MATH 381 HW#1

Jingyi Cui

Due: 4/10/20

Contents

Problem 1

In the past, there exist a special profession in ancient China: people who steal things from graves.[1] They would raid emperors' tomb to get treasures, such as a host of gold or jewelry. In this problem, we suppose they can only take 50 pieces of jewelry, since there is not enough space in the truck. The truck can only contain things whose total weight is less than 1000 kg and whose total volume is less than 1000 liters. We can choose to take the jewelry or nor. Since we have 50 objects(pieces of jewelry), we use i as the index for each object. So i is from 1 to 50. For each piece thieves can take away, it has its own value, weight and volume, defined in this way:

$$\text{value} = \text{floor}^2(100 + 50cos(i))$$

$$\text{weight} = \text{floor}(100 + 50cos(5 * i - 3)$$

$$\text{volume} = \text{floor}(100 + 50cos(3 * i + 11)$$

The thieves would finally sell all the treasure, so they want to put things that have highest total value(in thousand Yuan) into truck, with the constraints of weight and volume.

To solve this problem, we already know that we need to maximize the value of the objects, and we have two constraints of the weight and volume. Besides, we know that we can choose to take the jewelry or not.

The idea is to define a binary variable x_i for each object($1 \leq i \leq 50$). Notice i is the index for each object. For each x_i, $x_i = 0$ means that we do not take this piece away; $x_i = 1$ means that we will take this piece away. We use value[i] to denote the value for i-th object. Then as we wish, we maximize the sum:

$$\sum_{i=1}^{50} \text{value}[i] * x_i$$

subject to the constraints of the truck weight and volume capacity. The weight is less than 1000 kg and the volume is less than 1000 liters. We use weight[i] to denote the weight of i-th object and volume[i] as the volume of i-th object. We can express as

$$\sum_{i=1}^{50} \text{weight}[i] * x_i \leq 1000$$

$$\sum_{i=1}^{50} \text{volume}[i] * x_i \leq 1000$$

Also, x_i is binary variable as defined above:

$$x_i \in \{0, 1\}$$

Using Java, we generate the input file for LPSolve IDE (the output of this code file is four equations above that LPSolve can read and solve):

[1] If you are interested in this profession, there is a Chinese novel called *Ghost Blows Out the Light*.

[2] Floor function here means getting the greatest integer less than or equal to the real number we use

```
import static java.lang.Math.cos;
import java.io.PrintStream;
import java.io.IOException;

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

            // gets the total value that we need to maximize
            System.out.print("max: ");
            for (int i = 1; i <= 50; i++) {
                // gets value for each object
                double value = Math.floor(100 + 50 * cos(i));
                System.setOut(output);
                System.out.print("+" + value + " x_" + i +" ");
            }
            System.setOut(output);
            System.out.println(";");

            // first constraint: weight <= 1000
            for (int t = 1; t <= 50; t++){
                // gets weight for each object
                double weight = Math.floor(100 + 50 * cos(5 * t - 3));
                System.setOut(output);
                System.out.print("+" + weight + " x_" + t + " ");
            }
            System.out.println(" <= 1000" + ";");

            // second constraint: volume <= 1000
            for (int q = 1; q <= 50; q++){
                // gets volume for each object
                double volume = Math.floor(100 + 50 * cos(3 * q + 11));
                System.out.print("+" + volume + " x_" + q + " ");
            }
            System.out.println(" <= 1000" + ";");

            // third constraint: all the x should be binary integer
            System.setOut(output);
            System.out.print("bin ");
            for (int n = 1; n <= 49; n++){
                System.setOut(output);
                System.out.print("x_" + n + ", ");
            }
```

```
                System.setOut(output);
                System.out.print("x_" + 50 +";");
            }
}
```

The result is shown as below:

$$\begin{aligned}\max: &+ 127.0x_1 + 79.0x_2 + 50.0x_3 + 67.0x_4 + 114.0x_5 + 148.0x_6 + 137.0x_7 \\ &+ 92.0x_8 + 54.0x_9 + 58.0x_{10} + 100.0x_{11} + 142.0x_{12} + 145.0x_{13} + 106.0x_{14} \\ &+ 62.0x_{15} + 52.0x_{16} + 86.0x_{17} + 133.0x_{18} + 149.0x_{19} + 120.0x_{20} + 72.0x_{21} \\ &+ 50.0x_{22} + 73.0x_{23} + 121.0x_{24} + 149.0x_{25} + 132.0x_{26} + 85.0x_{27} + 51.0x_{28} \\ &+ 62.0x_{29} + 107.0x_30 + 145.0x_31 + 141.0x_{32} + 99.0x_{33} + 57.0x_{34} + 54.0x_{35} \\ &+ 93.0x_{36} + 138.0x_{37} + 147.0x_{38} + 113.0x_{39} + 66.0x_{40} + 50.0x_41 + 80.0x_{42} \\ &+ 127.0x_{43} + 149.0x_{44} + 126.0x_{45} + 78.0x_{46} + 50.0x_{47} + 67.0x_{48} + 115.0x_{49} \\ &+ 148.0x_{50};\end{aligned}$$

