Homework 1. Knapsack

MATH 381
Disrete Mathematical Modeling
Lucas Liu 2062671

January 14, 2023

There are many occasions where a company may need to collect their most valuable products but are restricted by a carrying capacity. For instance, imagine a furniture company wanting to move as much value as possible from one location to another. How can we determine which objects to choose?

To determine which objects should be shipped, we can think of each individual product(sofa, table, etc.) as having three different attributes: the **value** of the product, the **weight** of the product, and the **volume** or space the product occupies.

Consider a furniture store owning 100 different pieces of furniture $x_1, x_2, , x_{100}$ with each piece having attributes:

$$\text{Where } i \in [1, 100]$$

$$x_{i,value} = floor(50 + 41\cos(14i)) \text{ dollars}$$

$$x_{i,weight} = floor(22 + 13\cos(5i + 1)) \text{ kg}$$

$$x_{i,volume} = floor(25 + 19\cos(4i - 2)) \text{ L}$$

Where the function $floor(x)$ round the number down to the nearest whole integer.
The total value of the objects, and our objective function to be maximized, is:

$$V_{total} = \sum_{n=1}^{100} x_{n,value} \cdot x_n$$

Where x_n is a binary integer- either 0 or 1- that represents if object n‘ is not chosen or chosen, respectively.

However, the total volume and weight that can be held by the shipping container restricts the amount and size of products that can be included. Thus, for a container of size 200 Liters that can hold 200 kg, the restrictions are:

$$\sum_{n=1}^{100} x_{n,weight} \cdot x_n \leq 200$$

$$\sum_{n=1}^{100} x_{n,volume} \cdot x_n \leq 200$$

Our problem then becomes:

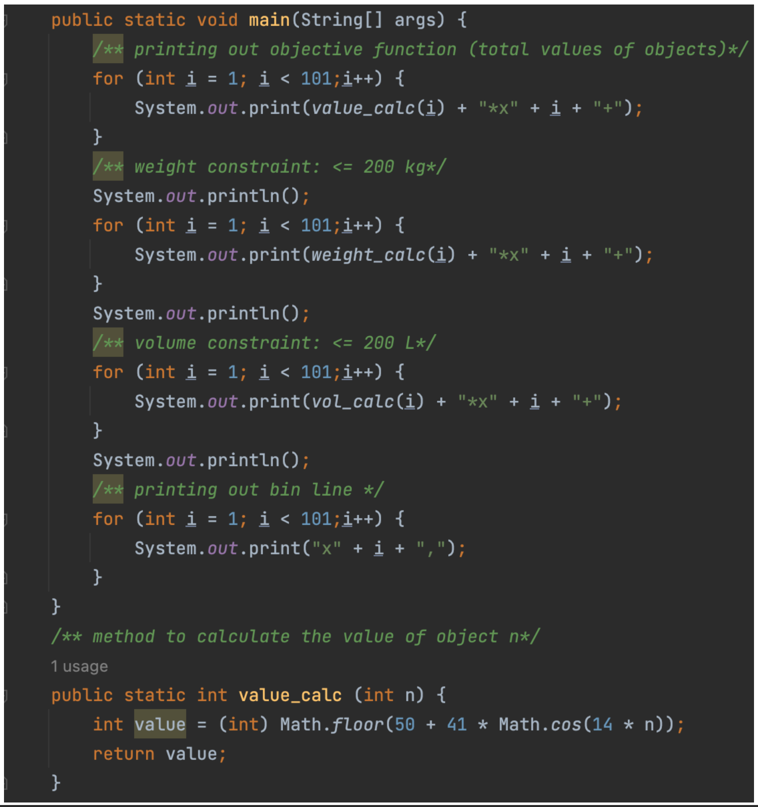
$$\text{Maximize} \sum_{n=1}^{100} x_{n,value} \cdot x_n$$

