Packing problems in one and more dimensions

Winter School on Network Optimization 2018

Silvano Martello

DEI "Guglielmo Marconi", Alma Mater Studiorum Università di Bologna, Italy

Contents

I. Bin Packing and Cutting Stock Problems

The Bin Packing Problem

One of the most famous problems in combinatorial optimization.

Attacked with all main theoretical and practical tools.

Packing problems have been studied since the Thirties (Kantorovich).

In 1961 Gilmore and Gomory introduced, for these problems, the concept of column generation.

The worst-case performance of approximation algorithms investigated since the early Seventies.

Lower bounds and effective exact algorithms developed starting from the Eighties.

[Many heuristic and metaheuristic approaches.]

This talk will also introduce many basic general techniques for Combinatorial Optimization.

The field is still very active:

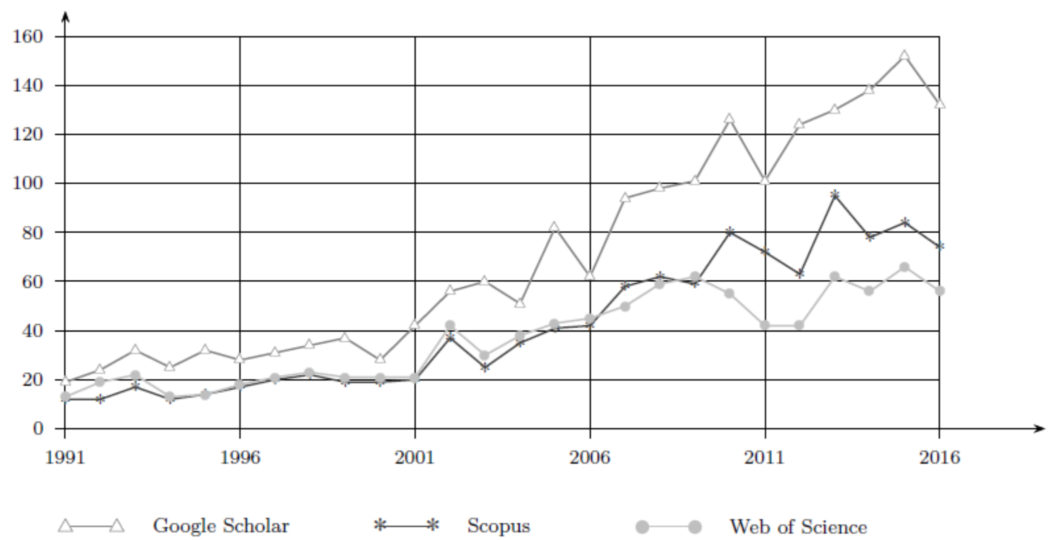


Figure 1.1: Number of papers dealing with bin packing and cutting stock problems, 1991-2016

Contents

Definitions

1) Given n *items*, each having an integer *weight* (or *size*) w_j $(j = 1, \ldots, n)$, and an unlimited number of identical *bins* of integer *capacity* c,

Bin Packing Problem (BPP): pack all the items into the minimum number of bins so that the total weight packed in any bin does not exceed the capacity.

We assume, with no loss of generality, that $0 < w_j < c$ for all j.

Main application (**generalization**):

2) Given m *item types*, each having an integer *weight* w_j and an integer *demand* d_j $(j = 1, \ldots, m)$, and

an unlimited number of identical *bins* of integer *capacity* c,

Cutting Stock Problem (CSP): produce (at least) d_j copies of each item type j using the minimum number of bins so that the total weight in any bin does not exceed the capacity.

Frequently interpreted as the process of *cutting pieces* (items) *from rolls of material* (bins).

Real world applications in

packing trucks with a given weight limit,

assigning commercials to station breaks

allocating memory in computers,

subproblems in more complex optimization problems ...

Polynomial models (textbooks)

- Let u be any upper bound on the minimum number of bins needed (e.g., approximate solution), assume that the potential bins are numbered as $1, \dots, u$.

$$y_i = \begin{cases} 1 & \text{if bin } i \text{ is used in the solution;} \\ 0 & \text{otherwise} \end{cases} \qquad (i = 1, \dots, u),$$

$$x_{ij} = \begin{cases} 1 & \text{if item } j \text{ is packed into bin } i; \\ 0 & \text{otherwise} \end{cases} \qquad (i = 1, \dots, u; j = 1, \dots, n),$$

- *Integer Linear Program* (ILP) for the **BPP** (Martello and Toth, 1990)

$$\min \quad \sum_{i=1}^{u} y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_j x_{ij} \le c y_i \quad (i = 1, \dots, u), \tag{2}$$

$$\sum_{i=1}^{u} x_{ij} = 1 \quad (j = 1, \dots, n), \tag{3}$$

$$y_i \in \{0, 1\} \quad (i = 1, \dots, u), \tag{4}$$

$$x_{ij} \in \{0, 1\} \quad (i = 1, \dots, u; j = 1, \dots, n). \tag{5}$$

- **Polynomial number of variables and constraints**

Polynomial models (cont'd)

- u and y_i as before;

$$\xi_{ij} = \text{number of items of type } j \text{ packed into bin } i \quad (i = 1, \dots, u; j = 1, \dots, m).$$

- *Integer Linear Program* (ILP) for the **CSP**

$$\min \quad \sum_{i=1}^{u} y_i \tag{6}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} w_j \xi_{ij} \le c y_i \quad (i = 1, \dots, u), \tag{7}$$

$$\sum_{i=1}^{u} \xi_{ij} = d_j \quad (j = 1, \dots, m), \text{ [or } \ge d_j \text{(equivalent)]} \tag{8}$$

$$y_i \in \{0, 1\} \quad (i = 1, \dots, u), \tag{9}$$

$$\xi_{ij} \ge 0, \text{ integer} \quad (i = 1, \dots, u; j = 1, \dots, m). \tag{10}$$

BPP $=$ special case of the CSP in which $d_j = 1$ for all j;

CSP $=$ a BPP in which the item set includes d_j copies of each item type j.

The BPP (and hence the CSP) has been proved to be $\mathcal{NP}$-hard in the strong sense (Garey and Johnson, 1979: transformation from 3-Partition).

Upper and lower bounds

- We will normally refer to the **BPP** (unless otherwise specified).
- **Worst-case performance**
- Given a minimization problem and an approximation algorithm A, let
  - $A(I)$ = solution value provided by A for an instance I;
  - $OPT(I)$ = optimal solution value for an instance I.

  Then

  Worst-case performance ratio (WCPR) of A =

  smallest real number $\overline{r}(A) > 1$ such that $A(I)/OPT(I) \leq \overline{r}(A)$ for all instances I, i.e.,

$$\overline{r}(A) = \sup_I \{A(I)/OPT(I)\}.$$

- Given a minimization problem and a lower bounding procedure L, let
  - $L(I)$ = lower bound provided by L for an instance I.

  Then

  Worst-case performance ratio (WCPR) of L =

  largest real number $\underline{r}(L) < 1$ such that $L(I)/OPT(I) \geq \underline{r}(L)$ for all instances I, i.e.,

$$\underline{r}(L) = \inf_I \{L(I)/OPT(I)\}.$$

Approximation algorithms

- Seminal results: David Johnson's PhD thesis, 1973.
- Huge literature (specific surveys, $\sim$ 200 references).
- Two main families:
  - **On-line algorithms:** sequentially assign items to bins, in the order encountered in input, without knowledge of items not yet packed.
  - **Off-line algorithms:** all items are known in advance, and are available for sorting, preprocessing, grouping, etc.
- Many other (less relevant) families:
  - semi on-line,
  - bounded space,
  - open-end,
  - conservative,
  - re-pack,
  - dynamic,
  - ...

On-line algorithms

- **Next-Fit (NF)**: pack the next item into the current bin if it fits, or into a new bin (which becomes the current one) if it doesn't;
  time complexity: $O(n)$;
  worst-case: $\overline{r}(NF) = 2$ (Hint: the contents of two consecutive bins is $> c$).
- **First-Fit (FF)**: pack the next item into the lowest indexed bin where it fits, or into a new bin if it does not fit in any open bin.
  time complexity: trivial implementation: $O(n^2)$. With special data structures: $O(n \log n)$.
- **Best-Fit (BF)**: pack the next item into the feasible bin (if any) where it fits by leaving the smallest residual space, or into a new one if no open bin can accommodate it;
  time complexity: same as FF.
- **Numerical example:**

  $n = 12$, $c = 100$, $(w_j) =$ (50 3 48 53 53 4 3 41 23 20 52 49)

  NF: $\{50 \quad 3\}$, $\{48\}$, $\{53\}$, $\{53 \quad 4 \quad 3\}$, $\{41 \quad 23 \quad 20\}$, $\{52\}$, $\{49\}$ **7 bins**

  FF: $\{50 \quad 3 \quad 4 \quad 3 \quad 23\}$, $\{48 \quad 41\}$, $\{53 \quad 20\}$, $\{53\}$, $\{52\}$, $\{49\}$ **6 bins**

  BF: $\{50 \quad 3 \quad 4 \quad 3 \quad 23\}$, $\{48 \quad 52\}$, $\{53 \quad 41\}$, $\{53 \quad 20\}$, $\{49\}$ **5 bins**
- The **exact WCPR of FF and BF** has been an open problem for forty years, until recently (2014) Dósa and Sgall proved that $\overline{r}(FF) = \overline{r}(BF) = \frac{17}{10}$.
- Other algorithms: **Worse-Fit** (**WF**, leave the largest residual space), Any-Fit, Almost Any-Fit, Bounded space, Next-k-Fit, Harmonic-Fit, Refined First-Fit, Modified Harmonic-Fit, ...

Off-line algorithms

- Most of the classical on-line algorithms achieve their worst-case performance when the items are packed in increasing order of size or if small and large items are merged, and hence
- main off-line category: sort the items in **decreasing order** of size (time $O(n \log n)$).
- **Next-Fit Decreasing**
  time complexity: $O(n \log n)$;
  Exact worst-case unknown. It has been proved that it is not more than $\frac{7}{4}$.
- **First-Fit Decreasing**; time complexity: $O(n \log n)$; worst-case: $\overline{r}(FFD) = \frac{3}{2}$.
- **Best-Fit Decreasing (BFD)**; time complexity: $O(n \log n)$; worst-case: $\overline{r}(BFD) = \frac{3}{2}$.
- **Numerical example** (resumed):

  $n = 12$, $c = 100$, $(w_j) =$ (50 3 48 53 53 4 3 41 23 20 52 49);
  Sorted items: $(w_j) =$ (53 53 52 50 49 48 41 23 20 4 3 3);

  NFD: $\{53\}$, $\{53\}$, $\{52\}$, $\{50 \quad 49\}$, $\{48 \quad 41\}$, $\{23 \quad 20 \quad 4 \quad 3 \quad 3\}$ **6 bins**

  FFD: $\{53 \quad 41 \quad 4\}$, $\{53 \quad 23 \quad 20 \quad 3\}$, $\{52 \quad 48\}$, $\{50 \quad 49\}$, $\{\ 3\}$ **5 bins**

  BFD: $\{53 \quad 41 \quad 3 \quad 3\}$, $\{53 \quad 23 \quad 20 \quad 4\}$, $\{52 \quad 48\}$, $\{50 \quad 49\}$, **4 bins, optimum**

Best polynomially achievable worst-case performance

- Worst-case of FFD and BFD: $\overline{r}(FFD) = \overline{r}(BFD) = \frac{3}{2}$.
- Can we find a better algorithm? **Bad news:**
- **No polynomial-time approximation algorithm for the BPP can have a WCPR smaller than $\frac{3}{2}$ unless $\mathcal{P} = \mathcal{NP}$.**
- PARTITION problem: is it possible to partition $S = \{w_1, \ldots, w_n\}$ into S_1, S_2 so that $\sum_{j \in S_1} w_j = \sum_{j \in S_2} w_j$?

  PARTITION is $\mathcal{NP}$-complete.

  Assume a polynomial-time approximation algorithm A for the BPP exists such that $OPT(I) > \frac{2}{3} A(I)$ for all instances I.

  Execute A for an instance $\hat{I}$ of the BPP defined by $(w_1, \ldots, w_n)$ and $c = \sum_{j=1}^{n} w_j / 2$.

  if $A(\hat{I}) = 2$ **then** we know that the answer to PARTITION is `yes`;

  else $(A(\hat{I}) \geq 3)$ we know that $OPT(\hat{I}) > \frac{2}{3} 3$, i.e., that $OPT(\hat{I}) > 2$,

  and hence the answer to PARTITION is `no`.

  In other words, we could solve PARTITION in polynomial time! $\square$

Asymptotic worst-case performance

- FFD and BFD provide the best possible WCPR $\Longrightarrow$ the study approximation algorithms focused on a different performance ratio.
- Already in the mid-Seventies D. Johnson proved that $FFD(I) \leq \frac{11}{9} OPT(I) + 4 \ \ \forall \ I$.
- **Asymptotic worst-case performance ratio** of an approximation algorithm $A =$

  smallest real number $\overline{r}^{\infty}(A) > 1$ such that, for some positive integer k, $A(I)/OPT(I) \leq \overline{r}^{\infty}(A)$ for all instances I satisfying $OPT(I) \geq k$.
- $\overline{r}^{\infty}(FFD) = \overline{r}^{\infty}(BFD) = \frac{11}{9}$.
- Impressive number of results, of mostly theoretical relevance (see surveys).
- "History" of the $11/9$ ratio:
  - Johnson (1974): $FFD(I) \leq \frac{11}{9} OPT(I) + 4 \ \ \forall I$. **Proof: 100 pages**;
  - Baker (1985): $FFD(I) \leq \frac{11}{9} OPT(I) + 3 \ \ \forall I$. **Proof: 20 pages**;
  - Yue (1991): $FFD(I) \leq \frac{11}{9} OPT(I) + 1 \ \ \forall I$. **Proof: 10 pages**.

Asymptotic worst-case of on-line algorithms

Algorithm	Time	$\overline{r}^{\infty}(A)$
NF	$O(n)$	2
WF	$O(n \log n)$	2
FF	$O(n \log n)$	1.7
BF	$O(n \log n)$	1.7

- **Any-Fit constraint:**

  If $B_1, \ldots, B_i$ are the current non-empty bins, then the current item

  will not be packed into B_{i+1} unless it does not fit in any of the bins $B_1, \ldots, B_i$.
- $\mathcal{AF}$ = class of on-line heuristics satisfying the Any-Fit constraint.
- FF, WF, BF $\in \mathcal{AF}$.
- It can be proved that

  For every algorithm $A \in \mathcal{AF}$, $\quad \overline{r}^{\infty}(FF) \le \overline{r}^{\infty}(A) \le \overline{r}^{\infty}(WF)$

Asymptotic worst-case of off-line algorithms

- *For any algorithm $A \in \mathcal{AF}$ that packs the items by nonincreasing size,*

$$\frac{11}{9} \leq \overline{r}^{\infty}(A) \leq \frac{5}{4}$$

Algorithm	Time	$\overline{r}^{\infty}(A)$	
NFD	$O(n \log n)$	1.691...	Johnson et al., 1973-1974
FFD	$O(n \log n)$	1.222...	Johnson et al., 1973-1974
BFD	$O(n \log n)$	1.222...	Johnson et al., 1973-1974
MFFD	$O(n \log n)$	1.183...	Garey & Johnson, 1985
B2F	$O(n \log n)$	1.25	Friesen & Langston, 1991
CFB	$O(n \log n)$	1.16410... $\leq \cdot \leq$ 1.2	Friesen & Langston, 1991
GXFG	$O(n)$	1.5	Johnson, 1974
H_4	$O(n)$	1.333...	Martel, 1985
H_7	$O(n)$	1.25	Bekesi & Galambos, 1997

MFFD (Modified FFD): Try to pack pairs of items with size in $(c/6, c/3]$ into bins containing a single item of size $> c/2$.

