Practical Large Scale Optimization with illustrations in Matlab

S. Elhedhli

Last updated: Jan. 2025

Contents

Chapter 1

Mixed Integer Programming Formulations and The Bin Packing Problem

1.1 The Bin-Packing Problem

Given a set of items, $l = 1, ..., L$ of sizes $D_1, ...D_L$ and a set of available bins $k = 1, ..., K$ of identical size V, the bin-packing problem seeks to find the minimum number of bins to pack the L items. Using the binary variables:

- z_k that take the value 1 when bin k is used, and 0 otherwise
- y_{kl} that take value 1, when item l is assigned to bin k, and 0 otherwise.

The bin-packing problem is formulated as:

$$[BP_d]: \quad \min \quad \sum_{k=1}^{K} z_k \tag{1.1}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} y_{kl} = 1 \qquad l = 1, ..., L \tag{1.2}$$

$$\sum_{l=1}^{L} D_l y_{kl} \leq V \qquad k = 1, ..., K \tag{1.3}$$

$$y_{kl} \leq z_k \qquad k = 1, ..., K, l = 1, ..., L \tag{1.4}$$

$$y_{kl}, z_k \in \{0, 1\} \qquad k = 1, ..., K, l = 1, ..., L. \tag{1.5}$$

The objective (1.1) minimizes the total number of bins used. Constraints (1.2)are assignment constraints that assign each item to exactly one bin. Constraints (1.3) are capacity constraints that prevent the bin capacity from being exceeded. Constraints (1.4) make sure that items are assigned to open bins only. This formulation has $L + K + KL$ constraints and $K + KL$ variables.

Constraints (1.3) and (1.4) can be aggregated, leading to:

$$
\begin{aligned}
[BP_a]: \quad \min \quad & \sum_{k=1}^{K} z_k & \\
\text{s.t.} \quad & \sum_{k=1}^{K} y_{kl} = 1 & l = 1, ..., L \\
& \sum_{l=1}^{L} D_l y_{kl} \leq V z_k & k = 1, ..., K \\
& y_{kl}, z_k \in \{0, 1\} & k = 1, ..., K, l = 1, ..., L
\end{aligned}
$$

Formulation $[BP_a]$, which we call the **aggregated formulation**, has $L+K$ constraints and $K+KL$ variables. $[BP_d]$ is called the **disaggregated formulation**.

1.2 Example

Consider the following items of sizes 20, 50, 50, 50 and 70 that have to be packed in bins of sizes 100. The optimal solution will obviously require 3 bins. We always assume that there is an infinite supply of bins, but obviously, we will not need more bins than the number of items to pack. So let us assume that we have 5 bins available.
The disaggregated formulation for this example is:

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{5} z_k = z_1 + z_2 + z_3 + z_4 + z_5 \\
\text{s.t.} \quad & y_{11} + y_{21} + y_{31} + y_{41} + y_{51} = 1; \\
& y_{12} + y_{22} + y_{23} + y_{24} + y_{25} = 1; \\
& y_{13} + y_{23} + y_{33} + y_{43} + y_{53} = 1; \\
& y_{14} + y_{24} + y_{34} + y_{44} + y_{54} = 1; \\
& y_{15} + y_{25} + y_{35} + y_{45} + y_{55} = 1; \\
& 20y_{11} + 50y_{12} + 50y_{13} + 50y_{14} + 70y_{15} \leq 100; \\
& 20y_{21} + 50y_{22} + 50y_{23} + 50y_{24} + 70y_{25} \leq 100; \\
& 20y_{31} + 50y_{32} + 50y_{33} + 50y_{34} + 70y_{35} \leq 100; \\
& 20y_{41} + 50y_{42} + 50y_{43} + 50y_{44} + 70y_{45} \leq 100; \\
& 20y_{51} + 50y_{52} + 50y_{53} + 50y_{54} + 70y_{55} \leq 100; \\
& y_{11} \leq z_1; y_{12} \leq z_1; y_{13} \leq z_1; y_{14} \leq z_1; y_{15} \leq z_1; \\
& y_{21} \leq z_2; y_{22} \leq z_2; y_{23} \leq z_2; y_{24} \leq z_2; y_{25} \leq z_2; \\
& y_{31} \leq z_3; y_{32} \leq z_3; y_{33} \leq z_3; y_{34} \leq z_3; y_{35} \leq z_3; \\
& y_{41} \leq z_4; y_{42} \leq z_4; y_{43} \leq z_4; y_{44} \leq z_4; y_{45} \leq z_4; \\
& y_{51} \leq z_5; y_{52} \leq z_5; y_{53} \leq z_5; y_{54} \leq z_5; y_{55} \leq z_5; \\
& y_{11}, y_{12}, ..., y_{55}, z_1, z_2, z_3, z_4, z_5 \in \{0, 1\}
\end{aligned}
$$