subject to

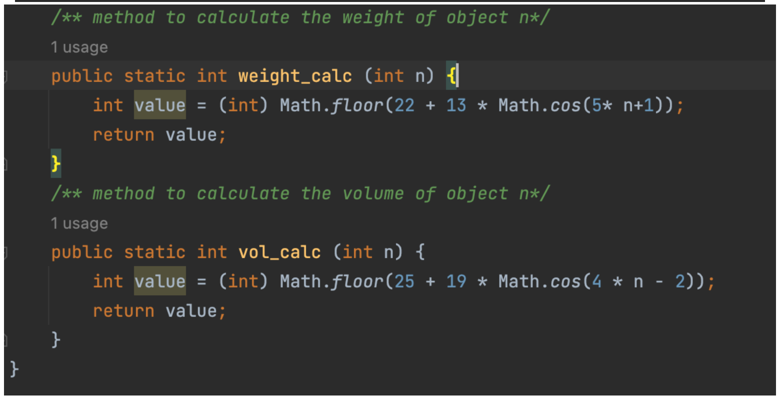
$$\sum_{n=1}^{100} x_{n,weight} \cdot x_n \leq 200 \text{ } \textbf{(The upper bound of weight held by container)}$$

$$\sum_{n=1}^{100} x_{n,volume} \cdot x_n \leq 200 \text{ } \textbf{(The upper bound of volume held by container)}$$

To solve this problem, the lpsolver by MIT can be used. The first step, however, is to obtain the linear functions to be inserted into the solver. This can be done in Java using the code:

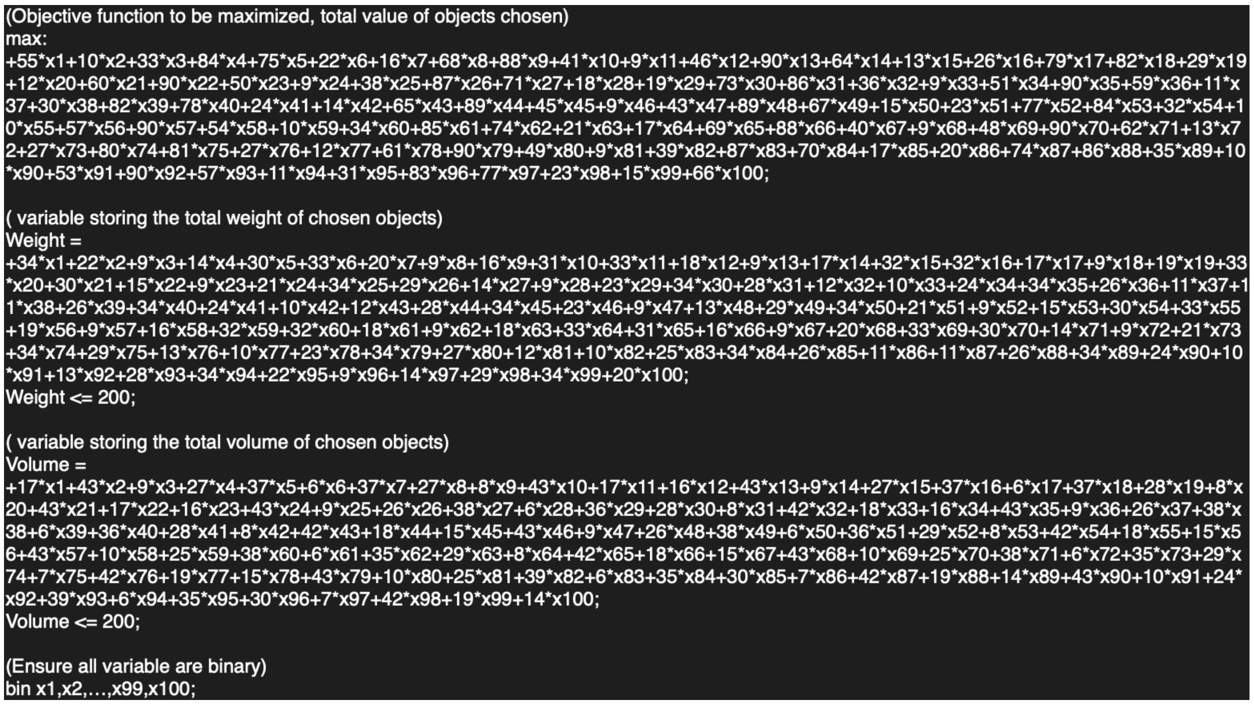


```java
    public static void main(String[] args) {
        /** printing out objective function (total values of objects)*/
        for (int i = 1; i < 101;i++) {
            System.out.print(value_calc(i) + "*x" + i + "+");
        }
        /** weight constraint: <= 200 kg*/
        System.out.println();
        for (int i = 1; i < 101;i++) {
            System.out.print(weight_calc(i) + "*x" + i + "+");
        }
        System.out.println();
        /** volume constraint: <= 200 L*/
        for (int i = 1; i < 101;i++) {
            System.out.print(vol_calc(i) + "*x" + i + "+");
        }
        System.out.println();
        /** printing out bin line */
        for (int i = 1; i < 101;i++) {
            System.out.print("x" + i + ",");
        }
    }
    /** method to calculate the value of object n*/
    1 usage
    public static int value_calc (int n) {
        int value = (int) Math.floor(50 + 41 * Math.cos(14 * n));
        return value;
    }
```



