Constraint Logic Programming Project

FEUP - - 2022/23 - M.EIC0023

Amanda de Oliveira Silva
up201800698

Adilson Silva
up202212352

1. Introduction

This project was carried out in the context of the Constraint Logic Programming M.EIC0023 class, taught at the Faculty of Engineering of Porto by Professor Daniel Silva, in the spring semester of 2023.

The project consisted of applying an optimization algorithm for sorting/allocation of products on shelves in a warehouse in order to use the smallest possible space, as described in the article published by [1]NP Hoare1 and JE Beasle, "Placing Boxes on Shelves: A Case Study ", 1999. For the execution of the project, tools and techniques of Constraint Programming were used. With a focus on Sicstus Prolog, using the provided materials proposed in the course.

2. Problem description

The problem at hand revolves around organizing a warehouse effectively. To solve this challenge, it is important to consider and address various factors related to conceptual frameworks and tangible objects. By exploring the details of this task, our goal is to create strategies that improve the efficiency and effectiveness of the warehouse.

To proceed with the concepts and objects, the following considerations need to be taken into account.

2.1 Products

All the products have a form or are packaged in a rectangular box. And the products come with a variety of these four attributes: **length, width, height and weight**.

For space utilization optimization, choosing the right orientation for a box is a crucial decision.Having this into account a box can be oriented in six different ways, and boxes containing the same product must be placed using the same orientation. Also the products can be classified with distinct families, and products of the same family must be stored together.

2.2 Bays

A bays is configured in a set of shelving which utilizes a rectangular area in the warehouse. The attributes of af a bay are a little more complex than the other objects, but can be easily assimilated with those key parameters:

- **width:** the space across the shelf.
- **Height** : the height of the bay.

■ **Depth**: how far back a shelf on the bay goes.
■ **Available Height:** products can be stacked above the top shelf, but can ultrapass the ceiling. So the available height is the height from the floor to ceiling of the stockhouse.

In the original problem proposed by the article the bays have two selections of bays, one with identical bays and other where the bays can differ. In this work, we only use equal bays, with the following dimensions: **1200 mm wide, 2400 mm high and 650 mm deep and the available height in the bay is 3000mm**.

2.3 Shelf

A shelf is a rectangular object where products will be laid. The parameters associated with each shelf are: **position, height of the shelves, and thickness.**

Another important parameter to consider when laing the products in the shelve are a obligatory gap around the stocked boxes, for facilitating the removal of the inventory. So a mustleft and mustright gap is a prerequisite besides an intergap between the boxes. Note that boxes of the product, placed with the same orientation, don't have an intergap between them, but boxes of the same family storing different products demand an intergap.

3. Literature review

Warehouse management is a complex process that can involve several optimization problems. One of the most important aspects of warehouse management is space usage. Constraint Logic Programming (CLP) has been proved to be useful to solve these kinds of optimization problems. Another paper , besides the one that inspired this project[1], demonstrates the use of CLP in this context , "A Constraint Logic Programming Model for Warehouse Space Allocation"[2]. In this literature review, we will compare and contrast these two papers.

3.1 Comparison

"A Constraint Logic Programming Model for Warehouse Space Allocation" proposes a CLP model that optimizes the allocation of warehouse space based on product characteristics and available storage space. The paper presents a mathematical model that considers various constraints such as product size, weight, and demand. The proposed model is tested on real-world data from a large distribution center, and the results show significant improvements in space utilization compared to traditional methods.

On the other hand, "Placing Boxes on Shelves: A Case Study" uses heuristics developed through CLP to minimize the space used when placing boxes on shelves.

The use of CLP in both papers demonstrates its effectiveness in optimizing warehouse management processes. The application of CLP allows for efficient and effective solutions to complex optimization problems in this field. The proposed models and heuristics provide valuable insights into different approaches to optimizing warehouse management processes using CLP, in both cases prolog.

3.2 Conclusion

In conclusion, the papers "A Constraint Logic Programming Model for Warehouse Space Allocation" and "Placing Boxes on Shelves: A Case Study" demonstrate the effectiveness of CLP in optimizing warehouse management processes. While both papers use CLP, they differ in their focus and approach.

