Optimization

Tutorial session 3 - Integer Linear Programming

Lucas Tramonte
Lucas Fernandes Martins
Joao Pedro Regazzi

March 2024

Contents

Introduction

1 Question 1

The greedy approach consists in first ordering the item in descending order. Then, we initialize a bin and start adding the items in the same order as they appear in the ordered list. We add as many items as possible, once an item does not fit in a bin, we initialize another bin and place it. Therefore, in a certain iteration, we take object j and try to fit it in the n bins previously created. If it doesn't fit in any of the bins, we create an empty one and add it. Although not optimal, this greedy strategy gives as an upper bound of the number of bins necessary.

```
1
2  %Greedy heuristic
3
4  % Step 1: Sort items by volume in non-increasing order
5  [sortedVolumes, sortedIndices] = sort(V, 'descend');
6
7  % Initialize variables
8  Box = struct('NumberBox', {}, 'Items', {}, 'NumberItems', {}, '
       UnusedVolume', {});
9  boxCounter = 0;
10
11
12
13      % Step 2: Greedily pack items into boxes
14 for i = 1:N
15      itemVolume = sortedVolumes(i);
16      placed = false;
17
18      % Try to place the item in existing boxes
19      for j = 1:boxCounter
20          if Box(j).UnusedVolume >= itemVolume
21              % Item can fit into this box
22              Box(j).Items = [Box(j).Items sortedIndices(i)];
23              Box(j).NumberItems = Box(j).NumberItems + 1;
24              Box(j).UnusedVolume = Box(j).UnusedVolume - itemVolume;
25              placed = true;
26              break;
27          end
28      end
29
30      % If the item couldn't be placed in any existing box, create a new box
31      if ~placed
32          % Check if the item itself exceeds the box capacity
33          if itemVolume <= C
34              boxCounter = boxCounter + 1;
35              Box(boxCounter).NumberBox = boxCounter;
36              Box(boxCounter).Items = sortedIndices(i);
37              Box(boxCounter).NumberItems = 1;
38              Box(boxCounter).UnusedVolume = C - itemVolume;
39          else
40              % Item cannot be placed in any box, as it exceeds box capacity
41              disp(['Item ', num2str(sortedIndices(i)), ' exceeds box
       capacity.']);
42          end
43      end
44 end
45
46 disp(['Number of boxes : ', int2str(boxCounter)])
```

Listing 1: Code for question 1 - heuristic

The solution found was 18 boxes, which is consistent as it is the same value as adding up all the volumes provided and dividing by 2.7:

Boxes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Boxes used	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Number of boxes = 18

2 Question 2

Let us introduce the binary variables, $\forall n \in \{1, \dots N\}$ and $b \in \{1, \dots B\}$:

$$x_{n,b} = \begin{cases} 1 & \text{, If item n is in box b} \\ 0 & \text{, otherwise} \end{cases} \tag{1}$$

$$y_b = \begin{cases} 1 & \text{, If box b is used} \\ 0 & \text{, otherwise} \end{cases} \tag{2}$$

We want to minimize the total number of boxes used ($\mathcal{O}_1$):

$$\underset{y \in \{0,1\}^B}{\text{Minimize}} \sum_{b=1}^{B} y_b \tag{3}$$

Let v_n be the volume of item n. To the total volume of the items in each box does not exceed the capacity C:

$$\begin{aligned} \sum_{n=1}^{N} v_n x_{n,b} \leq C y_b \quad \forall b \in \{1, \dots, B\} \\ \sum_{n=1}^{N} v_n x_{n,b} - C y_b \leq 0 \quad \forall b \in \{1, \dots, B\} \end{aligned} \tag{4}$$

Additionally, all items must be packed:

$$\sum_{b=1}^{B} x_{n,b} = 1 \quad \forall n \in \{1, \dots, N\} \tag{5}$$

3 Question 3

To formulate the above problem as an binary linear programming problem, we must first transform the binary matrix x into a vector:

$$
\begin{gathered}
x_{nb} \Leftrightarrow \hat{x}_{b+(n-1)B} \\
\begin{bmatrix} x_{11} & \cdots & x_{1B} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NB} \end{bmatrix} \Leftrightarrow \begin{bmatrix} x_{11} & \cdots & x_{1B} & x_{21} & \cdots & x_{2B} & \cdots & x_{N1} & \cdots & x_{NB} \end{bmatrix}^T
\end{gathered} \tag{6}
$$

In this way, we perform a flattening of the original $x_{n,b}$ matrix.