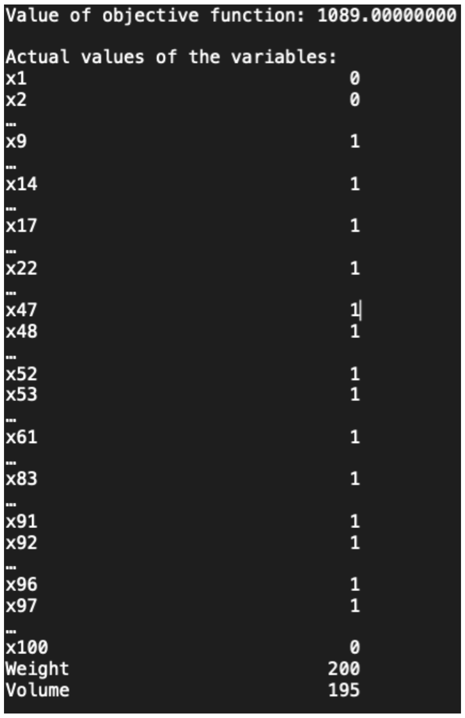
```java
    /** method to calculate the weight of object n*/
    1 usage
    public static int weight_calc (int n) {
        int value = (int) Math.floor(22 + 13 * Math.cos(5* n+1));
        return value;
    }
    /** method to calculate the volume of object n*/
    1 usage
    public static int vol_calc (int n) {
        int value = (int) Math.floor(25 + 19 * Math.cos(4 * n - 2));
        return value;
    }
}
```

