Knapsack

10/08/2021

By: Xinming Zhang

There is a Knapsack problem to solve by using Linear Programming to get the solutions under different constraints. I will try to solve it with linear programming. Here is the problem description:

Suppose I am carrying out the electronic parts for a car company in a container, and there will be different rules for each trip.

Suppose I have 100 unique objects with various values, weights and volumes.

Suppose object i (where $1 \leq i \leq 100$) has the following value, weight and volume:

$$\text{value} = \text{floor}(50+25\cos(7i)) \text{ thousands of dollars}$$

$$\text{weight} = \text{floor}(30+12\cos(6i+2)) \text{ kg}$$

$$\text{volume} = \text{floor}(40+8\cos(5i+4)) \text{ liters}$$

Please note that the floor function truncates a real value x to the largest integer less than or equal to x.

An example can be given as: floor(3.2)=3, and floor(3)=3.

Suppose I have a container that has a weight capacity of 400 kg and volume capacity of 500 liters.

a) For the first trip, every object is unique, and I want to decide what objects I should put into the container to maximize the total value to make the most profit for me, and check if any constraints are binding in this case.

First, I would start with setting up the variables and the LP formulation.

Let $value_i$, $weight_i$, and $volume_i$ represent the value, weight, and volume of the i_{th} object, and i is in an interval from 1 to 100.

I will use c_i to represent the coefficient of the i object. It also represents the number of that object in real life.

My object is maximizing the value of the objects I would bring in the container. They are unique, so I can only bring one or none of the objects.

The objective function would be following:

$$\text{Max: } \sum_{i=1}^{100} c_i * value_i$$

From the rules of the first trip, we can set up the constraints as:

$$\sum_{i=1}^{100} c_i * weight_i \leq 400$$

$$\sum_{i=1}^{100} c_i * volume_i \leq 500$$

c_i are binary numbers, which means they can either be one or zero.

I am going to use lp-solve to solve my problem, so after I know what LP formulation is, I chose to use **Java** code to generate my input file for lp_solve.

My code is following (green lines are the comments):

```
package math.pkg381;
/**
 * @author Xinming Zhang
 */
public class Math381 {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        //Defining the variables as three arrays to save the values, weights,
        //and volumes of each object.
        double[] value = new double[100];
        double[] weight = new double[100];
        double[] volume = new double [100];
        //A for loop to calculate those variables as described in the problem
        for (int i = 1; i <= 100; i++) {
            value[i-1] = Math.floor(50 + 25 * Math.cos(7*i));
            weight[i-1] = Math.floor(30 + 12 * Math.cos(6*i+2));
            volume[i-1] = Math.floor(40 + 8 * Math.cos(5*i+4));
        }
        //Print the "max: " in the output window as the beginning
        System.out.print("max:");
        //A for loop to print objective function by printing out the each term
        for (int i=1; i<=100; i++){
            System.out.print(" + " + Math.round(value[i-1]) + "*c_" + i);
        }
```

```
        //Finish the objective function of the input file
        System.out.println(";");
        //A for loop to print out the first constraint (weight capacity)
        for (int i=1; i<=100; i++){
            System.out.print("+ " + Math.round(weight[i-1]) + "*c_" + i +" ");
        }
        //Finish the first constraint
        System.out.println(" <= 400;");
        //A for loop to print out the second constraint (volume capacity)
        for (int i=1; i<=100; i++){
            System.out.print("+ " + Math.round(volume[i-1]) + "*c_" + i +" ");
        }
        //Finish the second constraint
        System.out.println(" <= 500;");
        //Last constraint (Binary numbers)
        System.out.print("int ");
        for (int i=1; i<=100; i++){
            if (i == 100) {
                System.out.print("c_" + i + ";");
            } else {
                System.out.print("c_" + i + ", ");
            }
        }
    }
}
```

I can get my **input file** for lp_solve by running the code given above as following:

```
max: + 68*c_1 + 53*c_2 + 36*c_3 + … + 60*c_98 + 43*c_99 + 29*c_100;

+ 28*c_1 + 31*c_2 + 34*c_3 + … + 39*c_98 + 37*c_99 + 34*c_100  <= 400;
```

+ 32*c_1 + 41*c_2 + 47*c_3 + … + 34*c_98 + 33*c_99 + 41*c_100 <= 500;

bin c_1, c_2, c_3, c_4, … , c_98, c_99, c_100;

Run lp-solve program on the input file above, I can get the **output** as following:

Value of objective function: 1000.00000000

Actual values of the variables:

c_1	1
c_9	1
c_17	1
c_26	1
c_35	1
c_44	1
c_45	1
c_61	1
c_69	1
c_70	1
c_78	1
c_79	1
c_88	1
c_89	1

All other variables are zero.

From reading the output, we know the objects I am going to carry out are objects 1, 9, 17, 26, 35, 44, 45, 61, 69, 70, 78, 79, 88, 89 to get a maximum value of 1000 thousands of dollars. There are 14 of them from the 100 objects.