B2F (Best Two Fit): Fill one bin at a time, in greedy way; when no further item fits into the current bin, if the bin contains more than one item, try to replace the smallest item in the bin with a pair of unpacked items with size $\geq c/6$.

CFB (combined FFD–B2F): run both B2F and FFD and take the better packing.

Approximation schemes

- **Approximation scheme** = parametric family of approximation algorithms that produces a prefixed worst-case behavior.
- **Question:** Does there exist an $\varepsilon > 0$ such that every $O(n)$-time algorithm A must satisfy $\overline{r}^\infty(A) \geq 1 + \varepsilon$?
- **Answer: No** (Fernandez de la Vega and Lueker, 1981):

  For any $\varepsilon > 0$ there exists a linear-time algorithm A_ε such that

$$\overline{r}^\infty(A_\varepsilon) \leq 1 + \varepsilon \;\; \forall \varepsilon$$

  A_ε is a **Polynomial-Time Approximation Scheme** based on:

  - partitioning of the items (depending on ε);
  - rounding techniques;
  - solution of an LP relaxation;
  - Next-Fit technique.

- The time complexity of A_ε is polynomial (linear) in n $\forall$ ε, but exponential in $\frac{1}{\varepsilon}$.
- Improved by Karmarkar and Karp, 1982: **Fully Polynomial-Time Approximation Scheme**; time complexity polynomial (linear) both in n and $\frac{1}{\varepsilon}$.

Lower bounds

- **Continuous relaxation**: $L_1 = \left\lceil \sum_{j=1}^{n} w_j / c \right\rceil$. Computable in $O(n)$ time.
- In the optimal solution at most one bin can have a total contents $\leq \dfrac{c}{2}$.

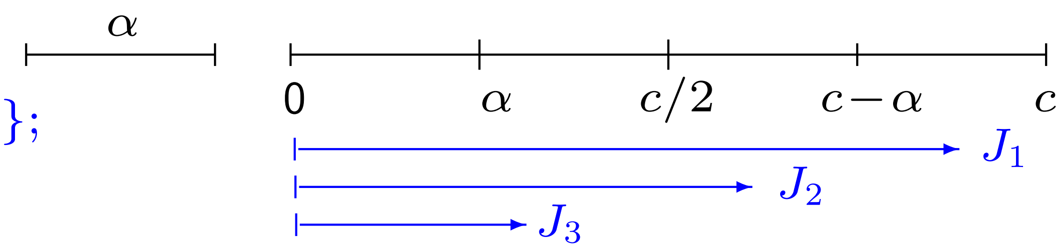
  $\Longrightarrow \sum_{j=1}^{n} w_j > \frac{OPT(I)-1}{2} c \Longrightarrow OPT(I) \leq 2 \frac{\sum_{j=1}^{n} w_j}{c} \leq 2L_1$

  $\Longrightarrow \underline{r}(L_1) = \frac{1}{2}$ (worst case: $(w) = (\frac{c}{2}+1, \frac{c}{2}+1, \dots))$
- **A better bound** (Martello and Toth, 1990). Given any integer α $(0 \leq \alpha \leq c/2)$, let

  $J_1 = \{j \in N : w_j > c - \alpha\}$;

  $J_2 = \{j \in N : c - \alpha \geq w_j > c/2\}$;

  $J_3 = \{j \in N : c/2 \geq w_j \geq \alpha\}$,



  each item in $J_1 \cup J_2$ needs a separate bin,

  no item of J_3 can go to a bin containing an item of J_1. Then

  $L(\alpha) = |J_1| + |J_2| + \max\left(0, \left\lceil \frac{\sum_{j \in J_3} w_j - (|J_2|c - \sum_{j \in J_2} w_j)}{c} \right\rceil\right)$ is a valid lower bound.

  The **overall bound** $L_2 = \max\{L(\alpha) : 0 \leq \alpha \leq c/2, \alpha \text{ integer}\}$

  (1) can be computed in $O(n \log n)$ time;

  (2) has **WCPR equal to** $\frac{2}{3}$.

Best polynomially achievable worst-case performance

- Worst-case of L_2: $\frac{2}{3}$.
- Can we find a better lower bound?
- **No lower bound, computable in polynomial time, for the BPP can have a WCPR greater than $\frac{2}{3}$ unless $\mathcal{P} = \mathcal{NP}$.**
- $\textsc{Partition}$ problem: is it possible to partition $S = \{w_1, \ldots, w_n\}$ into S_1, S_2 so that $\sum_{j \in S_1} w_j = \sum_{j \in S_2} w_j$? ($\mathcal{NP}$-complete).

  Assume a polynomial-time lower bound L exists such that $OPT(I) < \frac{3}{2}\, L(I)$ $\forall$ instances I.

  Compute L for instance $\hat{I}$ of the BPP defined by $(w_1, \ldots, w_n)$ and $c = \sum_{j=1}^{n} w_j/2$.

  if $L(\hat{I}) \geq 3$ **then** we know that the answer to $\textsc{Partition}$ is no.;

  else $(L(\hat{I}) = 2)$ we know that $OPT(\hat{I}) < \frac{3}{2}\,2$, i.e., $OPT(\hat{I}) = 2$ and the answer to $\textsc{Partition}$ is yes.

  In other words, we could solve $\textsc{Partition}$ in polynomial time! $\square$
- Other lower bounds can have better practical performance (Labbé et al., Martello and Toth) and have asymptotic WCPR equal to $\frac{3}{4}$.
- Different types of lower bound computations are based on *dual feasible functions* (Lueker, Fekete and Schepers).
- Methods to improve on a lower bound value (Dell'Amico and Martello, Alvim et al., Haouari and Gharbi, Jarboui et al.)

Reduction Algorithms

- **Reduction Algorithm** = preprocessing procedure used to determine the optimal value of a subset of variables.
- Numerical Example: $n = 12$, $c = 100$,
  $(w_j) = (99 \quad 93 \quad 90 \quad 88 \quad 80 \quad 10 \quad 10 \quad 6 \quad 5 \quad 5 \quad 4 \quad 4)$.

  99 alone in a bin;

  93 can be packed with at most one more item $\rightarrow$ packing it with 6 is dominating (largest item);

  reduced instance: $(w_j) = (90 \quad 88 \quad 80 \quad 10 \quad 10 \quad 5 \quad 5 \quad 4 \quad 4)$;

  90 can be packed with at most two more items $\rightarrow$ packing it with 10 is dominating (bin full);

  reduced instance: $(w_j) = (88 \quad 80 \quad 10 \quad 5 \quad 5 \quad 4 \quad 4)$;

  88 can be packed with at most two more items $\rightarrow$ packing it with 10 is dominating (10 $\geq$ maximum pair);

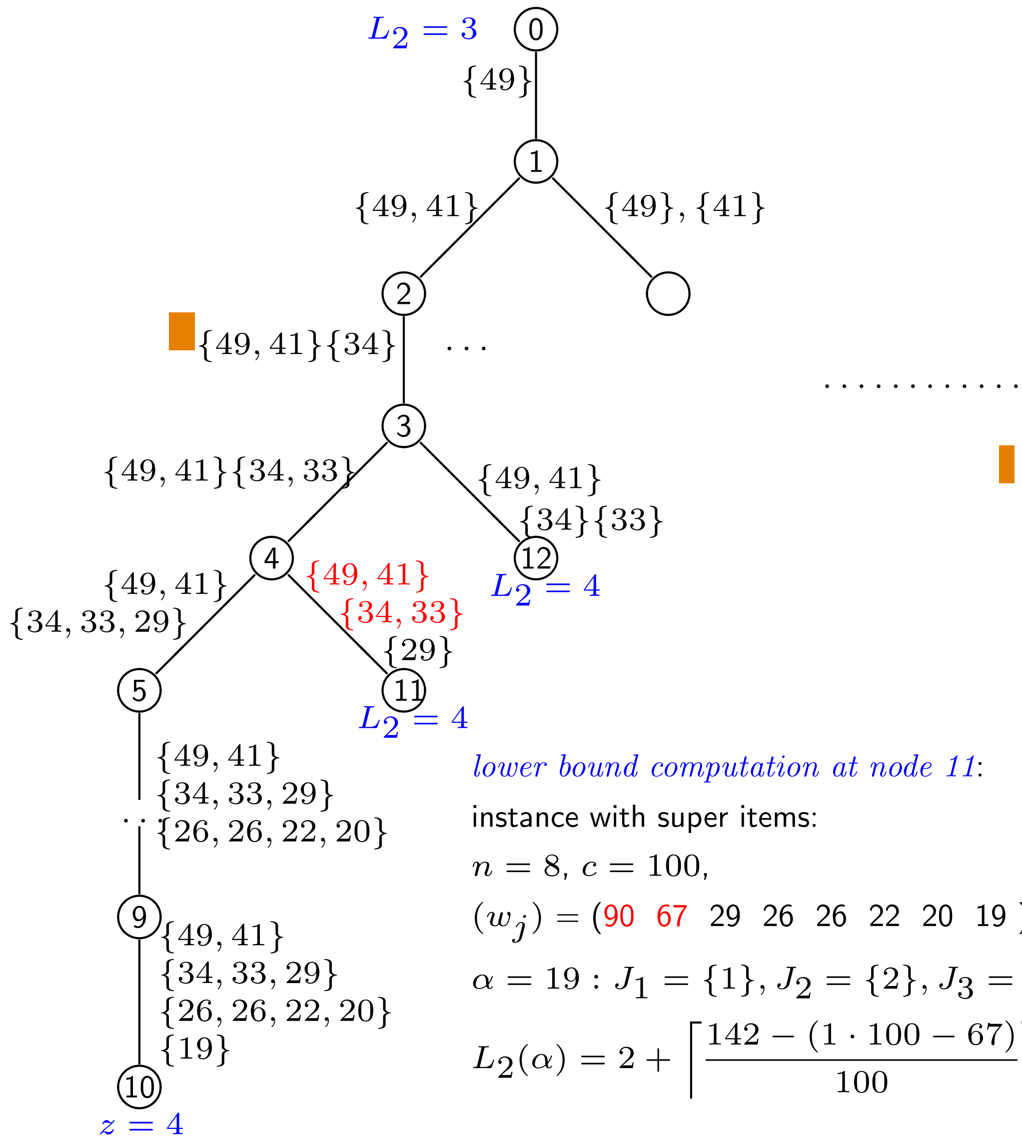
  reduced instance: $(w_j) = (80 \quad 5 \quad 5 \quad 4 \quad 4)$: one bin (optimal solution).
- Ideas generalized to a
  general **Dominance Criterion** between pairs of subsets of items (Martello and Toth)

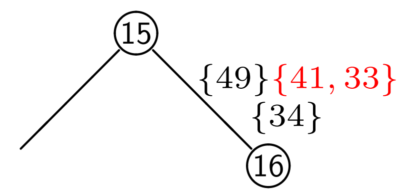
Exact Algorithms: Branch-and-Bound

- Eilon and Christofides, 1971 (enumerative algorithm);
  Hung and Brown, 1978 (branch-and-bound);
  Martello and Toth, 1990 (specifically tailored branch-and-bound: MTP, popular Fortran code);
  Scholl, Klein and Jurgens, 1997 (MTP + tabu search).
- Outline (MTP)
  - depth-first strategy;
  - items sorted by non-increasing size;
  - at each decision node, the first (largest) free item is assigned
    * to all feasible initialized bins,
    * and, possibly, to a new bin.
  - At any *forward step*
    * lower bound computations (L_2 and L_3 (improved bound));
    * reduction of the current instance;
    * if the node is not fathomed, FFD, BFD and WFD executed on the current problem to try and improve the incumbent solution.
  - Dominance criterion between decision nodes.
  - Computations at the decision nodes:
    * for each initialized bin, create a
      "super item" having size = sum of the sizes of the items in the bin;
    * lower bounds and reduction for the instance given by $\{$super items$\} \cup \{$free items$\}$.

Example

$n = 10$, $c = 100$, $(w_j) = ($49 41 34 33 29 26 26 22 20 19):





Reduction (super item 41+33=74):
$\{41, 33, 26\}$ removed from the instance
FFD on the residual instance:
$\{49, 29, 22\}$
$\{34, 26, 20, 19\}$
optimal

lower bound computation at node 11:

instance with super items:

$n = 8$, $c = 100$,

$(w_j) = ($90 67 29 26 26 22 20 19):

$\alpha = 19 : J_1 = \{1\}, J_2 = \{2\}, J_3 = \{3, \ldots, 8\};$

$$L_2(\alpha) = 2 + \left\lceil \frac{142 - (1 \cdot 100 - 67)}{100} \right\rceil = 4$$

Exact Algorithms: Branch-and-Bound/-and-Price/-and-Cut

- Classical **Branch-and-Bound** for ILP/MILP: At each node of the branch-decision tree:
  - solve the LP relaxation of the (sub-)problem associated with the the current decision node;
  - if the solution is not integer, separate a fractional variable to get 2 new sub-problems (2 new decision nodes);
  - continue until all decision nodes have been explored.
- **Branch-and-Cut:** ...
  - if the solution is not integer, before separating, add cutting planes to strengthen the relaxation, possibly finding an integer solution or improving on the lower bound value.
  - if the solution remains not integer, separate.
- **Branch-and-Price:** ...
  - use *column generation* to solve the LP relaxation at each node:
  - initially, only a subset of columns is included in the LP relaxation (*Restricted Master Problem*);
  - an auxiliary problem (*Pricing Problem*) is used to check optimality and to find columns to be added to improve the LP solution value;
- **Branch-and-Price-and-Cut** = Branch-and-Bound + column generation + cutting planes.
- For the BPP and the CSP, all Branch-and-Price (-and-Cut) algorithms are based on the *set covering formulation* and the solution of its continuous relaxation through column generation (seminal work by Gilmore and Gomory).