The code includes methods to calculate the value, weight, and volume of each object using the previously mentioned calculations. These methods are used in for-loops to print out the input equations one after the other. Using the output from this code, we can obtain the LP input file:



```
(Objective function to be maximized, total value of objects chosen)
max:
+55*x1+10*x2+33*x3+84*x4+75*x5+22*x6+16*x7+68*x8+88*x9+41*x10+9*x11+46*x12+90*x13+64*x14+13*x15+26*x16+79*x17+82*x18+29*x19
+12*x20+60*x21+90*x22+50*x23+9*x24+38*x25+87*x26+71*x27+18*x28+19*x29+73*x30+86*x31+36*x32+9*x33+51*x34+90*x35+59*x36+11*x
37+30*x38+82*x39+78*x40+24*x41+14*x42+65*x43+89*x44+45*x45+9*x46+43*x47+89*x48+67*x49+15*x50+23*x51+77*x52+84*x53+32*x54+1
0*x55+57*x56+90*x57+54*x58+10*x59+34*x60+85*x61+74*x62+21*x63+17*x64+69*x65+88*x66+40*x67+9*x68+48*x69+90*x70+62*x71+13*x7
2+27*x73+80*x74+81*x75+27*x76+12*x77+61*x78+90*x79+49*x80+9*x81+39*x82+87*x83+70*x84+17*x85+20*x86+74*x87+86*x88+35*x89+10
*x90+53*x91+90*x92+57*x93+11*x94+31*x95+83*x96+77*x97+23*x98+15*x99+66*x100;

( variable storing the total weight of chosen objects)
Weight =
+34*x1+22*x2+9*x3+14*x4+30*x5+33*x6+20*x7+9*x8+16*x9+31*x10+33*x11+18*x12+9*x13+17*x14+32*x15+32*x16+17*x17+9*x18+19*x19+33
*x20+30*x21+15*x22+9*x23+21*x24+34*x25+29*x26+14*x27+9*x28+23*x29+34*x30+28*x31+12*x32+10*x33+24*x34+34*x35+26*x36+11*x37+1
1*x38+26*x39+34*x40+24*x41+10*x42+12*x43+28*x44+34*x45+23*x46+9*x47+13*x48+29*x49+34*x50+21*x51+9*x52+15*x53+30*x54+33*x55
+19*x56+9*x57+16*x58+32*x59+32*x60+18*x61+9*x62+18*x63+33*x64+31*x65+16*x66+9*x67+20*x68+33*x69+30*x70+14*x71+9*x72+21*x73
+34*x74+29*x75+13*x76+10*x77+23*x78+34*x79+27*x80+12*x81+10*x82+25*x83+34*x84+26*x85+11*x86+11*x87+26*x88+34*x89+24*x90+10
*x91+13*x92+28*x93+34*x94+22*x95+9*x96+14*x97+29*x98+34*x99+20*x100;
Weight <= 200;

( variable storing the total volume of chosen objects)
Volume =
+17*x1+43*x2+9*x3+27*x4+37*x5+6*x6+37*x7+27*x8+8*x9+43*x10+17*x11+16*x12+43*x13+9*x14+27*x15+37*x16+6*x17+37*x18+28*x19+8*x
20+43*x21+17*x22+16*x23+43*x24+9*x25+26*x26+38*x27+6*x28+36*x29+28*x30+8*x31+42*x32+18*x33+16*x34+43*x35+9*x36+26*x37+38*x
38+6*x39+36*x40+28*x41+8*x42+42*x43+18*x44+15*x45+43*x46+9*x47+26*x48+38*x49+6*x50+36*x51+29*x52+8*x53+42*x54+18*x55+15*x5
6+43*x57+10*x58+25*x59+38*x60+6*x61+35*x62+29*x63+8*x64+42*x65+18*x66+15*x67+43*x68+10*x69+25*x70+38*x71+6*x72+35*x73+29*x
74+7*x75+42*x76+19*x77+15*x78+43*x79+10*x80+25*x81+39*x82+6*x83+35*x84+30*x85+7*x86+42*x87+19*x88+14*x89+43*x90+10*x91+24*
x92+39*x93+6*x94+35*x95+30*x96+7*x97+42*x98+19*x99+14*x100;
Volume <= 200;

(Ensure all variable are binary)
bin x1,x2,…,x99,x100;
```

Inserting this into the LP solver yields the result:



```
Value of objective function: 1089.00000000

Actual values of the variables:
x1                              0
x2                              0
…
x9                              1
…
x14                             1
…
x17                             1
…
x22                             1
…
x47                             1
x48                             1
…
x52                             1
x53                             1
…
x61                             1
…
x83                             1
…
x91                             1
x92                             1
…
x96                             1
x97                             1
…
x100                            0
Weight                        200
Volume                        195
```

The results indicate that, under the provided restrictions, the furniture store should put objects 9,14,17,22,47,48,52,53,61,83,91,92,96, and 97 into the knapsack (shipping container). For a total value of 1089 dollars, a total weight of 200 kg, and a total volume of 195 Liters.