Then we can use the solution above to know which objects are the best to carry out. Using a calculator to get the total weight and volume of those objects as 400kg and 499 liters.

According to the problem, the volume capacity is 500 liters, 499 liters is smaller than 500 liters, so the second constraint (volume capacity constraint) is not binding. On

the other hand, the total weight of the objects is equal to the weight capacity, so the weight capacity constraint is binding.

Above all, the constraint $\sum_{i=1}^{100} c_i * volume_i \leq 500$ is not binding.

b) For the second trip, I am trying to put a lower bound on the number of objects I am carrying out, but my objective does not change. I want to see what will happen with adding a lower bound?

From the previous part, we know without a lower bound, there will be 14 objects we are going to carry out. Then, when we are putting a lower bound on the number of objects, the first try will be 15. (Any positive number smaller than or equals to 14 will not change the solution or output.)

One constraint will be added to LP as:

$$\sum_{i=1}^{100} c_i \geq 15$$

I have to add some lines of code to regenerate the input file, I will add the following lines after the second constraint as my third one.

Code added after the second constraint:

```
//A for loop to print out the third constraint (lower bound of the number of objects)
for (int i=1; i<=100; i++){
    //Finish the third constraint using if statement
    if (i == 100) {
        System.out.println("+ c_" + i + " >= 15;");
    } else {
        System.out.print("+ c_" + i + " ");
    }
```

}

Then, the input file have a new line after the third line as following:

max: + 68*c_1 + 53*c_2 + 36*c_3 + … + 60*c_98 + 43*c_99 + 29*c_100;

+ 28*c_1 + 31*c_2 + 34*c_3 + … + 39*c_98 + 37*c_99 + 34*c_100 <= 400;

+ 32*c_1 + 41*c_2 + 47*c_3 + … + 34*c_98 + 33*c_99 + 41*c_100 <= 500;

+ c_1 + c_2 + c_3 +... + c_98 + c_99 + c_100 >= 15;

bin c_1, c_2, c_3, c_4, … , c_98, c_99, c_100;

The output from lp_solve is:

Value of objective function: 985.00000000

Actual values of the variables:

c_1	1
c_16	1
c_20	1
c_25	1
c_35	1
c_36	1
c_44	1
c_45	1
c_59	1
c_60	1
c_69	1
c_70	1

c_79 1

c_88 1

c_89 1

All other variables are zero.

Then the total volume decreased compared to the one we got from the previous part, and the number of objects changed to 15 because of the lower bound we just added. The objects 1, 16, 20, 25, 35, 36, 44, 45, 59, 60, 69, 70, 79, 88, 89 are chosen. Other than that, the total weight and volume has changed to 394 and 500. The total weight is decreased , and the total volume is increased to the maximum. From that, the weight constraint is binding, and the volume constraint is not binding.

After all that, we can increase the lower bound to 16 using the same method, but the output will be “This problem is infeasible”, then it will not have an optimal solution for the LP. For this LP to be solvable, the greatest lower bound will be 15. The reason for that is when we are setting the lower bound of the number of objects to 16 or more, the volume capacity will be broken, and it will violate the volume constraint, so that the LP will not be solved.

c) On the third and last trip, I will not limit the number of objects anymore, which also means that the objects are no longer unique.

According to the information above, I am going to the change the meaning of c_i, it will number of object will be taken from class i. On the other hand, c_i is not binary

anymore, they will be non-negative integers. That will be the new constraint edited to the LP from part a.

I will edit the last constraint from part a as:

```
//Last constraint (non-negative numbers)
System.out.print("int ");
```

The change is to switch “bin “ to “int “.

Run the code to get the new input file as:

max: + 68*c_1 + 53*c_2 + 36*c_3 + … + 60*c_98 + 43*c_99 + 29*c_100;

+ 28*c_1 + 31*c_2 + 34*c_3 + … + 39*c_98 + 37*c_99 + 34*c_100 <= 400;

+ 32*c_1 + 41*c_2 + 47*c_3 + … + 34*c_98 + 33*c_99 + 41*c_100 <= 500;

int c_1, c_2, c_3, c_4, … , c_98, c_99, c_100;

The output form lp_solve is:

Value of objective function: 1110.00000000

Actual values of the variables:

c_35	10
c_88	5

All other variables are zero.

The solution above shows that 10 objects from class 35, and 5 objects from class 88 are the chosen ones to carry out. The total value of the chosen object is 1110 thousands of dollars, and that is larger than the two solutions from part a and part b. That does make sense, because we are not limiting the number of objects anymore From looking at the value, weight, and volume of objects 35 and 88. Object 88 has the most valuable one in all 100 objects. That is the reason why object 88 is in the solution, and object 55 has the least volume and middle weight with good value. Therefore, the container could bring lots of those to maximize the total value. The solution makes more sense when we are looking at those three variables of the objects.