The aggregated formulation is:

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{5} z_k = z_1 + z_2 + z_3 + z_4 + z_5 \\
\text{s.t.} \quad & y_{11} + y_{21} + y_{31} + y_{41} + y_{51} = 1; \\
& y_{12} + y_{22} + y_{23} + y_{24} + y_{25} = 1; \\
& y_{13} + y_{23} + y_{33} + y_{43} + y_{53} = 1; \\
& y_{14} + y_{24} + y_{34} + y_{44} + y_{54} = 1; \\
& y_{15} + y_{25} + y_{35} + y_{45} + y_{55} = 1; \\
& 20y_{11} + 50y_{12} + 50y_{13} + 50y_{14} + 70y_{15} - 100z_1 \leq 0; \\
& 20y_{21} + 50y_{22} + 50y_{23} + 50y_{24} + 70y_{25} - 100z_2 \leq 0; \\
& 20y_{31} + 50y_{32} + 50y_{33} + 50y_{34} + 70y_{35} - 100z_3 \leq 0; \\
& 20y_{41} + 50y_{42} + 50y_{43} + 50y_{44} + 70y_{45} - 100z_4 \leq 0; \\
& 20y_{51} + 50y_{52} + 50y_{53} + 50y_{54} + 70y_{55} - 100z_5 \leq 0; \\
& y_{11}, y_{12}, ..., y_{55}, z_1, z_2, z_3, z_4, z_5 \in \{0, 1\}
\end{aligned}
$$

The coefficient matrix corresponding to the disaggregated formulation shows a nice structure as illustrated in Figure 1.1.

1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
20	50	50	50	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-100	0	0	0	0
0	0	0	0	0	20	50	50	50	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-100	0	0	0
0	0	0	0	0	0	0	0	0	0	20	50	50	50	70	0	0	0	0	0	0	0	0	0	0	0	0	-100	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	50	50	50	70	0	0	0	0	0	0	0	0	-100	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	50	50	50	70	0	0	0	0	-100

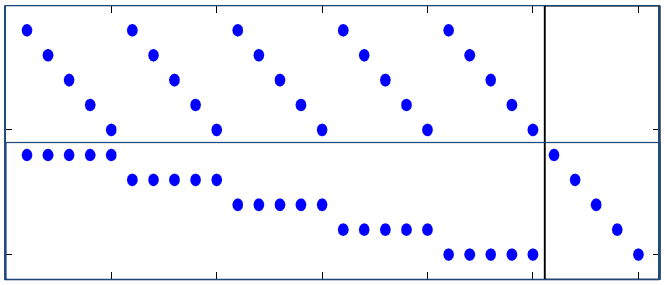

Figure 1.1: The coefficient matrix and its structure for $[BP_d]$

1.2.1 Strength of Formulations

The question that comes to mind is which formulation is better? This obviously depends on the criteria used. A natural criteria is the computational time it takes each to solve. For this, let us recall that MIP are typically solved using branch-and-bound where at each node of the branch-and-bound tree, a linear program is solved. Let us also recall that the tighter the feasible regions of the linear programs, the less nodes are expected to be explored. So, MIPs whose Linear programming relaxations has a tighter feasible region solve faster. Hence, a good criteria is the tightness of the LP relaxation of the formulation. See Figure 1.2 for an illustration. We note that:

- The tightest feasible region is given by the convex hull of integer points(case 4 in Figure 1.2). The formulation that corresponds to the convex hull will solve the MIP at the root node as

every extreme point of the convex hull is an integer solution.