4. Approaches

4.1 Common concept: "Grouped products"

This is a core concept in both of our approaches. As said before, the same products must stick together. The idea is to treat them like a single larger box, making the overall problem easier. For example: in **Fig 1** below, the first four boxes in the second shelf are shelved as one larger box only. In this case, we have 4 instances of the product, with 2 stacks on the height axis (top/down) and 2 on the length axis (left/right).

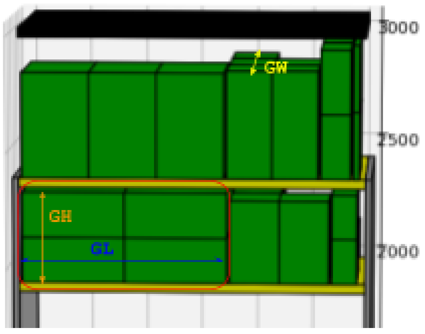
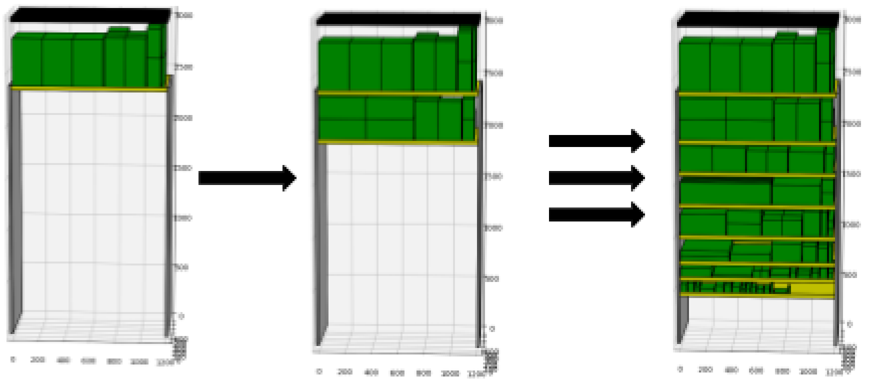


Fig 1 - Example of "grouped product" box

4.2 "Paper" approach

This approach is based on "random storage" design algorithm of the *Beasle* paper [1] applied for one family at a time.

- For each list of products ("family"), choose one shelf height position (from top to down) and shelve, seeking to optimize the space used. For that, it must be decided, for each product, which orientation to use and how the products will be stacked ("grouped products", as explained before).
- Recursively apply the same algorithm, for remaining products and vertical space on the bay.
- When there is no room for shelving more products on that bay, another one is pick-up.
- The process continues until all products are shelved.



(a) after the first shelf *(b) one more shelf* *(c) all products shelved*

Fig.2 - Applying the approach steps

4.2.1 Variables/Domains

For each product we need several sets of variables. Most of them have 3 distinct suffixes: L, W, H (length, width and height axis).

Variables	Domain	
`IL,IW,IH`	`1..3 (distincts)`	indexes used to rotate products and get desired orientation. (IL=1, IW=2, IH=3) are the original orientation.
`RL,RW,RH`	`{L,W,H}`	size of each rotated product on each direction
`NL,NW,NH`	`1..Q`	number of stacks in each direction (Q is the quantity of products).
`GL,GW,GH (Gs)`	`positive`	size of grouped product, calculated after rotation and stacks
`Chosen (Cs)`	`{0,1}`	product will be chosen to be shelved (1) or

Variables	Domain	
		not (0)
`"Anti- Chosen" (AntiCs)`	`{1,2}`	auxiliary var (defined on "AntiC #=1 ⇔ C") used for allow a easy change value selection order of Cs (1 for shelved product and 2 for not). Used on the best results alternative for this approach.
`MaxH`	`{0,100,...2950, 3000}`	next shelf height (maximum available height to a "grouped product" box).
`MaxL`	`{0..1200}`	space used in length axis of the shelf

Table 1 - "Paper approach" main variables and domains

4.2.2 Core Constraints

Variables	Constraints
`IL,IW,IH`	all distincts. Indexes of product sizes (element).
`RL,RW,RH`	smaller than available shelf space in each direction.
`NL,NW,NH`	product must be equal or bigger than "Q".
`GL,GW,GH (Gs)`	each one must be smaller than available shelf space (considering needed gaps).
`Chosen (Cs)` `MaxL`	the scalar product of all "Chosen" and "GL" must be smaller than shelf length (1200 in our case). $\sum_{i=1}^{n} Cs_i * GL_i = MaxL$
`"Anti-Chosen" (AntiCs)`	AntiC #=1 ⇔ C

Table 2 -"Paper approach" main constraints

4.3 "Cumulative" approach

In the second approach we use two "cumulative" global constraints to shelve the products. The "cumulative" here is used as "bin packing", setting duration equals 1. We can't use the "bin_packing" global constraint because, in our case, the resources are domain variables, not integers.

