



HAL
open science

Pattern based diving heuristics for a two-dimensional guillotine cutting stock problem with leftovers

François Clautiaux, Ruslan Sadykov, François Vanderbeck, Quentin Viaud

► To cite this version:

François Clautiaux, Ruslan Sadykov, François Vanderbeck, Quentin Viaud. Pattern based diving heuristics for a two-dimensional guillotine cutting stock problem with leftovers. Matheuristics 7th International workshop, Jun 2018, Tours, France. hal-01958165

HAL Id: hal-01958165

<https://inria.hal.science/hal-01958165>

Submitted on 17 Dec 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Pattern based diving heuristics for a two-dimensional guillotine cutting stock problem with leftovers

François Clautiaux^{2,1} **Ruslan Sadykov**^{1,2}
François Vanderbeck^{2,1} Quentin Viaud^{1,2}

¹ Inria Bordeaux,
France



² Université Bordeaux,
France



Matheuristics 2018, Tours, France, June 19

Contents

Introduction

Column generation and standard diving heuristic

“Non-proper” diving heuristic

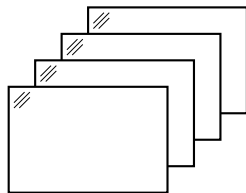
Solving the pricing problem

Partial enumeration technique

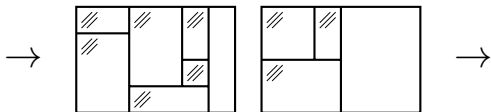
Computational results

Context: production of windows

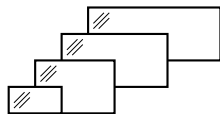
1. Initial storage



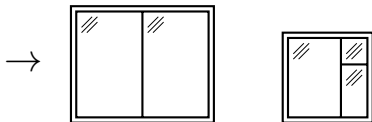
2. Cutting table



3. Intermediate storage

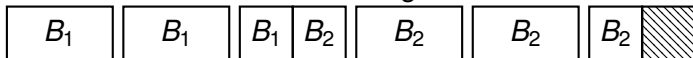


4. Assembly of windows



Specific industrial constraints

- ▶ 4-stage guillotine-cut process
- ▶ **Restricted cuts** (size of the cut part coincides with the width or height of a piece)
- ▶ Daily production is decomposed into **independent batches** (fitting to the intermediate storage)
- ▶ The **order of batches is fixed** because of the due dates of the customer orders
- ▶ The **leftover of only the last bin** of a batch **can be reused** for the next batch due to the organisational costs

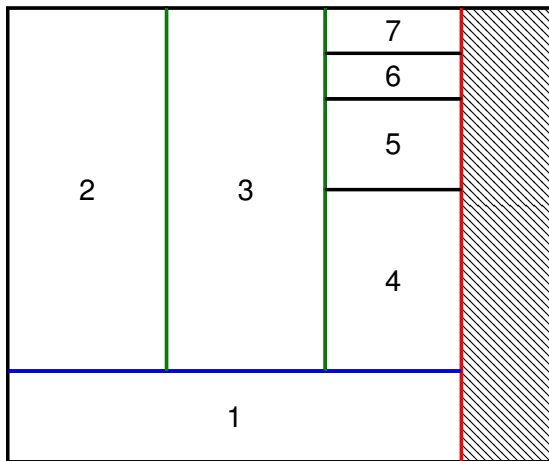


One-batch problem : 2D guillotine cutting stock with leftovers

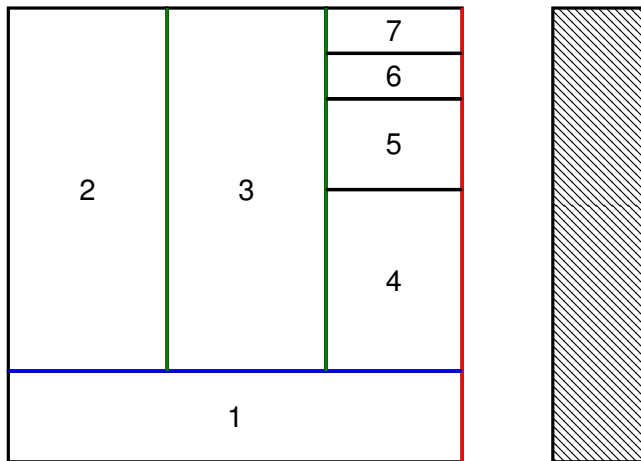
- ▶ Unlimited set of identical rectangular bins of size $W \times H$
- ▶ Additional leftover bin of size $\bar{W} \times H$, $\bar{W} < W$.
- ▶ Set \mathcal{I} of items with fixed demand d_i (number of pieces to cut) and size $w_i \times h_i$, $i \in \mathcal{I}$
- ▶ Each item copy can be rotated by 90°
- ▶ Each item copy should be cut in at most 4 stages
- ▶ Each cut is restricted and guillotine (from one side to the opposite side)
- ▶ The objective function is to minimize the total width of used bins and the width of the used part of the last bin.



