Bin packing problems

François Clautiaux

Paris, February,2024

Introduction

The bin packing problem (BPP) is one of the first studied combinatorial optimization problems

Packing problems

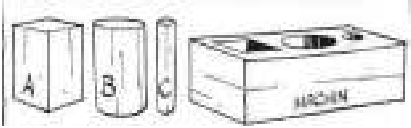
Given a set of **containers** with a limited capacity, find the minimum number of containers needed to pack a set of **items** in such a way that **some geometric constraints are satisified**.

Two challenges ROADEF (one on cutting, one on packing), one challenge ESICUP (packing)

The bin packing problem (BPP) is one of the first studied combinatorial optimization problems

Packing problems

Given a set of **containers** with a limited capacity, find the minimum number of containers needed to pack a set of **items** in such a way that **some geometric constraints are satisified**.



Two challenges ROADEF (one on cutting, one on packing), one challenge ESICUP (packing)

EWG ESICUP CUTTING AND PACKING

Introduction

The bin packing problem (BPP) is one of the first studied combinatorial optimization problems

Packing problems

Given a set of **containers** with a limited capacity, find the minimum number of containers needed to pack a set of **items** in such a way that **some geometric constraints are satisified**.

Two challenges ROADEF (one on cutting, one on packing), one challenge ESICUP (packing)

EWG ESICUP CUTTING AND PACKING

Cutting and packing

Cutting and packing are different activities

But their combinatorial structure are the same

Cutting/packing problems occur when the capacity constraints are the main constraints of the problem

Main differences: side constraints

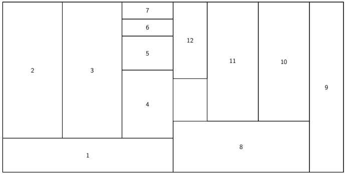


Figure: A packing problem

Cutting and packing

Cutting and packing are different activities

But their combinatorial structure are the same

Cutting/packing problems occur when the capacity constraints are the main constraints of the problem

Main differences: side constraints

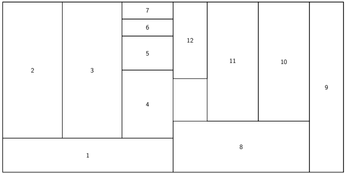


Figure: A packing problem

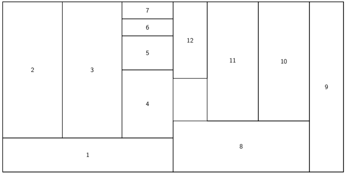
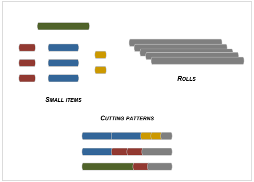
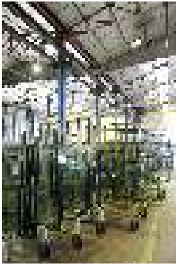


Figure: A cutting problem

Some packing problems





Bingo card of this presentation

an exponential model that pretends to be compact	complicated algorithm for "simple" problems	simple algorithms for complicated problems
a $\frac{239091}{148304}$ performance ratio	an irritating 2D packing problem	the new complexity concept of *combo-hardness*
fake news (about Kantorovich) debunked	an unexpected 2D circle packing problem	a donkey

Bingo card of this presentation

an exponential model that pretends to be compact	complicated algorithm for "simple" problems	simple algorithms for complicated problems
a $\frac{239091}{148304}$ performance ratio	an irritating 2D packing problem	the new complexity concept of *combo-hardness*
fake news (about Kantorovich) debunked	an unexpected 2D circle packing problem	a donkey (*)

(*) a real one, not a new metaheuristic

The mother of all C&P problems

The **knapsack problem** is weakly NP-hard (one of the 21 Karp's problems).

Used in all OR / combinatorial optimization courses.

Few breakthroughs in the last 20 years!

Best available method/software: the combo method of Martello et al., 1999 (linear relaxation, dynamic programming, branch-and-bound).

But MIP solvers are now catching up!

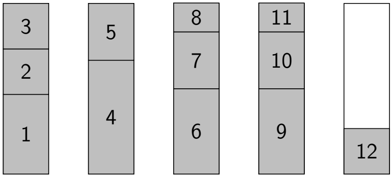
A trend of research: finding "hard" KP instances (= hard for combo) *[Smith-Miles et al., 2021]*

Bin packing and cutting stock

Bin packing Problem

Given a list of orders $i = 1, \ldots, n$, each having a size $c_i \in \mathbb{R}_+$, and an integer value C (roll size), find the minimum number of cutting rolls to pack all items in such a way that the sum of the item sizes in one roll is always smaller than C.

The problems is NP-complete (NP-hard in the strong sense)



Bin packing and cutting-stock

Cutting-Stock Problem

Given a list of orders $i = 1, \ldots, n$, each having a size $c_i \in \mathbb{R}_+$ **and a demand** d_i, and an integer value C (roll size), find the minimum number of cutting rolls **to satisfy the item demand** in such a way that the sum of the item sizes in one roll is always smaller than C.

The problem is NP-hard in the strong sense.

Only known to be NP-complete since [Eisenbrand and Shmonin, 2006]

The problem becomes a **bin packing problem** if it is "reasonable" to specify the size of each item individually (even if they have the same size).

Classical heuristics

Classical heuristics are ordered-based algorithms

Initially, an empty bin is created. At each step, the next item is selected and packed in a bin. A new bin may be created et each step.

- next fit: choose the current (last) bin
- first fit : choose the first possible bin
- best fit : choose the bin with the largest remaining capacity
- worst fit : choose the bin with the smallest remaining capacity

The race to the ratio

Algorithm	Guarantee	Complexity
Next-fit	2	$O(n)$
First-fit	1.7	$O(n \log n)$
Refined first-fit	5/3	$O(n \log n)$
Harmonic-k	1.69103	$O(n \log n)$
Refined harmonic	$373/228 = 1.63597$	$O(n)$
Modified Harmonic	$538/33 = 1.61562$	
Modified Harmonic 2	$239091/148304 = 1.61217$	
Harmonic ++	1.58889	

Introduction

One dimensional bin packing problem

Classical results

Integer programming formulations

Two-dimensional cutting-stock problems

Three-dimensional problems

Recent trends in cutting and packing

Conclusion

Classical model for CSP

First model for the cutting-stock problem, due to Kantorovich.