In order to formulate the problem as an ILP problem, we must transform the matrix $x_{n,b}$ and the vector y_b into a single one column vector $z \in \{0,1\}^{B(N+1)}$:

$$
z = \begin{bmatrix} y_1 & \cdots & y_B & X_{11} & \cdots & X_{1B} & \cdots & X_{N1} & \cdots & X_{NB} \end{bmatrix}^T \tag{7}
$$

Taking $c \in \mathbb{R}^{B(N+1)}$ as the vector:

$$
c = \begin{bmatrix} \{\mathbf{1}^T\}^B & \{\mathbf{0}^T\}^B & \cdots & \{\mathbf{0}^T\}^B \end{bmatrix}^T \tag{8}
$$

Therefore, we have the following binary linear programming problem:

$$
\underset{z \in \{0,1\}^{B(N+1)\times 1}}{\text{Minimize}} \quad \langle c, z \rangle \tag{9}
$$

Subject to

$$
\begin{cases} Lz \leq d \\ L_{eq} z = d_{eq} \\ l_b \leq z \leq u_b \end{cases} \tag{10}
$$

with $L_{eq} \in \{0,1\}^{N \times B(N+1)}$:

$$
L_{eq} z = \begin{bmatrix} \{\mathbf{0}^T\}^B & \{\mathbf{1}^T\}^B & \{\mathbf{0}^T\}^B & \cdots & \{\mathbf{0}^T\}^B \\ \{\mathbf{0}^T\}^B & \{\mathbf{0}^T\}^B & \{\mathbf{1}^T\}^B & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \{\mathbf{0}^T\}^B & \{\mathbf{0}^T\}^B & \{\mathbf{0}^T\}^B & \cdots & \{\mathbf{1}^T\}^B \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_B \\ x_{11} \\ \vdots \\ x_{1B} \\ \vdots \\ x_{N1} \\ \vdots \\ x_{NB} \end{bmatrix} = \{\mathbf{1}\}^{B(N+1)} = d_{eq} \tag{11}
$$

For the inequality constraints, we can aggregate them as follows:

$$Lz = \begin{bmatrix} g_1^T & l_1^T & \cdots & l_1^T \\ g_2^T & l_2^T & \cdots & l_2^T \\ \vdots & \vdots & \ddots & \vdots \\ g_B^T & l_B^T & \cdots & l_B^T \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_B \\ x_{11} \\ \vdots \\ x_{1B} \\ \vdots \\ x_{N1} \\ \vdots \\ x_{NB} \end{bmatrix} \leq \{\mathbf{0}\}^{B(N+1)} = d \tag{12}$$

where, $\forall k \in \{1, \ldots, B\}$, $g_k = (g_k^{(i)})_{1 \leq i \leq B} \in \mathbb{R}^B N$ are vectors such that their elements i have the following relationship:

$$g_k^{(i)} = \begin{cases} -C & \text{, if i=k} \\ 0 & \text{, otherwise} \end{cases} \tag{13}$$

and $\forall k \in \{1, \ldots, B\}$, $l_k = (l_k^{(i)})_{1 \leq i \leq B} \in \mathbb{R}^B$ are vectors such that their elements i have the following relationship:

$$l_k^{(i)} = \begin{cases} v_i & \text{, if i=k} \\ 0 & \text{, otherwise} \end{cases} \tag{14}$$

Using intlinprog, we get a optimal solution of **18 bins**. So we've reached the theoretical lower bound: as seen previously, if all elements were considered to be 'liquid', we would need sum(V)/2.7 = 17.43 boxes, so, 18 is our integer lower bound.

Boxes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Boxes used	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Total Cost = 18

```
L_y = -C*eye(B, B);
Litem = zeros(B, N^2);
Lbin = zeros(N, N^2);

for i=1:B
    for j=1:N
        Litem(i, (j-1)*B+i) = V(j);
    end
end


for i=1:N
    Lbin(i, (i-1)*B+1: (i-1)*B+B) = ones(1,B);
end

%A = [L_y -Litem]

Aeq = [zeros(N, B) Lbin];
A = [L_y Litem]
beq = ones(N, 1)
b = zeros(B, 1)
f = [ones(B, 1) ; zeros(N^2, 1)]
S = N^2 + B
[x_min, f_min] = intlinprog(f, 1:S, A, b, Aeq, beq, zeros(1,S), ones(1,S))
```

Listing 2: Code for question 3

4 Question 4

So that the items numbered from 1 to 5 are within the same group (group G1), they must be packed in at most 2 boxes:

$$\begin{aligned} x_{1,b} + x_{1,b+1} &= 1 \\ x_{2,b} + x_{2,b+1} &= 1 \\ x_{3,b} + x_{3,b+1} &= 1 \\ x_{4,b} + x_{4,b+1} &= 1 \\ x_{5,b} + x_{5,b+1} &= 1 \end{aligned} \tag{15}$$

Why is it possible to consider without loss of optimality that these items are packed in boxes 1 and 2 ? Because the order in which the items are packet does not affect the overall quality of the packing. That is, permuting the items within the same group does not change the minimum number of bins required to pack them.

We are going to add 5 new constraints, of the form:

$$x_{n,1} + x_{n,2} = 1 \quad \forall n \in [1, 5] \tag{16}$$

Which means object i (i for 1 to 5) must be found on either box 1 or box 2.