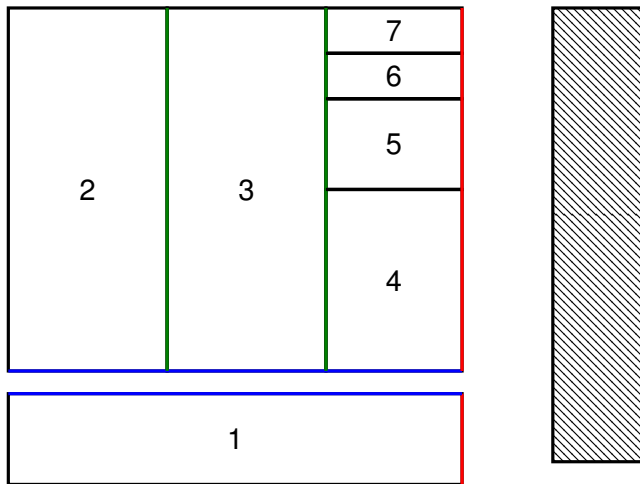
Example of a feasible cutting pattern



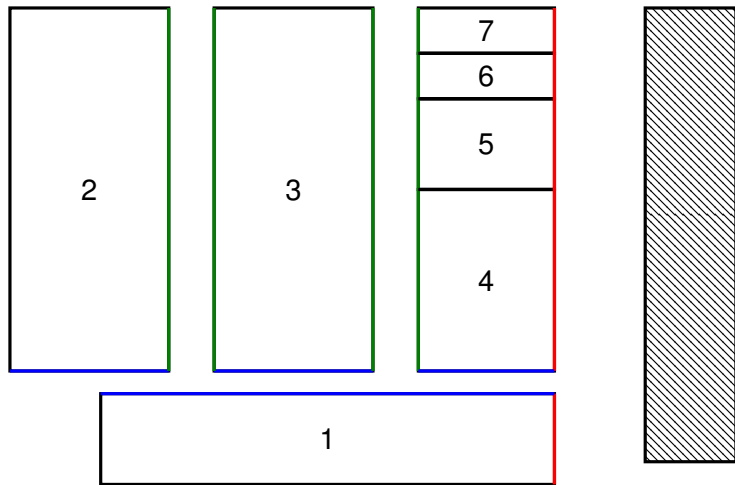
Example of a feasible cutting pattern



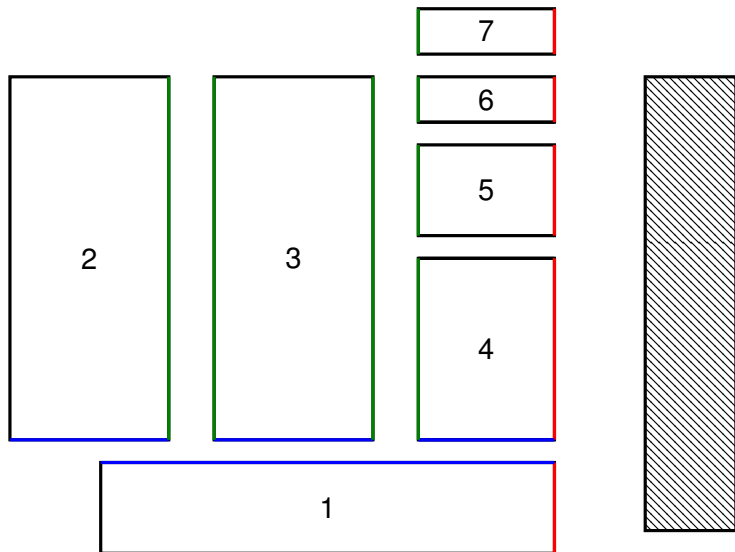
Example of a feasible cutting pattern



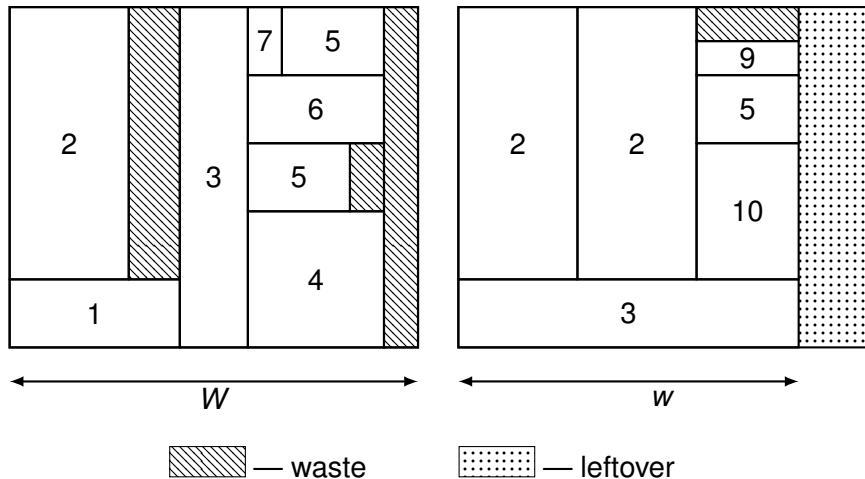
Example of a feasible cutting pattern



Example of a feasible cutting pattern



Example of a valid solution



$$\text{Solution value} = W + w$$

2D guillotine cutting stock : literature review

- ▶ Column generation + rounding (3 stages) [Vanderbeck, 2001]
- ▶ Branch and price (3 stages) [Puchinger and Raidl, 2007]
- ▶ Column generation + heuristics for the residual problem after the rounding [Cintra et al., 2008]
- ▶ Arc-flow MIP formulation (3 stages) [Silva et al., 2010]
- ▶ Column generation + diving (2 stages) [Furini et al., 2012]
- ▶ Dynamic MIP formulation [Furini et al., 2016]
- ▶ With leftovers (2 and 3 stages) [Puchinger et al., 2004]
[Dusberger and Raidl, 2014] [Dusberger and Raidl, 2015]
[Andrade et al., 2016]

