The Knapsack Problem

Ryan Mouw

January 13, 2021

1 Introduction

This paper describes how to solve a problem using linear programming. Linear programming is the mathematics of optimizing a linear objective function, given a set of linear constraint functions.

2 The Problem

Suppose Sally is down on her luck and decides to rob a bank. After months of preparation, Sally finds herself in a bank vault surrounded by 50 different varieties of precious metals of various sizes, weights, and values. Each metal bar is labeled 1-50, and there is only one bar of each type of metal. Preferring to work alone, Sally only brought one knapsack with which to stow the loot and escape. The four questions that need to be answered are as follows;

a) Which metal bars should Sally choose to maximize her payout, given that she can only fit a finite amount of loot in her knapsack?

b) Which metals bars should Sally choose in order to maximize her payout if Sally wants prime numbered bars to compose at least 50% of the take?

c) Which metals bars should Sally choose in order to maximize her payout if Sally wants prime numbered bars to compose at least 75% of the take?

d) Which metal bars should Sally choose to maximize her payout if there were an unlimited supply of each type of metal bar, but still only had finite space in her knapsack?

3 The Constraints

In an unimpeded world, Sally would take every metal bar that was in the vault, maximizing her profit. But their is a problem. Sally's only knapsack can hold no more than 200kg and has a maximum volume of 100 liters. The volume and weight capacity of Sally's knapsack are two constraints that Sally must contend with in order to maximize the value of the metal bars she takes from the vault. Each of the metal bars in the vault has a value, weight, and volume that

corresponds to the following formulas, rounded down to the nearest integer.

$value(thousands\ of\ dollars) = 50 + 30\cos(i)$
$weight(kg) = 14 + 9\cos(11i + 2)$
$volume(liters) = 10 + 2\cos(4i - 1)$

For example, for metal bar $i = 15$:

$value(thousands\ of\ dollars) = 50 + 30\cos(15) = 27.2$ which rounds down to \$27.
$weight(kg) = 14 + 9\cos(11 \times 15 + 2) = 6.08$ which rounds down to 6kg.
$volume(liters) = 10 + 2\cos(4 \times 15 - 1) = 8.5$ which rounds down to 8 liters.

4 Assigning Variables and Creating Functions

First we create the objective function. This is the thing that needs to be maximized or minimized in a linear programming problem. For Sally, the objective function is the sum of all the values of the metal bars that are placed in her knapsack. In this case, Sally wants to maximize this value.
To account for the binary nature of the value of each metal bar (Take it, or leave it) our objective function multiply the value of each metal bar by an integer, x_i, which is either 0 or 1. If Sally decides to put metal bar i into her knapsack, the value of x_i will be 1, if she leaves it behind, the value of x_i will be 0.

$$Objective\ function = \sum_{i=1}^{50}(50 + 30\cos(i))x_i \quad [1]$$

To enforce the binary constraint of the x_i variables in LPSolve, we list each variable, x_i, separated by commas. The row is headed with "bin" as follows;

$$bin\ x_1,\ x_2, ..., x_i$$

The constraints are expressed as inequalities.

Constraint 1 (weight(kg)):

$$\sum_{i=1}^{50}(14 + 9\cos(11i + 2))x_i \ \leq\ 200 \quad [2]$$

Constraint 2 (volume(liters)):

$$\sum_{i=1}^{50}(10 + 2\cos(4i - 1))x_i \ \leq\ 100 \quad [3]$$

5 Creating the LPSolve input file

LPSolve IDE was used to solve this linear programming problem. LPSolve IDE takes in an objective function and constraint functions. The objective function must be in the particular form;

$$max\ :\ +c_1x_1\ +c_2x_2\ +...+c_ix_i;$$

note: "max" above can be replaced with "min" depending on whether the objective function needs to be minimized or maximized