Set covering formulation

- Enumeration of the set P of all *patterns* p (combinations of items that can fit into a bin).
- **For the CSP:** pattern $p \equiv$ integer array $(a_{1p}, a_{2p}, \dots)$, with a_{jp} = number of copies of item j contained in pattern p, satisfying $\sum_{j=1}^{m} a_{jp} w_j \le c$ and $a_{jp} \ge 0$, integer $\forall j$.
- Let y_p = number of times pattern p is used. **Set covering** formulation of the CSP:

$$\min \quad \sum_{p \in P} y_p \tag{11}$$

$$\text{s.t.} \quad \sum_{p \in P} a_{jp} y_p \ge d_j \qquad (j = 1, \dots, m), \tag{12}$$

$$y_p \ge 0 \text{ and integer } (p \in P). \tag{13}$$

- Similarly **for the BPP**: (i) $p \equiv$ binary array, y_p binary ($= 1$ iff pattern p is used for a bin):

$$\sum_{p \in P} a_{jp} y_p \ge 1 \; (j = 1, \dots, n) \tag{14}$$

- **the number of feasible patterns is exponential**
  $\Longrightarrow$ **the number of columns of the LP relaxation is exponential** $\Longrightarrow$ **Column generation**

Column generation

- heuristically initialize the LP relaxation it with a subset of patterns $P' \subset P$ (**Restricted Master Problem (RMP)**):

$$\min \sum_{p \in P'} y_p \tag{15}$$

$$\text{s.t.} \quad \sum_{p \in P'} a_{jp} y_p \geq d_j \quad (j = 1, \ldots, m), \tag{16}$$

$$y_p \geq 0 \quad (p \in P'). \tag{17}$$

- Solve (15)-(17) and let π_j be the dual variables associated with the jth constraint (16).
- **Pricing:** find a column $p \notin P'$ that could reduce the objective function value:
  - find the column with the most negative reduced cost (**Slave Problem** (SP)) by solving an associated *knapsack problem* in the dual variables.
  - if solution of SP > 1, then add the corresponding column (pattern) to the RMP.
- Iterate until no column with negative reduced cost is found (optimal solution).
- Huge number of **Branch-and-Price(-and-Cut) algorithms** in the Nineties and the Noughties;
- Most efficient algorithm (and computer code): Belov and Scheithauer (2006).

Parenthesis: IRUP and MIRUP (BPP and CSP)

- L_{LP} = solution value of the LP relaxation of the set covering formulation;
- z_{opt} = optimal solution value;
- **IRUP** (*Integer Round-Up Property*) **conjecture:** $z_{\mathsf{opt}} = \lceil L_{\mathsf{LP}} \rceil$.
- **Disproved** by Marcotte (1986) (instance with $n = 24$ and $c = 3,397,386,255$).
- **MIRUP** (*Modified IRUP*) **conjecture:** $z_{\mathsf{opt}} \leq \lceil L_{\mathsf{LP}} \rceil + 1$.
- **Conjecture open.**

Pseudo-Polynomial Formulations

- The number of variables and constraints depends on the number of items **and** on the **bin capacity**.

One-cut formulation

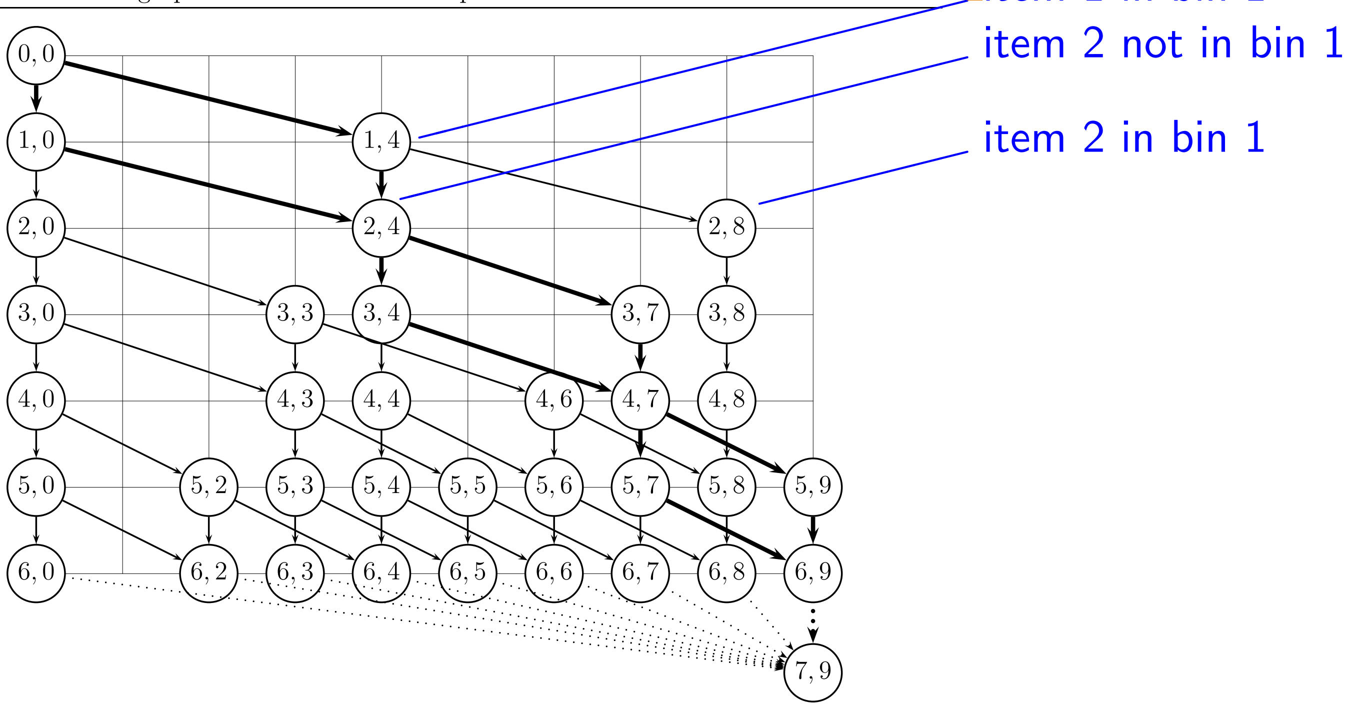
- Independently developed by Rao in 1976 and by Dyckhoff in 1981.
- Basic idea (for the **CSP**): simulate the physical cutting process:
  - divide an ideal bin into **two pieces**, where
  - the **left piece** is an item that has been cut;
  - the **right piece** is either another item

    or a residual that can be re-used to produce other items.
  - Iterate the process on cutting residuals or new bins, until all demands are fulfilled.
  - **Integer variables** x_{pq} **=** number of times a bin, or a residual of width p, is cut into a left piece of width q and a right piece of width $p - q$.
- The resulting ILP model has $O(mc)$ variables and $O(c)$ constraints.

DP-flow formulation

- Cambazard and O'Sullivan (2010): Basic idea (for the **BPP**):
  - associate variables with the decisions taken in a classical *dynamic programming (DP) table*;
  - DP states $\leftrightarrow$ graph: **path** from initial to terminal node = feasible filling of a bin.
- Example: $n = 6$, $c = 9$, $w = (4, 4, 3, 3, 2, 2)$:
  $[j, d]$ $(j = 0, .., n; d = 0, .., c)$: [decisions taken up to item j, partial bin filling d units].

Figure 1 DP-flow graph construction for Example 1



- Network Flow-type model to minimize the number of paths. $O(nc)$ variables and constraints.

Arc-flow formulation

- Valério de Carvalho (1999) anticipated DP-flow (but Wolsey (1977) anticipated everybody):

  vertically shrunk the DP graph: states with the same partial bin filling $\rightarrow$ single state:

Figure 1 DP-flow graph construction for Example 1

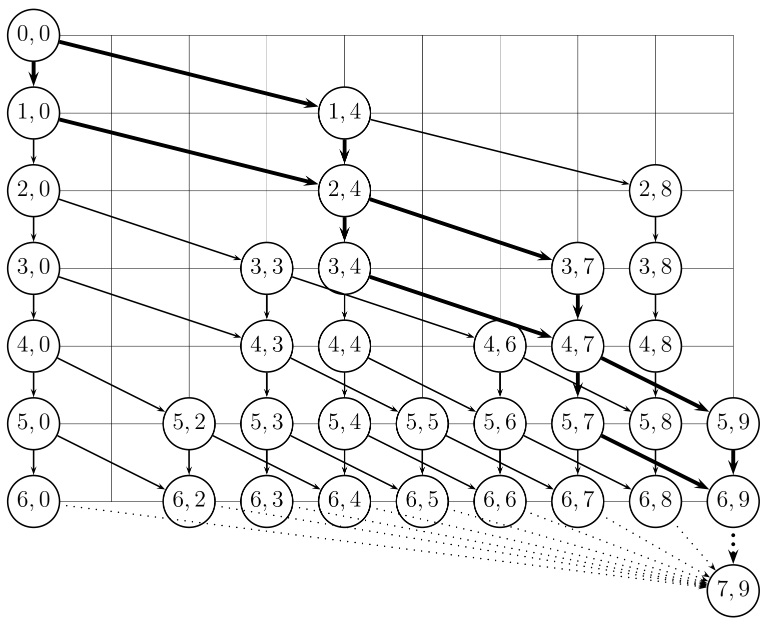
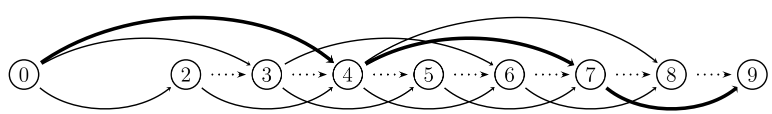
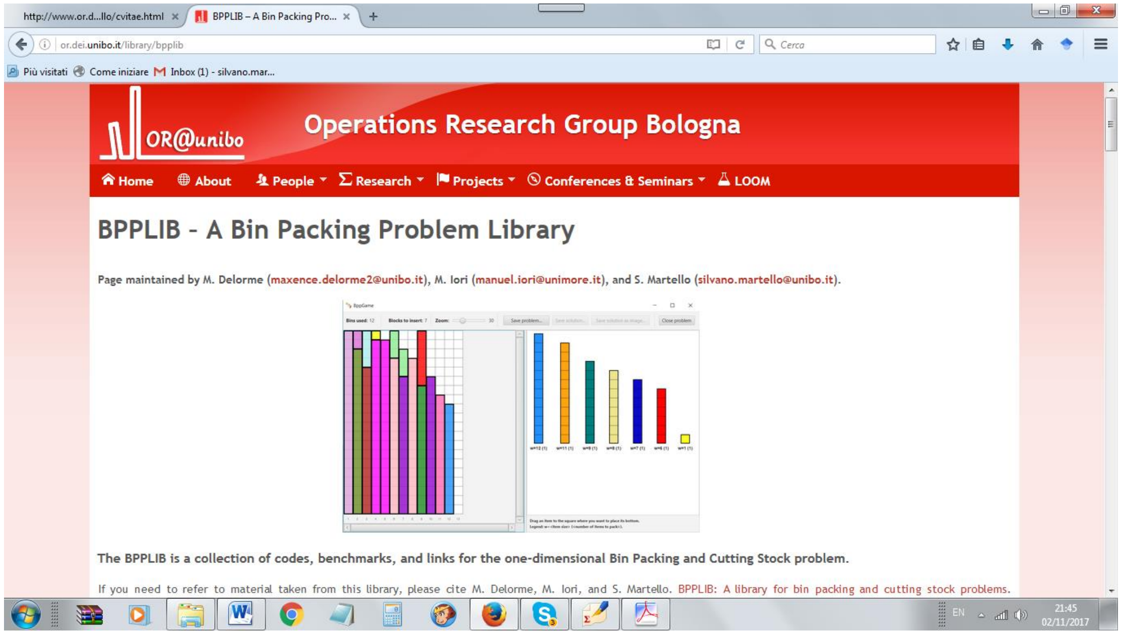


Figure 2 Arc-flow representation of the graph of Figure 1



- **CSP** modeled as a network flow problem;
- Brandão and Pedroso (2016): alternative arc-flow formulation, very effective code VPSOLVER.

Computer codes and the BPPLIB



From this talk:
MTP (branch-and-bound, Fortran)
BISON (branch-and-bound, MTP + Tabu Search, Pascal)
BELOV (branch-and-cut-and-price, C++ & Cplex)
ONECUT (pseudo-polynomial, C++ & Cplex/SCIP))
ARCFLOW (pseudo-polynomial, C++ & Cplex/SCIP)
DPFLOW (pseudo-polynomial, C++ & Cplex/SCIP)
VPSOLVER (pseudo-polynomial, C++ & Gurobi)

Other codes, benchmarks, links, BibTeX file, interactive visual solver.

Experimental evaluation (BPP)

Number of **literature** instances solved in less than **10 minutes**

Set	# inst.	BISON	BELOV	ARCFLOW	VPSOLVER
Falkenauer U	74	50	**74**	**74**	**74**
Falkenauer T	80	47	**80**	**80**	**80**
Scholl 1	323	290	**323**	**323**	**323**
Scholl 2	244	234	**244**	231	242
Scholl 3	10	3	**10**	0	**10**
Wäscher	17	10	**17**	4	13
Schwerin 1	100	**100**	**100**	**100**	**100**
Schwerin 2	100	63	**100**	**100**	**100**
Hard28	28	0	**28**	26	26
Total	976	797	**976**	938	968

Number of **random** instances solved in less than **10 minutes**

n	# inst.	BISON	BELOV	ARCFLOW	VPSOLVER
50	165	**165**	**165**	**165**	**165**
100	271	261	**271**	**271**	**271**
200	359	299	**359**	**359**	**359**
300	393	269	**393**	**393**	**393**
400	425	250	**425**	**425**	**425**
500	414	212	**414**	**414**	**414**
750	433	217	**433**	431	**433**
1000	441	200	**441**	434	**441**
Total	2901	1873	**2901**	2892	**2901**

Difficult instances

Number of **difficult (ANI)** instances, out of **50**, solved in less than **1 hour (average absolute gap)**

n	$\overline{c}$	BISON	BELOV	ARCFLOW	VPSOLVER
201	2500	0 (1.0)	**50** (0.0)	16 (0.7)	47 (0.1)
402	10000	0 (1.0)	1 (1.0)	0 (1.0)	6 (0.9)
600	20000	-	(1.0)	-	0 (1.0)
801	40000	-	(1.0)	-	0 (1.0)
1002	80000	-	-	-	-
Overall		0 (1.0)	51 (0.7)	16 (0.8)	53 (0.7)

A final comment:

- Originally (ARCFLOW, 1999) pseudo-polynomial formulations were seen as theoretical results and rarely directly used in practice as ILP formulations (too many variables and constraints).
- Nowadays they are extremely competitive in practice. **Why?**
- 20 selected random instances ($n \in [300, 1000]$, $c \in [400, 1000]$;
  ARCFLOW: $\#$ constraints $\in [482, 1093]$, #variables $\in [32\,059, 111\,537]$);
  8 versions of CPLEX: number of solved instances [average CPU time]:

Time	inst.	6.0 (1998)	7.0 (1999)	8.0 (2002)	9.0 (2003)	10.0 (2006)	11.0 (2007)	12.1 (2009)	12.6.0 (2013)
10 minutes	20	13 [366]	10 [420]	5 [570]	17 [268]	19 [162]	**20** [65]	19 [117]	**20** [114]
60 minutes	20	16 [897]	15 [1210]	15 [2009]	**20** [343]	**20** [186]	**20** [65]	19 [267]	**20** [114]

End of Part I

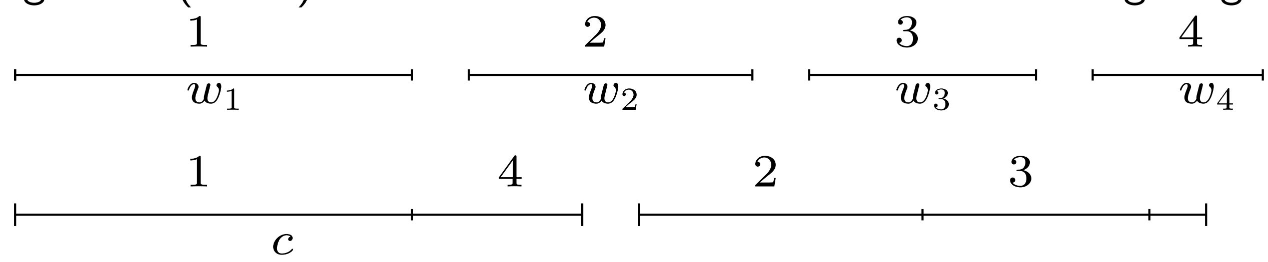
II. Two-Dimensional Packing Problems

Contents

- Definitions, applications, and variants
- Approximation algorithms
  - Two-phase algorithms
  - One-phase algorithms
- Lower bounds
- Exact algorithms
- Three-dimensional packing problems (brief outline)

Definitions

- Geometrical interpretation of the (one-dimensional) **BPP**:
  pack a set of segments (*items*) into the minimum number of identical large segments (*bins*):



- **Two possible two-dimensional extensions.**

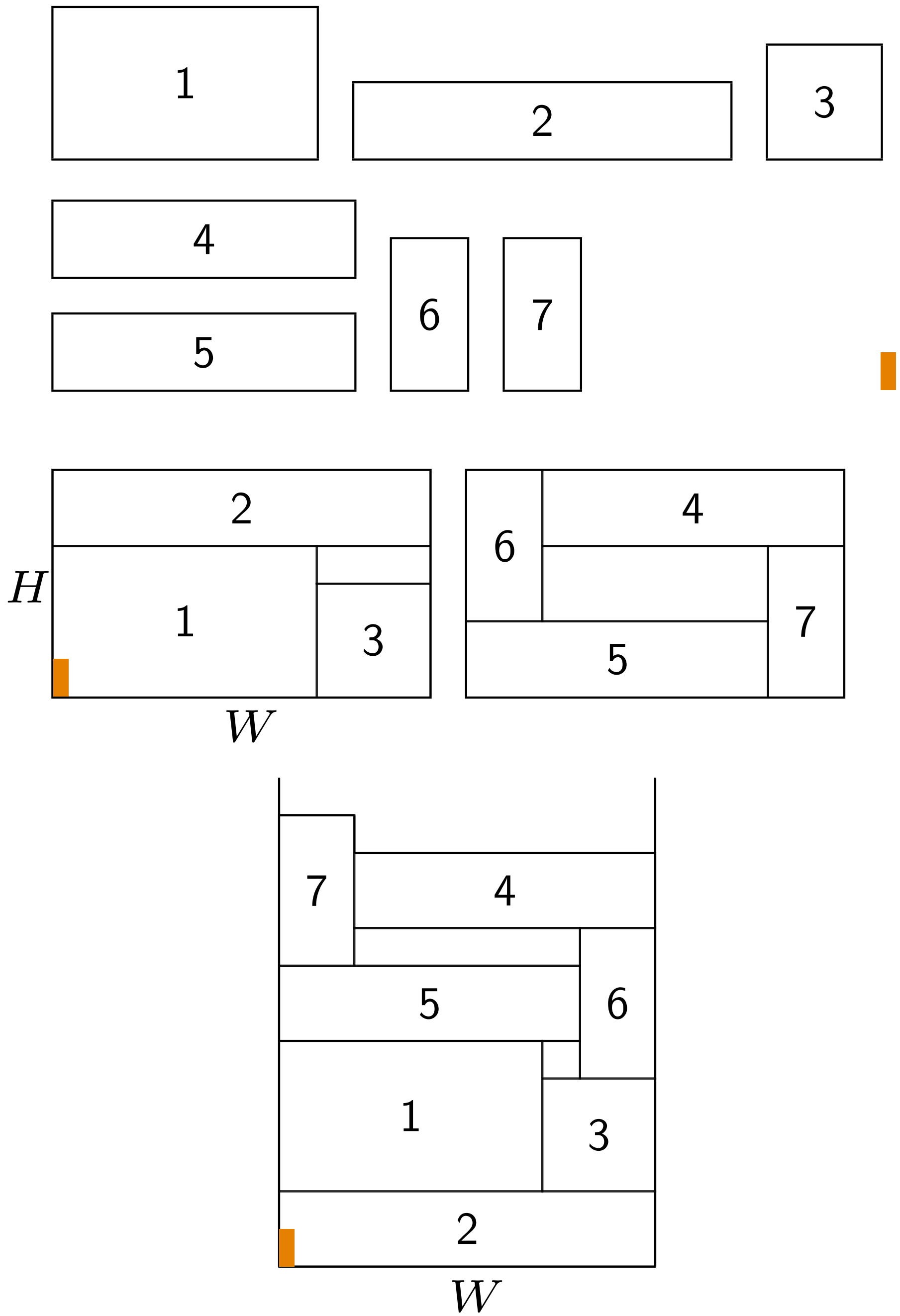
  Given n **rectangular** *items*, each having integer **height** h_j and **width** w_j $(j = 1, \ldots, n)$,

---

1) **Two-Dimensional Bin Packing Problem (2BPP)**:
given an unlimited number of identical **rectangular bins** of integer **height** H and **width** W,
pack all the items, without overlapping, into the minimum number of bins
(find the minimum number of **cutting patterns** providing all the items).

---

2) **Two-Dimensional Strip Packing Problem (2SPP)**:
given a single **open-ended bin (strip)** of **width** W and **infinite height**
determine a cutting pattern providing all the items
such that the height to which the strip is filled is minimized.
(Also called **1.5-dimensional packing**.)


1
2
3
4
5
6
7
2
H
1
3
W
6
4
7
5
7
4
5
6
1
3
2
W

Applications

- Industrial cutting. Cutting from:
  - standardized stock pieces (glass industry, wood industry, ...) $\Longrightarrow$ 2BPP;
  - *rolls* (textile industry, paper industry, ...) $\Longrightarrow$ 2SPP;
- Transportation:
  - packing on floors, shelves, truck beds, ...
  - packing into containers (3-Dimensional Bin Packing Problem, reduction to a series of 2BPP)
- Memory sharing: shared storage multiprocessor system: 2SPP with

$$\begin{array}{rcl} \text{job } j & \longleftrightarrow & \text{rectangle } j \\ \text{memory requirement} & \longleftrightarrow & w_j \text{ (contiguous locations)} \\ \text{time requirement} & \longleftrightarrow & h_j \\ \text{system} & \longleftrightarrow & \text{strip} \\ \text{available memory} & \longleftrightarrow & W \\ \text{time} & \longleftrightarrow & \text{height} \end{array}$$

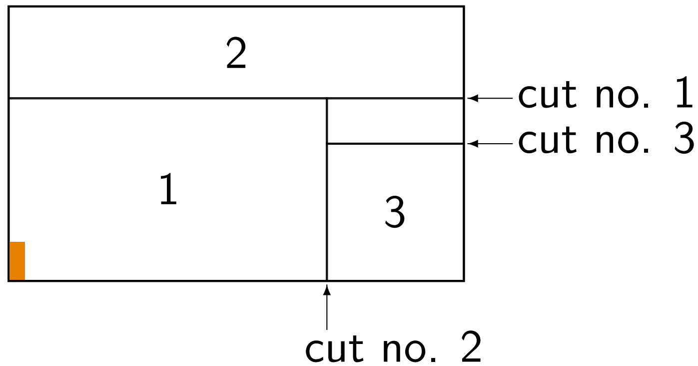
Complexity

- Both the 2BPP and the 2SPP are special cases of the BPP;
- both are **strongly $\mathcal{NP}$-hard**.

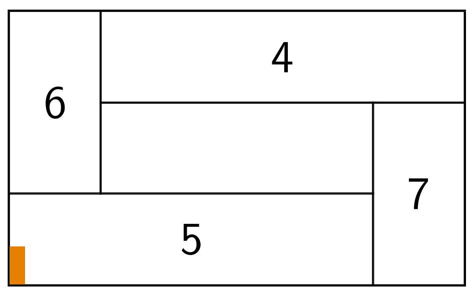
Variants

- **Guillotine Cuts**: In cutting applications it may be imposed that the patterns be such that the items can be obtained by sequential edge-to-edge cuts parallel to the edges of the bin.

guillotine-cuts:



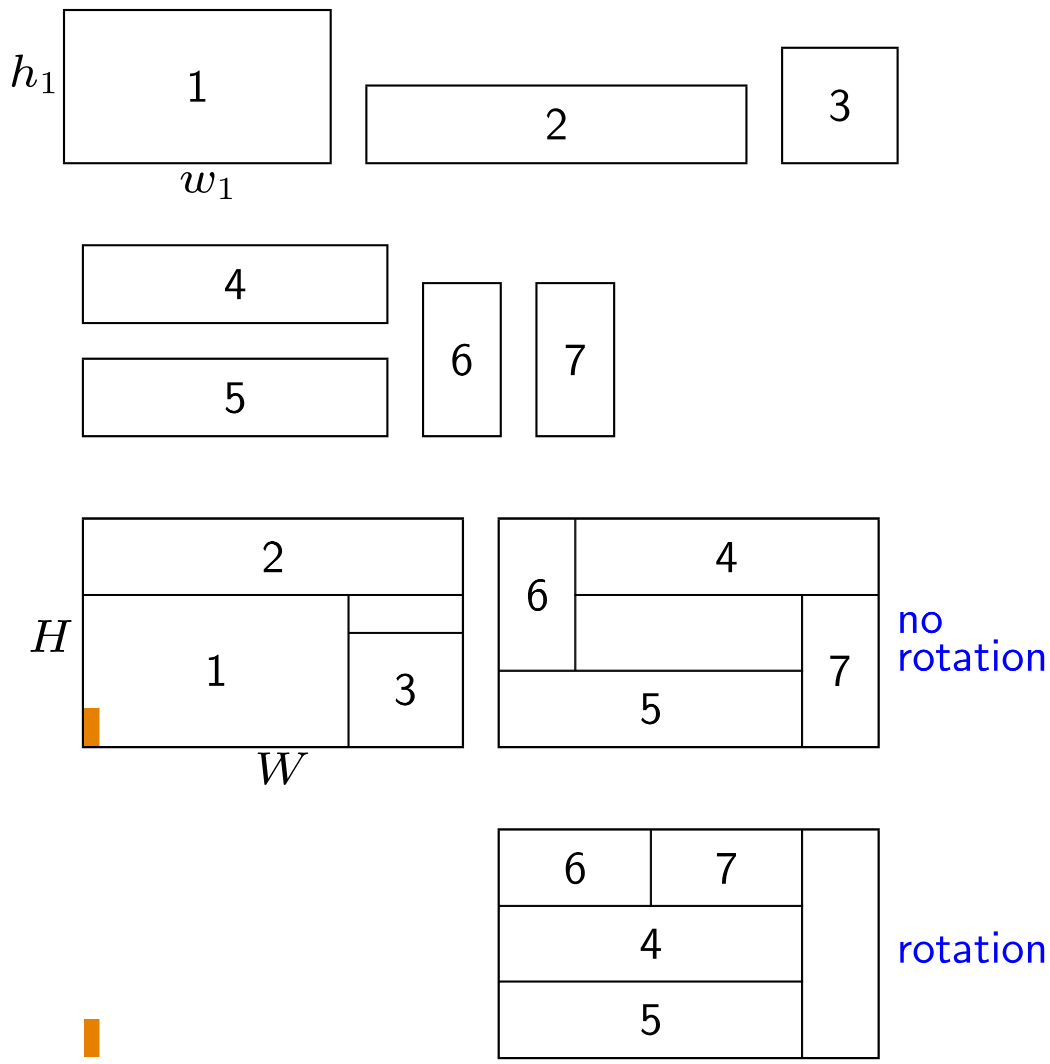
non guillotine-cuts:

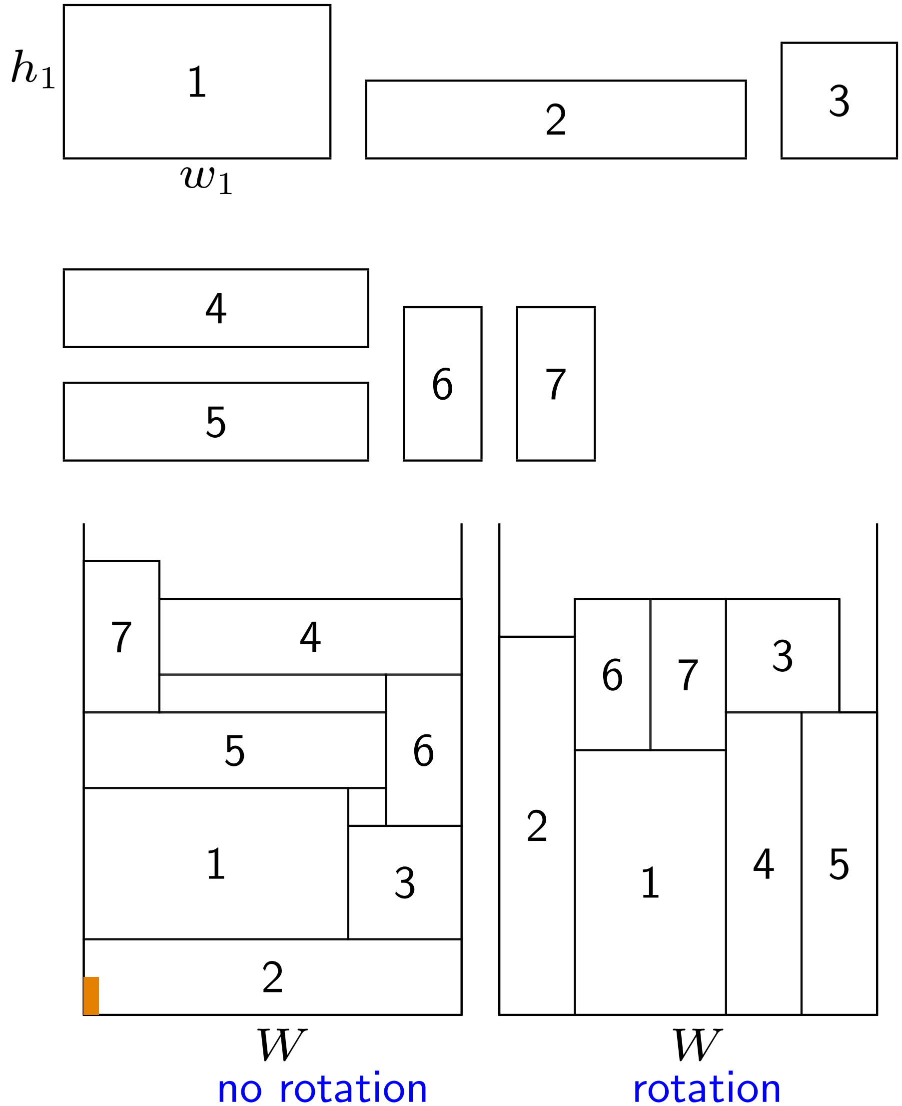


- additional constraints: limit on the number of cuts per bin (2,3).

