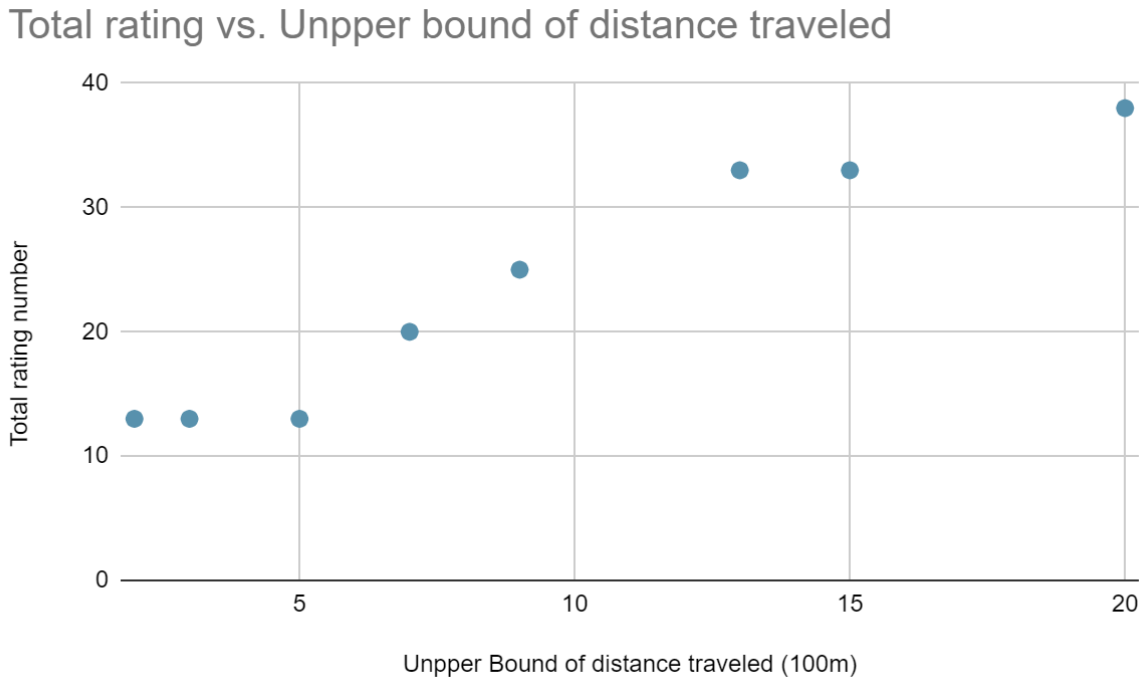
For second new one, we can write it as:

```
//Step in step, no skipping steps
for (int i = 2; i <= n; i++) {
        int k = i - 1;
        for (int j = 1; j <= n; j++) {
            System.out.print(" + " + "x_" + i + "_" + j);
        }
            System.out.print(" <= ");
        for (int j = 1; j <= n; j++) {
            System.out.print(" + " +"x_" + k + "_" + j);
        }
        System.out.println(" ;");
    }
```

Using lp_solve to solve the input file from running the code to get our results as the table:

Upper Bound B (Units: 100 m)	Number of Locations traveled	Result path represented in number form	Total ratings	Runtime (seconds)
2	2	2-5	13	0.171
3	2	3-10	13	0.343
5	2	3-1	13	0.809
7	3	3-2-5	20	1.538
9	4	5-2-4-8	25	4.836
13	5	3-10-5-2-4	33	66.632

15	5	3-1-2-5-4	33	199.089
20	6	3-1-2-4-5-10	38	1685.615



Finding: I was trying to increase the Upper Bound of the traveled distance (B) one at each time to get a better idea about the changing of the locations. However, from B = 12, the runtime starts to be pretty long. For B = 12, the runtime is getting longer than a minute. When B = 15, the runtime is more than three minutes, so I started to increase B by 5 each time. The next and the last try is B = 20, and it took a long time to get the solution (28 minutes). I did not go for the next try, because ideally it would have an even longer runtime than B = 20. From the plot, we can see the total rating might not be increasing by the increase of B. The table told us, even if the total rate might not be changed the path will be different. My ratings on each place are in integer, and some of them are the same. That could be a factor, and another one is the distance between locations (buildings). Some distances are the same as well, and those two factors made the plot like that.