In the same way as the other approach, first we need the “grouped products” boxes. Then we set one “cumulative” to put “grouped products” boxes on each shelf, all at once. The resources are the lengths of the boxes and the “limit” is the total length of the shelf.

$$\forall\ Product\ in\ Family,\ \exists\ task(Shelf, 1, \_,\ GL, 1)\text{, where limit = 1200}$$

Another “cumulative” is used to put shelves on each bay. The resources are the height of shelves (MaxHs) and the “limit” are the available height of the bays.

$$\forall\ Shelf\ in\ Shelves,\ \exists\ task(Bay, 1, \_,\ MaxH,\ 1)\text{, where limit = 3000}$$

4.3.1 Variables/Domains

Variables	Domain	
`IL,IW,IH`	`1..3 (distincts)`	indexes used to rotate products and get desired orientation. (IL=1, IW=2, IH=3) are the original orientation.
`RL,RW,RH`	`{L,W,H}`	size of each rotated product, in each direction.
`NL,NW,NH`	`1..Q`	number of stacks in each direction (Q is the quantity of products).
`GL,GW,GH`	`positive`	size of grouped product, calculated after rotation and stacks.
`MaxHs`	`{0, 50, 100...2950, 3000}`	list of the shelf heights (max. available height for “grouped product” boxes).
`Shelves (Vs)`	`{1..100}`	list with the shelf number of each product .
`Bays`	`{1..50}`	list with the bay number of each shelf.

Table 3 - Cumulative approach - main variables and domains

4.3.2 Constraints

Variables	Constraints
`IL,IW,IH`	all distincts.
`RL,RW,RH`	smaller than available shelf space in each direction.
`NL,NW,NH`	product must be equal or bigger than “Q”.
`GL,GW,GH`	are the products of each rotated size and number of products in that direction. Each one must be smaller than available shelf space (considering gaps). GL is the resource of the first cumulative.
`MaxHs`	for each bay, at least one needs to be greater than diff between bay height and bay available height (first shelf on top of the bay). Resource of second cumulative.
`Shelves`	Result of first cumulative
`Bays`	Result of second cumulative

Table 4 - Cumulative approach - main constraints

4.4 Other proofs of concept

Besides the two presented approaches, we try several other proofs of concepts, with different degree of effort:

- Fixing some variables like the shelves heights and/or product orientation. The results were far way worse than expected.
- Using global constraints for placement (ex: “disjoint2”, “diffn”). The problem with them was the difficulty of restriction in terms of same product placement.
- “Paper” approach with grounded “grouped products” domain variables. Slower than the final version, mainly because it needs a labeling call for all products, in each iteration.
- Using global constraints (ex: “cumulative”, “bin_packing”, scalar product, etc) with the “paper” approach. Didn't work either because of the need of integer/grounded variables or because of doing more work and being slower than our simple self coded “scalar product”.
- “Cumulatives” instead of “cumulative” (on second approach). It was just slower, but with the same results in terms of number of bays.
- “Paper” approach implementations in docplex (and or-tools). Dropped in favor of adding a “cumulative” approach in Prolog.

5. Experimental setup

5.1 “Paper” approach search strategies

We run several tests with different search strategies and initial orders of variables. As we want to make as many combinations as possible, first tests are made with a “small” family of products (family 1, with 50 products). After that, we use some of the best results to test on a larger one (family 66, with 605 products).

As we can see (Appendix I), the best results with “leftmost, bisect, down” options , with cost function based only on shelf height (“Cost #= MaxH”) and putting domain variables of each product together.