Variants

- **Item Rotation**: if the items in demand do not have a prefixed orientation with respect to the bins then they may be rotated (usually by 90°).



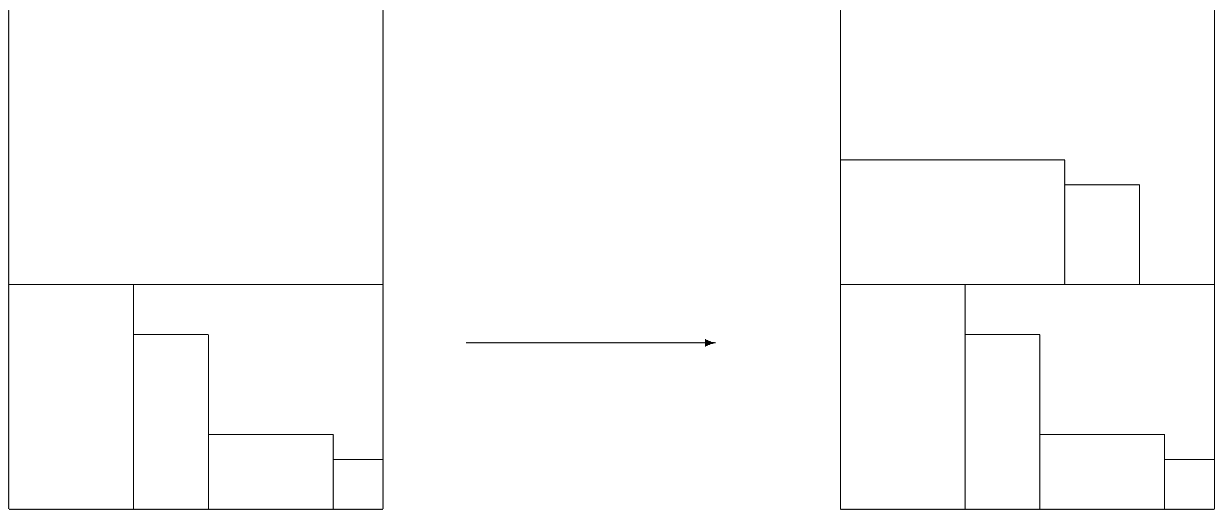


- Guillotine-cuts and rotations are frequent in other two-dimensional packing problems (Two-Dimensional Cutting Stock, Two-Dimensional Knapsack)
- For two-dimensional bin (strip) packing problems most results concern the case: **no guillotine-cut required**, **no rotation allowed** (implicitly assumed in the following).

Approximation algorithms

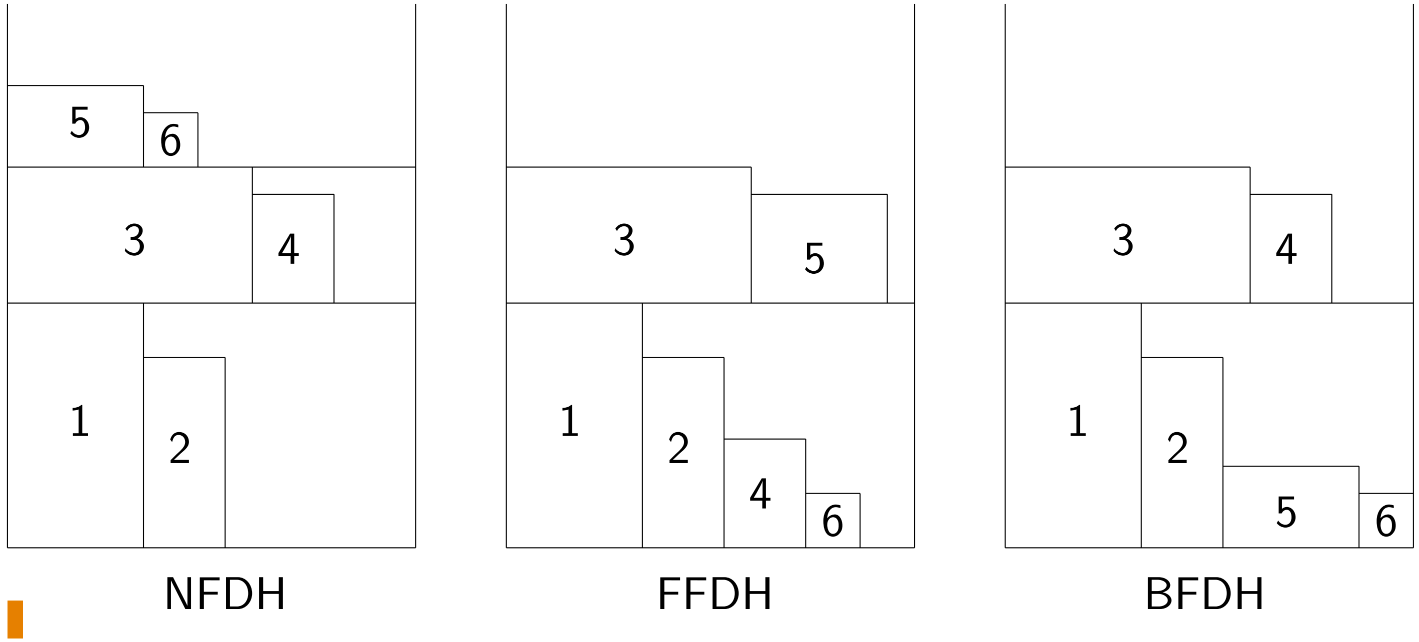
Two main families of heuristic algorithms:

- **one-phase algorithms:** directly pack the items into the bins;
- **two-phase algorithms:**
  - *Phase 1*: pack the items into a single strip;
  - *Phase 2*: use the strip solution to construct a packing into bins.
- **Shelf algorithms:** in most of the approaches the bin/strip packing is obtained by placing the items, from left to right, in rows forming levels ($shelves$):
  - 1st shelf = bottom of the bin/strip;
  - subsequent shelves = horizontal line given by the top of the tallest item in the shelf below.



Shelf packing strategies (2SPP)

- sort the items by nonincreasing height (assumed in the following);
- j = current item, s = last created shelf:
- **Next-Fit Decreasing Height (NFDH):** pack j left justified in shelf s, if it fits; otherwise, create a new shelf $(s+1)$, and pack j left justified into it.
- **First-Fit Decreasing Height (FFDH):** pack j left justified in the first shelf where it fits, if any; if no shelf is feasible, initialize a new shelf as in NFDH.
- **Best-Fit Decreasing Height (BFDH):** pack j left justified in the feasible shelf which minimizes the unused horizontal space; if no shelf is feasible, initialize a new shelf as in NFDH.



Worst-Case Performance (2SPP)

- If the heights are **normalized** so that $\max_j\{h_j\} = 1$, for the **Strip packing** we have (Coffman, Garey, Johnson, and Tarjan, 1980):

$$NFDH(I) \leq 2 \cdot OPT(I) + 1 \;\; \forall\, I$$

and

$$FFDH(I) \leq \frac{17}{10} \cdot OPT(I) + 1 \;\; \forall\, I$$

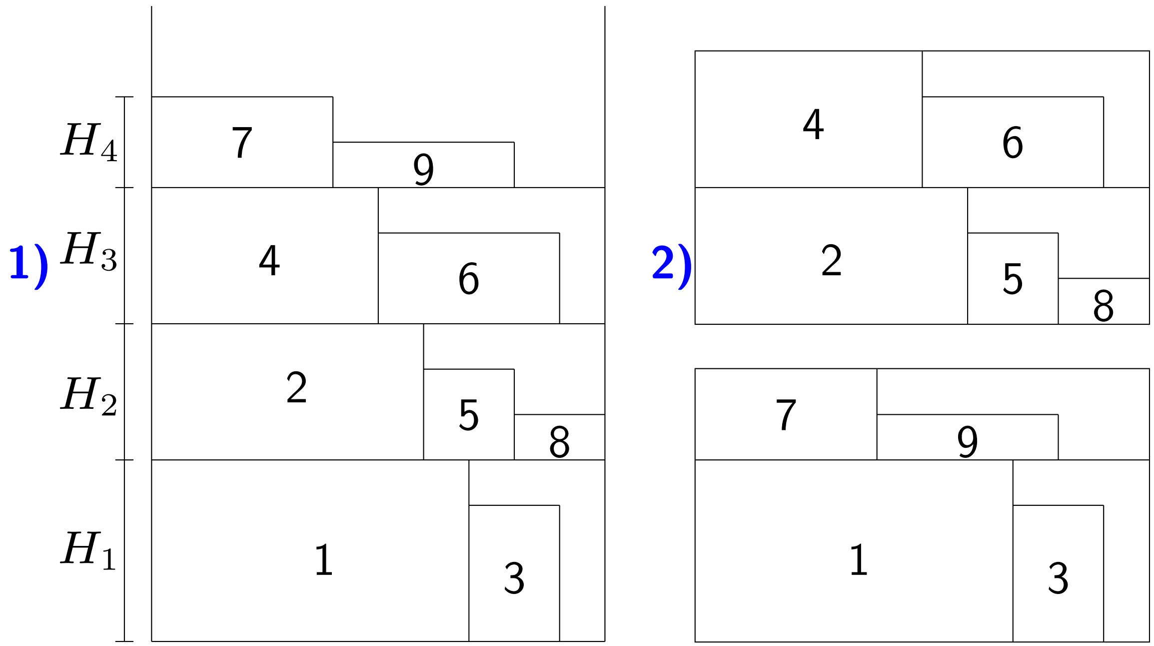
- Remind: for the BPP, $\overline{r}(NF) = 2$, $\overline{r}(FF) = \frac{17}{10}$.

- Both bounds are tight.
- If the h_j's are **not normalized**, only the additive term is affected.

- Both algorithms can be implemented so as to require $O(n \log n)$ time, through the appropriate data structures used for the 1BPP.

Two-phase algorithms (2BPP)

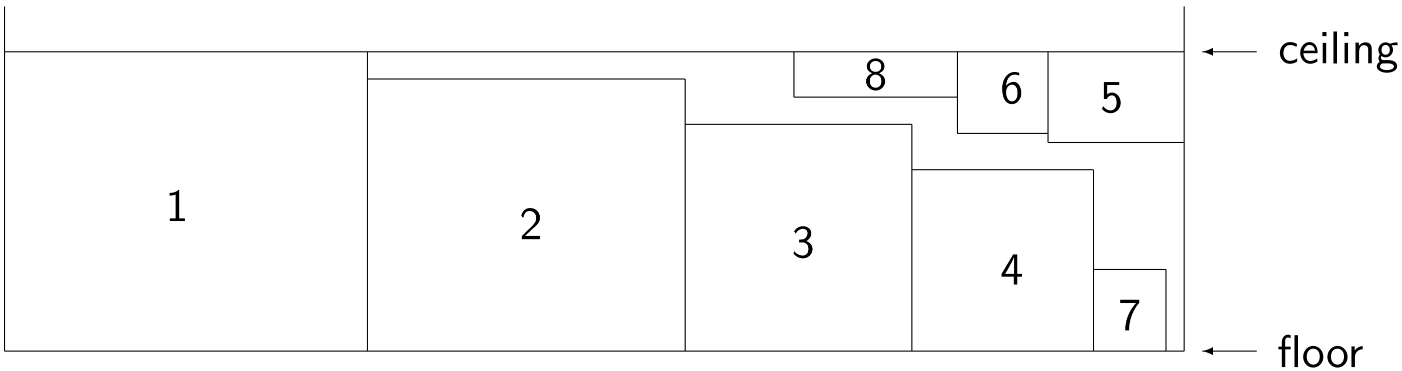
- **Hybrid First-Fit (HFF,** Chung, Garey, and Johnson, 1982):
  - *Phase 1*: strip packing through **FFDH** $\rightarrow H_1, H_2, \ldots$ =heights of the resulting shelves ($H_1 \geq H_2 \ldots$ by construction).
  - *Phase 2*: one-dimensional bin packing problem over the shelves: item sizes H_i, bin capacity H: solve through the **FFD** algorithm (BPP):
    – initialize bin 1 to pack shelf 1;
    – **for** $i := 2, \ldots$ **do** pack shelf i into the lowest indexed bin where it fits, if any (**otherwise** initialize a new bin).



- If the heights are **normalized to 1**, $HFF(I) \leq \frac{17}{8} \cdot OPT(I) + 5 \;\; \forall \, I$

Other two-phase algorithms (2BPP)

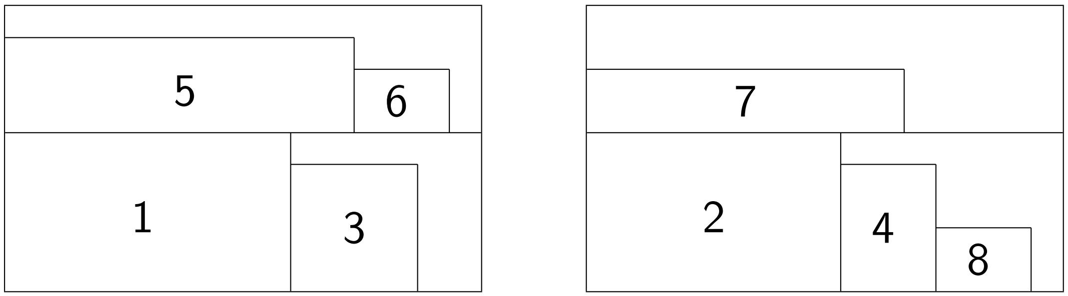
- **Hybrid Best-Fit (HBF,** Berkey and Wang, 1982):
  - *Phase 1*: strip packing through the BFDH strategy;
  - *Phase 2*: 1BPP solved through the *Best-Fit Decreasing* algorithm.
- **Hybrid Next-Fit (HNF,** Frenk and Galambos, 1987):
  - *Phase 1*: strip packing through the NFDH strategy;
  - *Phase 2*: 1BPP solved through the *Next-Fit Decreasing* algorithm.
- Both $O(n \log n)$ time.
- **Floor-Ceiling (FC,** Lodi, Martello, and Vigo, 2000):
  - *ceiling* $=$ horizontal line defined by the top edge of the tallest item packed in the shelf;
  - pack on the shelf floor (left to right) and with the top edge on the ceiling (right to left).
  - $O(n^3)$ time but better experimental performance.