Remarks

- ▶ Small instances $(W, H) = (300, 300) \Rightarrow$ Exact methods
- ▶ Large instances $(W, H) = (1000, 1000) \Rightarrow$ Heuristics
- ▶ Our one-batch instances $(W, H) = (6000, 3000)$,
up to 150 items and ≈ 400 pieces to cut

Contents

Introduction

Column generation and standard diving heuristic

“Non-proper” diving heuristic

Solving the pricing problem

Partial enumeration technique

Computational results

Extended (pattern-based) formulation

- ▶ 3 bin types : leftover bin ($W' \times H$), normal bin ($W \times H$), last bin ($W \times H$)
- ▶ \mathcal{P}_t — set of valid cutting patterns for a bin of type $t = 1, 2, 3$
- ▶ a_i^p — number of pieces of item i cut in pattern p
- ▶ w^p — width of pattern p

$$\begin{aligned} \min \quad & \sum_{p \in \mathcal{P}_1} W' \lambda_p + \sum_{p \in \mathcal{P}_2} W \lambda_p + \sum_{p \in \mathcal{P}_3} w^p \lambda_p \\ & \sum_{p \in \mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3} a_i^p \lambda_p = d_i, \quad \forall i \in \mathcal{I}, \\ & \sum_{p \in \mathcal{P}_t} \lambda_p = 1, \quad \forall t \in \{1, 3\}, \\ & \lambda_p \in \mathbb{Z}_+, \quad \forall p \in \mathcal{P}_2, \\ & \lambda_p \in \{0, 1\}, \quad \forall p \in \mathcal{P}_t, t \in \{1, 3\}. \end{aligned}$$

Pricing problem

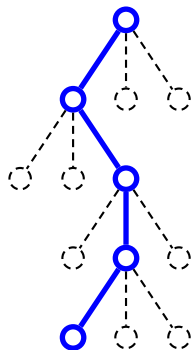
- ▶ $\pi \in \mathbb{R}^{|\mathcal{I}|}$ — dual values for the demand constraints
- ▶ $\mu = (\mu_1, \mu_3) \in \mathbb{R}^2$ — dual values for the bin number constraints
- ▶ Reduced cost of a pattern p :

$$\bar{c}_p = - \sum_{i \in \mathcal{I}} a_i^p \pi_i + \begin{cases} W' - \mu_1, & p \in \mathcal{P}_1, \\ W, & p \in \mathcal{P}_2, \\ w^p - \mu_3, & p \in \mathcal{P}_3. \end{cases}$$

- ▶ The pricing problem decomposes into three **2D guillotine integer knapsack problems**, one for each bin type
- ▶ Can be solved by
 - ▶ a branch-and-bound [Puchinger and Raidl, 2007]
 - ▶ a MIP [Furini et al., 2012]
 - ▶ a dynamic program with bounds [Dolatabadi et al., 2012]
 - ▶ a labelling algorithm [Clautiaux et al., 2018]

Standard diving heuristic [Furini et al., 2012] [Sadykov et al., 2018]

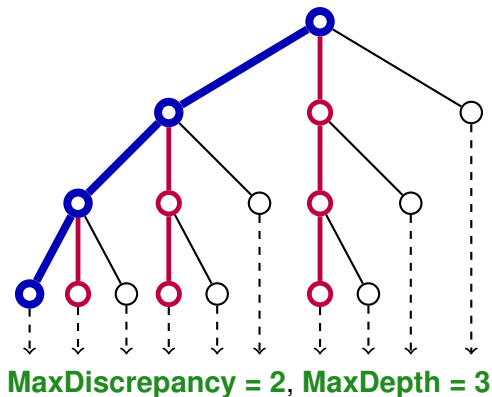
- ▶ use Depth-First Search
- ▶ at each node of the tree
 - ▶ solve the master LP by column generation
 - ▶ select a pattern $p \in \mathcal{P}$ with its value $\bar{\lambda}_p$ closest to a non-zero integer $\lceil \bar{\lambda}_p \rceil$
 - ▶ add $\lceil \bar{\lambda}_p \rceil$ to the partial solution
 - ▶ update the master LP:
 - ▶ update demands d of the items
 - ▶ remove “non-proper” patterns p ($\exists i \in \mathcal{I} : a_i^p > d_i$)
- ▶ repeat until a complete solution is obtained



The heuristic assumes that the pricing generates **proper patterns** ($a_i^p \leq d_i, \forall i \in \mathcal{I}$)!

Diving with LDS [Sadykov et al., 2018]

Idea: add some **diversification** through limited backtracking
(Limited Discrepancy Search by [Harvey and Ginsberg, 1995])



At each node, we have a tabu list of columns forbidden to be added to the partial solution.