The constraint functions take on the form;

$$+c_1x_1\ +c_2x_2\ +...+c_ix_i \leq or \geq j = some\ constant$$

The following is python code that will produce an LPSolve input filed described above.

```
#Create the objective function
obj_func = ""
#iterate through 50 times, increasing i each time
for i in range(50):
    #equation for value-floor function rounds down to nearest integer
    temp = "+ "+str(math.floor(50+30*np.cos(i+1))) + "*x_"+str(i+1)+" "
    #append each new string to get a long sequence of variables
    obj_func = obj_func + temp
#add appropriate syntax for LPSolve
obj_func = "max: "+ obj_func + ";"

#create weight constraint text
constraint_1 = ""
#50 weights for 50 items (i)
for i in range(50):
    #equation for weight
    temp = "+ "+str(math.floor(14+9*np.cos(11*(i+1)+2))) + "*x_"+str(i+1)+" "
    #concatenate weight strings
    constraint_1 = constraint_1 + temp
constraint_1 = constraint_1+" <=200;"

#create volume constraint text
constraint_2 = ""
#iterate 50 times for 50 items
```

```
for i in range(50):
    #equation for volume
    temp = "+ "+str(math.floor(10+2*np.cos(4*(i+1)-1))) + "*x_"+str(i+1)+" "
    constraint_2 = constraint_2 + temp
constraint_2 = constraint_2+" <=100;"
print(constraint_2)

#create text for "bin" constraint in LPsolve
binary = ""
for i in range(49):
    binary = binary + ("x_"+str(i+1)+", ")
binary = "bin " +binary + str("x_50;")
print(binary)

#This code writes the python print statements to a txt file
with open("knapsack.txt", "w") as text_file:
    print(obj_func, file=text_file)
    print(file=text_file)
    print(constraint_1, file=text_file)
    print(file=text_file)
    print(constraint_2, file=text_file)
    print(file=text_file)
    print(binary, file=text_file)
```

The result of the python code above is a .txt LPSolve input file. The form is as follows;

Base LPSolve input file [4]

```
max: + 66*x_1 + 37*x_2...+ 78*x_50 ;
+ 22*x_1 + 17*x_2...+ 19*x_50  <=200;
+ 8*x_1 + 11*x_2...+ 9*x_50  <=100;
bin x_1, x_2,...x_50;
```

The above LPSolve file has everything needed to solve the first question.
Questions b, c, and d will require new constraints.

Question b requires Sally's loot bag be filled with at least 50% prime indexed items (e.g. i = 2, 3, 5, 7...47).
This constraint requires that the total combination of metal bars that are labeled with a prime number be greater than or equal to the total value of the loot bag contents multiplied by .5.

Let P be the set of primes less than 50. Then;

$$prime\ constraint\ 50\% = .5 \sum_{i=1}^{50}(50 + 30\cos(i))x_i \ \leq \ \sum_{i \in P}(50 + 30\cos(i))x_i \qquad [5]$$

In order to facilitate this, a new constraint must be added to the Base LPSolve input file 4 in the form;

```
+ 37*x_2 + 20*x_3 +...+ 20*x_47 >= + 33*x_1 + 18.5*x_2...+ 39*x_50 ;
```

This new constraint multiplies the entire objective function by .5 and sets it as being less than or equal to the sum of all the prime labeled variables in the value function.
Below is the python code that adds the prime constraint line to the .txt file.

```
prime_50 = ""
for i in range(50):
    var_prime_50 = "+ "+str(.5*math.floor(50+30*np.cos(i+1)))+"*x_"+str(i+1)+" "
    prime_50 = prime_50 + var_prime_50
prime_50 = " >= "+ prime_50
```

The above code creates the right hand side of the necessary prime constraint.
To create the left hand side of the constraint, simply copy and paste the prime labeled variables and their coefficients from the objective function to the new prime constraint.
This is not a recommended tactic for larger data sets.