"Compact" formulation

$$\min \sum_{j \in \mathcal{M}} y_j$$
$$\sum_{i \in \mathcal{N}} c_i x_{ij} \leq C y_j, \quad j \in \mathcal{M}$$
$$\sum_{j \in \mathcal{M}} x_{ij} \geq d_i, \quad i \in \mathcal{N}$$
$$x_{ij} \in \mathbb{N}, \quad i \in \mathcal{N}; j \in \mathcal{M}$$
$$y_j \in \{0, 1\}, \quad j \in \mathcal{M}$$

y_j : 1 if bin j is open, 0 otherwise

x_{ij} : how many times item i is cut from bin j

m : an upper bound on the number of bins needed

$\mathcal{N} = 1, \dots, n$ and $\mathcal{M} = 1, \dots, m$.

Classical model for CSP

First model for the cutting-stock problem, due to ~~Kantorovich~~ .

"Compact" formulation

$$\min \sum_{j \in \mathcal{M}} y_j$$

$$\sum_{i \in \mathcal{N}} c_i x_{ij} \le C y_j, \quad j \in \mathcal{M}$$

$$\sum_{j \in \mathcal{M}} x_{ij} \ge d_i, \quad i \in \mathcal{N}$$

$$x_{ij} \in \mathbb{N}, \quad i \in \mathcal{N}; j \in \mathcal{M}$$

$$y_j \in \{0,1\}, \quad j \in \mathcal{M}$$

y_j : 1 if bin j is open, 0 otherwise

x_{ij} : how many times item i is cut from bin j

m : an upper bound on the number of bins needed

$\mathcal{N} = 1, \dots, n$ and
$\mathcal{M} = 1, \dots, m$.

Classical model for CSP

First model for the cutting-stock problem, due to Kantorovich.

"Compact" formulation

$$\min \sum_{j \in \mathcal{M}} y_j$$

$$\sum_{i \in \mathcal{N}} c_i x_{ij} \leq C y_j, \quad j \in \mathcal{M}$$

$$\sum_{j \in \mathcal{M}} x_{ij} \geq d_i, \quad i \in \mathcal{N}$$

$$x_{ij} \in \mathbb{N}, \quad i \in \mathcal{N}; j \in \mathcal{M}$$

$$y_j \in \{0, 1\}, \quad j \in \mathcal{M}$$

Bad linear relaxation.
always equal to $\sum_{i=1}^{n} \frac{d_i c_i}{C}$.

Asymptotic worst case: $1/2$

Classical model for CSP

First model for the cutting-stock problem, due to Kantorovich.

"Compact" formulation

$$\min \sum_{j \in \mathcal{M}} y_j$$

$$\sum_{i \in \mathcal{N}} c_i x_{ij} \leq C y_j, \quad j \in \mathcal{M}$$

$$\sum_{j \in \mathcal{M}} x_{ij} \geq d_i, \quad i \in \mathcal{N}$$

$$x_{ij} \in \mathbb{N}, \quad i \in \mathcal{N}; j \in \mathcal{M}$$

$$y_j \in \{0,1\}, \quad j \in \mathcal{M}$$

Bad linear relaxation.
always equal to $\sum_{i=1}^{n} \frac{d_i c_i}{C}$.

Asymptotic worst case: $1/2$

Is it really compact?

$|\mathcal{M}|$ is the number of bins in the solution $\implies \Theta(\sum_{i=1}^{n} d_i)$

Size of the instance:
$O(\sum_{i=1}^{n} \log d_i)$.

Pseudo-polynomial model!

Classical model for CSP

First model for the cutting-stock problem, due to Kantorovich.

"Compact" formulation

$$\min \sum_{j \in \mathcal{M}} y_j$$

$$\sum_{i \in \mathcal{N}} c_i x_{ij} \leq C y_j, \quad j \in \mathcal{M}$$

$$\sum_{j \in \mathcal{M}} x_{ij} \geq d_i, \quad i \in \mathcal{N}$$

$$x_{ij} \in \mathbb{N}, \quad i \in \mathcal{N}; j \in \mathcal{M}$$

$$y_j \in \{0, 1\}, \quad j \in \mathcal{M}$$

Bad linear relaxation.
always equal to $\sum_{i=1}^{n} \frac{d_i c_i}{C}$.

Asymptotic worst case: $1/2$

Is it really compact?

$|\mathcal{M}|$ is the number of bins in the solution $\implies \Theta(\sum_{i=1}^{n} d_i)$

Size of the instance:
$O(\sum_{i=1}^{n} \log d_i)$.

Pseudo-polynomial model!

Many symmetries.

A polynomial size version of (SC)

Theorem (Eisenbrand and Shmonin, 2006)

The number of different cutting patterns in an optimal solution is no larger than $\mathrm{LD} = 2\sum_{i=1}^{n} \log d_i$.

Remark

A pattern cannot be used more than $d_{\max} = \max\{d_i : i \in I\}$

- binary variable y_{jk} means that bin number j is replicated 2^k times
- integer variable x_{ijk} is the number of times i is used in each bin j replicated 2^k times

For the sake of simplicity, assume that all numbers are powers of 2.

A polynomial size version of (SC)

A smaller model

$$\min \sum_{j=1}^{\mathrm{LD}} \sum_{k=0}^{\log d_{\max}} 2^k d_{jk}$$

$$\sum_{i \in \mathcal{N}} c_i x_{ijk} \le C y_{jk}, \quad j = 1, \dots, \mathrm{LD};\, k = 0, \dots, \log d_{\max}$$

$$\sum_{j=1}^{\mathrm{LD}} \sum_{k=0}^{\log d_{\max}} 2^k x_{ijk} = d_i, \quad i \in \mathcal{N}$$

$$x_{ijk} \in \mathbb{N}, \quad i \in \mathcal{N};\, j = 1, \dots, \mathrm{LD};\, k = 0, \dots, \log d_{\max}$$

$$y_{jk} \in \{0, 1\}, \quad j = 1, \dots, \mathrm{LD};\, k = 0, \dots, \log d_{\max}$$

Removing symmetries from (SC)

For the binary case, variables x_{ij}: i is packed in a bin whose item of smallest index is j (x_{ii} when i is the smallest item in its bin).