- If the LP feasible region of one formulation is contained in an other, then the formulation corresponding to the tighter feasible region is better. See case 2 in Figure 1.2.

To find if one formulation's feasible region is contained in another's, one will have to prove that every point from the LP relaxation of the tighter formulation is contained in the loose formulation, and there is at least one point feasible to the loose formulation, but not the tighter one. So the question is: which one of $[BP_a]$ and $[BP_d]$ is tighter. The answer is none. The feasible regions of the LP relaxations of the two formulations intersect, but none is contained in the other (see case 3 in Figure 1.2). To prove this, it suffices to find a point that is feasible to the LP relaxation of $[BP_a]$, denoted $[\overline{BP}_a]$:

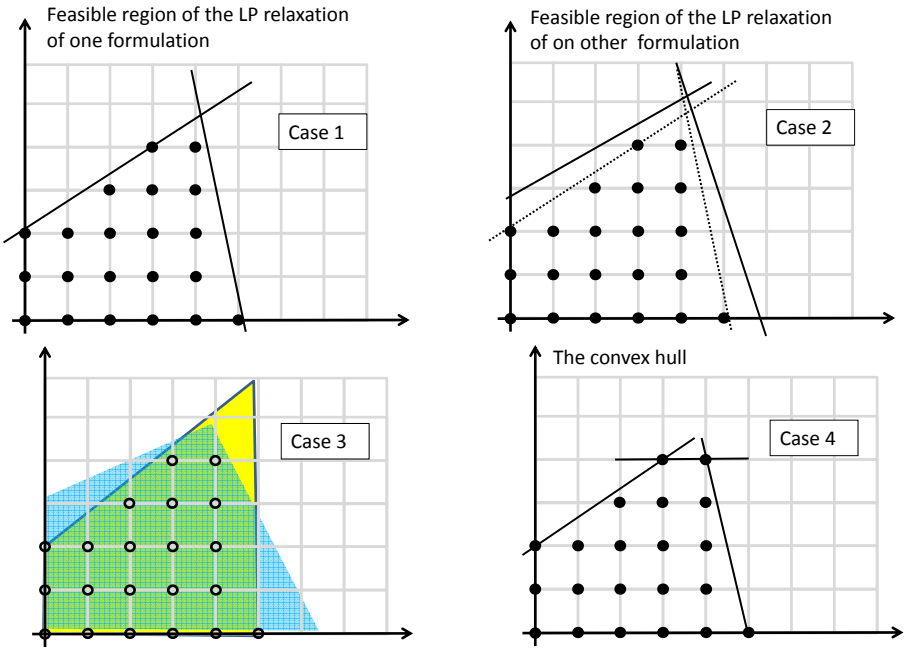

Figure 1.2: The Illustration of the LP relaxations and the convex hull.

$$
\begin{aligned}
[\overline{BP}_a]: \quad \min \quad & \sum_{k=1}^{K} z_k & & \\
\text{s.t.} \quad & \sum_{k=1}^{K} y_{kl} = 1 & & l = 1, ..., L \\
& \sum_{l=1}^{L} D_l y_{kl} \leq V z_k & & k = 1, ..., K \\
& 0 \leq y_{kl}, z_k \leq 1 & & k = 1, ..., K, l = 1, ..., L
\end{aligned}
$$

but not to the LP relaxation of $[BP_d]$, denoted $[\overline{BP}_d]$:

$$
\begin{aligned}
[\overline{BP}_d]: \quad \min \quad & \sum_{k=1}^{K} z_k & \\
\text{s.t.} \quad & \sum_{k=1}^{K} y_{kl} = 1 & l = 1, ..., L \\
& \sum_{l=1}^{L} D_l y_{kl} \leq V & k = 1, ..., K \\
& y_{kl} \leq z_k & k = 1, ..., K, l = 1, ..., L \\
& 0 \leq y_{kl}, z_k \leq 1 & k = 1, ..., K, l = 1, ..., L.
\end{aligned}
$$

For that let us consider the following two solutions displayed in Figures 1.3 and 1.4.