Combining those two parts together. The second problem has a relationship with the first part. The upper bound B from the second part must be affected by the shorts path as well. The pair 5-2 or 2-5 appeared again in the second part. I rated them medium to medium-high, and they are the closest buildings in those ten buildings. That makes sense. However, when B = 20, the pair 2-5 broke up. I think that is caused by the setting of B. The lowest number of B to get a total rating of 38 should not be 20. Ideally if we are setting upper bound (B) lower to a lowest number to get total rating of 38, 5-2 path should be included again.

Revisiting:
I changed a few mistakes according to the feedback from assignment 4 for the previous part. Other than that, I am trying to shorten the run time for solving the LP in the second part of the assignment which is the rating problem.

At first, I added one constraint as following:
The purpose of the constraint is setting a lower bound of the number of locations traveled, and the constraint in the LP is:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{i,j} \geq C,\ C\ is\ positive\ integer$$

Set variable C as the lower bound of the number of locations traveled, and C will always be positive integers. Then, it will depend on the total rate we got from the previous run. For example, from the table of the result we got for the rating problem. When the upper bound of distance traveled (B) is 900 meters, the total rating is 25. Then the total rating from any higher upper bound of distance traveled will have at least 25 rating.According to the rating table on page 8, the three highest rating is 8, 9 and 10, so we will need to travel at least 3 places to get a total rating higher than 25 (8+9+10 = 27 > 25). Then we can set the lower bound of locationst travel (C) to 3, when we are solving B = 10 problem.

The code to generate this constraint is following (C = 3 case):

```
int C = 3;
for (int j = 1; j <= n; j++) {
        for (int i = 1; i <= n; i++) {
           System.out.print(" + " + "x_" + i + "_" + j);
        }
    }
System.out.println(" >= " + C + ";");
```

After adding the new constraint, the new LP will be:

$$\text{Maximize: } \sum_{1 \leq i,j \leq n} R_j x_{i,j}$$

$$\text{Subject to: } \sum_{i=1}^{n}\sum_{j=1}^{n} x_{i,j} \geq C,\ C\ is\ positive\ integer$$

$$\sum_{1 \leq i,j \leq n} e_{i,j} d_{i,j} \leq B$$

$$\sum_{j=1}^{n} x_{i,j} \leq 1 \quad , i = 1,\ldots,n$$

$$\sum_{j=1}^{n} x_{i,j} \leq 1 \quad , j = 1,\ldots,n$$

$$e_{a,b} \geq x_{i,a} + x_{i+1,b} - 1 \quad , i = i,\ldots,n-1,\ 1 \leq a,b \leq n$$

$$\sum_{j=1}^{n} x_{i,j} \leq \sum_{j=1}^{n} x_{i-1,j} \quad , i = 2,\ldots, n-1$$

$$e_{a,b}, x_{i,a} \in \{0,1\} \quad , 1 \leq a, b, i, j \leq n$$

Then, I tried to run B = 10, and C = 4. After solving the input file from generating the code using lpsolve, the output is:

Value of objective function: 25.00000000

Actual values of the variables:

x_1_2	1
x_2_5	1
x_3_1	1
x_4_7	1
e_1_7	1
e_2_5	1
e_5_1	1

All other variables are zeros.

From the output, we know the path is from location 2-5-1-7 for B = 10, and the total rating is 25 which is the same as B = 9 case. The runtime of this is 1.0 second, so it is faster 4.8 seconds which is the runtime of B = 9 from the result table on page 10. Then I went ahead to check if we can run a larger case quickly, so I tried to solve the upper bound of distance traveled of 1600 meters (B = 16), and C = 4.

It took 351 seconds for lpsolve to get the optimal solution, and the path is 7-1-2-5-4-8 (6 locations traveled), and the total rating is 38. The runtime is still pretty long, so the runtime got improved but not much after adding the lower bound of locations traveled. Just to be sure, my idea is right, I tried to solve the B = 21 case, and it did not return an optimal solution after 20 minutes, so the runtime is still extremely long.