Similar methodology is used to answer question c, all that is needed is to change the coefficient in the inequality;

$$prime\ constraint\ 75\% = .75 \sum_{i=1}^{50}(50 + 30\cos(i))x_i \ \leq \ \sum_{i \in P}(50 + 30\cos(i))x_i \qquad [6]$$

The new constraint will need to be added to the Base LPSolve input file shown in figure 4.
The new line will be of the form;

```
+ 37*x_2 + 20*x_3 +...+ 20*x_47 >= + 49.5*x_1 + 27.75*x_2...+ 58.5*x_50 ;
```

The code to create the right hand side of the above inequality is below. To create the left hand side of the constraint, simply copy and paste the prime labeled variables and their coefficients from the objective function to the new prime constraint.
This is not a recommended tactic for larger data sets.

```
#intitialize string variable
prime_75 = ""
#iterate through all 50 items
for i in range(50):
    #multiply each variable in objective function by .75
```

```
    var_prime_75 = "+ "+str(.75*math.floor(50+30*np.cos(i+1))) + "*x_"+str(i+1)+'
    prime_75 = prime_75 + var_prime_75
#create final string output
prime_75 = prime_75 + ";"
```

The final question, d, asks us to consider what Sally should put in her knapsack if there was an unlimited number of each type of metal.
To consider this question only requires a simple change in the LPSolve input file.
In the final line of the LPSolve Base input file in figure 4, the variables are designated as binary.
In order to change the variables to allow more than one of each type to be taken, the word "bin" needs to be changed to "int".
This change allows an integer number of each variable to be chosen.
The form of the new line will look as follows;

```
int x_1, x_2,...,x_50;
```

6 LPSolve Output file and answers

We will now discuss the LPSolve outputs for the various problems and how they correspond to Sally's decision making in the vault. For each output file shown below, only the variables that had nonzero answers are shown. All variables with a value of 0 are suppressed.

6.1 Question a

The LPSolve output file gives the following answer to our Base LPSolve input file (figure 4);

```
x_1                             1
x_6                             1
x_7                             1
x_12                            1
x_18                            1
x_20                            1
x_26                            1
x_31                            1
x_37                            1
x_39                            1
x_45                            1
x_50                            1

Optimal solution                840 after
```

The output files shows that to maximize the value of available metal bars, Sally should fit metal bar 1, 6, 7, 12, 18, 20, 26, 31, 37, 39, 45, 50 into her knapsack. The total value of the bars will be $840 thousand dollars.

6.2 Question b

The LPSolve output file gives the following answer to our first prime constraint (50% prime labeled metal bars) input file;

```
 x_6                             1
x_7                              1
x_12                             1
x_13                             1
x_19                             1
x_25                             1
x_26                             1
x_31                             1
x_37                             1
x_43                             1
x_50                             1

Optimal solution                 822 after
```

The output file shows that to maximize the value of available metal bars, while maintaining that 50% of the bars have a prime label, Sally must put bars 6, 7, 12, 13, 19, 25, 26, 31, 37, 43, and 50 into her knapsack for a total of $822 thousand dollars.
We can see that the added constraint has lowered the maximum value that Sally can obtain.

6.3 Question c

The LPSolve output file gives the following answer to our second prime constraint (75% prime labeled metal bars) input file;

```
x_5                              1
x_6                              1
x_7                              1
x_12                             1
x_13                             1
x_17                             1
x_19                             1
x_23                             1
x_31                             1
x_37                             1
x_43                             1

Optimal solution                 729 after
```

The output file shows that to maximize the value of available metal bars, while maintaining that 75% of the bars have a prime label, Sally must put bars 5, 6, 7, 12, 13, 17, 19, 23, 31, 37, and 43 into her knapsack with a value total of $729 thousand dollars.
We can see that the added constraint has lowered the maximum value that Sally can obtain.

6.4 Question d

The final question removes the prime constraints and the binary constraint of the variables, and replaces them with an integer constraint.
The LPSolve output file gives the following result from the final input file;

```
x_6                             10
x_19                             1
x_50                             1

Optimal solution                  937 after
```

The output suggests Sally take 10 metal bars that are labeled 6, 1 bar of label 19, and 1 bar of label 50, resulting in a total value of $937 thousand dollars.
Removing the binary constraint on the variables translated to a much higher objective function maximization.

7 End comments

I believe that prime constraints are the more interesting ones in this problem.
Personally, the inequality used to describe the prime constraint was a novel solution and one I will take forward with me.
An interesting addition to this problem would be to add a time constraint to each variable.
In the scenario I used, perhaps each metal bar takes a different amount of time to load into the knapsack, and there is only so much time available before the police respond to the robbery.
I suspect this new constraint would not behave too differently than the ones introduced, but it would be a fun anecdotal addition.