Contents

Introduction

Column generation and standard diving heuristic

“Non-proper” diving heuristic

Solving the pricing problem

Partial enumeration technique

Computational results

“Proper” vs. “non-proper” pricing

Proper case (item bounds are imposed in the pricing)

- + Standard diving heuristic can be applied
- Exact pricing is expensive
- Heuristic pricing makes diving less efficient

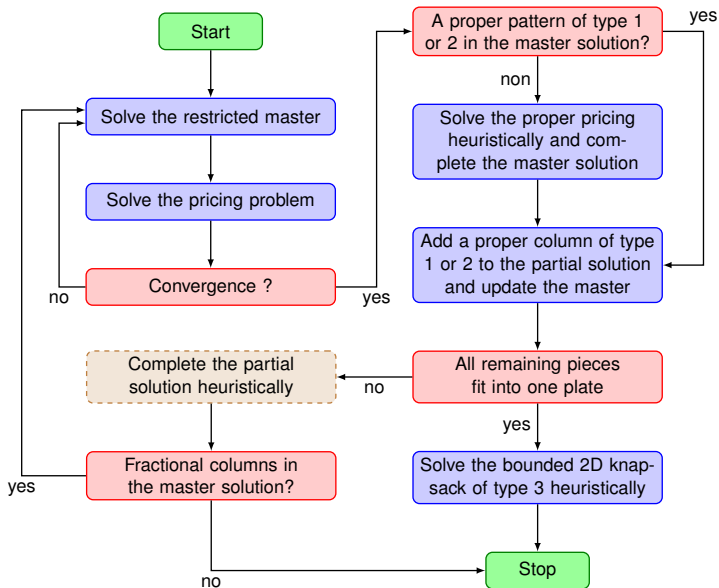
Non-proper case (unbounded pricing)

- + Exact pricing by dynamic programming is relatively efficient
- + Column generation dual bound is almost as tight
[Cintra et al., 2008]
- One needs to adapt the diving heuristic

Diving heuristic adaptations for the “non-proper” case

- ▶ If there are **not enough proper columns** in the master solution, then choose ones with the smallest reduced cost
 - ▶ among all proper column in the restricted master
 - ▶ and proper columns generated with a heuristic pricing
- ▶ **Never fix patterns of type 3** (for the last bin)
- ▶ When all remaining pieces fit into one bin, **heuristically generate a cutting pattern minimizing its width**
- ▶ Every time a partial solution is augmented, complete it heuristically (**hybridization with the evolutionary heuristic**)

“Non-proper” diving heuristic for our problem



Contents

Introduction

Column generation and standard diving heuristic

“Non-proper” diving heuristic

Solving the pricing problem

Partial enumeration technique

Computational results

Unbounded 2D guillotine knapsack: dynamic program

- ▶ Application of [Beasley, 1985] and [Russo et al., 2014]
- ▶ $\mathcal{W}(w, h), \mathcal{H}(w, h)$ — set of all possible widths and heights
- ▶ $U(w, h, s)$ ($U(\overline{w}, h, s)$) — max. value of a pattern of size $w \times h$ cut at stage s (next cut should cut a piece)

$$U(w, h, 1) = \max \left\{ 0, \max_{w' \in \mathcal{W}(w, h)} \{ U(\overline{w'}, h, 2) + U(w - w', h, 1) \} \right\}$$

$$U(w, h, 2) = \max \left\{ 0, \max_{h' \in \mathcal{H}(w, h)} \{ U(\overline{w}, h', 3) + U(w, h - h', 2) \} \right\}$$

$$U(w, h, 3) = \max \left\{ 0, \max_{w' \in \mathcal{W}(w, h)} \{ U(\overline{w'}, h, 4) + U(w - w', h, 3) \} \right\}$$

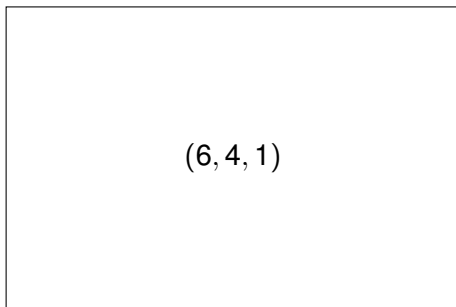
$$U(\overline{w}, h, 2) = \max_{i \in \mathcal{I}: w_i = w, h_i \leq h} \{ \pi_i + U(w, h - h_i, 2) \}$$

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

$$U(\overline{w}, h, 4) = \max \left\{ 0, \max_{i \in \mathcal{I}: w_i = w, h_i \leq h} \{ \pi_i + U(\overline{w}, h - h_i, 4) \} \right\}$$

Dynamic program: example

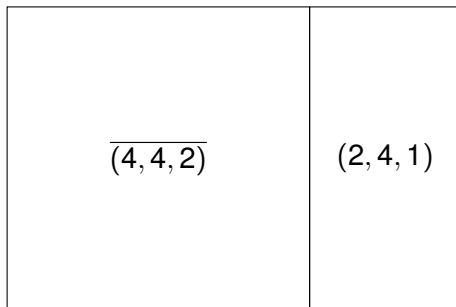
$$U(w, h, 1) = \max\{0, \max_{w' \in \mathcal{W}(w, h)} \{U(\overline{w'}, h, 2) + U(w - w', h, 1)\}\}$$