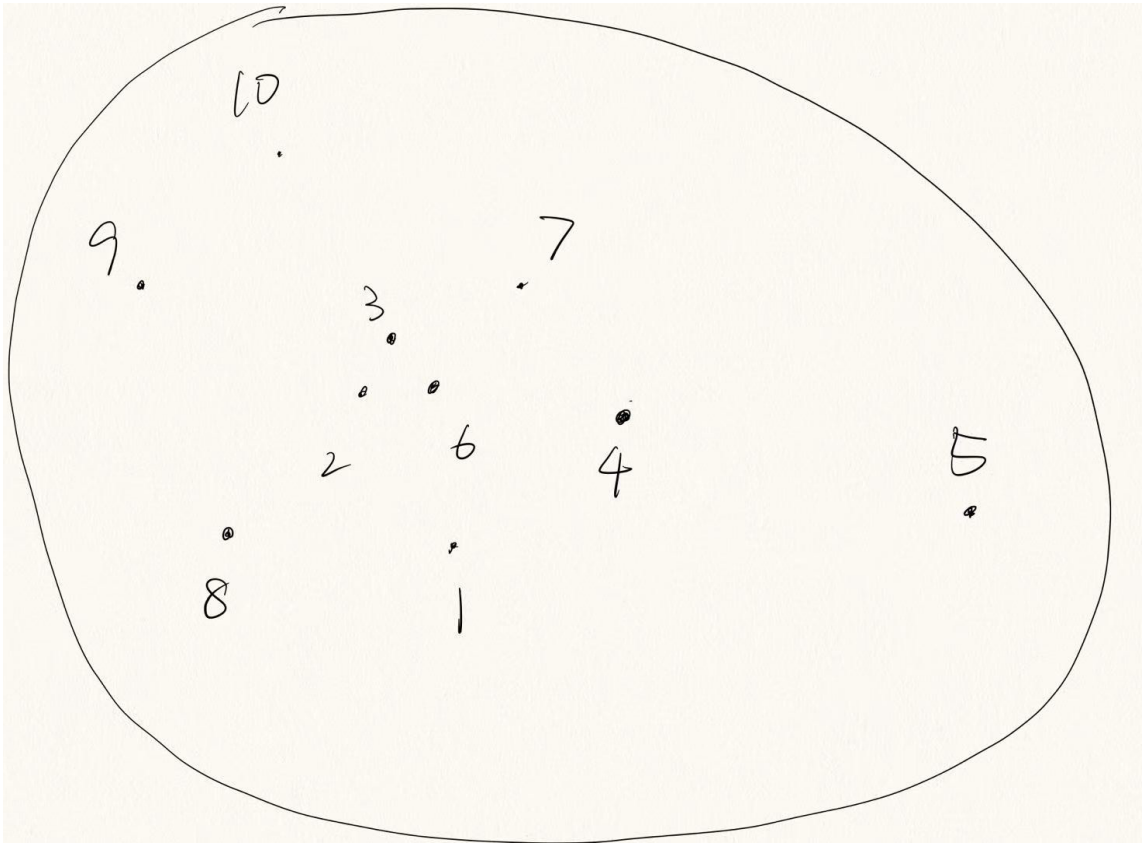
After realizing that, I went ahead and changed all the ratings of 10 locations to 10. They all have the same and full rating, then put it in lpsolve to solve the B = 20 case. It still took 95 seconds to get the optimal solution, it is a lot faster compared to B = 20 case (1685 seconds) in the result table on page 10 (previous results). That means the runtime is influenced by the different ratings. On the other hand, I think my distance matrix might be another factor influencing the runtime, because the distances are too similar. From looking at the distance matrix (table) on the first page, we can see there are lots of similar or even same distances between some places. To see if my ideas are right. Then, I tried to find a totally different distance matrix as following (next page):

Locations	1	2	3	4	5	6	7	8	9	10
1	0	4	4.3	7.2	17	3.1	8.7	7	9.9	12
2	4	0	2.1	9.2	21	1.7	8.2	6.1	7.6	8
3	4.3	2.1	0	9.5	20	1.3	6.2	8.6	8.8	8.9
4	7.2	9.2	9.5	0	12	8.7	6.8	15	22	18
5	17	21	20	12	0	20	18	28	34	29
6	3.1	1.7	1.3	8.7	20	0	6.3	6.7	9.1	7.8
7	8.7	8.2	6.2	6.8	18	6.3	0	14	14	9.7
8	7	6.1	8.6	15	28	6.7	14	0	7.8	10
9	9.9	7.6	8.8	22	34	9.1	14	7.8	0	7.3
10	12	8	8.9	18	29	7.8	9.7	10	7.3	0

New distance matrix (Unit: Kilometers)

The locations are the 10 places of tourist attraction in Jinan, China. This is a totally different dataset from the one I did in assignment 4. This distance matrix has less same distances between places than the one from the first page.

Hand Drawn Map:



I will skip the first part which is finding the shortest path, and go ahead to do the rating part.