And three constraints are:

$$\begin{aligned}&+ 79.0x\_1 + 137.0x\_2 + 142.0x\_3 + 86.0x\_4 + 50.0x\_5 + 85.0x\_6 + 141.0x\_7 \\ &+ 138.0x\_8 + 80.0x\_9 + 50.0x\_10 + 91.0x\_11 + 144.0x\_12 + 133.0x\_13 + 74.0x\_14 \\ &+ 51.0x\_15 + 98.0x\_16 + 147.0x\_17 + 128.0x\_18 + 68.0x\_19 + 53.0x\_20 + 105.0x\_21 \\ &+ 149.0x\_22 + 122.0x\_23 + 63.0x\_24 + 56.0x\_25 + 111.0x\_26 + 149.0x\_27 + 116.0x\_28 \\ &+ 59.0x\_29 + 60.0x\_30 + 117.0x\_31 + 149.0x\_32 + 110.0x\_33 + 56.0x\_34 + 64.0x\_35 \\ &+ 123.0x\_36 + 148.0x\_37 + 103.0x\_38 + 53.0x\_39 + 69.0x\_40 + 129.0x\_41 + 147.0x\_42 \\ &+ 97.0x\_43 + 51.0x\_44 + 75.0x\_45 + 134.0x\_46 + 144.0x\_47 + 90.0x\_48 + 50.0x\_49 \\ &+ 81.0x\_50 \leq 1000;\end{aligned}$$

$$\begin{aligned}&+ 106.0x\_1 + 86.0x\_2 + 120.0x\_3 + 73.0x\_4 + 132.0x\_5 + 62.0x\_6 + 141.0x\_7 \\ &+ 54.0x\_8 + 147.0x\_9 + 50.0x\_10 + 149.0x\_11 + 50.0x\_12 + 148.0x\_13 + 54.0x\_14 \\ &+ 142.0x\_15 + 61.0x\_16 + 133.0x\_17 + 71.0x\_18 + 122.0x\_19 + 84.0x\_20 + 108.0x\_21 \\ &+ 98.0x\_22 + 94.0x\_23 + 112.0x\_24 + 80.0x\_25 + 125.0x\_26 + 68.0x\_27 + 136.0x\_28 \\ &+ 59.0x\_29 + 144.0x\_30 + 52.0x\_31 + 149.0x\_32 + 50.0x\_33 + 149.0x\_34 + 51.0x\_35 \\ &+ 146.0x\_36 + 56.0x\_37 + 139.0x\_38 + 65.0x\_39 + 129.0x\_40 + 76.0x\_41 + 116.0x\_42 \\ &+ 90.0x\_43 + 102.0x\_44 + 104.0x\_45 + 88.0x\_46 + 117.0x\_47 + 75.0x\_48 + 130.0x\_49 \\ &+ 64.0x\_50 \leq 1000;\end{aligned}$$

bin $x\_1, x\_2, x\_3, x\_4, x\_5, x\_6, x\_7, x\_8, x\_9, x\_10, x\_11, x\_12, x\_13, x\_14, x\_15, x\_16, x\_17, x\_18, x\_19,$
$x\_20, x\_21, x\_22, x\_23, x\_24, x\_25, x\_26, x\_27, x\_28, x\_29, x\_30, x\_31, x\_32, x\_33, x\_34, x\_35, x\_36,$
$x\_37, x\_38, x\_39, x\_40, x\_41, x\_42, x\_43, x\_44, x\_45, x\_46, x\_47, x\_48, x\_49, x\_50;$

We run the linear objective function and constraints above in LPSolve IDE, and we get the result:

Value of objective function: 1680.00000000

Actual values of the variables:

x_1	1
x_6	1
x_10	1
x_12	1
x_14	1
x_19	1
x_20	1
x_25	1
x_31	1
x_39	1
x_44	1
x_45	1
x_50	1

Optimal solution 1680 after 1519 iter, 1210 nodes (gap 1.3%).
Relative numeric accuracy $||*|| = 5.55112\text{e-}017$

Our solution shows that the thieves only need to take the first piece, the sixth piece, the tenth piece, the twelfth piece, the fourteenth piece, the nineteenth, the twentieth piece, the twenty-fifth piece, the thirty-first piece, the thirty-ninth piece, the fortieth piece, the forty-fifth piece and the fiftieth piece. All other variables are zero, meaning that thieves do not need to take those pieces. And the value will be the largest: 1680(in thousand Yuan).

Problem 2

Suppose there will be volcano eruptions in 6 hours, so we have to escape the city as soon as possible. There is not enough time for us to pack all the things and we will drive a truck to escape. However, we know the truck has weight capacity and volume capacity. In this problem, the bus can only contain things whose total weight is less than 1000 kg and things' total volume is less than 1000

liters. Due to lack of bus capacity and time, we can only choose take things among 50 classes, such as food, water, clothes, books, etc. This time we can take multiple objects from each class, such as five jackets or two bottle of water. For each object we can take away, it has its own value(in dollars), weight(in kg) and volume, defined in the same way as problem 1.