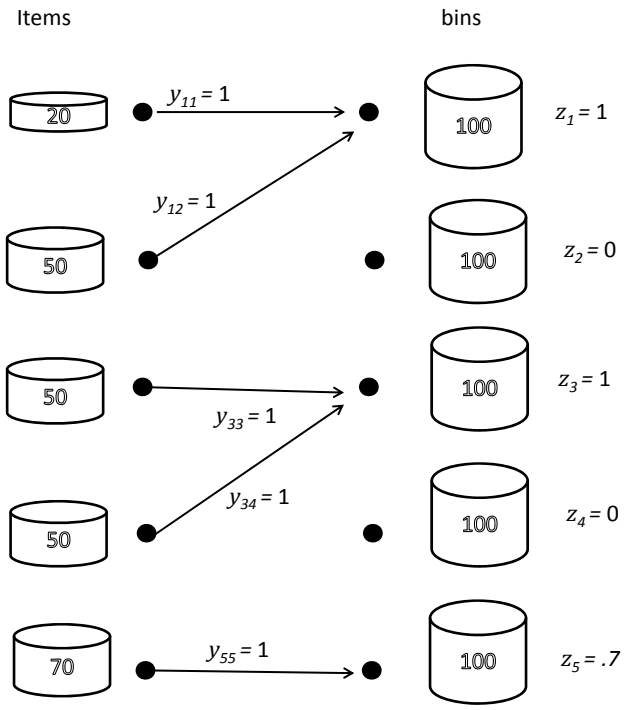

Figure 1.3: A solution that is feasible to $[\overline{BP}_a]$ but not $[\overline{BP}_d]$

The first is feasible to $[\overline{BP}_a]$, but is infeasible to $[\overline{BP}_d]$ as $1 = y_{55} > z_5 = .7$. The second is feasible to $[\overline{BP}_d]$, but is infeasible to $[\overline{BP}_a]$ as $\sum_{l=1}^{L} D_l y_{4l} = 50y_{44} + 70y_{45} = 50 \times .7 + 70 \times .7 = 84 > Vz_4 = .7$. Therefore, none of the two feasible regions is contained in the other; and so none of the two formulations is better than the other. The intersection of the feasible regions of the two formulations, however, will lead to a formulation that dominates both of the previous ones. This lead to

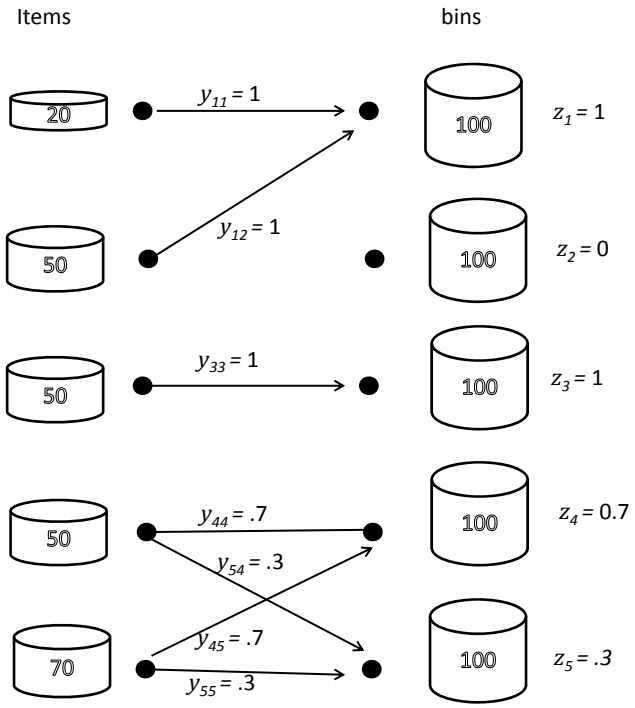

Figure 1.4: A solution that is feasible to $[\overline{BP}_d]$ but not $[\overline{BP}_a]$

$$
\begin{aligned}
[BP_{ad}]: \quad \min \quad & \sum_{k=1}^{K} z_k & & \\
\text{s.t.} \quad & \sum_{k=1}^{K} y_{kl} = 1 & & l = 1, ..., L \\
& \sum_{l=1}^{L} D_l y_{kl} \leq V z_k & & k = 1, ..., K \\
& y_{kl} \leq z_k & & k = 1, ..., K, l = 1, ..., L \\
& y_{kl}, z_k \in \{0, 1\} & & k = 1, ..., K, l = 1, ..., L.
\end{aligned}
$$