Because the total weight of the maximum combination of objects is equal to the max weight (200 kg), the constraint related to the weight of the objects **is a binding constraint**. Decreasing the maximum weight will lead to a different result. On the other hand, the total volume of the objects is less than the maximum volume so the volume **is not a binding constraint**

But what if there is a minimum number of pieces of furniture that needs to be moved. To address this, a new constraint can be introduced into the LP that ensures the total number of objects is at greater than or equal to the lower bound of the number of objects:

New Constraint:

$$N_{total} = \sum_{n=1}^{100} x_n \geq N_{\text{lower bound}}$$

Our problem then becomes:

$$\text{Maximize } \sum_{n=1}^{100} x_{n,value} \cdot x_n$$

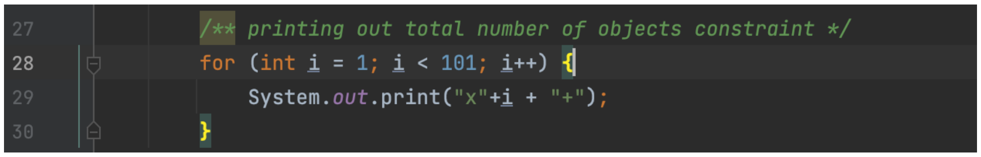
subject to

$$\sum_{n=1}^{100} x_{n,weight} \cdot x_n \leq 200 \ \textbf{(The upper bound of weight held by container)}$$

$$\sum_{n=1}^{100} x_{n,volume} \cdot x_n \leq 200 \ \textbf{(The upper bound of volume held by container)}$$

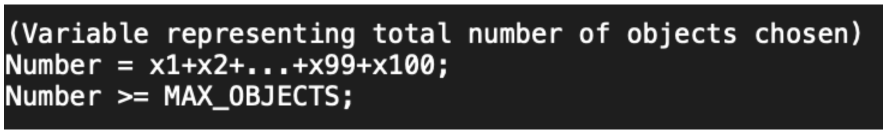
$$\sum_{n=1}^{100} x_n \geq N_{\text{lower bound}} \ \textbf{(The lower bound of total objects held by container)}$$

Like the previous constraints, the new constraint can also be generated using equivalent code:



```
/** printing out total number of objects constraint */
for (int i = 1; i < 101; i++) {
    System.out.print("x"+i + "+");
}
```

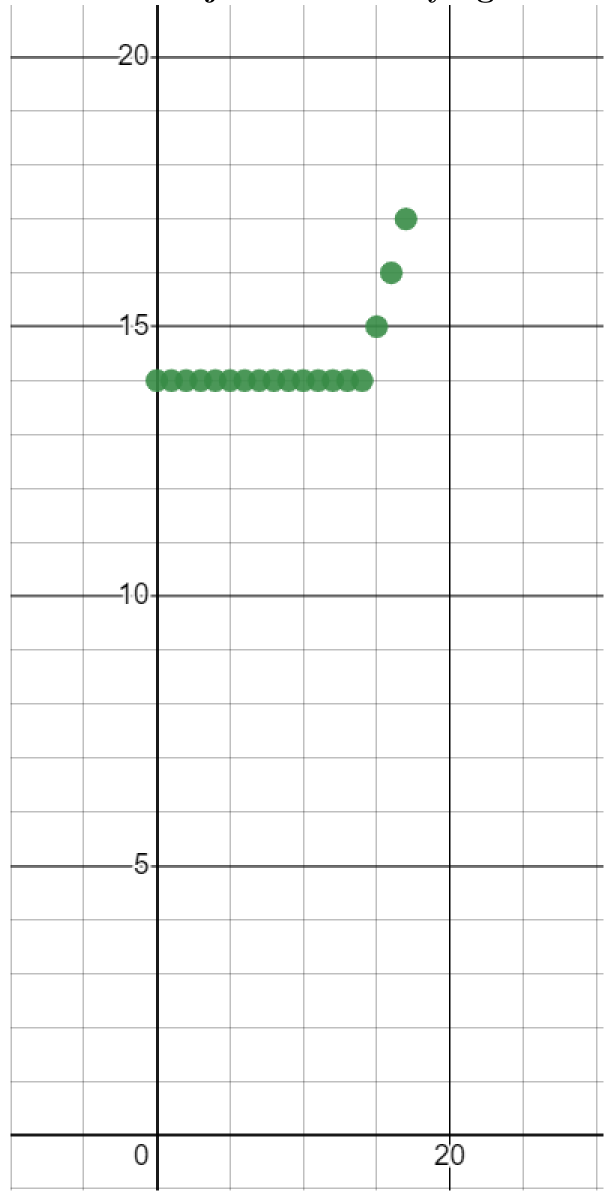
Using the output, we can add the additional constraint to our LP input file:



```
(Variable representing total number of objects chosen)
Number = x1+x2+...+x99+x100;
Number >= MAX_OBJECTS;
```