Bin $(W, H) = (6, 4)$ and an item $a = (4, 3)$

Dynamic program: example

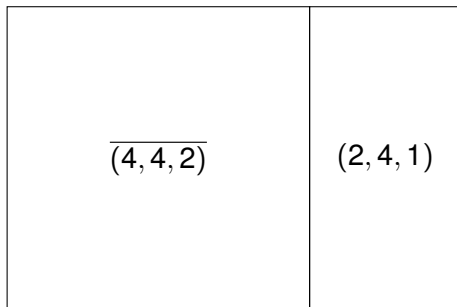
$$U(w, h, 1) = \max\{0, \max_{w' \in \mathcal{W}(w, h)} \{U(\overline{w'}, h, 2) + U(w - w', h, 1)\}\}$$



Bin $(W, H) = (6, 4)$ and an item $a = (4, 3)$

Dynamic program: example

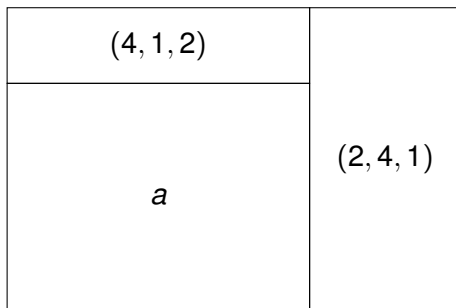
$$U(\overline{w}, \overline{h}, 2) = \max_{i \in \mathcal{I}: w_i = w, h_i \leq h} \{\pi_i + U(w, h - h_i, 2)\}$$



Bin $(W, H) = (6, 4)$ and an item $a = (4, 3)$

Dynamic program: example

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



Bin $(W, H) = (6, 4)$ and an item $a = (4, 3)$

Constructive and evolutionary heuristics

Bounded 2D guillotine knapsack

- ▶ Heuristic modification of the dynamic program
 - ▶ State $U(w, h, s)$ is associated with its best partial solution
 - ▶ In the DP, we combine only the states which together satisfy the item bounds
 - ▶ Heuristic is embedded in a local search (a cut in the solution is replaced by another one)
- ▶ Evolutionary algorithm, based on [Hadjiconstantinou and Iori, 2007]
 - ▶ Every individual is a sequence of glass pieces
 - ▶ **First-Fit heuristic** is used to produce a complete solution
 - ▶ Two-point crossover operator

2D guillotine cutting-stock

- ▶ Iteratively call above evolutionary algorithm for every bin
- ▶ List heuristics: Next-Fit, Best-Fit, First-Fit, Bottom-Fit
- ▶ List heuristics to create the vertical strips + list heuristics for the 1D bin-packing

Contents

Introduction

Column generation and standard diving heuristic

“Non-proper” diving heuristic

Solving the pricing problem

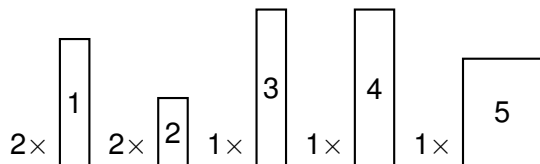
Partial enumeration technique

Computational results

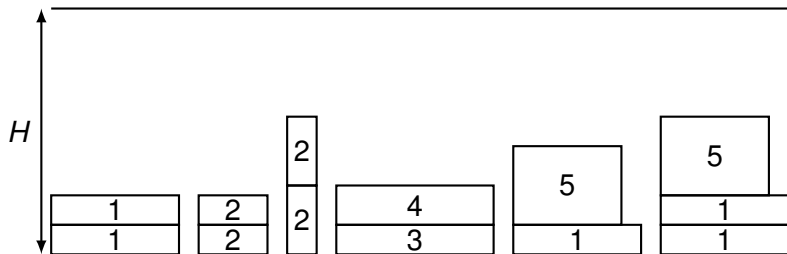
Making the pricing more “proper” : partial enumeration

- ▶ Often, only “non-proper” cutting patterns in the master solution, especially deep in the dive
⇒ last fixing decisions may be bad
- ▶ Our idea is to **partly take into account the item bounds** and to modify accordingly the dynamic program
- ▶ Implementation is done using so-called **meta-items** representing stacks of item pieces satisfying item bounds
- ▶ $\mathcal{M}(w, h)$ — set of vertical meta-items containing items $i \in \mathcal{I}$, $w - \delta < w_i \leq w$, $h_i \leq h$, where $\delta = \min_{i \in \mathcal{I}} w_i$,
similar definition for set $\mathcal{M}(h, w)$ of horizontal meta-items
- ▶ Sets of meta-items are **generated by enumeration**
- ▶ In practice, the size of $\mathcal{M}(w, h)$ is limited

Example of meta-items



All vertical meta-items different from a single item piece



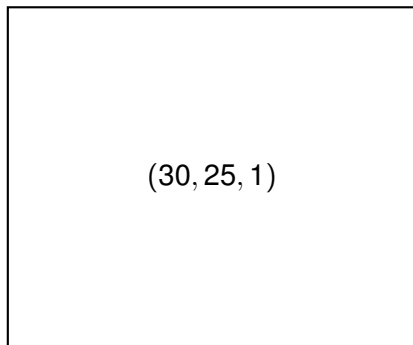
Modified dynamic program

$$U(\overline{w}, h, 2) = \max_{m \in \mathcal{M}(w, h): \bar{h}_m \leq h} \{ \bar{\pi}_m + U(w, h - \bar{h}_m, 2) \},$$

$$U(\overline{w}, h, 3) = \max_{m \in \mathcal{M}(h, w - \delta): \bar{w}_m \leq w} \{ \bar{\pi}_m + U(w - \bar{w}_m, h, 3) \},$$

$$U(\overline{w}, h, 4) = \max \left\{ 0, \max_{m \in \mathcal{M}(w, h - \delta): \bar{w}_m = w} \{ \bar{\pi}_m \} \right\}.$$