Representative model

$$\min \sum_{i=\mathcal{M}} x_{ii}$$

$$\sum_{i=j+1,n} c_i x_{ij} \leq (C - c_i) x_{jj}, \quad j = 1, \ldots, n-1$$

$$\sum_{j=1,\ldots,i} x_{ij} = 1, \quad i = 1, \ldots, n$$

$$x_{ij} \in \mathbb{N}, \quad i = 1, \ldots, n; j = 1, \ldots, i$$

Removing symmetries from (SC)

For the binary case, variables x_{ij}: i is packed in a bin whose item of smallest index is j (x_{ii} when i is the smallest item in its bin).

Representative model

$$\min \sum_{i=\mathcal{M}} x_{ii}$$

$$\sum_{i=j+1,n} c_i x_{ij} \leq (C - c_i) x_{jj}, \quad j = 1, \dots, n-1$$

$$\sum_{j=1,\dots,i} x_{ij} = 1, \quad i = 1, \dots, n$$

$$x_{ij} \in \mathbb{N}, \quad i = 1, \dots, n; j = 1, \dots, i$$

Not always better (sometimes $m << n$).

Gilmore-Gomory model for CSP

P: set of all possible *patterns* (valid set of items for one bin).
a_i^p: number of times item i appears in p.

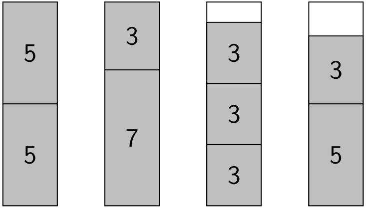
$\forall p \in P$, λ_p indicates the number of times p is used in the solution.

Gilmore-Gomory model

$$\min \sum_{p \in P} \lambda_p$$

$$\text{s.t.} \quad \sum_{p \in P} a_i^p \lambda_p \geq d_i \quad \forall i \in \mathcal{N}$$

$$\lambda_p \in \mathbb{N} \quad \forall p \in P$$



$\lambda_1 = 1$ $\lambda_2 = 2$ $\lambda_3 = 1/3$ $\lambda_4 = 0$

Column generation

Master problem

$$\min \sum_{p \in P} \lambda_p$$
$$\text{s.t.} \quad \sum_{p \in P} a_i^p \lambda_p \geq d_i, \quad i \in \mathcal{N} \quad (\pi)$$
$$\lambda_p \in \mathbb{R}_+, \quad p \in P$$

Dual

$$\max \sum_{i \in \mathcal{N}} d_i \pi_i$$
$$\text{s.t.} \quad \sum_{i \in \mathcal{N}} a_i^p \pi_i \leq 1, \quad p \in P$$
$$\pi_i \geq 0, \quad i \in \mathcal{N}$$

Variable of minimum reduced cost:
$\min\{1 - \pi_i a_i : \sum_{i \in \mathcal{N}} c_i a_i \leq C; a_i \in \mathbb{N}^n\}$.

Column generation

Master problem

$$\min \sum_{p \in P} \lambda_p$$

$$\text{s.t.} \sum_{p \in P} a_i^p \lambda_p \geq d_i, \quad i \in \mathcal{N} \quad (\pi)$$

$$\lambda_p \in \mathbb{R}_+, \quad p \in P$$

Dual

$$\max \sum_{i \in \mathcal{N}} d_i \pi_i$$

$$\text{s.t.} \sum_{i \in \mathcal{N}} a_i^p \pi_i \leq 1, \quad p \in P$$

$$\pi_i \geq 0, \quad i \in \mathcal{N}$$

Variable of minimum reduced cost:
$\min\{1 - \pi_i a_i : \sum_{i \in \mathcal{N}} c_i a_i \leq C; a_i \in \mathbb{N}^n\}$.

Where are the convexity constraints ?

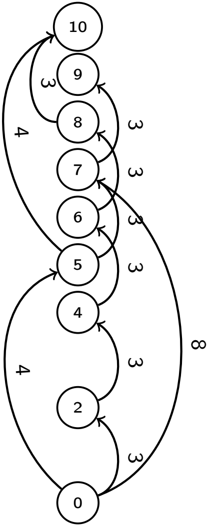
Worst model to explain Dantzig-Wolfe decomposition!

Pricing subproblem (unbounded)

$$\max \sum_{i\in\mathcal{N}} \pi_i a_i$$
$$\sum_{i\in\mathcal{N}} c_i a_i \leq C$$
$$a_i \in \mathbb{N}, i \in \mathcal{N}$$

$$\alpha(c) = \max_{i\in\mathcal{N}: c\leq c_i}\{p_i + \alpha(c_i + c)\}$$

Better to use Combo in general, but useful for my talk.



Obtaining integer solutions

Classical branching in CG: using the original variables.

Cutting-stock, subproblems are aggregated, how can we get x_{ij} values from the λ_p variables?

Solution: branching on the subproblem's variables.

Branching decision : number of times item i is chosen before (or after) some position in the bin.

Only changes the cost of the corresponding arcs

Obtaining integer solutions

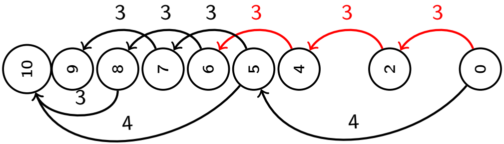
Classical branching in CG: using the original variables.

Cutting-stock, subproblems are aggregated, how can we get x_{ij} values from the λ_p variables?

Solution: branching on the subproblem's variables.

Branching decision : number of times item i is chosen before (or after) some position in the bin.

Only changes the cost of the corresponding arcs



MIRUP property

The linear relaxation of this model is always excellent.

MIRUP conjecture: $\lceil LP \rceil \geq OPT - 1$?