Note that the constraints $y_{kl} \leq z_k, k = 1, ..., K$ is redundant for $[BP_{ad}]$ but not $[\overline{BP}_{ad}]$.

1.3 The Bin Packing Problem with Identical Items

In the above example, all items were treated as being different. If some items are identical, then this information could be used to improve the formulation. Let us suppose that there are c_l copies of item l. The disaggregated formulation for the bin packing problem with identical items is:

$$\begin{aligned}
\min \quad & \sum_{k=1}^{K} z_k & \\
\text{s.t.} \quad & \sum_{k=1}^{K} y_{kl} = c_l & l = 1, ..., L \\
& \sum_{l=1}^{L} D_l y_{kl} \leq V & k = 1, ..., K \\
& y_{kl} \leq c_l z_k & k = 1, ..., K, l = 1, ..., L \\
& y_{kl} \geq 0, \text{ integer}, k = 1, ..., K, l = 1, ..., L; z_k \in \{0, 1\} & k = 1, ..., K.
\end{aligned} \tag{1.6}$$

where y_{kl} is now defined as the number of copies of item l that are assigned to bin k. The aggregated formulation is:

$$\begin{aligned}
\min \quad & \sum_{k=1}^{K} z_k & \\
\text{s.t.} \quad & \sum_{k=1}^{K} y_{kl} = c_l & l = 1, ..., L \\
& \sum_{l=1}^{L} D_l y_{kl} \leq V z_k & k = 1, ..., K \\
& 0 \leq y_{kl} \leq c_l \text{ and integer}, k = 1, ..., K, l = 1, ..., L; z_k \in \{0, 1\} & k = 1, ..., K.
\end{aligned} \tag{1.7}$$

and the stronger formulation is:

$$\begin{aligned}
\min \quad & \sum_{k=1}^{K} z_k & \\
\text{s.t.} \quad & \sum_{k=1}^{K} y_{kl} = c_l & l = 1, ..., L \\
& \sum_{l=1}^{L} D_l y_{kl} \leq V z_k & k = 1, ..., K \\
& y_{kl} \leq c_l z_k & k = 1, ..., K, l = 1, ..., L \\
& y_{kl} \geq 0, \text{ integer}, k = 1, ..., K, l = 1, ..., L; z_k \in \{0, 1\} & k = 1, ..., K.
\end{aligned}$$

1.4 A Set Partitioning/Covering Formulation

Before tackling the formulation,one can think of the different possible ways of packing items to bins. Each pattern would represent a feasible allocation that does not exceed the capacity of the bin. An example of *p*acking/pattern is putting the first item of size 20 with the last item of size 70. The packing could be represented as a vector of size L with binary entries to indicate if the item is include. For example, the previous packing would be represented as

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Note that the order corresponds to the order of items given, i.e.

$$\begin{bmatrix} 20 \\ 50 \\ 50 \\ 50 \\ 70 \end{bmatrix}.$$

Let $\mathbf{a^h} = (a_1^h, a_2^h, ..., a_l^h, ..., a_L^h)$ be defined as a possible *packing* for a bin. For the example above there are 12 feasible patterns/packings:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Also let

$$\alpha_h = \begin{cases} 1 & \text{if packing } h \text{ is used;} \\ 0 & \text{o.w.} \end{cases}$$

Then the set partitioning formulation is:

$$\begin{aligned} [BP_{sc}]: \quad \min \quad & \sum_{h=1}^{H} \alpha_h & & \\ \text{s.t.} \quad & \sum_{h=1}^{H} a_l^h \alpha_h = 1 & & l = 1, ..., L \\ & \alpha_h \in \{0,1\} & & h = 1, ..., H \end{aligned}$$