5.2 “Cumulative” approach search strategies

As we can see in (Appendix I), for working combinations of labeling options, the results are very close. So, we define default options (“leftmost”, “step” and “up”) as best, for use in the next part, where we compare the two strategies.

In this approach, in terms of the number of bays used, we got the best results (or very close ones) at the first solution, probably due to the use of “cumulative” global constraint.

To verify this, we ran a test with the biggest family 66 with 10hs of timeout and got the same solution we got with a few seconds: 25 bay used. For the purpose of comparison between the different labeling options, time to get this first one is more than enough.

By the way, in the tables we present only the default one (“leftmost”) and “impact”, because “ff” and “ffc” always end up with timeouts.

6. Results

Finally, we got the best search strategies from both approaches and ran with all 67 families. The result can be seen in the following graphics.

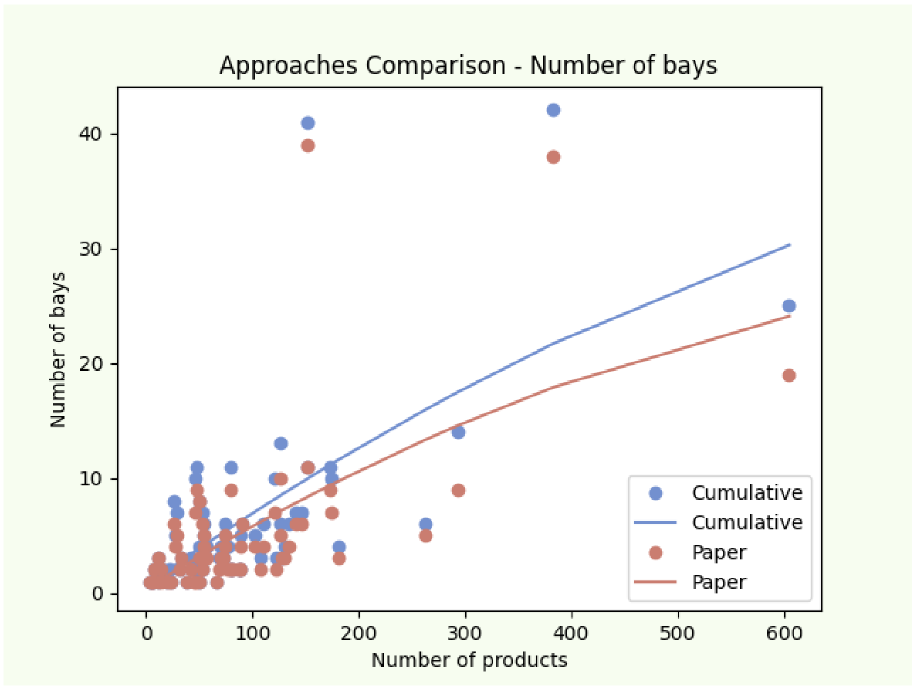


Fig. 5 - All families results: comparing number of bays

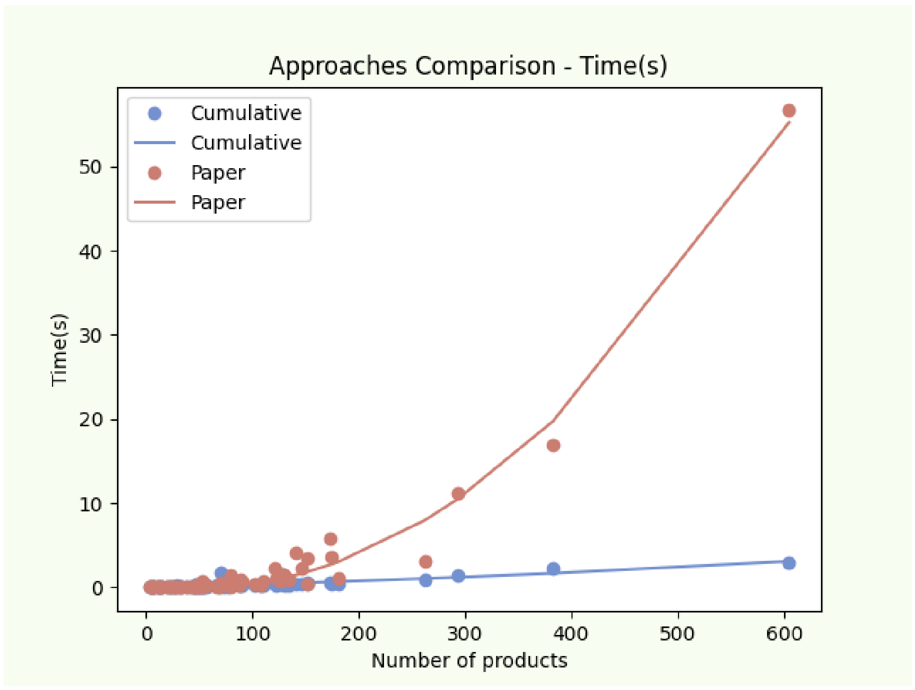


Fig. 6 - All families results: comparing execution times

7. Conclusion

As we can see in the figures 5 and 6, the paper approach got the best results in terms of bays used, while the “cumulative” has best times, as the problem size increases.

We think this happens because the two “cumulative” add more tighter constraints to the problem. Furthermore, the recursive nature of the “paper” approach leads to an expressive increase of the time spent.

In practical terms, the “paper” approach should be chosen because the space needed is more important than time spent by the approaches. Besides, the time of execution of the “paper” approach is not that bad. And, as we can see in figure 1 of the Appendix II, the results are clearly more compact, with less space wasted between shelves.

8. Bibliography

[1] NP Hoare1 and JE Beasle. (2001)"Placing Boxes on Shelves: A Case Study" . Journal: The Logistics Business Ltd., Birmingham, UK; and PImperial College of Science, Technology and Medicine, London, UK