This explains partially why so many papers were devoted to techniques for accelerating the column generation procedure for cutting stock

Lagrangian bound for CSP

For any vector $\pi \in R_+^n$, the lagrangian lower bound is valid.

$$L(\pi) = \sum_{i \in \mathcal{N}} d_i \pi_i + m * \{ \min y - \sum_{i \in \mathcal{N}} x_i \pi_i$$

$$\sum_{i \in \mathcal{N}} c_i x_i \leq Cy$$

$$x_i \in \mathbb{N}, \quad i \in \mathcal{N}$$

$$y \in \{0, 1\} \quad \}$$

$\sum_{i \in \mathcal{N}} d_i \pi_i^* \quad \rightarrow$ optimal value of the RMP
minimization problem with $\pi^* \quad \rightarrow$ optimal value of the subproblem

OPT(RMP) + m SP is a valid lower bound

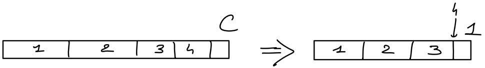
Dual solutions

Dual problem

$$\max \sum_{i \in \mathcal{N}} d_i \pi_i$$

$$\sum_{i \in \mathcal{N}} a_{ip} \pi_i \leq 1, \quad \forall p \in P$$

$$\pi_i \geq 0, \quad \forall i \in \mathcal{N}$$



$$c_1 + c_2 + c_3 + c_4 \leqslant C$$

$$\pi_1 + \pi_2 + \pi_3 + \pi_4 \leqslant 1$$

Farley's bound

SP$= \min_{p \in P} \{1 - \sum_{i \in \mathcal{N}} \bar{\pi}_i a_{ip}\}$

$\implies \forall p \in P,\ 1 - \sum_{i \in \mathcal{N}} \bar{\pi}_i a_{ip} \geq$ SP

$\implies \forall p \in P,\ \sum_{i \in \mathcal{N}} \frac{\bar{\pi}_i}{1-SP} a_{ip} \leq 1$

$\implies \frac{\bar{\pi}}{1-SP}$ is dual-feasible $\equiv OPT(RMP)/(1-SP)$ is a valid lower bound

Arc-flow model

Another way of producing an extended formulation for CSP is to reformulate the knapsack constraints are shortest path constraints.

Possible since knapsack feasibility can be computed recursively through dynamic programming.

Arc-flow model [Carvalho, 1999]

The model is based on a graph $G = (V, A)$.

V : positions in the bin $\{0, 1, \ldots, W\}$
A : set of arcs (packing an item at a given position)

(i, j) : packing at position i an item of size $j - i$
$(i, i+1)$: packing nothing at position i (to allow solutions with slack)

$A(i)$: set of arcs "covering" item i

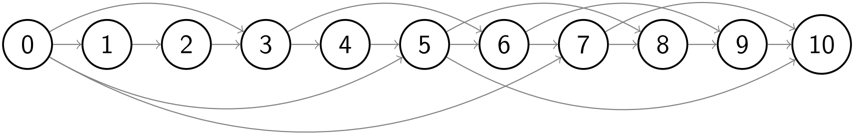
Variables

- z : flow value = number of paths = number of bins
- λ_a : value of the flow through arc a

Arc-flow formulation: model

$$
\begin{aligned}
&\min z \\
\text{s.c.} \quad &\sum_{a \in A(i)} \lambda_a \geq d_i, \qquad \forall i \in I \\
\sum_{a \in \delta^+(v)} \lambda_a - \sum_{a \in \delta^-(v)} \lambda_a = &\begin{cases} z, & \text{if v=0} \\ 0, & \text{if v=1,...,W-1} \\ -z & \text{if v=W} \end{cases} \\
&z \in \mathbb{N} \\
&\lambda_a \in \mathbb{N}, \qquad \forall a \in A
\end{aligned}
$$

Arc-flow formulation: example



Data

Bin of size 10

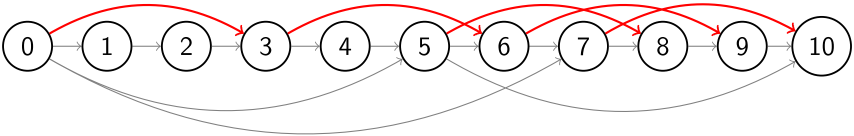
- 2 items of size 7
- 3 items of size 5
- 7 items of size 3

Objective : minimize the value of the flow

Constraints :

- flow conservation
- demand for each item

Arc-flow formulation: example



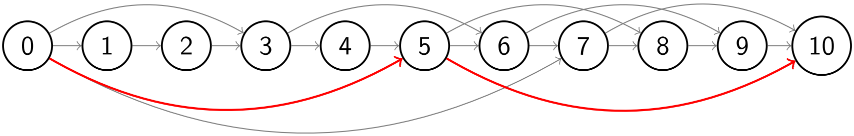
Data

Bin of size 10

- 2 items of size 7
- 3 items of size 5
- 7 items of size 3

Arcs covering items of size 3

Arc-flow formulation: example



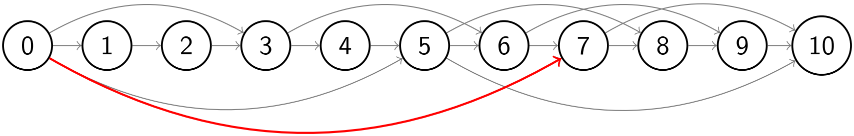
Data

Bin of size 10

- 2 items of size 7
- 3 items of size 5
- 7 items of size 3

Arcs covering items of size 5

Arc-flow formulation: example



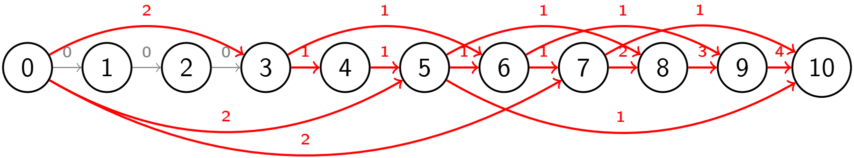
Data

Bin of size 10

- 2 items of size 7
- 3 items of size 5
- 7 items of size 3

Arcs covering items of size 7

Arc-flow formulation: example



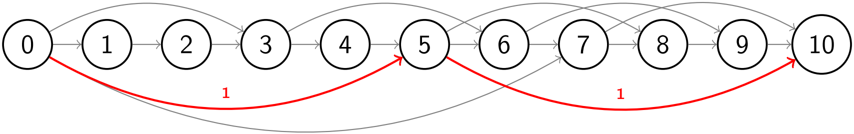
Data

Bin of size 10

- 2 items of size 7
- 3 items of size 5
- 7 items of size 3

A solution for the problem

Arc-flow formulation: example



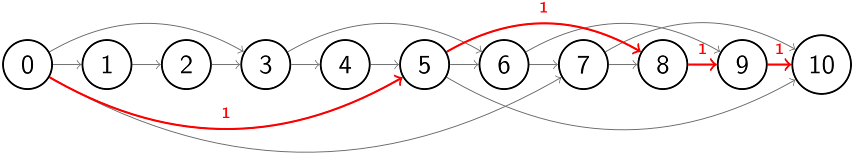
Data

Bin of size 10

- 2 items of size 7
- 3 items of size 5
- 7 items of size 3

Configuration $5, 5$

Arc-flow formulation: example



Data

Bin of size 10

- 2 items of size 7
- 3 items of size 5
- 7 items of size 3

Configuration $5, 3$

From SC to Arc-Flow

Constraints $\sum_{i \in \mathcal{N}} c_i x_{ij} \leq C y_j$ are knapsack constraints.