Example



Modified dynamic program

$$U(\overline{w}, h, 2) = \max_{m \in \mathcal{M}(w, h): \bar{h}_m \leq h} \{ \bar{\pi}_m + U(w, h - \bar{h}_m, 2) \},$$

$$U(\overline{w}, h, 3) = \max_{m \in \mathcal{M}(h, w - \delta): \bar{w}_m \leq w} \{ \bar{\pi}_m + U(w - \bar{w}_m, h, 3) \},$$

$$U(\overline{w}, h, 4) = \max \left\{ 0, \max_{m \in \mathcal{M}(w, h - \delta): \bar{w}_m = w} \{ \bar{\pi}_m \} \right\}.$$

Example

$(\overline{13}, 25, 2)$	$(17, 25, 1)$
--------------------------	---------------

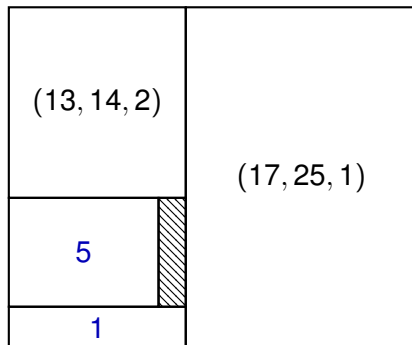
Modified dynamic program

$$U(\overline{w}, h, 2) = \max_{m \in \mathcal{M}(w, h): \bar{h}_m \leq h} \{ \bar{\pi}_m + U(w, h - \bar{h}_m, 2) \},$$

$$U(\overline{w}, h, 3) = \max_{m \in \mathcal{M}(h, w - \delta): \bar{w}_m \leq w} \{ \bar{\pi}_m + U(w - \bar{w}_m, h, 3) \},$$

$$U(\overline{w}, h, 4) = \max \left\{ 0, \max_{m \in \mathcal{M}(w, h - \delta): \bar{w}_m = w} \{ \bar{\pi}_m \} \right\}.$$

Example



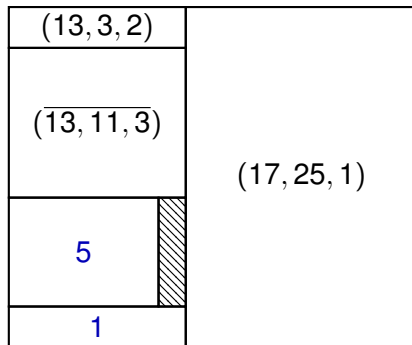
Modified dynamic program

$$U(\overline{w}, h, 2) = \max_{m \in \mathcal{M}(w, h): \bar{h}_m \leq h} \{ \bar{\pi}_m + U(w, h - \bar{h}_m, 2) \},$$

$$U(\overline{w}, h, 3) = \max_{m \in \mathcal{M}(h, w - \delta): \bar{w}_m \leq w} \{ \bar{\pi}_m + U(w - \bar{w}_m, h, 3) \},$$

$$U(\overline{w}, h, 4) = \max \left\{ 0, \max_{m \in \mathcal{M}(w, h - \delta): \bar{w}_m = w} \{ \bar{\pi}_m \} \right\}.$$

Example



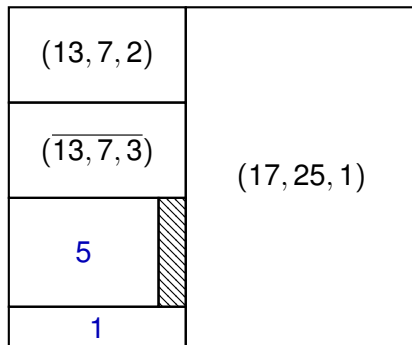
Modified dynamic program

$$U(\overline{w}, h, 2) = \max_{m \in \mathcal{M}(w, h): \bar{h}_m \leq h} \{ \bar{\pi}_m + U(w, h - \bar{h}_m, 2) \},$$

$$U(\overline{w}, h, 3) = \max_{m \in \mathcal{M}(h, w - \delta): \bar{w}_m \leq w} \{ \bar{\pi}_m + U(w - \bar{w}_m, h, 3) \},$$

$$U(\overline{w}, h, 4) = \max \left\{ 0, \max_{m \in \mathcal{M}(w, h - \delta): \bar{w}_m = w} \{ \bar{\pi}_m \} \right\}.$$

Example



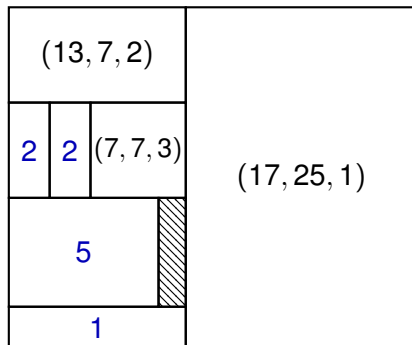
Modified dynamic program

$$U(\overline{w}, h, 2) = \max_{m \in \mathcal{M}(w, h): \bar{h}_m \leq h} \{ \bar{\pi}_m + U(w, h - \bar{h}_m, 2) \},$$

$$U(\overline{w}, h, 3) = \max_{m \in \mathcal{M}(h, w - \delta): \bar{w}_m \leq w} \{ \bar{\pi}_m + U(w - \bar{w}_m, h, 3) \},$$