Note that an equivalent formulation is the *s*et covering formulation where the constraints are replaced by $\sum_{h=1}^{H} a_l^h \alpha_h \geq 1, \quad l = 1, ..., L$. [1]

The set of possible packings is typically large. For L items the number is bounded by $2^L - 1$. Generating them all at once, could be very time consuming. Apart from a brute force approach where all possible packings of size L are generated an then filtered to only keep a^h that satisfy $\sum_{l=1}^{L} D_l a_l^h \leq V$, one could generate some packings by solving a *Binary Knapsack Problem* of the

[1] There three variants of the The set covering, set packing, and set partitioning formulations are binary integer programs with the right-hand-side being a vector of ones $\mathbf{1}$, of the form

set covering : min	$c^T x$	set packing : min	$c^T x$	set partitioning : min	$c^T x$
s.t.	$Ax \geq \mathbf{1}$	s.t.	$Ax \leq \mathbf{1}$	s.t.	$Ax = \mathbf{1}$
	$x \in \{0,1\}$		$x \in \{0,1\}$		$x \in \{0,1\}$

form:

$$\begin{aligned} \max \quad & \sum_{l=1}^{L} w_l y_l \\ \text{s.t.} \quad & \sum_{l=1}^{L} D_l y_l \leq V \\ & y_l \in \{0,1\} \end{aligned}$$

where w_l, $l = 1, ..., L$ are weights that could be changed randomly to generate different packings.

1.5 A Formulation with a large number of constraints (based on cover inequalities)

Let us recall the example given above, and note that we can write a constraint for every set of items that do not fit into a bin. To elaborate on this, consider the first, second, and fifth items of sizes 20, 50 and 70 respectively. They obviously do not all fit in one bin, and that at most two could fit. Based on this, we can write

$$y_{k1} + y_{k2} + y_{k5} \leq 2.$$

This constraint is called a **cover inequality**. In general, given a set of C items that can not fit in a bin of capacity V i.e. $\sum_{l \in C} D_l > V$, one can write a constraint to eliminate this possibility as a cover inequality of the form

$$\sum_{l \in C} y_l \leq |C| - 1 \qquad y_l \in \{0,1\}.$$

A **minimal cover inequality** is obtained when removing any item from the set C makes the remaining items fit into a bin. Using this fact, the capacity constraints

$$\sum_{l=1}^{L} D_l y_{kl} \leq V$$

can be replaced by a large set of cover inequalities of the form

$$\sum_{l \in C} y_{kl} \leq (|C| - 1) \qquad \text{for } C \subseteq \{1, 2, .., L\}, \text{where} \sum_{l \in C} D_l > V$$

leading to

$$\begin{aligned} [BP_{ci}]: \quad \min \quad & \sum_{k=1}^{K} z_k & & \\ \text{s.t.} \quad & \sum_{k=1}^{K} y_{kl} = 1 & & l = 1, ..., L \\ & \sum_{l \in C} y_{kl} \leq (|C| - 1) & & C \subseteq \{1, 2, .., L\}, \text{such that} \sum_{l \in C} D_l > V \\ & y_{kl} \leq z_k & & k = 1, ..., K, l = 1, ..., L. \\ & y_{kl}, z_k \in \{0,1\} & & k = 1, ..., K, l = 1, ..., L. \end{aligned}$$

This formulation has a very large number of constraints, as opposed to the set partitioning formulation that has a large number of variables. The generation of all cover inequalities is time consuming. As with the set partitioning formulation, the cover inequalities should be generated as needed. One way to take advantage of the formulation is to use a constraint generation method where a relaxed version of the problem with only a few set of constraints is solved. If the resulting solution turns out to be feasible to the bin packing problem, i.e. no capacity constraints are violate, then it is optimal. If not, then a cover inequality is introduced for every bin capacity that is violated. For the above example, we start by solving