Knapsack problem can be solved by dynamic programming

The dynamic program is equivalent to seeking a path of largest profit in a graph

The shortest path problem can be expressed exactly by a linear program with the total unimodularity property

By replacing the knapsack constraints by path constraints, one obtains the arc-flow formulation.

Since the same set of constraints is convexified, it has the same linear relaxation as the Gilmore-Gomory model.

Pros and cons of AF

AF has the same linear relaxation as GG

It can be entered directly in a general purpose MIP solver

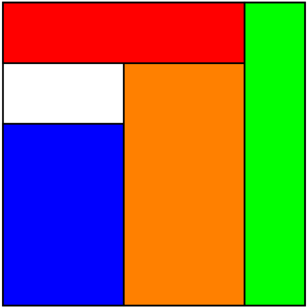
It has more symmetries (ordered configurations instead of sets)

For large sizes of bin, the model does not even load!

Several works address methods for compacting the graph [Delorme and Iori, 2017], [Brandao and Pedroso. 2013]

Two-dimensional packing

Two-dimensional can be understood two ways



2D orthogonal

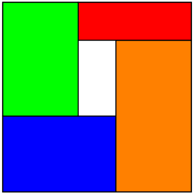
2D vector-packing

In this presentation: orthogonal packing.

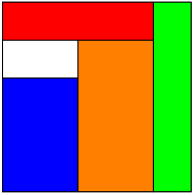
Feasibility problem

Main difference with 1D packing: the bin feasibility problem.

Straightforward in 1D, strongly NP-hard in 2D.

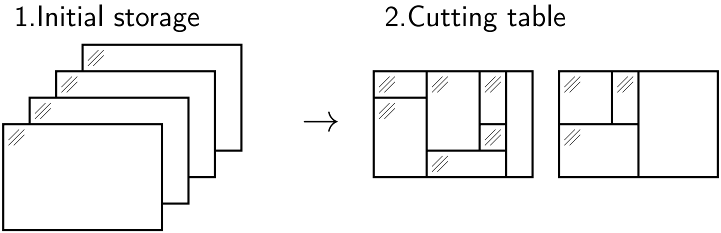


Non-guillotine



Guillotine

2D guillotine CSP



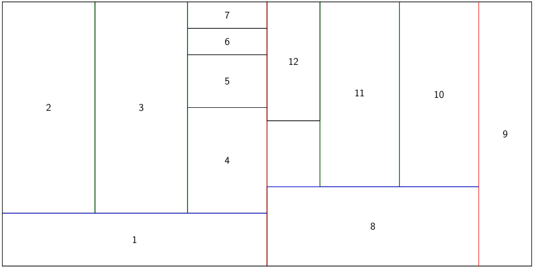
DW decomposition again!

Let P be the set of all possible patterns (valid set of items for one bin). Let us consider for each $p \in P$ a variable λ_p indicating the number of times p is used in the solution.
We denote by a_i^p the number of times item i appears in p.

$$\begin{aligned} \min \sum_{p \in P} \lambda_p & \\ \text{s.t.} \quad \sum_{p \in P} a_i^p \lambda_p \geq d_i, & \quad \forall i \in \mathcal{N} \\ \lambda_p \in \mathbb{N}, & \quad \forall p \in P \end{aligned}$$

Now, set P is now defined as the set of possible item placement in a rectangle.

The subproblem to solve is a two-dimensional unbounded knapsack problem.



Restricted case

DP inspired by Beasley

$$U((w,h)^1) = \max\{0, \max_{h' \le h, \exists i \in \mathcal{I}: h_i = h', w_i \le w} \{U(\overline{(w,h')}^2) + U((w, h-h')^1)\}\}$$

$$U((w,h)^2) = \max\{0, \max_{w' \le w: \exists i \in \mathcal{I}: w_i = w', h_i \le h} \{U(\overline{(w',h)}^3) + U((w-w', h)^2)\}\}$$

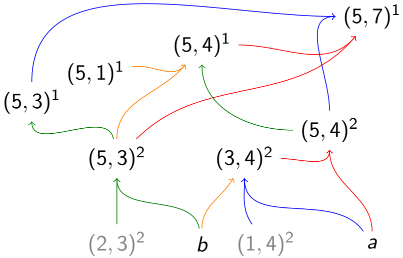
$$U((w,h)^3) = \max\{0, \max_{h' \le h, \exists i \in \mathcal{I}: h_i = h', w_i \le w} \{U(\overline{(w,h')}^4) + U((w, h-h')^3)\}\}$$

$$U(\overline{(w,h)}^2) = \max_{i \in \mathcal{I}: h_i = h, w_i \le w} \{p_i + U((w-w_i, h)^2)\}$$

$$U(\overline{(w,h)}^3) = \max_{i \in \mathcal{I}: w_i = w, h_i \le h} \{p_i + U((w, h-h_i)^3)\}$$

$$U(\overline{(w,h)}^4) = \max\{0, \max_{i \in \mathcal{I}: h_i = h, w_i \le w} \{p_i + U(\overline{(w-w_i, h)}^4)\}\}$$

Practical implementation



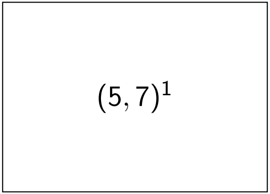
More complex dynamic program: no more a path in a graph, but a flow in an hypergraph.