[2]Perera D, Mirando U, & Fernando A. (2022). Warehouse space optimization using linear programming model and goal programming model, Sri Lanka Journal of Economics, Statistics, and Information Management, 1(1), 103-124

Appendix I - Search strategies results

"Paper" Approach - Small family test

Familie 1: 50 products minimize(MaxH + RemainL)							
Variable Selection		Value Selection				Time	Bays
id	order	id	labeling options				
1	`MaxH, Gs, Cs` `leftmost (default)`	1	step (default)	up (default)	1.3	2
		2		down	1410.6	48
		3	enum	up (default)	3.2	2
		4		down	1440.6 3	48
		5	bisect	up (default)	0.98	2
		6		down	1410.65	48
1	`MaxH, Gs, Cs` `ff`	11	step (default)	up (default)	312.2	3
		12		down	90.11	1
		13	enum	up (default)	335.11	3
		14		down	90.13	1
		15	bisect	up (default)	310.17	3
		16		down	90.04	1
1	`MaxH, Gs, Cs` `impact`	21	step (default)	up (default)	1.13	2
		22		down	220.52	2
		23	enum	up (default)	1.66	2
		24		down	1325.65	244
		25	bisect	up (default)	0.08	2
		26		down	635.15	21
1	`MaxH, Gs, Cs`	31	step (default)	up (default)	314.47	3

Familie 1: 50 products minimize(MaxH + RemainL)						
Variable Selection		Value Selection			Time	Bays
id	order	id	labeling options			
	`ffc`	32		down	90.1	1
		33	enum	up (default)	335.12	3
		34		down	90.1	1
		35	bisect	up (default)	311.05	3
		36		down	90.06	1

Familie 1: 50 products minimize(MaxH + RemainL)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			

2	`Gs, Cs, MaxH`
	
	`leftmost (default)`
		1	step (default)	up (default)	1.07	2

		2		down	0.52	2
		3	enum	up (default)	2.63	2
		4		down	0.46	2
		5	bisect	up (default)	0.79	2
		6		down	0.11	2

2	`Gs, Cs, MaxH`
	
	`ff`
		11	step (default)	up (default)	311.88	3

		12		down	90.09	1
		13	enum	up (default)	335.16	3
		14		down	90.11	1
		15	bisect	up (default)	310.21	3
		16		down	90.04	1

Familie 1: 50 products minimize(MaxH + RemainL)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			

2	`Gs, Cs, MaxH`
	
	`impact`
		21	step (default)	up (default)	0.43	2

		22		down	0.32	2
		23	enum	up (default)	30.4	2
		24		down	0.49	2
		25	bisect	up (default)	0.3	2
		26		down	0.1	2