The below graph shows the total value of the objects and the total number objects chosen as a function of the lower bound of objects:

Total Number of Objects with Varying Lower Bounds

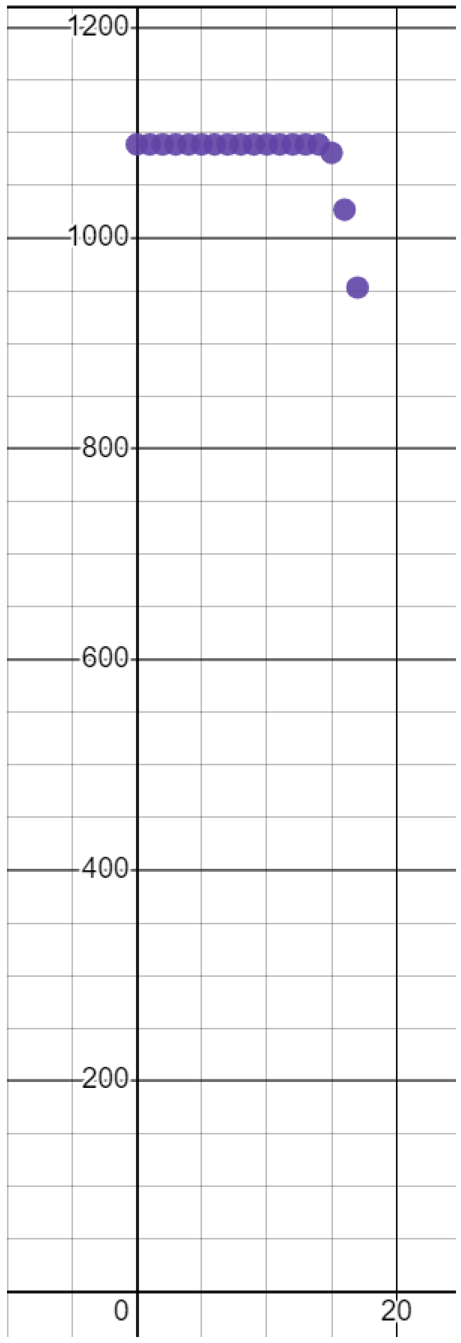


The function can be represented by:

$$N_{total}(x) = \begin{cases} 14 & 0 \le x \le 14 \\ x & 15 \le x \le 17 \end{cases}$$

This tells us that, for a container with weight limit 200 kg and volume limit 200 L, a combination including 14 objects yield the maximum total value. However, when the lower bound is increased to a value above 14, the combination yielding the maximum value contains the lower-bound objects in the combination. Finally, when the lower bound $x \ge 18$, there are no possible combinations of the objects that meet our constraint- there is no sum of 18 or more objects who's volumes are less than 200 L and who's weight is less than 200 kg.