We want to stay as long as possible with the things we choose. Thus, we want to maximize the value of all those objects. For example, since we cannot live for long without water, it may have a higher value than books.

To solve this problem, we already know we need to maximize the value. At the same time, we have two constraints: maximum weight we can have is 1000 kg and maximum volume is 1000 liters. In this problem, we can bring many objects from classes as long as they can be put into the truck. Then, there is no constraint for each object's quantity.

The idea is to define an integer variable x_i for each class($1 \leq i \leq 50$). i here is the index for each object and x_i means the i-th object. In this problem, x_i is a positive integer. We wish to maximize the sum

$$\sum_{i=1}^{50} \text{value}[i] * x_i$$

subject to the constraints of the truck weight and volume capacity. The weight is less than 1000 kg and the volume is no more than 1000 liters. We can express as

$$\sum_{i=1}^{50} \text{weight}[i] * x_i \leq 1000$$

$$\sum_{i=1}^{50} \text{volume}[i] * x_i \leq 1000$$

Also, x_i is an integer as defined above:

$$x_i \in \mathbb{Z}$$

The LPSolve input file is the same as Problem 1 except for the last constraint, since in this problem, we can take multiple objects from each class, x_i does not need to be 1 or 0:

int $x\_1, x\_2, x\_3, x\_4, x\_5, x\_6, x\_7, x\_8, x\_9, x\_10, x\_11, x\_12, x\_13, x\_14, x\_15, x\_16, x\_17, x\_18, x\_19,$
$x\_20, x\_21, x\_22, x\_23, x\_24, x\_25, x\_26, x\_27, x\_28, x\_29, x\_30, x\_31, x\_32, x\_33, x\_34, x\_35, x\_36,$
$x\_37, x\_38, x\_39, x\_40, x\_41, x\_42, x\_43, x\_44, x\_45, x\_46, x\_47, x\_48, x\_49, x\_50;$

We run the linear objective function and constraints above and get the output:

Value of objective function: 2079.00000000

Actual values of the variables:

x_6 6

x_25 7

x_50 1

Optimal solution 2079 after 468 iter, 416 nodes (gap 1.0%).
Relative numeric accuracy $||*|| = 0$

Our solution shows that we only need to bring six objects from class 6, seven objects from class 25 and one object from class 25. All other variables are zero, meaning that we do not need to take those things for escaping. Thus, we can achieve the maximum value: 2079(in dollars). In this case, those classes may be food, water and lighters, since we need to drink water and eat food for staying long; also, we need the lighter to burn wood for keeping warm.

Problem 3

Assume we will drive a truck to shop some seeds in the city. We plan to buy 50 kinds of seeds, including tomatoes, potatoes, apples, etc. The truck we use has a limit of weight(in kg) and volume(in liters): both needs to be less than 1000. For each type of seed, we can either buy 3 pounds which is an integer quantity or 3.35 pounds which is a double type number. For each type of seed, it has its own value, weight and volume, defined as the same way in Problem 1.
We want to get the seeds which may have the highest value, since those kinds of seeds may have the highest quality. For example, they may have better ability to grow up which facing some severe weathers.

To solve this problem, we already know we need to maximize the total value of the seeds. Meantime, we need to ensure that the total weight and total volume is less than 1000.

The idea is to define a float number x_i for each type of seeds($1 \leq i \leq 50$). i here is the index for each object and x_i here is the i-th object. We wish to maximize the total value(value[i] means ith objects's value):

$$\sum_{i=1}^{50} \text{value}[i] * x_i$$

subject to the constraints of the truck weight and volume capacity. The weight is less than 1000 kg and the volume is no more than 1000 liters. Then, volume[i] means volume of i-th object and weight[i] means weight of i-th object. We can express as

$$\sum_{i=1}^{50} \text{weight}[i] * x_i \leq 1000$$

$$\sum_{i=1}^{50} \text{volume}[i] * x_i \leq 1000$$

Also, x_i is an integer as defined above:

$$x_i \in \mathbb{R}, x_i \geq 0$$

The LPSolve input file is the same as Problem 1 except for the last constraint. In this question, since x_i is a positive real number, we do not have to define it specifically. And we run the linear objective function and constraints above in LPSolve IDE and get the result below:

Value of objective function: 2101.17403315

Actual values of the variables:

x_25	5.87017
x_50	8.28729

Optimal solution 2101.17403315 after 4 iter.
Relative numeric accuracy $|| * || = 0$

This result shows that we only need to buy 5.87017 pounds of type 25 and 8.28729 pounds of type 50. All other variables are zero, meaning that we do not need to buy those types of seeds. In this way, the value can be maximized: it will be 2101.17403315.