- **Knapsack packing** (Lodi, Martello, and Vigo, 1999):
optimize the packing on the shelves by solving associated knapsack problems ($\mathcal{NP}$-hard).

One-phase algorithms (2BPP)

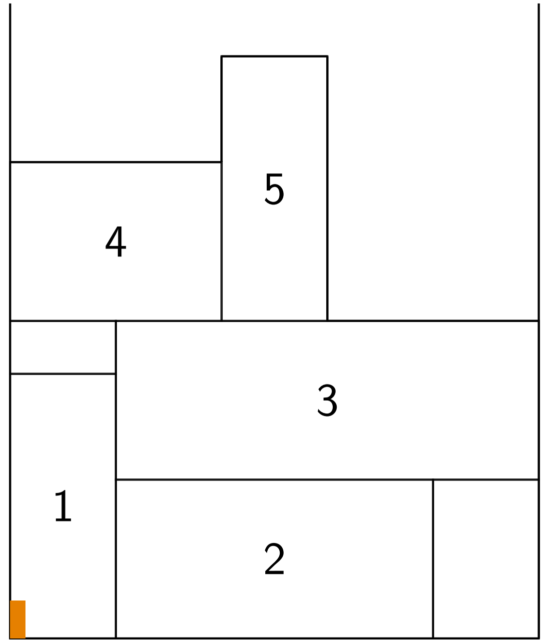
- **Finite Next-Fit (FNF)**:
  - pack the current item in the current shelf of the current bin, if it fits;
  - otherwise, create a new (current) shelf
    either in the current bin (if enough vertical space is available)
    or by initializing a new bin.
- **Finite First-Fit (FFF)**:
  - pack the current item in the lowest shelf of the first bin where it fits;
  - if no shelf can accommodate it, create a new shelf
    either in the first suitable bin
    or by initializing a new bin



- Both $O(n \log n)$ time (Berkey and Wang, 1982).

One-phase algorithms (2BPP and 2SPP)

- Main non-shelf strategy:
  Bottom-Left (BL): pack the current item in the lowest possible position, left justified.



- Complicated $O(n^2)$ time implementation (Chazelle).
- Worst-case performance for the 2SPP (Baker, Coffman, and Rivest, 1980):
  - if no item ordering is used, then BL may be arbitrarily bad;
  - if the items are sorted by nonincreasing width, then

$$BL(I) \leq 3 \cdot OPT(I) \;\; \forall \; I \text{ (tight)}$$

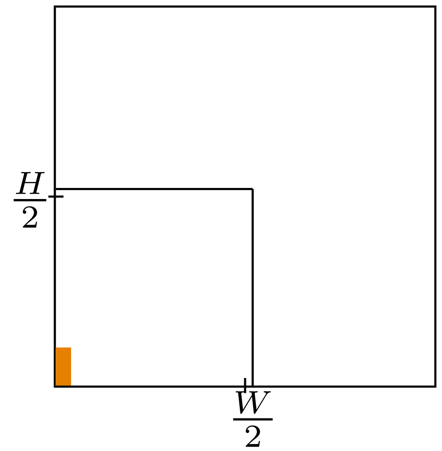
Approximation algorithms and schemes

- Mostly theoretical relevance.
- **Asymptotic approximability**:
  - First asymptotic fully polynomial-time approximation scheme for the **2SPP**: Kenyon and Remila (2000).
  - Asymptotic fully polynomial-time approximation scheme for a restricted version of the **2BPP**: Caprara, Lodi and Monaci (2002).
  - Bansal and Sviridenko (2004): No asymptotic polynomial time approximation scheme (APTAS) can exist for the **2BPP** unless $\mathcal{P} = \mathcal{NP}$.
  - Best result: General framework for approximation algorithms: asymptotic approximation guarantees arbitrarily close to 1.525 for the **2BPP** (Bansal, Caprara and Sviridenko,2006)
- **Absolute approximability**:
  - Zhang (2005): 3-approximation algorithm for the **2BPP**;
  - Harren and van Stee (2009): **2-approximation algorithm for the 2BPP**; best possible polynomial time approximation for **2BPP**, unless $\mathcal{P} = \mathcal{NP}$.
  - Harren, Jansen, Prädel, and van Stee (2014): $\frac{5}{3} + \varepsilon$-approximation algorithm for the **2SPP**.

Lower bounds

Continuous Lower Bound

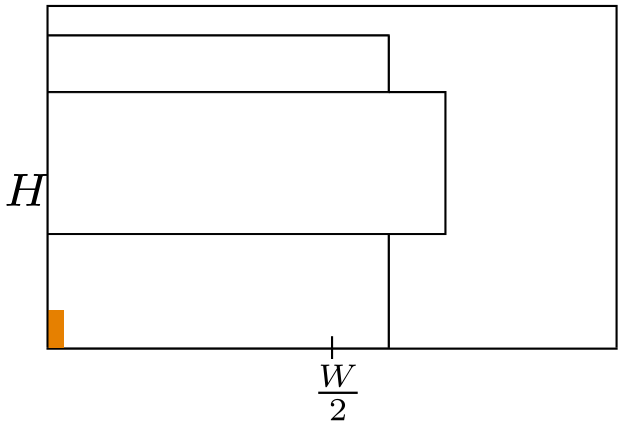
- **2BPP:** $L_0 = \left\lceil \frac{\sum_{j=1}^{n} h_j w_j}{HW} \right\rceil$;
  - $L_0(I) \geq \frac{1}{4} \cdot OPT(I) \;\; \forall\, I$. Tight:



- **2SPP:** $L_0 = \left\lceil \frac{\sum_{j=1}^{n} h_j w_j}{W} \right\rceil$;
  - Arbitrarily bad! $(n = 1,\ w_1 = 1\ h_1 = W\colon\ L_0 = 1,\ z = W)$;
  - better bound: $\overline{L}_0 = \max(L_0, \max_{j=1,\dots,n}\{h_j\})$;
  - $\overline{L}_0(I) \geq \frac{1}{2} \cdot OPT(I) \;\; \forall\, I$ (Tight) (Lodi, Martello, Monaci, Vigo 2003).

Lower bounds from the (one-dimensional) BPP

1. $J^W := \{j \in J : w_j > \frac{1}{2}W\}$



no two items of J^W may be packed side by side into a bin

One dimensional BPP instance: $h_j,\ j \in J^W$, bin capacity H.

$L_1^W =$ **one dimensional lower bound for the BPP instance.**

2. $J^H := \{j \in J : h_j > \frac{1}{2}H\}$: no two items of J^H may be packed one over the other;
   Analogous bound L_1^H

3. $L_1 = \max\{L_1^W, L_1^H\}$.

   No dominance between L_0 and L_1.

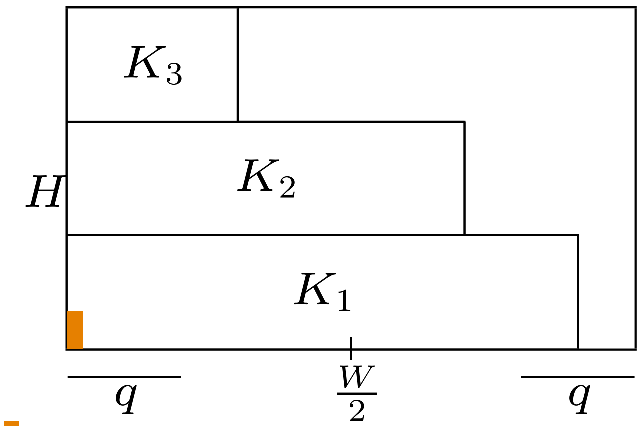
Better lower bounds

- Given an integer value q, $1 \le q \le \frac{1}{2}W$, let

  $K_1 = \{j \in J : w_j > W - q\}$

  $K_2 = \{j \in J : W - q \ge w_j > \frac{1}{2}W\}$

  $K_3 = \{j \in J : \frac{1}{2}W \ge w_j \ge q\}$



- Each item in $K_1 \cup K_2$ needs a separate bin

  $\Rightarrow L_1^W$ = valid lower bound for the items $\in K_1 \cup K_2$;
- $\Rightarrow$ **better bound:** $L_2^W(q) = L_1^W$ + lower bound for the items $\in K_3$.
- No item $\in K_3$ may be packed beside an item $\in K_1 \Rightarrow \ldots$

$$L_2^W(q) = L_1^W + \max\left\{0, \left\lceil \frac{\sum_{j \in K_2 \cup K_3} h_j w_j - (H L_1^W - \sum_{j \in K_1} h_j) W}{HW} \right\rceil \right\}$$

- valid for any $q \Rightarrow L_2^W = \max_{1 \le q \le \frac{1}{2}W} \left\{ L_2^W(q) \right\}$ (computable in $O(n^2)$ time)
- Similar bound $L_2^H \Rightarrow$ **better bound:** $L_2 = \max\{L_2^W, L_2^H\}$; (**dominates both** L_1 **and** L_0).
- **Other lower bounds** derived from the 1BPP, or based on relationships represented through graphs (Caprara, Lodi, and Rizzi (2004); Fekete and Schepers (2004)).

Exact Algorithms

2BPP:

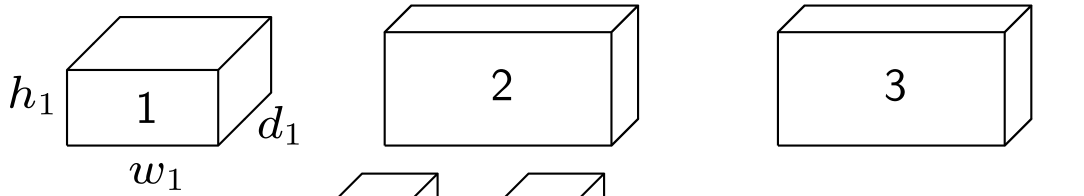
- ***Nested*** **branch-and-bound algorithm** (Martello & Vigo, 1998). Depth-first strategy:
  - *outer tree* (from the 1BPP): items assigned to bins without specifying their actual position: at level k, item k is assigned, in turn, to all active bins and, possibly, to a new bin;
  - *inner tree*: find a a feasible packing (if any) for the items assigned to the bin through
    - approximation algorithms (the packing is feasible if $z = 1$);
    - lower bounds (no packing exists if $LB > 1$); if these fail,
    - enumeration of all possible patterns.
- **Branch-and-price algorithms:**
  Pisinger and Sigurd (2007): decomposition + constraint programming;
- Enumerative approach for the **single bin 2BPP** (Fekete, Schepers, and van der Veen (2007));

2SPP :

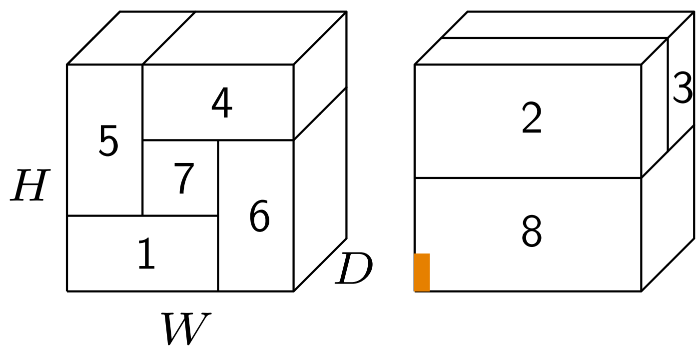
- **Branch-and-bound algorithm:** (Martello, Monaci, and Vigo (2003));
  improvements by Boschetti and Montaletti (2010).
- **2SPP with 90° item rotation** (*Stock Cutting Problem*):
  - branch-and-bound algorithm (Arahori, Imamichi, and Nagamochi (2012);
  - Benders' decomposition (Delorme, Iori, and Martello (2017).

Three-dimensional packing problems (brief outline)

- Given n rectangular-shaped *boxes* with integer **height** h_j, **width** w_j, and **depth** d_j ...



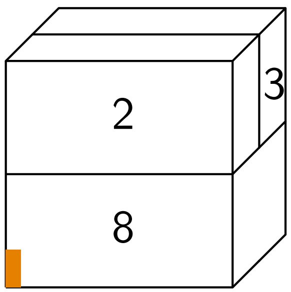
- ... and an unlimited number of identical **rectangular 3-dimensional bins** having **height** H, **width** W and **depth** D, orthogonally pack all the boxes into the minimum number of bins (**Three-Dimensional Bin Packing Problem, 3BPP**):



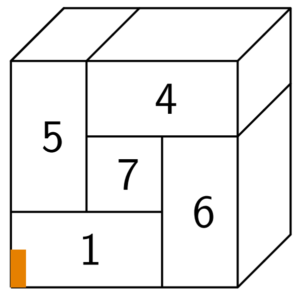
- ... and a single open-ended strip of **width** W, **depth** D, and **infinite height**, orthogonally pack all the boxes by minimizing the height to which the strip is filled (**Three-Dimensional Strip Packing Problem, 3SPP**).

Three-dimensional packing problems (brief outline)

- **Complexity**: obviously NP-hard in the strong sense.
- **Applications:**
  - *Loading*: containers, vehicles (trucks, freight cars), pallets;
  - *Packaging design*: boxes, cases;
  - *Cutting*: foam rubber (arm-chair production).
- **Variants and additional constraints:**
  - Guillotine cuts:



guillotine-cuts



non guillotine-cuts

  - Boxes rotation;
  - Layers;
  - Limit on superposed weights;
  - Stability of the load . . .

End of Part II

III. Real world applications:

Routing problems with loading constraints

- A classical problem: **Capacitated Vehicle Routing Problem (CVRP):**
  find at most K routes of minimum total cost to deliver goods demanded by a set of clients i (each requiring goods of total weight d_i), for a fleet of K vehicles of limited capacity D, based at a central depot.

- **Applications to real world problems** limited by additional constraints:
  - **CVRP**: client demands $=$ total weight of the items to be delivered;
  - **Real-world**: demands $=$ sets of items with a weight and a shape $\Longrightarrow$
  - **Combination of CVRP with loading/packing problems**.

- **2-Dimensional case:**
  - Transportation of rectangular-shaped items that <u>cannot</u> be stacked one on top of the other (big refrigerators, food trolleys, . . .):
  - feasibility of packing on the truck bed;
  - feasibility of the loading and unloading operations.

- **3-Dimensional case:**
  - Transportation of rectangular-shaped boxes that <u>can</u> be stacked one on top of the other;
  - feasibility of box stacking ($\Longleftarrow$ fragility);
  - constraints on the stability of the loading;
  - feasibility of the loading and unloading operations.

- These problems are very difficult to solve in practice:
  branch-and cut algorithms for the exact solution of small-size instances;
  metaheuristics for the approximate solution of instances of realistic size.