No more T.U. matrix, but TDI (see [Martin et al., 1991])

Flow in an hypergraph

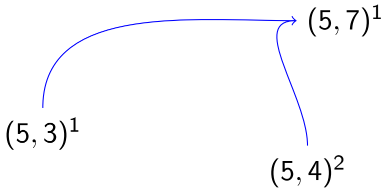
Bin : $(5,7)$ - 6 items to cut: $2 \times a = (2,4)$ and $4 \times b = (3,3)$

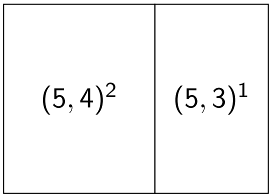
$(5,7)^1$



Flow in an hypergraph

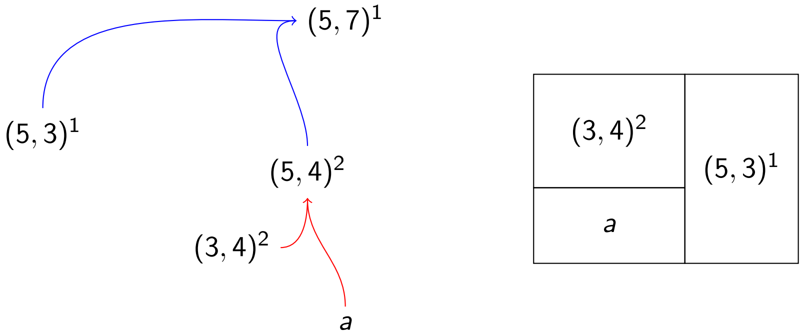
Bin : (5, 7) - 6 items to cut: $2 \times a = (2, 4)$ and $4 \times b = (3, 3)$





Flow in an hypergraph

Bin : $(5,7)$ - 6 items to cut: $2 \times a = (2,4)$ and $4 \times b = (3,3)$



Arc-flow model again! (1)

2D arc flow: (almost) same as 1D

$$\min z$$

$$\text{s.c.} \quad \sum_{a \in A(i)} \lambda_a \geq d_i, \qquad \forall i \in I$$

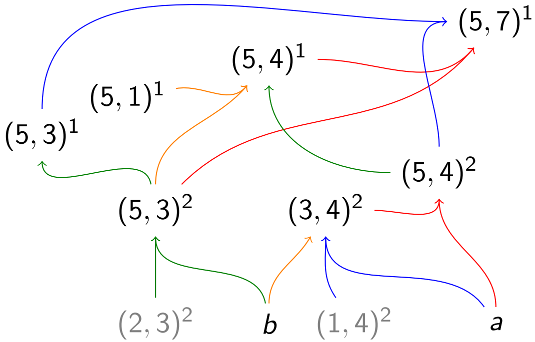
$$\sum_{a \in \delta^+(v)} \lambda_a - \sum_{a \in \delta^-(v)} \lambda_a = \begin{cases} z, & \text{if v=0} \\ 0, & \text{if v=1,...,W-1} \\ -z & \text{if v=W} \end{cases}$$

$$z \in \mathbb{N}$$

$$\lambda_a \in \mathbb{N}, \qquad \forall a \in A$$

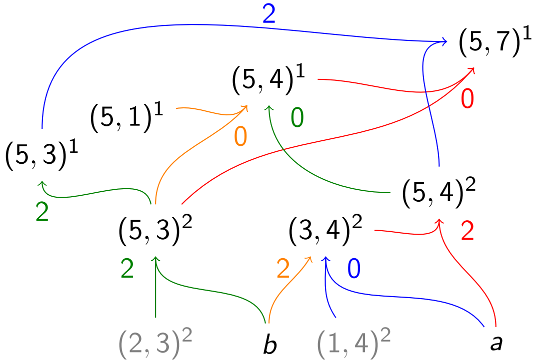
Arc-flow model again! (2)

Bin : $(5,7)$ - 6 articles Ã couper : $2 \times a = (2,4)$ et $4 \times b = (3,3)$



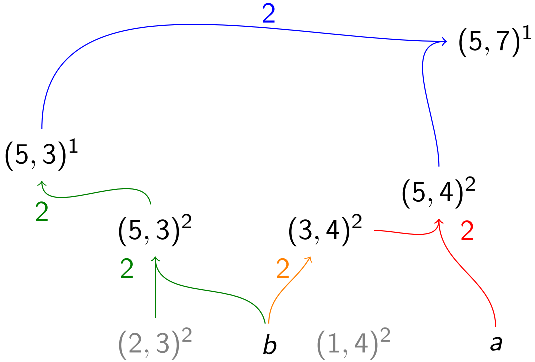
Arc-flow model again! (2)

Bin : $(5, 7)$ - 6 articles Ã couper : $2 \times a = (2, 4)$ et $4 \times b = (3, 3)$



Arc-flow model again! (2)

Bin : $(5,7)$ - 6 articles Ã couper : $2 \times a = (2,4)$ et $4 \times b = (3,3)$