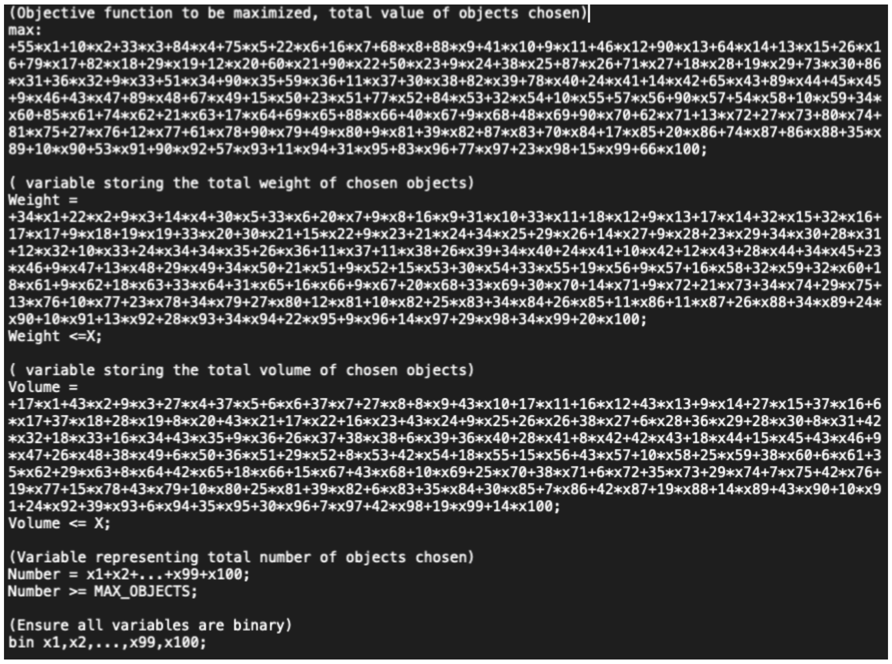
Total Value of Objects with Varying Lower Bounds



Similarly, the plot above reinforces the idea that a combination including 14 objects yields the maximum total value (1089 dollars). Once the lower bound of total objects exceeds 14, the total value of the objects decreases linearly. Since there are no possible combinations with at least 18 objects or more, this function is also undefined on the domain $x \geq 18$

However, our current findings assume that the container has a maximum weight and volume limit of 200 kg and 200 Liters. How do our results change as the upper bound of these parameters change?