2	`Gs, Cs, MaxH`
	
	`ffc`
		31	step (default)	up (default)	313.66	3

		32		down	90.09	1
		33	enum	up (default)	335.13	3
		34		down	90.11	1
		35	bisect	up (default)	310.86	3
		36		down	90.06	1

Familie 1: 50 products minimize(MaxH + RemainL)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			

3	`(Weaving)`
	
	`MaxH, GL1,GW1,GH1,C1`
	…
	`GL50,GW50,GH50, C50`
		1	step (default)	up (default)	-	-

		2		down	1410.79	48
		3	enum	up (default)	-	-
		4		down	1440.7	48

Familie 1: 50 products minimize(MaxH + RemainL)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			
	leftmost (default)	5	bisect	up (default)	-	-
		6		down	1385.63	47
3	(Weaving) GL1,GW1,GH1,C1 … GL50,GW50,GH50, C50, MaxH leftmost (default)	11	step (default)	up (default)	90.08	2
		12		down	2.96	2
		13	enum	up (default)	90.07	2
		14		down	4.19	2
		15	bisect	up (default)	90.05	2
		16		down	0.22	2

Familie 1: 50 products minimize(MaxH)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			
2	Gs, Cs, MaxH leftmost (default)	1	step (default)	up (default)	-	-
		2		down	0.15	2
		3	enum	up (default)	-	-
		4		down	0.14	2
		5	bisect	up (default)	-	-
		6		down	0.05	2
2	Gs, Cs, MaxH ff	11	step (default)	up (default)	-	-
		12		down	90.79	1
		13	enum	up (default)	-	-
		14		down	91.15	1

Familie 1: 50 products minimize(MaxH)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			
		15	bisect	up (default)	-	-
		16		down	90.64	1

2	`Gs, Cs, MaxH`
	`impact`
		21	step (default)	up (default)	-	-

		22		down	0.22	2
		23	enum	up (default)	-	-
		24		down	0.14	2
		25	bisect	up (default)	-	-
		26		down	0.05	2

2	`Gs, Cs, MaxH`
	`ffc`
		31	step (default)	up (default)	-	-

		32		down	90.83	1
		33	enum	up (default)	-	-
		34		down	91.19	1
		35	bisect	up (default)	-	-
		36		down	90.67	1

Familie 1: 50 products minimize(MaxH)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			

3	`(Weaving)`
	`MaxH, GL1,GW1,GH1,C1`
	…
	`GL50,GW50,GH50, C50,`
		1	step (default)	up (default)	0.07	1

		2		down	-	-
		3	enum	up (default)	0.10	1
		4		down	-	-

Familie 1: 50 products minimize(MaxH)							
Variable Selection		Value Selection				Time	Bays
id	order	id	options				
	`leftmost (default)`	5	bisect	up (default)		0.05	1
		6		down		-	-

3	`(Weaving)`
	
	`GL1,GW1,GH1,C1`
	…
	`GL50,GW50,GH50, C50, MaxH`
	
	`leftmost (default)`
		1	step (default)	up (default)		0.28	1

		2		down		-	-
		3	enum	up (default)		0.77	1
		4		down		-	-
		5	bisect	up (default)		0.13	1
		6		down		-	-

“Paper” Approach - Large family tests

Family 66: 605 products minimize(MaxH + RemainL)							
Variable Selection		Value Selection				Time	Bays
id	order	id	options				

2	`Gs, Cs, MaxH`
	
	`impact`
		21	step (default)	up (default)		289.53	44

		22		down		1966.59	77
		23	enum	up (default)		259.69	43
		24		down		2111.08	74
		25	bisect	up (default)		51.11	35
		26		down		1392.09	69

Family 66: 605 products minimize(MaxH)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			

2	`Gs, Cs, MaxH`
	
	`impact`
		21	step
			(default)
				up (default)	-	-

		22		down	470.13	34
		23	enum	up (default)	-	-
		24		down	69.08	34
		25	bisect	up (default)	-	-
		26		down	69.68	34

Family 66: 605 products minimize(MaxH)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			