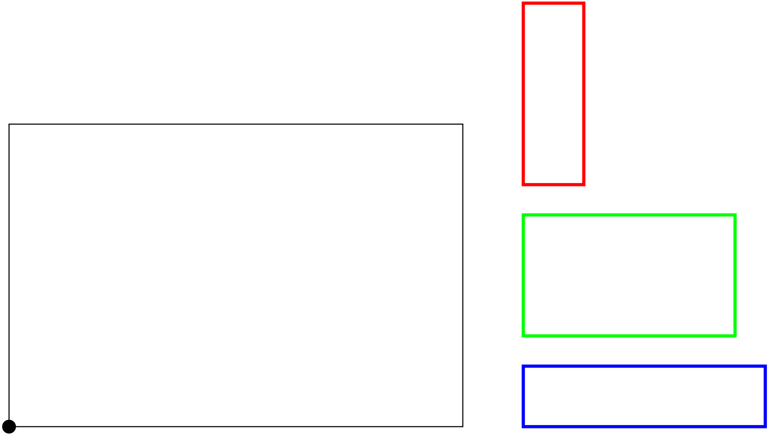


Building 2D solutions

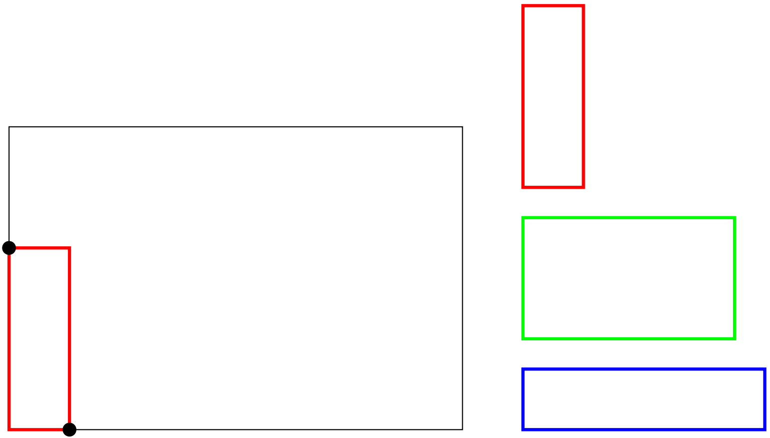
Several ways to model solutions.

- Coordinates
  - Values (X, Y)
  - Discretized values $x_{ip} \in \{0, 1\}$, $y_{iq} \in \{0, 1\}$
- Relative position
  - Oriented (left/right, top-down)
  - Non-oriented (interval graphs)

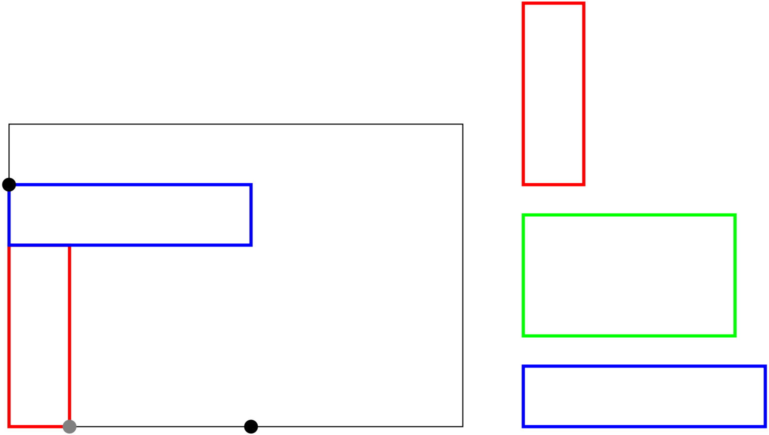
Ensuring 2D feasibility



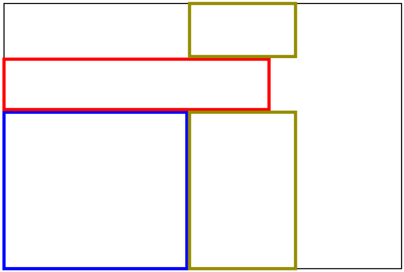
Ensuring 2D feasibility



Ensuring 2D feasibility



A scheduling relaxation



Relaxation into a cumulative scheduling problem.

- Only the x-coordinates have to be determined
- The relaxation is strong

- Branch and bound
- Constraints Programming
- Integer programming, cutting plans

An irritating open problem

Definition (Pallet-loading problem)

Let W, H, w, h, n be five integer values. Is it possible to pack n rectangular items of size (w, h) in a rectangle of size (W, H) if a rotation of 90 degrees of small items is allowed?

An irritating open problem

Definition (Pallet-loading problem)

Let W, H, w, h, n be five integer values. Is it possible to pack n rectangular items of size (w, h) in a rectangle of size (W, H) if a rotation of 90 degrees of small items is allowed?

- Is the problem in P?
- is the problem NP-hard?

An irritating open problem

Definition (Pallet-loading problem)

Let W, H, w, h, n be five integer values. Is it possible to pack n rectangular items of size (w, h) in a rectangle of size (W, H) if a rotation of 90 degrees of small items is allowed?

- Is the problem in P?
- is the problem NP-hard?
- is it even in NP?

Variants

Multiple bin size

Rotation / no rotation

Squares / rectangles / circles

convex items

non-convex items $\implies$ nesting

Variants

Multiple bin size

Rotation / no rotation

Squares / rectangles / **circles**

convex items

non-convex items $\Longrightarrow$ nesting

Source : Au Château Carbonnieux, F. Clautiaux, collection personnelle

Three-dimensional problems

Source : ISMP, F. Clautiaux, collection personnelle

Same structure as 2D, more diff

Often come with practical constraints

Three-dimensional problems

Source : ISMP, F. Clautiaux, collection personnelle

Same structure as 2D, more diff

Often come with practical constraints

Pallet loading

In logistics, boxes are generally packed on pallets.

The problem decomposes into two subproblems.

- build the pallets (3D)
- pack the pallets in the truck (2D bin packing)

In certain cases, pallets are built using *levels*.

Practical constraints

Three-dimensional packing problems come from logistic operations.

Many pratical constraints.

- Maximum weight, weight distribution
- Priorities, visibility
- Stacking constraints, fragility
- Stability (static vs. dynamic)

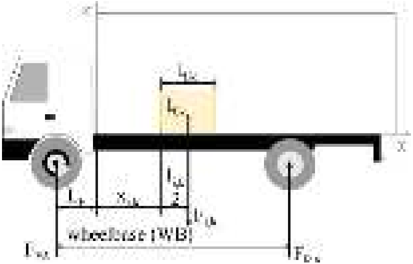


Figure borrowed from Corinna Krebs, Jan Fabian Ehmke, Axle Weights in combined Vehicle Routing and Container Loading Problems, EJTL 10, 2021

Classical 3D heuristics

Classical 3D heuristics restrict the problem to some highly structured solutions

- Wall building
- Layer building

Most heuristics are based on a static order of the items, and local searches

May be linked to simulation / mechanics

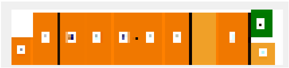
Example of practical 3D problem

Challenged proposed in 2014 by ESICUP, Renault, Université de Bordeaux.

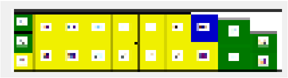
Real data from Renault

"Simplified" truck-loading problem.