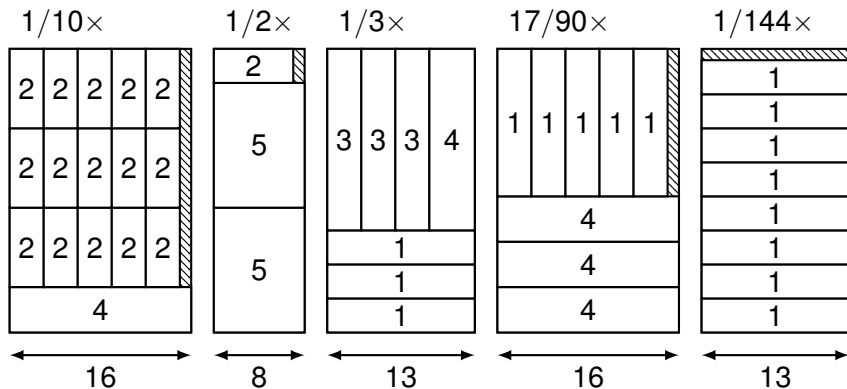
$$U(\overline{w}, h, 4) = \max \left\{ 0, \max_{m \in \mathcal{M}(w, h - \delta): \bar{w}_m = w} \{ \bar{\pi}_m \} \right\}.$$

Example



Fractional solution with standard dynamic program

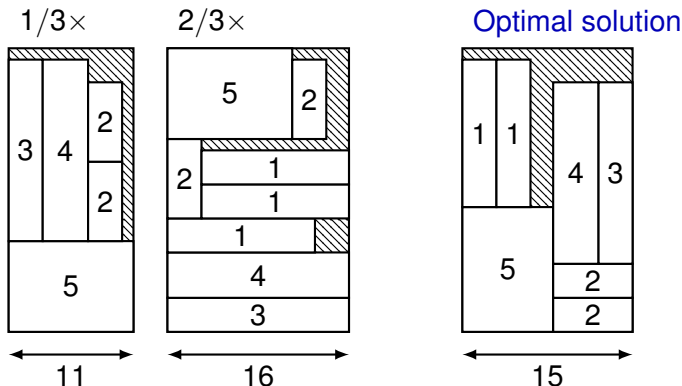
$$d_1 = d_2 = 2, d_3 = d_4 = d_5 = 1$$



Objective function : 13.046

Fractional solution with partial enumeration

$$d_1 = d_2 = 2, d_3 = d_4 = d_5 = 1$$



Objective function : 14.333

66% of the gap is closed

Contents

Introduction

Column generation and standard diving heuristic

“Non-proper” diving heuristic

Solving the pricing problem

Partial enumeration technique

Computational results

Instances

Academic bin-packing instances

- ▶ $\mathcal{I} \in \{20, 40, 60, 80, 100\}$
- ▶ Six classes by [Berkey and Wang, 1987]
 - ▶ $W = H \in \{10, 30, 40, 100, 300\}$
 - ▶ average piece/bin area ratio $\in \{3\%, 4\%, 20\%, 25\%\}$
- ▶ Four classes by [Martello and Vigo, 1998]
 - ▶ $W = H = 100$
 - ▶ 70% of long/wide/small/large items, 10% of each other type
 - ▶ average piece/bin area ratio $\in \{6\%, 22\%, 56\%\}$

Industrial cutting-stock instances

- ▶ $\mathcal{I} \in \{25, 50, 100\}$
- ▶ average demand ≈ 2.5
- ▶ $W \times H \in \{100 \times 50, 1000 \times 500, 6000 \times 3000\}$
- ▶ average piece/bin area ratio is 4.5%

Impact of partial enumeration technique

Dynamic program size, in thousands

Instances	# of states		# of transitions	
	stand.	p.enum.	stand.	p.enum.
Academ	2.6	3.5	25.2	29.6
Indust	8.3	10.4	123.9	162.0

Column generation gap, % from the BKS

Instances	$gap_{p.enum.}$	$gap_{stand.}$	$t_{p.enum.}$	$t_{stand.}$
Academ	1.88%	2.22%	3.9s	3.8s
Indust	1.14%	1.30%	11.2s	10.8s

Comparison with the proper bound

Instances	#conv.	t_{proper}	gap_{proper}	$gap_{p.enum.}$	$gap_{stand.}$
Academ	170/500	12m	1.29%	1.82%	2.27%
Indust	24/135	26m	0.65%	0.70%	0.90%

Comparison of heuristics (gap in % from the BKS)

evol — evolutionary algorithm

evlist — evolutionary + list heuristics

divStand — diving with standard dynamic program

divPE — diving with DPPE (dynamic program with partial enumeration)

divE — hybrid diving with DPPE \ evolutionary algorithm

divLDS — diving with DPPE and LDS

divELDS — hybrid diving with DPPE and LDS \ evolutionary algorithm

	evol		evlist		divStand		divPE		divE		divLDS		divELDS	
Inst.	gap	t	gap	t	gap	t	gap	t	gap	t	gap	t	gap	t
Acad.	3.88	2	1.90	23	1.31	5	1.22	5	0.79	14	0.60	12	0.46	57
Indst.	2.03	4	1.92	65	1.12	20	0.75	23	0.56	59	0.38	143	0.26	448
Large	3.55	5	1.69	99	0.71	22	0.48	24	0.28	73	0.13	134	0.01	497