3	`(Weaving)`
	
	`MaxH, GL1,GW1,GH1,C1`
	…
	`GL50,GW50,GH50, C50`
	
	`leftmost (default)`
		1	step
			(default)
				up (default)	81.46	19

		2		down	-	-
		3	enum	up (default)	252.18	19
		4		down	-	-
		5	bisect	up (default)	38.09	19
		6		down	-	-

“Cumulative” Approach - Small family test

Familie 1: 50 products Cost function: minimize(NBay)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			
1	`leftmost (default)`	1	step (default)	up (default)	0.09	1
		2		down	-	-
		3	enum	up (default)	0.09	1
		4		down	-	-
		5	bisect	up (default)	0.31	1
		6		down	-	-
1	`impact`	21	step (default)	up (default)	0.09	1
		22		down	-	-
		23	enum	up (default)	0.09	1
		24		down	-	-
		25	bisect	up (default)	3.86	1
		26		down	-	-

“Cumulative” Approach - Large family test

Familie 66: 605 products Cost function: minimize(NBay)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			
1	`leftmost (default)`	1	step (default)	up (default)	3.16	25
		2		down	-	-
		3	enum	up (default)	3.1	25
		4		down	-	-

Familie 66: 605 products Cost function: minimize(NBay)						
Variable Selection		Value Selection			Time	Bays
id	order	id	options			
		5	bisect	up (default)	3.61	25
		6		down	-	-
1	`impact`	21	step (default)	up (default)	3.17	25
		22		down	-	-
		23	enum	up (default)	3.19	25
		24		down	-	-
		25	bisect	up (default)	3.86	25
		26		down	-	-

Appendix II - Visualizing and comparing results

Comparison of both approaches

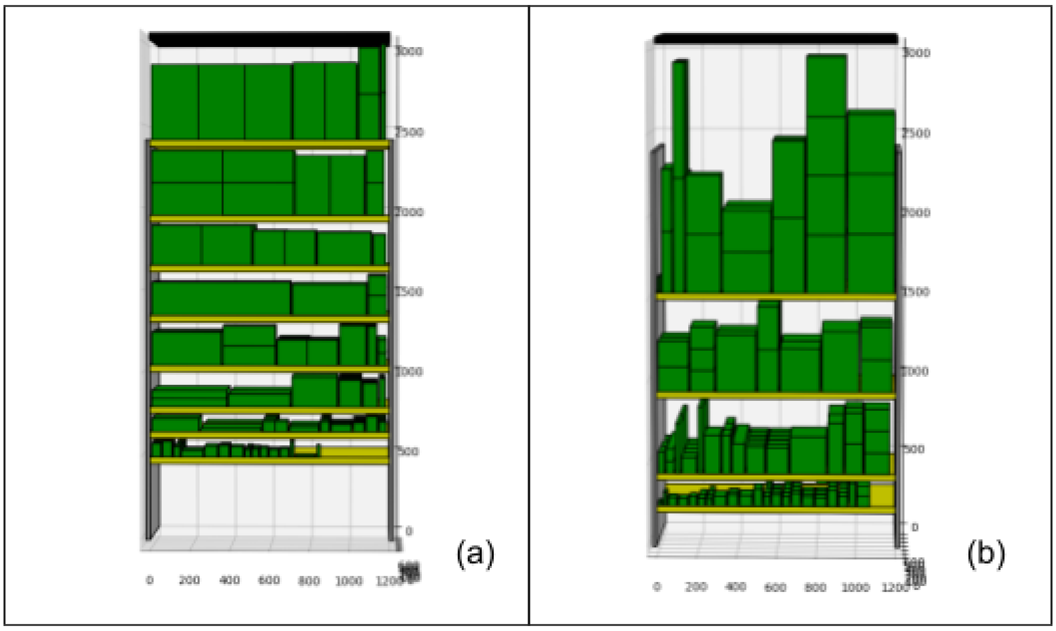


Fig 1 - Comparing best solutions of approaches for family 1.
(a) Paper approach
(b) Cumulative approach

Comparison of different search strategies

A can be seen in the fig 2, for the smallest family, the solution “a”, result of the best strategy, is the most compact of all. Solution “b”, despite using only one bay too, has more space between shelves. On the other hand, solution“c”, consuming 3 bays, is a clear example of a very poor shelving, with lots of wasted space.