Using linprog, we get once again a total of 18 bins necessary. That means this new constraint does not affect the previous optimal result. The minimal number of boxes as well as how items from group G1 that are distributed in boxes 1 and 2:

Boxes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Boxes used	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Number of boxes = 18

```
constraint_1 = zeros(1, S);

constraint_1(B+1) = 1;
constraint_1(B+2) = 1;


constraint_2 = zeros(1, S);

constraint_2(B+1) = 1;
constraint_2(B+2) = 1;

constraint_3 = zeros(1, S);

constraint_3(B+1) = 1;
constraint_3(B+2) = 1;
constraint_4 = zeros(1, S);

constraint_4(B+1) = 1;
constraint_4(B+2) = 1;
constraint_5 = zeros(1, S);

constraint_5(B+1) = 1;
constraint_5(B+2) = 1;
new_constraints = [constraint_1;constraint_2;constraint_3;constraint_4;
    constraint_5];
b_new = ones(5, 1)
Aeq_new = [Aeq; new_constraints];
beq_new = [beq;b_new];
[x_min_4, f_min_4] = intlinprog(f, 1:S, A, b, Aeq_new, beq_new, zeros(1,S)
    , ones(1,S))
```

Listing 3: Code for question 4

5 Question 5

The expression:

$$\sum_{1<=b<=B} bx_{n,b} \tag{17}$$

Gives us the number of the box the item is in. E.g., if item is in bin 1, all $x_{1,b}$ with $b\,!=1$ will be zero, so the expression will yield 1.

Why is it no longer possible to set the boxes where items from both groups will be packed ? We allow boxes to contain objects from the two groups. As we have two constraints now, we cannot for instance suppose for C3 that the objects are in bins 1 and 2 as those could be one of three boxes used by C4. Therefore, we must consider all individual cases (C3 uses two arbitrary bins and C4 uses three arbitrary bins).

Now items 6 to 11 cannot be in boxes 1 and 2. So that the items numbered from 6 to 11 are within group G2 and must be packed in at most 3 boxes:

$$\sum_{b=3}^{5} bx_{n,b} \leq 3 \quad \forall n \in [6, 11] \tag{18}$$

$$\begin{aligned} x_{6,3} + 2x_{6,4} + 3x_{6,5} &\leq 3 \\ x_{7,3} + 2x_{7,4} + 3x_{7,5} &\leq 3 \\ x_{8,3} + 2x_{8,4} + 3x_{8,5} &\leq 3 \\ x_{9,3} + 2x_{9,4} + 3x_{9,5} &\leq 3 \\ x_{10,3} + 2x_{10,4} + 3x_{10,5} &\leq 3 \\ x_{11,13} + 2x_{11,4} + 3x_{11,5} &\leq 3 \end{aligned} \tag{19}$$

We must also add the constraints:

$$\begin{aligned} x_{6,3} + x_{6,4} + x_{6,5} &= 1 \\ x_{7,3} + x_{7,4} + x_{7,5} &= 1 \\ x_{8,3} + x_{8,4} + x_{8,5} &= 1 \\ x_{9,3} + x_{9,4} + x_{9,5} &= 1 \\ x_{10,3} + x_{10,4} + x_{10,5} &= 1 \\ x_{11,13} + x_{11,4} + x_{11,5} &= 1 \end{aligned} \tag{20}$$

6 Question 6

To obtain a unique solution, we can, after having access to a given solution x, restrict the possibility of repeating the previous optimal solution by altering at least one decision variable:

$$\sum_{b=1}^{B} x_{n,b} \leq B - 1 \tag{21}$$

If the value of the function applied in this second solution is equal to the value of the function applied in the previous optimum, then we can conclude that we didn't have a unique solution. On the other hand, if the new solution has a higher value than the previous one, then our previous solution is optimal and unique.

We can run the following code multiple times, and test if the new solution x_min_unique has the boxes with the same items or not. If they have, we should add a new constraint with the new solution and run again. If the new solution has at least one new bin with a different set of items, then the search is finished. We tested it for the V data and found out that the solution is not unique.

```
new_constraint = zeros(1, N*N+B);
b_new_constraint = N-1;

for i=1:N*N+B
    new_constraint(i) = x_min(i);
end

A_unique = [new_constraint];
b_unique = [b_new_constraint];

[x_min_unique, f_min_unique] = intlinprog(f, 1:S, [A; A_unique], [b;
   b_unique], Aeq, beq, zeros(1, S), ones(1, S));
```

Listing 4: Code for question 6

7 Question 7

To select the least filled box, we do :

$$min \sum_{b=1}^{B} v_n x_{n,b} \tag{22}$$

So the new cost function becomes:

$$\underset{y \in \{0,1\}^B}{\text{Minimize}} \sum_{b=1}^{B} y_b + min \sum_{b=1}^{B} v_n x_{n,b} \tag{23}$$

The constraints remain the same as in Question 5.