One-day instances (several batches)

$ \mathcal{B} $	$ \mathcal{I} $	LB	Solution, # glass plates				Time, minutes			
			evol	evlist	divPE	divE	evol	evlist	divPE	divE
10	100	123.3	126.5	126.4	124.4	124.4	1	32	24	48
10	150	191.5	195.5	195.3	192.8	192.5	4	144	83	194
15	100	191.7	196.4	196.2	193.5	193.2	2	50	36	72
15	150	277.3	283.3	282.9	279.2	279.0	6	208	126	291

- ▶ *divPE* saves up 1.4% of plates in comparison with the evolutionary heuristic
- ▶ The gap of *divPE* with the lower bound is at most 0.9%

Conclusions

- ▶ **An industrial variant** of the 2D guillotine **cutting-stock problem** is considered
- ▶ **Very large practical instances** (available online)
- ▶ Column generation-based **“non-proper” diving heuristic** is proposed
- ▶ **Partial enumeration technique** for the pricing DP allows us to improve the heuristic quality at virtually no cost
- ▶ **Significant raw material savings** due to the diving heuristic in comparison with the evolutionary one

References I



Andrade, R., Birgin, E., and Morabito, R. (2016).

Two-stage two-dimensional guillotine cutting stock problems with usable leftover.

International Transactions in Operational Research, 23(1-2):121–145.



Beasley, J. E. (1985).

Algorithms for unconstrained two-dimensional guillotine cutting.

Journal of the Operational Research Society, 36(4):297–306.



Berkey, J. O. and Wang, P. Y. (1987).

Two-dimensional finite bin-packing algorithms.

Journal of the Operational Research Society, 38(5):423–429.



Cintra, G., Miyazawa, F., Wakabayashi, Y., and Xavier, E. (2008).

Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation.

European Journal of Operational Research, 191(1):61 – 85.

References II



Clautiaux, F., Sadykov, R., Vanderbeck, F., and Viaud, Q. (2018).
Combining dynamic programming with filtering to solve a four-stage
two-dimensional guillotine-cut bounded knapsack problem.
Discrete Optimization, (In Press).



Dolatabadi, M., Lodi, A., and Monaci, M. (2012).
Exact algorithms for the two-dimensional guillotine knapsack.
Computers & Operations Research, 39(1):48–53.







Dusberger, F. and Raidl, G. R. (2014).
*A Variable Neighborhood Search Using Very Large Neighborhood
Structures for the 3-Staged 2-Dimensional Cutting Stock Problem*,
pages 85–99.
Springer International Publishing, Cham.



Dusberger, F. and Raidl, G. R. (2015).
Solving the 3-staged 2-dimensional cutting stock problem by dynamic
programming and variable neighborhood search.
Electronic Notes in Discrete Mathematics, 47:133 – 140.

References III

-  Furini, F., Malaguti, E., DurÄjn, R. M., Persiani, A., and Toth, P. (2012).
A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size.
European Journal of Operational Research, 218(1):251 – 260.
-  Furini, F., Malaguti, E., and Thomopoulos, D. (2016).
Modeling two-dimensional guillotine cutting problems via integer programming.
INFORMS Journal on Computing, 28(4):736–751.
-  Hadjiconstantinou, E. and Iori, M. (2007).
A hybrid genetic algorithm for the two-dimensional single large object placement problem.
European Journal of Operational Research, 183(3):1150 – 1166.
-  Harvey, W. D. and Ginsberg, M. L. (1995).
Limited discrepancy search.
In Proceedings of the 14th international joint conference on Artificial intelligence (IJCAI'95), volume 1, pages 607–615, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

References IV



Martello, S. and Vigo, D. (1998).

Exact solution of the two-dimensional finite bin packing problem.

Management Science, 44(3):388–399.



Puchinger, J. and Raidl, G. R. (2007).

Models and algorithms for three-stage two-dimensional bin packing.

European Journal of Operational Research, 183(3):1304–1327.



Puchinger, J., Raidl, G. R., and Koller, G. (2004).

Solving a Real-World Glass Cutting Problem, pages 165–176.

Springer Berlin Heidelberg, Berlin, Heidelberg.



Russo, M., Sforza, A., and Sterle, C. (2014).

An exact dynamic programming algorithm for large-scale unconstrained two-dimensional guillotine cutting problems.

Computers & Operations Research, 50:97 – 114.

References V



Sadykov, R., Vanderbeck, F., Pessoa, A., Tahiri, I., and Uchoa, E. (2018).

Primal heuristics for branch-and-price: the assets of diving methods.
INFORMS Journal on Computing, (Forthcoming).



Silva, E., Alvelos, F., and Valério de Carvalho, J. M. (2010).

An integer programming model for two-and three-stage two-dimensional cutting stock problems.
European Journal of Operational Research, 205(3):699–708.



Vanderbeck, F. (2001).

A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem.
Management Science, 47(6):864–879.