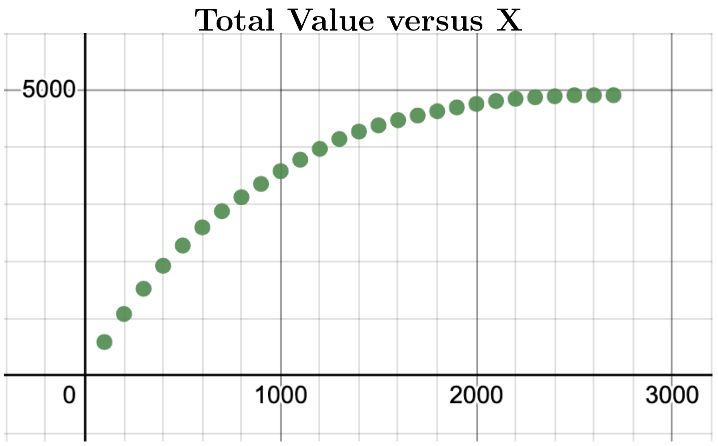
To examine the behavior, we can make minor changes to our LP input file. Specifically, changing the upper limits of our Weight and Volume variables. Where our new upper bounds are represented by **X**, the LP input file takes the form:



```
(Objective function to be maximized, total value of objects chosen)
max:
+55*x1+10*x2+33*x3+84*x4+75*x5+22*x6+16*x7+68*x8+88*x9+41*x10+9*x11+46*x12+90*x13+64*x14+13*x15+26*x1
6+79*x17+82*x18+29*x19+12*x20+60*x21+90*x22+50*x23+9*x24+38*x25+87*x26+71*x27+18*x28+19*x29+73*x30+86
*x31+36*x32+9*x33+51*x34+90*x35+59*x36+11*x37+30*x38+82*x39+78*x40+24*x41+14*x42+65*x43+89*x44+45*x45
+9*x46+43*x47+89*x48+67*x49+15*x50+23*x51+77*x52+84*x53+32*x54+10*x55+57*x56+90*x57+54*x58+10*x59+34*
x60+85*x61+74*x62+21*x63+17*x64+69*x65+88*x66+40*x67+9*x68+48*x69+90*x70+62*x71+13*x72+27*x73+80*x74+
81*x75+27*x76+12*x77+61*x78+90*x79+49*x80+9*x81+39*x82+87*x83+70*x84+17*x85+20*x86+74*x87+86*x88+35*x
89+10*x90+53*x91+90*x92+57*x93+11*x94+31*x95+83*x96+77*x97+23*x98+15*x99+66*x100;

( variable storing the total weight of chosen objects)
Weight =
+34*x1+22*x2+9*x3+14*x4+30*x5+33*x6+20*x7+9*x8+16*x9+31*x10+33*x11+18*x12+9*x13+17*x14+32*x15+32*x16+
17*x17+9*x18+19*x19+33*x20+30*x21+15*x22+9*x23+21*x24+34*x25+29*x26+14*x27+9*x28+23*x29+34*x30+28*x31
+12*x32+10*x33+24*x34+34*x35+26*x36+11*x37+11*x38+26*x39+34*x40+24*x41+10*x42+12*x43+28*x44+34*x45+23
*x46+9*x47+13*x48+29*x49+34*x50+21*x51+9*x52+15*x53+30*x54+33*x55+19*x56+9*x57+16*x58+32*x59+32*x60+1
8*x61+9*x62+18*x63+33*x64+31*x65+16*x66+9*x67+20*x68+33*x69+30*x70+14*x71+9*x72+21*x73+34*x74+29*x75+
13*x76+10*x77+23*x78+34*x79+27*x80+12*x81+10*x82+25*x83+34*x84+26*x85+11*x86+11*x87+26*x88+34*x89+24*
x90+10*x91+13*x92+28*x93+34*x94+22*x95+9*x96+14*x97+29*x98+34*x99+20*x100;
Weight <=X;

( variable storing the total volume of chosen objects)
Volume =
+17*x1+43*x2+9*x3+27*x4+37*x5+6*x6+37*x7+27*x8+8*x9+43*x10+17*x11+16*x12+43*x13+9*x14+27*x15+37*x16+6
*x17+37*x18+28*x19+8*x20+43*x21+17*x22+16*x23+43*x24+9*x25+26*x26+38*x27+6*x28+36*x29+28*x30+8*x31+42
*x32+18*x33+16*x34+43*x35+9*x36+26*x37+38*x38+6*x39+36*x40+28*x41+8*x42+42*x43+18*x44+15*x45+43*x46+9
*x47+26*x48+38*x49+6*x50+36*x51+29*x52+8*x53+42*x54+18*x55+15*x56+43*x57+10*x58+25*x59+38*x60+6*x61+3
5*x62+29*x63+8*x64+42*x65+18*x66+15*x67+43*x68+10*x69+25*x70+38*x71+6*x72+35*x73+29*x74+7*x75+42*x76+
19*x77+15*x78+43*x79+10*x80+25*x81+39*x82+6*x83+35*x84+30*x85+7*x86+42*x87+19*x88+14*x89+43*x90+10*x9
1+24*x92+39*x93+6*x94+35*x95+30*x96+7*x97+42*x98+19*x99+14*x100;
Volume <= X;

(Variable representing total number of objects chosen)
Number = x1+x2+...+x99+x100;
Number >= MAX_OBJECTS;

(Ensure all variables are binary)
bin x1,x2,...,x99,x100;
```

Here **MAX_OBJECTS** = 0 as we are no longer concerned with how a minimum total object bound affects the total value. x

Solving the LP from **X** = 100 to **X** = 2700 in steps of 100 reveals the following graph:



The graph reveals a quadratic relationship between the total value of the objects and the maximum weight and volume that can be carried. The total value increases with **X** until **X** = 2500, where it reaches a constant value of 4916 dollars. This tells us that under no restrictions with regard to total weight, total volume, or total number, the maximum total value of a combination the objects is 4916 dollars. In other words, this is the total value of all the pieces of furniture in the inventory.