CVRP + 2-Dimensional packing

- Complete undirected **graph** $G = (V, E)$: $V = \{0\}$ (depot) $\bigcup$ $\{1, \ldots, n\}$ (clients); edge set $E = \{(i, j)\}$, with c_{ij} = cost of edge (i, j);

- K identical **vehicles**, each having
  - **weight capacity** D;
  - **rectangular loading surface** of width W and height H;

- demand of **client** i $(i = 1, \ldots, n)$:
  - m_i items of total weight d_i;
  - item $I_{i\ell}$ $(\ell = 1, \ldots, m_i)$ has width $w_{i\ell}$ and height $h_{i\ell}$;
  - the items must be orthogonally packed on the loading surface;

- each client must be served by a **single vehicle**;

- let $S(k) \subseteq \{1, \ldots, n\}$ be the set of clients served by vehicle k:
  - **Weight constraint**: total weight $\sum_{i \in S(k)} d_i \leq D$;
  - **Loading constraint**: there must be a feasible (non-overlapping) loading of all the transported items into the $W \times H$ loading area.

CVRP + 2-Dimensional packing (cont'd)

Objective:

- find a partition of the clients into at most K subsets and,
- $\forall$ subset, a route starting and ending at the depot such that
  - all client demands are satisfied;
  - the weight constraint is satisfied;
  - **the loading constraint is satisfied** (feasible packing on the loading area);
  - the total cost of the edges is a minimum.

Two variants:

- **Unrestricted**: no further constraint;
- **Sequential**: the loading of each vehicle must be such that

  when a client is visited, the items of its lot can be downloaded **through a sequence of straight movements (one per item) parallel to the H-edge** of the loading area.

(a) Dashed strip = forbidden area for clients visited after client i
Sequential (b) and non-sequential (c,d) packings

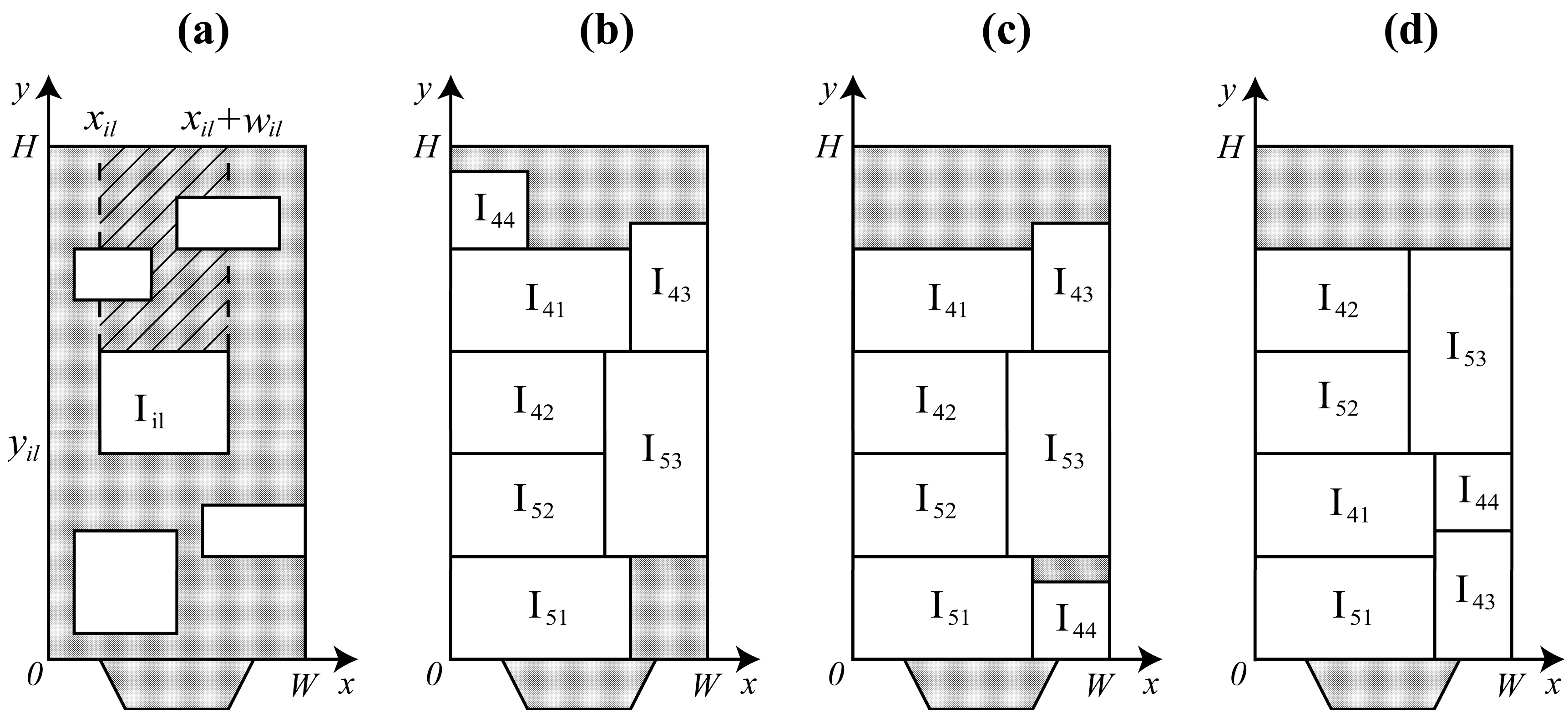
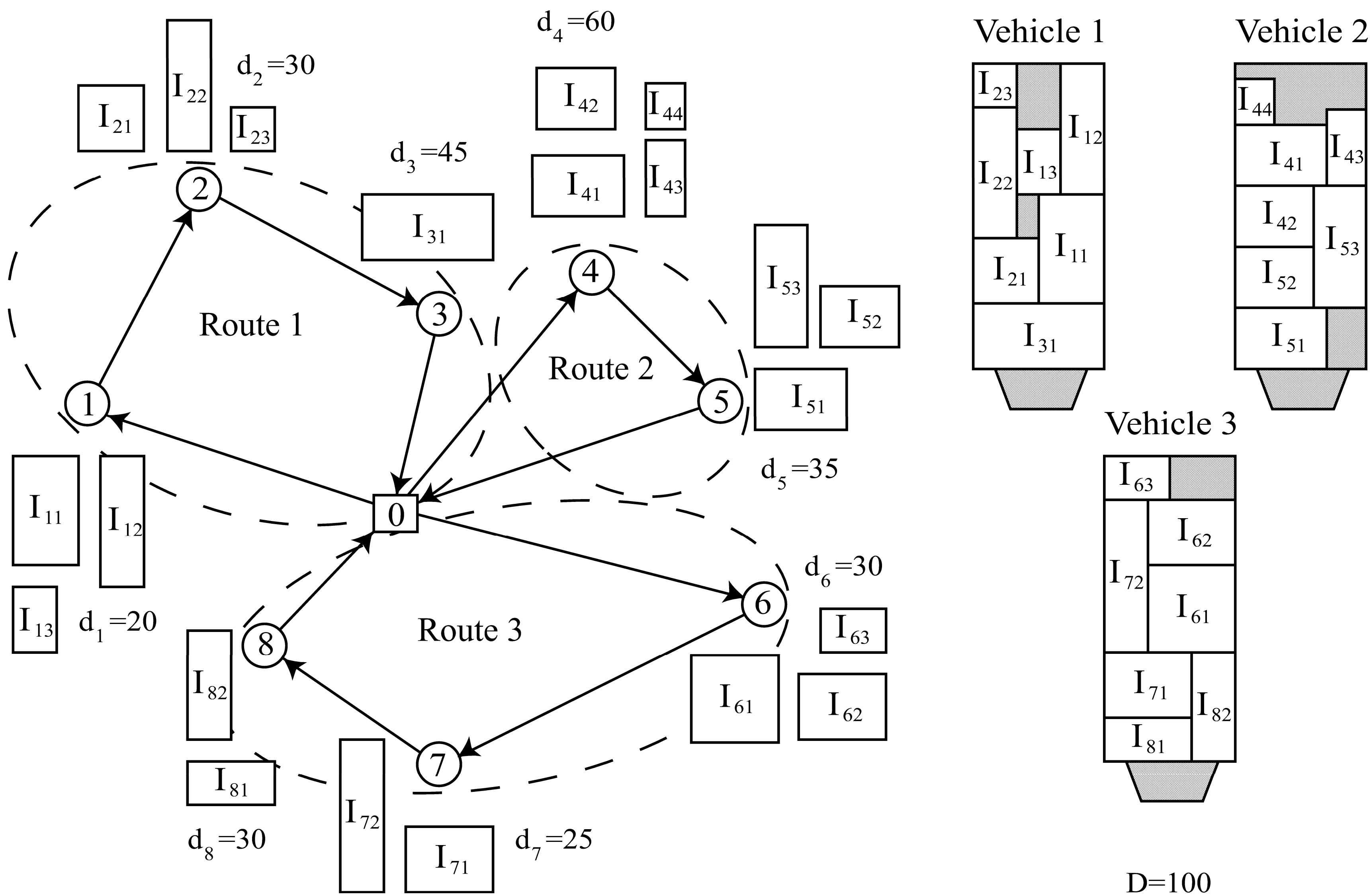


Figura 1: Client 4 visited before Client 5

An instance with 3 vehicles and 8 clients (D = 100)



CVRP + 2-Dimensional packing: example of Tabu search

- **Neighborhood**
  - the algorithm can accept moves producing infeasible tours. Two infeasibilities:
    * *weight-infeasible*: total weight $> D$;
    * *load-infeasible*: height of the loading surface $> H$.
    * Infeasible moves are assigned a proportional *penalty*.
- **Feasibility check of the candidate tour:**
  - *weight-infeasiblity*: immediate;
  - *load-infeasiblity*: NP-hard problem $\Longrightarrow$ **heuristic algorithm** derived from heuristics for 2BP (Lodi, Martello and Vigo 1999), and 2SP (Iori, Martello and Monaci 2003).
- **Tabu search objective function (infeasibilities = penalties):**
  - solution s with $c(k)$ = total edge cost in route k:

$$Z(s) = \sum_{k=1}^{K} c(k) + \alpha q(s) + \beta h(s)$$

  - $q(s)$ = total weight excess;
  - $h(s)$ = total height excess in the infeasible loadings;
  - α and β = self-adjusting parameters.
- Gendreau, Iori, Laporte, Martello (2010).

CVRP + 3-Dimensional packing

Same constraints as the 2-dimensional case, but

- the items are **three-dimensional** boxes;
- the boxes can be rotated by 90° degrees on the horizontal plane;
- some items can be fragile;
- no non-fragile item be placed over a fragile one;
- when boxes are stacked, the supporting surface must be large enough to guarantee stability;
- the loading of each vehicle must be such that

  when a client is visited, the items of its lot can be downloaded

  without shifting the items requested by other clients.

A sequential three-dimensional vehicle loading

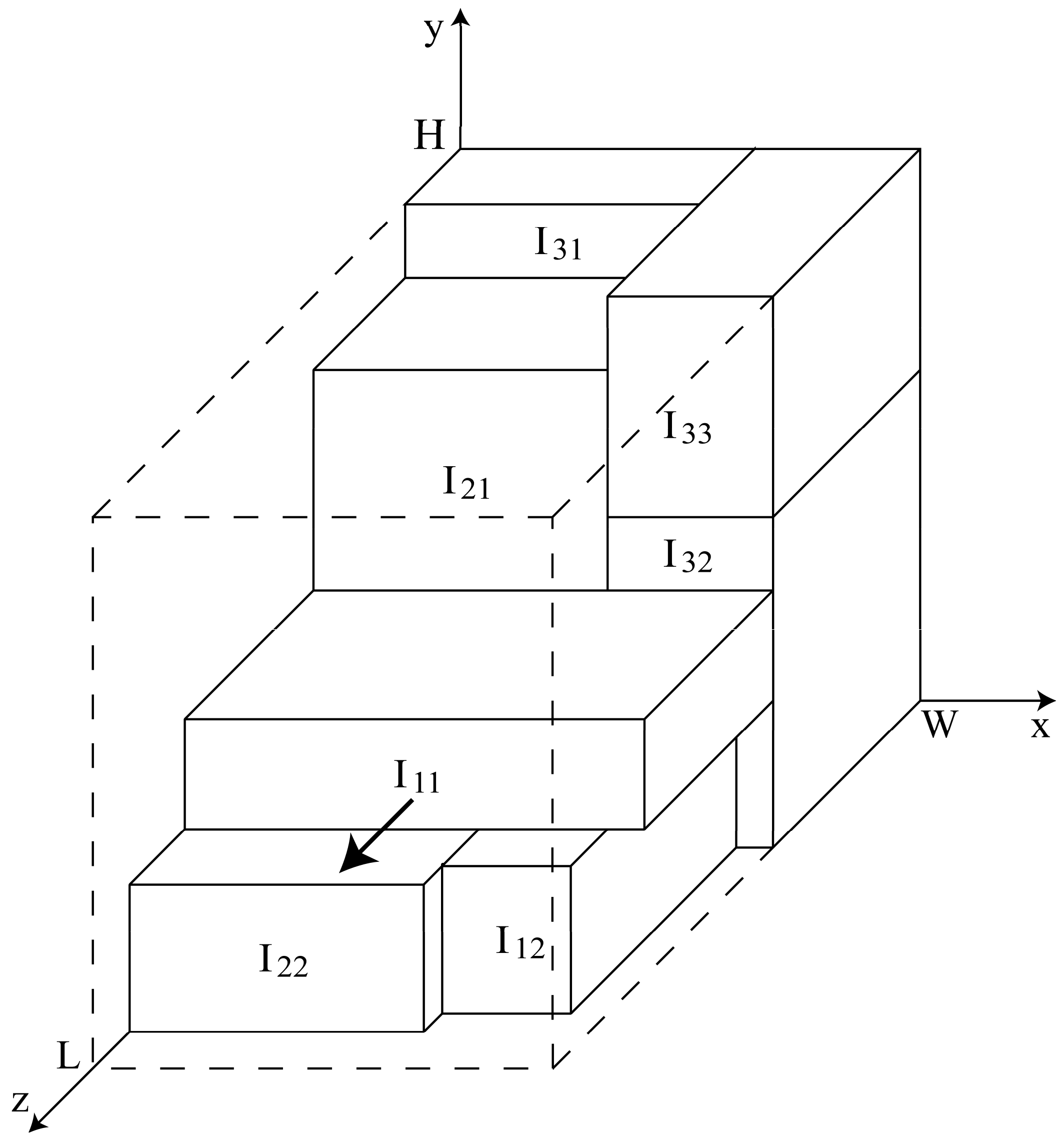
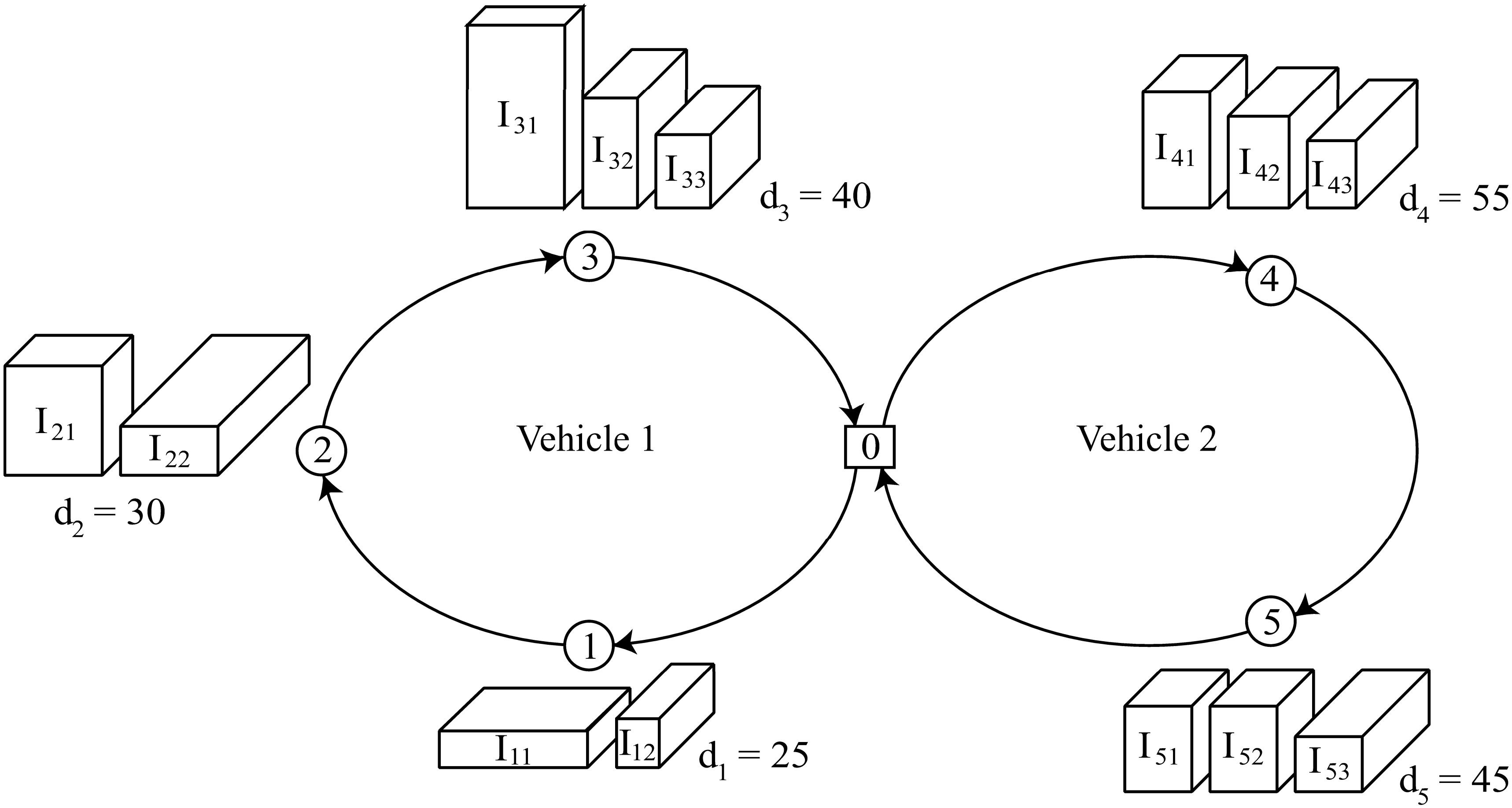