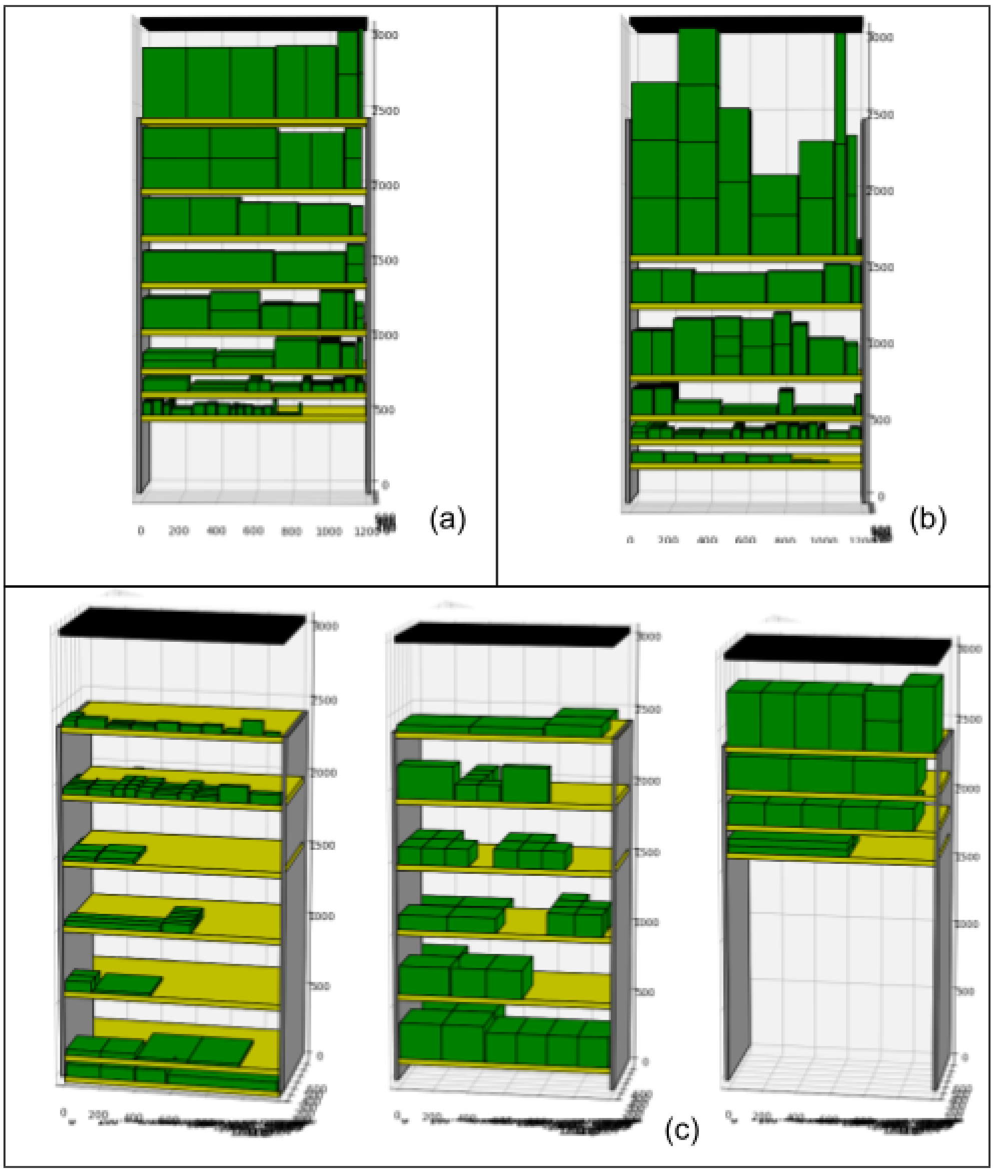


Fig 2 - Differents results for family 1 (50 products)
(a) - Weaving, leftmost, step, up - Cost: MaxH
(b)- Gs,Cs,MaxH ffc, step, down - Cost: MaxH+RemainL
(c) - Gs,Cs,MaxH ffc, bisect, up - Cost: MaxH+RemainL