Packing problems

Logistics

Giovanni Righini

Università degli Studi di Milano

Packing

In general, packing problems arise when items must be put into bins:

- goods in boxes
- boxes on pallets
- pallets in containers
- containers in trains, barges, ships, airplanes...

Objective:

- minimize the number or the cost of the used bins;
- maximize the number or the value of the packed items.

Items and bins

Items can be characterized by

- size (1, 2, 3 dimensions)
- weight
- value
- compatibilty/incompatibility with other items or with bins
- et cetera...

Bins can be characterized by

- size (1, 2, 3 dimensions)
- weight capacity
- cost for usage
- number of available bins
- et cetera...

In on-line packing problems, items must be loaded as they arrive according to a given sequence, with no knowledge about the next items to come.

The knapsack problem

Given a set N of items with a value c_i and a weight a_i and a knapsack with capacity b, select a maximum value subset of items that fits into the knapsack.

A binary variable x_i indicates whether item i is selected or not $\forall i \in N$.

$$\begin{aligned}
\text{maximize } z = \sum_{i \in N} c_i x_i & \\
\text{subject to } \sum_{i \in N} a_i x_i \leq b & \\
x_i \in \{0, 1\} & \qquad \forall i \in N
\end{aligned}$$

Variations

Many possible variations may occur:

- multiple capacities (weight, volume, value...),
- multiple knapsacks,
- several copies of identical items for each type $i \in N$ (*Integer KP*),
- incompatible items,
- selection of one item from each given group of items (*Multiple Choice KP*)...

Bin packing

Pack a set N of items of given weight a_i $\forall i \in N$ in a minimum number of identical bins of capacity b.

A binary variable x_{ij} indicates whether item i is assigned to bin j.
A binary variable y_j indicates whether bin j is used or not.

$$
\begin{aligned}
\text{minimize } z = & \sum_{j \in M} y_j & \\
\text{subject to } & \sum_{j \in M} x_{ij} = 1 & \forall i \in N \\
& \sum_{i \in N} a_i x_{ij} \leq b & \forall j \in N \\
& x_{ij} \leq y_j & \forall i \in N, \forall j \in M \\
& x_{ij} \in \{0, 1\} & \forall i \in N, \forall j \in M \\
& y_j \in \{0, 1\} & \forall j \in M
\end{aligned}
$$

Bin packing: an alternative formulation

Constraints $x_{ij} \leq y_j$ can be replaced by a modification to capacity constraints:

$$\sum_{i \in N} a_i x_{ij} \leq b y_j ,$$

yielding a more compact model but a weaker linear relaxation.

$$\begin{aligned}
\text{minimize } z = & \sum_{j \in M} y_j & \\
\text{subject to } & \sum_{j \in M} x_{ij} = 1 & \forall i \in N \\
& \sum_{i \in N} a_i x_{ij} \leq b y_j & \forall j \in N \\
& x_{ij} \in \{0, 1\} & \forall i \in N, \forall j \in M \\
& y_j \in \{0, 1\} & \forall j \in M
\end{aligned}$$

Variations

Many possible variations may occur:

- multiple capacities (weight, volume, value...),
- heterogeneous bins,
- costs associated with bins,
- incompatibility constraints,
- level packing/strip packing,
- bi-dimensional, tri-dimensional packing...

The same models describe cutting problems.

Complexity

From a computational complexity point of view, packing problems are in general difficult to solve (*NP*-hard), because they translate into integer/binary linear programming models.

The most commonly used techniques are

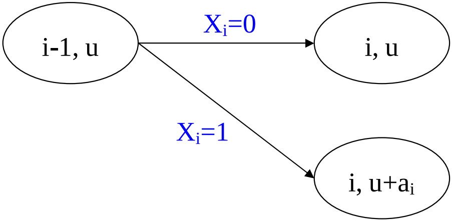
- dynamic programming (for single bin problems),
- branch-and-price (for multi-bin problems),
- heuristic algorithms (for 2D and 3D problems).

Dynamic programming for the KP

- **Policy**: sort the items in N (the variables) from x_1 to x_n.
- **State**:
  - **Feasibility** depends on the residual capacity;
  - **Cost** does not depend on previous decisions.

  Hence the state is given by the last item considered (i) and the capacity used so far (u).
- **R.E.F.**:
  - **Initialization**: $z(0,0) = 0$;
  - **Extension**:
    $z(i,u) = \max\{z(i-1,u), z(i-1,u-a_i)+c_i\} \;\; \forall i \in N, \forall u = 1,\dots,b.$

Dynamic programming for the KP



The state graph has a layer for each item (variable) $j \in N$ and $b+1$ nodes per layer.

Complexity: The graph has $O(nb)$ nodes and each of them has only two predecessors. Then the D.P. algorithm has complexity $O(nb)$, which is pseudo-polynomial.

Heuristics for on-line packing problems

First Fit: Put each object in the first bin that can accommodate it. If no such bin exists, then initialize a new bin.

Best Fit: Put each object in the bin with minimum residual capacity among those than that can accommodate it. If no such bin exists, then initialize a new bin.

Both of them have a constant approximation factor equal to 1.7 (2013,2014).

If the objects are sorted by decreasing weight, then both *First Fit Decreasing* and *Best Fit Decreasing* have constant approximation factor equal to $1.\overline{2}$ (2007).

2D and 3D packing problems

Both items and bins are usually represented by rectangles (2D) or parallelepiped (3D).

- with/without rotation
- with/without overlap
- bin packing/strip packing
- with/without levels
- with balance constraints (ships, artificial satellites...)
- loading/unloading constraints (LIFO, guillotine cutting,...)

Heuristics for 2D packing problems

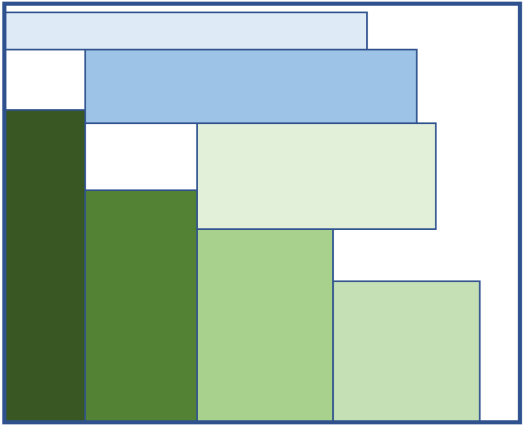
In some heuristics item are accommodates in layers within each bin.

Finite First Fit: sort the items by non-increasing height. For each item, insert it in the leftmost position of the first layer of the first bin that can accommodate it. If no suitable position exists, then initialize a new layer in the first bin that can accommodate it. If no suitable bin exists, then initialize the first layer of a new bin.

Finite Best Fit: same as FFF but selecting the layer and the bin with minimum residual capacity.

Heuristics for 2D packing problems

Bottom-Left: same as before but without layers. Insert each bin in the bottom-most and left-most position in the first bin where this is possible.



Heuristics for 3D packing problems

3D packing problems can be solved in a heuristic way by transforming them in into a set of bi-dimensional packing problems, disregarding the height of items and bins, followed by a 1D packing problem, where 2D sets of items must be stacked one above the other to form 3D arrangements.