Figura 2: the vehicle is unloaded in the direction of the z axis

An instance with 2 vehicles and 5 clients (D = 100)



Other routing problems with loading constraints

- **Basis: Traveling Salesman Problem with Pickup and Delivery (TSPPD):**

  single vehicle must visit a set of customers, each associated with an origin location where some items must be picked up, and a destination location where such items must be delivered;

  find a shortest Hamiltonian cycle through all locations while ensuring that the pickup of any given request is performed before the corresponding delivery.

  - **TSPPD and LIFO loading:**
    pickups and deliveries must be performed in LIFO order (vehicles with a single access point);
  - **TSPPD and FIFO loading:**
    pickups and deliveries must be performed in FIFO order (AGVs that load items on one end and unload them at the other end);
- **CVRP + 2-dimensional loading + pickup and delivery constraints**;
- **3-dimensional container loading problems with multi-drop constraints** (special sequences);
- **CVRP with time windows and three-dimensional container loading**;
- **CVRP with pickup and delivery, delivery due dates and 3-dimensional loading**:
  auto-carrier transportation problem;
- **TSP with pickup and delivery and handling costs** (when the loading is not sequential);
- **...**

An industrial case (CVRP + 3-Dimensional packing)

- Italian company (furniture for bedrooms);
- fleet of private-owned vehicles paid per mileage;
- demands: three-dimensional rectangular items (to be assembled);
- identical vehicles (standard ISO containers);
- time windows neglected;
- typical solutions: one-day tours or multi-day tours;
- volumes between 1% and 4% of the vehicle volume;
- heights between 10% and 50% of the vehicle height.

Transportation Science 40(3), pp. 342–350, ©2006 INFORMS

Figure 3 Distribution of Clients in Italy (Instance F01)

					Tabu s. **1** hour		Tabu s. **10** hours		Tabu s. **24** hours	
Instance	n	M	K	Greedy	z	sec_z	z	sec_z	z	sec_z
F01	44	141	4	7711	3723	2839.4	3694	32133.9	3694	32133.9
F02	49	152	4	7167	4182	1993.8	4182	1993.8	3941	86046.8
F03	55	171	4	6111	3674	3478.5	3650	31776.5	3650	31776.5
F04	57	159	4	7059	4686	2520.5	4543	5049.7	4509	5995.1
F05	64	181	4	7408	7235	2366.3	6886	33917.9	6241	75441.1
Average				7091	4700	2639.7	4591	20974.3	4407	46278.7

M = total number of items to deliver.

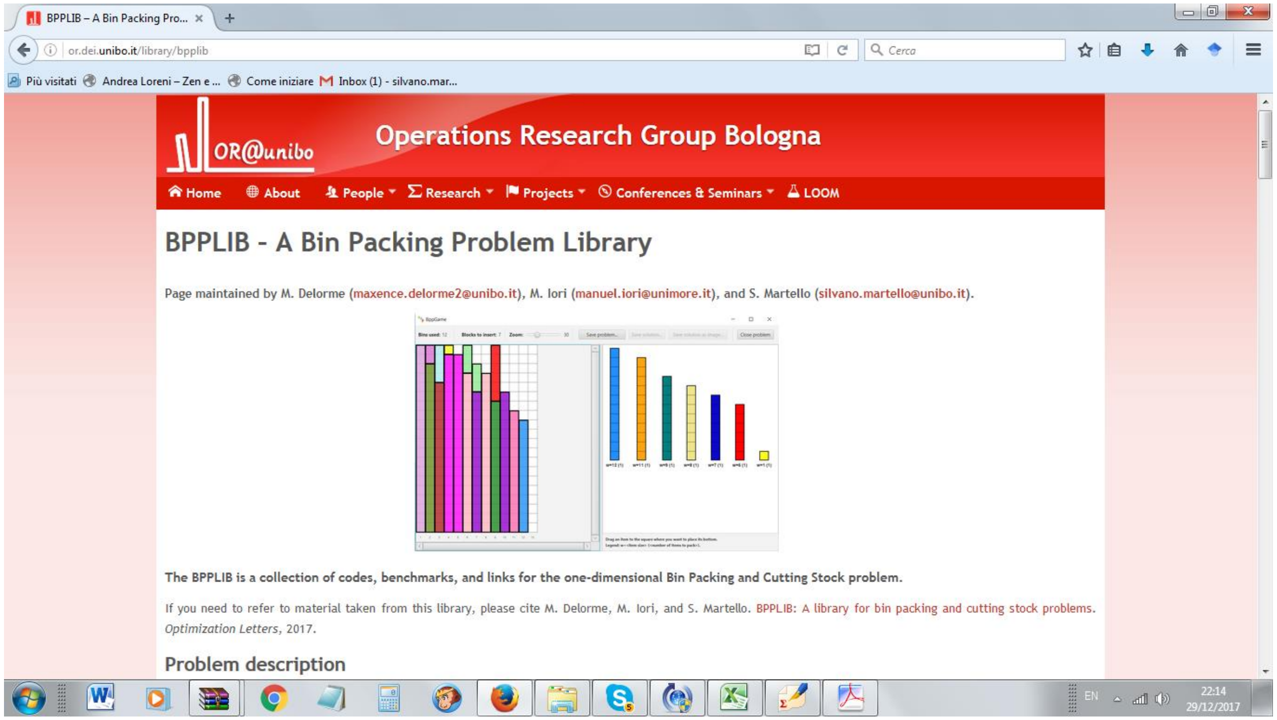
End of Part III

IV. Interactive visual solvers for one- and two-dimensional packing

One-Dimensional Bin Packing Problem

The BPPLIB

- http://or.dei.unibo.it/library/bpplib

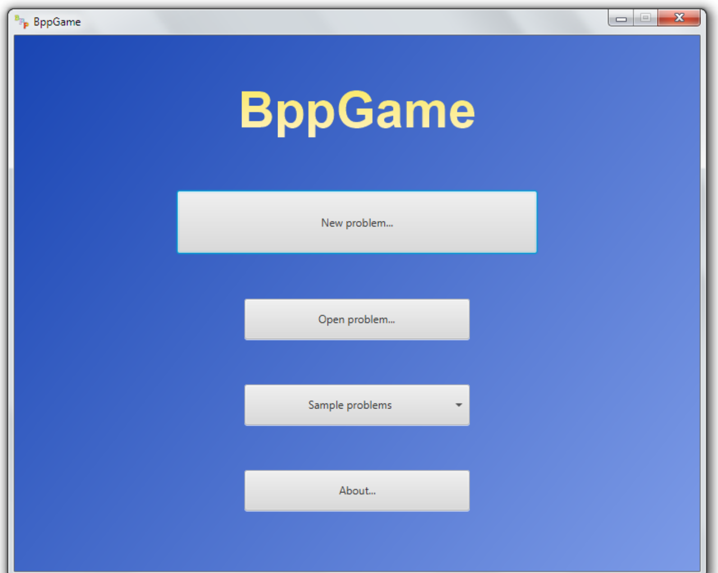


The BPPLIB

- surveys;
- computer codes:
  - branch-and-bound;
  - branch-and-price;
  - pseudo-polynomial formulations (CPLEX or SCIP solver);
- benchmarks;
- problem generators;
- bibliography (BibTeX file);
- **BppGame**: An interactive visual solver.

BppGame

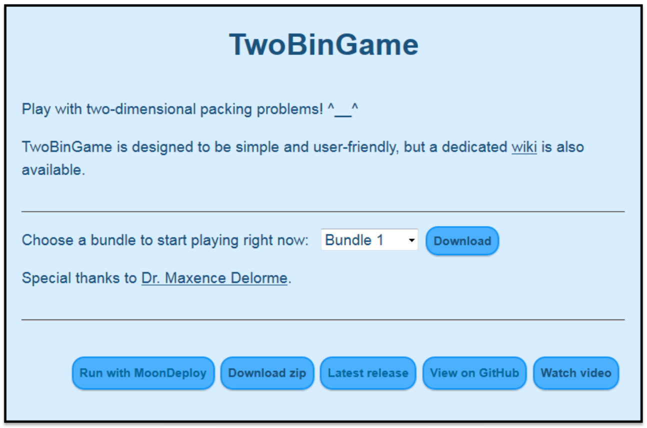
- Go to `http://www.or.deis.unibo.it/staff_pages/martello/Tools/T.html`;
- Click “Bin Packing Problem” to download BppGame-1.3.2.zip;
- save the zipfile and extract its contents in a folder;
- access the folder and click folder “bin”;
- double click “BppGame.bat” (Windows) or “BppGame” (Linux);



- click “Sample problems”;
- select the problems in sequence, using “Zoom” to adjust the screen.

Two-Dimensional Packing Problems

- Go to `http://www.or.deis.unibo.it/staff_pages/martello/Tools/T.html`;
- click `http://gianlucacosta.info/TwoBinPack/`;



- in the "TwoBinGame" box, click "Download zip" to download TwoBinGame-3.1.zip;
- save the zipfile and extract its contents in a folder;
- access the folder and click "bin";
- double click "TwoBinGame.bat" (Windows) or "TwoBinGame" (Linux);
- in the new "TwoBinGame" box, click "Demo problem" to see how it works. (The "Nickname" field can be left empty. Note that, as rotation is allowed, the two yellow blocks appear twice.);
- in the old "TwoBinGame" box, click "Download" to download Bundle 1. Save it to the folder;
- in TwoBinGame, click "Play" → "Open local file" and select the bundle (Problems_1.tbprob2).

Essential Bibliography (Surveys and books)

I. One-dimensional bin packing problem

- M. Delorme, M. Iori, S. Martello (2016). Bin Packing and Cutting Stock Problems: Mathematical Models and Exact Algorithms, *European Journal of Operational Research*.
- E.G. Coffman, Jr, J. Csirik, G. Galambos, S. Martello, D. Vigo (2013). Bin packing approximation algorithms: Survey and classification *Handbook of Combinatorial Optimization*, Springer.
- G. Wäscher, H. Haußner, and H. Schumann (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*.
- H. Kellerer, U. Pferschy, D Pisinger (2004). *Knapsack problems*, Springer, Berlin.
- S. Martello, P. Toth (1990). *Knapsack Problems: Algorithms and Computer Implementations* (Ch. 6), John Wiley & Sons, Chichester-New York. **Free download @ my home page**.

II. Two-dimensional bin packing problems

- A. Lodi, S. Martello, M. Monaci, D. Vigo (2014). Two-dimensional bin packing problems. In *Paradigms of Combinatorial Optimization: Problems and New Approaches*, ISTE and John Wiley & Sons.
- A. Lodi, S. Martello, M. Monaci (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research*. (> 800 citations on Google Scholar)
- G. Scheithauer (2018). *Introduction to Cutting and Packing Optimization Problems*, Springer, Berlin.

Essential Bibliography (Surveys and books)

III. Routing problems with loading constraints

- H. Pollaris, K. Braekers, A. Caris, G.K. Janssens, S. Limbourg (2015). Vehicle routing problems with loading constraints: State-of-the-art and future directions. *OR Spektrum*.
- M. Iori, S. Martello (2013). An annotated bibliography of combined routing and loading problems. *Yugoslav Journal of Operations Research*
- M. Iori, S. Martello (2010). Routing problems with loading constraints. *TOP*.

- **Hot area:** ~ 100 references since 2006.

IV. Interactive visual solvers for packing problems

- M. Delorme, M. Iori, S. Martello (2018). BPPLIB: A library for bin packing and cutting stock problems. *Optimization Letters* (to appear).
- G. Costa, M. Delorme, M. Iori, E. Malaguti, S. Martello (2017). Training software for orthogonal packing problems. *Computers and Industrial Engineering*.