$$\begin{aligned}
\min \quad & \sum_{k=1}^{5} z_k = z_1 + z_2 + z_3 + z_4 + z_5 \\
\text{s.t.} \quad & y_{11} + y_{21} + y_{31} + y_{41} + y_{51} = 1; \\
& y_{12} + y_{22} + y_{23} + y_{24} + y_{25} = 1; \\
& y_{13} + y_{23} + y_{33} + y_{43} + y_{53} = 1; \\
& y_{14} + y_{24} + y_{34} + y_{44} + y_{54} = 1; \\
& y_{15} + y_{25} + y_{35} + y_{45} + y_{55} = 1; \\
& y_{11} \leq z_1; y_{12} \leq z_1; y_{13} \leq z_1; y_{14} \leq z_1; y_{15} \leq z_1; \\
& y_{21} \leq z_2; y_{22} \leq z_2; y_{23} \leq z_2; y_{24} \leq z_2; y_{25} \leq z_2; \\
& y_{31} \leq z_3; y_{32} \leq z_3; y_{33} \leq z_3; y_{34} \leq z_3; y_{35} \leq z_3; \\
& y_{41} \leq z_4; y_{42} \leq z_4; y_{43} \leq z_4; y_{44} \leq z_4; y_{45} \leq z_4; \\
& y_{51} \leq z_5; y_{52} \leq z_5; y_{53} \leq z_5; y_{54} \leq z_5; y_{55} \leq z_5; \\
& y_{11}, y_{12}, ..., y_{55}, z_1, z_2, z_3, z_4, z_5 \in \{0, 1\}
\end{aligned}$$

This leads to the solution: $y_{11} = y_{12} = y_{13} = y_{14} = y_{15} = 1$ and the rest being zero, meaning that all items are assigned to bin 1. Based on this, we ca generate the cover inequality $y_{11} + y_{12} + y_{13} + y_{14} + y_{15} \leq 4$. But given that bins are identical, this applies to all other bins. Hence we add 5 constraints $y_{k1} + y_{k2} + y_{k3} + y_{k4} + y_{k5} \leq 4$, $k = 1, ..., 5$. We can even account for the fact that items could not be assigned to open bins by making the set of added cuts

$$y_{k1} + y_{k2} + y_{k3} + y_{k4} + y_{k5} \leq 4z_k, \quad k = 1, ..., 5.$$

Note that

- one further improvement is to realize that at most 3 of the 5 items could fit one bin and hence change the rhs from $4z_k$ to $3z_k$.
- the above cover inequalities are not minimal, to make them minimal we have to remove items one by one starting from the largest and see if the bin capacity is violated.

Adding the above inequalities and resolving, leads to the solution $y_{11} = y_{12} = y_{14} = y_{15} = 1; y_{23} = 1, z_1 = z_2 = 1$ and the rest being zeros. Obviously bin 1 can not hold items 1, 2, 4 and 5. So, we add these set of inequalities:

$$y_{k1} + y_{k2} + y_{k4} + y_{k5} \leq 3z_k, \quad k = 1, ..., 5.$$

The Matlab code to execute this is given below

```
clc;
clear all;

Aeq(1:5,:)=[eye(5),eye(5),eye(5),eye(5),eye(5),zeros(5,5)];
```

```
c=[zeros(25,1);ones(5,1)];
beq=[ones(5,1)];
D=[20,50,50,50,70];
bineq=zeros(5,1);
Aineq=[];bineq=[];
lb=zeros(30,1); ub=ones(30,1);
optimal=0;
Aeq
pause
while optimal==0

x=intlinprog(c,[1:30],Aineq,bineq,Aeq,beq,lb,ub)


fprintf(' y = (rows--> items; columns --> bins) \n')
y=reshape(x(1:25),5,5)
z=reshape(x(26:30),5,1)'


 violated=find(D*y>100);
 if ~isempty(violated)
   for i=violated
      cut=y(:,i)'
     rhs=sum(cut)-1;   %this generates cover cuts;, but not minumum cover cuts
     Acut=[];
     for i=1:5
      Acut=[[Acut;zeros(1,size(Acut,2))],[zeros(size(Acut,1),5);cut]]
     end
     Aineq=[Aineq;[Acut,-rhs*eye(5)]];
     bineq=[bineq;zeros(5,1)];
 end
 else
   optimal=1;
 end
 pause
end
```