Using the same order of constraints as what I did from the assignment 4, and same ratings for corresponding places, but different distances. The result table can be shown as below:

Upper Bound B (Units: 100 m)	Number of Locations traveled	Result path represented in number form	Total ratings	Runtime (seconds)
2	2	6-3	12	0.122
3	3	3-6-2	18	0.140
5	3	3-6-1	19	0.288
7	4	2-3-6-1	25	0.520
9	4	6-3-2-1	25	0.880
13	5	8-2-3-6-1	33	6.303
15	5	1-3-6-2-9	35	15.819
20	6	4-7-3-2-6-1	39	219.53
30	unknown	unknown	52	More than 30 mins

At B = 30 case, the runtime starts to be extremely long, so I did not go further than that. The solution is the improved solution from lpsolve, and it did not provide the path.
Then, compare the runtime with the old result table from page 10, we can get table as below:

Upper Bound B (Units varys)	Runtime for first distance matrix on the first page (seconds)	Runtime for new distance matrix on page 14 (seconds)
2	0.171	0.122
3	0.343	0.140
5	0.809	0.288
7	1.538	0.520
9	4.836	0.880

13	66.632	6.303
15	199.089	15.819
20	1685.615	219.53
30	unknown	More than 30 mins

Comparing the runtimes for each case between the old distances matrix and the new one, the runtime of the new distance matrix is a lot shorter. Distance matrix can influence the runtime, but when the lower bound of distance traveled got up to 30 units, the runtime for either distance matrix is extremely long. The distance matrix is one of the factors that influence the runtime, but the lower bound of the distance traveled is the most important one.

Just for fun, I tried the different ratings on the new distance matrix and tried to solve the LP. The new rating table is following:

Locations	Rating
1	5
2	8
3	6
4	8
5	6
6	9
7	7
8	10
9	5
10	5

I used the best LP (has least runtime) so far which is the one mentioned on page 12 (the beginning of revisiting). Then I used code to generate the corresponding LP input file, and use lpsolve to get the following result table (next page):

Upper Bound B (Units: km)	Number of Locations traveled	Result path represented in number form	Total ratings	Runtime (seconds)
2	2	2-6	17	0.059
3	3	3-6-2	23	0.079
5	3	2-3-6	23	0.079
7	4	1-2-6-3	28	0.255
10	4	6-3-2-8	33	0.974
15	5	1-3-2-6-8	38	67.18
20	6	4-1-6-3-2-8	46	54.983
30	unknown	unknown	53	More than 1hr
40	unknown	unknown	59	More than 1hr
50	unknown	unknown	64	More than 1hr
60	10	1-3-2-6-8-9-10-7-4-5	69	0.318
70	10	1-3-2-6-7-10-9-8-4-5	69	0.089
80	10	1-3-2-6-7-4-5-8-9-10	69	0.027

The runtime of this distance matrix is quicker than the one I had before on page 15 with different ratings except for the B = 15 case. For the B = 15 case, the runtime for this set of ratings is longer than the other set of ratings. I still do not know the reason why that is. It might be caused by the lpsolve (just a guess). Then the interesting thing is that the B = 20 case is quicker than the B = 15 case, and the constraints and order of constraints did not change.

When the upper bound of distance travel (B) gets to 20, the time starts to get long again. For this LP, it is a huge one, and there are only a few places we can improve it. Even after the change, it is still slow.

For B = 30, 40 ,and 50, the runtime starts to be so huge that I run it for about an hour, and it still did not give me the optimal solution that lpsolve likes. I wrote down the improved solution for those cases. Then, when B is greater than or equal to 60, the total rate becomes 69

which is the greatest we can get. I did not set up the starting location and the first path. When I change B, even though the total rating is still the same, the paths are different.

Setting up a starting point or the first path might make the runtime shorter, but the result might be a lot different than the ones in the table, so I did not do that. Changing the order of constraints helped a lot, and adding some constraints might help sometimes, but it depends.

Overall, I did improve the runtime, but not as much as I was hoping for. The distance matrix and the ratings can influence the runtime. However, when the lower bound of distance (B) travels gets larger, the runtime starts to be extremely long. From my results, when B gets larger than 20 (but not larger than the shortest path between those 10 places), the runtime is most likely longer than one hour using lpsolve. Then, when the lower bound of distance traveled gets long enough that is longer than the shortest path between those places, then the runtime is extremely short (few seconds or even less). That is because the problem has changed. It will not be a rating problem anymore, it is just giving a path that goes through every location under the lower bound of distance traveled.