Above view



left view



Constraints

1. One or two possible orientations
2. Maximum total weight in the bin
3. Stacks non overlap
4. Stacks lie entirely into the bins
5. Each item is packed.
6. Maximum number of items that can be packed in the last bin.
7. Bin 0 is the one with the smallest volume
8. The height of a stack is the sum of the heights of its layer
9. Maximum total height.
10. Layers of almost equal dimensions in a stack.
11. The envelope of a stack is the envelop of the orthogonal projection of the layers it contains
12. Metal packages are packed together in stacks.
13. Maximum density for each stack.
14. The layers in a stack are sorted by decreasing weight.
15. Layers are composed of contiguous rows
16. Maximum number of rows in a layer.
17. Same sizes of rows in a layer.
18. All items in a layer have the same height.
19. Rows are justified in a layer

18. The dimension of a layer is the envelope of its rows.
19. Rows are composed of contiguous items
20. Maximum number of items in a row.
21. Same horizontal size of items in a row.
22. Items are justified in a row.
23. The dimension of a row is the envelope of its items.
24. Consecutive layers are contiguous in the vertical dimension
25. The top of a stack is the top of its highest layer
26. layers composed of metal packages can only contain one item
27. maximum weight on the base layer items

Introduction

One dimensional bin packing problem

Two-dimensional cutting-stock problems

Three-dimensional problems

Recent trends in cutting and packing

Conclusion

Classical 1D variants

- Conflicts
- Multiple length (several sizes of bin, minimize the total length used)
- Bin packing with fragile objects
- Cardinality constraints
- Bin packing with fragmentation
- Temporal bin packing

For all of them: branch-and-price, heuristics

Problems: integrated problems

In the industry, stock rolls do not pop from nowhere, and items are not shipped as soon as they are produced

Inventory (lot-sizing) and packing

Also packing + routing (not so recent), visibility constraints, order constraints.

Problems: leftovers

When glass, metal, wood are cut, the leftovers are either thrown away, recycled, sold, or reused

The question of how the leftovers are handled is getting more and more interest from the community.

Several approaches, generally with an implicit assumption on the relative costs of storage and raw material.

Problems/method: uncertainty

The bin packing problem is historically treated in a stochastic version (i.e. on-line, semi on-line algorithms).

The works on robust bin packing are more recent

Not much work on stochastic / robust optimization with recourse (too hard?)

Conclusion

The bin packing problem is still a benchmark problem for many algorithms

Exists in many variants with many hard practical constraints

Like many classical problems, the new variants involve either non-linear aspects, or uncertainty

Thank you!

References (knapsack)

[Pisinger, 2005] D. Pisinger, **Where are the hard knapsack problems?** Computers and Operations Research Volume 32, Issue 9, Pages 2271-2284, 2005

[Smith-Miles et al., 2021] Kate Smith-Miles, Jeffrey Christiansen, Mario Andrés Muñoz, **Revisiting where are the hard knapsack problems? via Instance Space Analysis**, Computers and Operations Research, Volume 128, 105184 2021

References (DP)

[Beasley, 1985] Beasley, J. E, **Algorithms for unconstrained two-dimensional guillotine cutting**, Journal of the Operational Research Society 36 (4), 297–306, 1985

[Martin et al., 1990] Martin, R. K., Rardin, R. L., Campbell, B. A., **Polyhedral characterization of discrete dynamic programming.** Operations Research 38 (1), 127–138, 1990

[FC. et al. 2018] Clautiaux F., Sadykov R., Vanderbeck F., Viaud Q. **Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem.** Discrete Optimization, 29:18-44, 2018

[FC et al. 2019] Clautiaux F., Sadykov R., Vanderbeck F., Viaud Q. **Pattern based diving heuristics for a two-dimensional guillotine cutting-stock problem with leftovers** Report, Université de Bordeaux, Inria, 2019

References (Models)

[Kantorovich, 1960] Kantorovich, L. V., **Mathematical methods of organizing and planning production.** Management Science 6 (4), 366–422, 1960

[Eisenbrand and Shmonin, 2006] Eisenbrand F., Shmonin G.: **Carathéodory bounds for integer cones.** Oper. Res. Lett. 34(5): 564-568, 2006

[Gilmore and Gomory, 1961] Gilmore, P. C., Gomory, R. E., **A linear programming approach to the cutting-stock problem.** Operations research 9 (6), 849–859, 1961

[Pessoa et al., 2018] Pessoa A., Sadykov R., Uchoa E., F. Vanderbeck. **Automation and combination of linear-programming based stabilization techniques in column generation.** INFORMS Journal on Computing, 30(2):339-360, 2018.

[Wentges, 1997] Wentges P., **Weighted Dantzig-Wolfe decomposition for linear mixed-integer programming.** International Transactions in Operational Research 42 (2), 151-162, 1997

References (Arc-flow formulation)

[Carvalho, 2000] Valério de Carvalho, J. M., **LP models for bin packing and cutting stock problems**. European Journal of Operational Research 141 (2), 253 – 273, 2002

[Delorme and Iori, 2017] Delorme M., Iori M., **Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems**, Technical Report, 2017

[Brandao and Pedroso. 2013] Brandao F., Pedroso J.P., **Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression,** Technical report, 2013

References (surveys)

Coffman, E.G., Garey, M.R., Johnson, D.S., **Approximation algorithms for bin-packing - an updated survey.** In Ausiello, G., Lucertini, M., Serafini, P. (eds) Journal of Mechanical Design. Springer, Vienna, 3, pp. 49-106, 1984.

Dowsland, K.A., Dowsland, W.B., **Packing problems**, European Journal of Operational Research 56 (1), pp. 2-14, 1992

Lodi, A., Martello, S., Monaci, M., **Two-dimensional packing problems: A survey**, European Journal of Operational Research 141 (2), pp. 241-252, 2002

Lodi, A., Martello, S., Vigo, D., **Recent advances on two-dimensional bin packing problems**, Discrete Applied Mathematics 123 (1-3), pp. 379-396, 2002

Delorme, M., Iori, M., Martello, S., **Bin packing and cutting stock problems: mathematical models and exact algorithms**, European Journal of Operational Research 255 (1), pp. 1-20, 2016

Iori, M., de Lima, V.L., Martello, S., Miyazawa, F.K., Monaci, M., **Exact solution techniques for two-dimensional cutting and packing**, European Journal of Operational Research 289 (2) pp. 399-415, 2021