Heuristics for a 2D Guillotine Cutting Problem: A Wardrobes Application

Joel Gomes and Maria Cândida Mourão

CIO – Working Paper 1/2012

Heuristics for a 2D Guillotine Cutting Problem: A Wardrobes Application [1]

Joel Gomes [a; b; 2] (joelargomes@gmail.com)

M. Cândida Mourão [a; b] (cmourao@iseg.utl.pt)

[a] Centro de Investigação Operacional, Faculdade de Ciências-UL,

Campo Grande, Bloco C6, Piso 4, 1749-016 LISBOA, PORTUGAL.

[b] Instituto Superior de Economia e Gestão, Universidade Técnica de Lisboa,

DM (gab 203Q); Rua do Quelhas, nº 6; 1200-781 LISBOA; PORTUGAL.

[1] This work is supported by National Funding from FCT - Fundação para a Ciência e a Tecnologia, under the project: PEst-OE/MAT/UI0152.

[2] Corresponding author. Address: Centro de Investigação Operacional, Faculdade de Ciências-UL; Campo Grande, Bloco C6, Piso 4, 1749-016 Lisboa, PORTUGAL. Phone: (+ 315) 217 500 027; Fax: (+351) 217 500 022

ABSTRACT

The guillotine cutting problem with two-dimensional rectangular packaging consists of allocating small items in one or more bins – objects – with such a pattern as to minimize the waste of raw materials.

The rational for this paper resides in solving a real life situation where a Portuguese company gets confronted with the problem of deciding how to efficiently cut raw materials in the process of streamlining production in a wardrobe factory.

With this aim, two new heuristics are developed and presented, the Bottom-Left First-Fit Decreasing Height (BLFFDH) and the Guillotinable Bottom-Left First-Fit Decreasing Height (BLFFDHg). Based on well-known methods - First-Fit Decreasing Height (FFDH) and Bottom-up left-justified (BL) - these new heuristics are level-oriented. More specifically, a level is first filled using FFDH, and before a new level gets opened, empty spaces in the current one are thoroughly examined, and fulfilled, if possible, with the help of BL heuristic. Only after exploring the current level can a new one be open. The difference between the two new heuristics resides in BLFFDHg ensuring a pattern which is cuttable by a guillotine. With these new methods items may not be rotated or overlapped.

The quality of the new methods is tested and analysed on both real and benchmark instances and offer quite promising results. In fact, when compared against some well-known algorithms from literature, they perform competitively. Execution times of proposed heuristics are also analysed.

Key Words – Cutting and packing problems; Level-oriented Heuristics; Guillotinable Heuristics.

1. INTRODUCTION

Cutting and packing (C&P) are similar combinatorial optimization problems, and address the allocation of a set of small items (e.g., parts, rectangles, output, demand) into a set of objects - bins (e.g., plates, sheets, input, offer), with equal or greater size. Items should be allocated in the larger objects without overlapping, and each item must be entirely contained in only one object. The pattern (or layout) produced should, for instance, minimize the total waste of raw material or input. Even considering this same objective, several problem characteristics may influence the studies. Amongst these, we should emphasize (i) the materials dimensions, which may be defined as one (1D), two (2D), three (3D) or N – dimensional (ND, N>3), all being dependent on the geometric characteristics which should be used; (ii) the number of items and objects; and (iii) the kind of patterns allowed. The recent paper of (Wäscher, Haussner, & Schumann, 2007) present a typology for C&P problems were such differences are typified.

C&P problems may address many real world situations. In fact, applications exist ranging from industry and commerce (the disposal of the articles in a newspaper; the loading of materials into containers, pallets or trucks; or cutting raw materials in industry), through areas as diverse as information technology and engineering (e.g., distribution of the computer memory). Even more abstract applications, involving non-spatial dimensions, such as weight, time or financial problems may be the focus of C&P (Silva, 2003). The present study was motivated by a real life application in a wardrobes factory. Its aim is to identify how to cut the wood needed to manufacture the wardrobes, while minimizing the total waste of raw material. The problem was classified as one of bin packing, as next explained.

This paper proposes two new heuristics for bin packing problems. When adopting (Wäscher, Haussner, & Schumann, 2007)'s typology, they can then be classified as Two-Dimensional Rectangular Single Bin Size Bin Packing Problem (2DRSBSBPP). Actually, we consider an unlimited number of objects of finite and equal dimensions, with width W and height H, and a finite number of rectangular items ($i \in N$) to pack, with widths $w_i \leq W$ and heights $h_i \leq H$ (the dimensions of the different pieces of a wardrobe). As it will be seen, the heuristics hence presented are level-oriented, since the packaging is done over a sequence of levels, a new one being opened whenever the current one cannot accommodate more items. The height of a level is given by the height of the highest item allocated to it - the first one to get allocated in the level (Hopper, 2000). The second heuristic differs from the first as it guarantees a guillotine pattern, an imposition of the real world wardrobe instances considered.

According to (Ortmann, Ntene, & Vuuren, 2010) level algorithms pack items into levels, such that the bottom edges of all items are on the floor of their respective levels. Pseudo-level algorithms also pack items into levels, but without the constraint of floor packing. Plane-packing algorithms may pack items anywhere in the plane defined by the boundaries of either the strip or the bin.

Classic strip packing level-oriented heuristics are due to (Coffman, Garey, Johnson, & Tarjan, 1980) and named as: Next-Fit Decreasing Height (NFDH), First-Fit Decreasing Height (FFDH) and Split-Fit (SF). In these methods items are left-justified in each level with no overlaps, and differ (for instance) on the sequence of items selected from an ordered list N of items.

(Coffman & Shor, 1990) introduced and studied in detail the Best-Fit Decreasing Height (BFDH), which, although similar to FFDH, instead packs the items on a level for which the residual horizontal space is the minimum. (Rode & Rosenberg, 1987) proposed the Modified First-Fit Decreasing Height (MFFDH), differing from the classic FFDH in the definition of rectangular sub-objects where smaller items may be allocated, and hence use empty spaces between different levels. With this same objective, and based on the MFFDH, (Silva, 2003) presented the Adaptive First-Fit Decreasing Height (AFFDH) and the Split First-Fit Decreasing Height (SFFDH). These methods, being more flexible, may produce sub-objects with irregular shapes. This may be the case when packed items are allocated in sub-objects with the AFFDH, or with the SFFDH where more than one sub-object can be considered at the same level and for the same purpose.

(Berkey & Wang, 1987) adapted classical approaches and present the Finite Next-Fit (FNF) and the Finite First-Fit heuristics (FFF) to solve bin packing problems. These authors also introduce the Finite Best-Strip (FBS), in which items first get allocated on a best-fit packing approach (usually used in strip packing problems), and levels are afterwards matched to an object of finite dimensions; the Hybrid First-Fit (HFF), similar to FBS but with the FFDH strategy; and the Finite Bottom-Left (FBL), a non-level-oriented heuristic.

Amongst non-level-oriented heuristics, we stress out, the Bottom-up Left-justified (BL), proposed by (Baker, Coffman, & Rivest, 1980) and improved in 1999 by Liu and Teng, according to (Bortfeldt, 2006); the Bottom-Left Fill (BLF), by (Chazelle, 1983); the Improved Lowest Gap Fill (LGFi) for oriented case and the Improved Lowest Gap Fill non-oriented (LGFi$_{OF}$) for the non-oriented case, both proposed by (Wong & Lee, 2009). These methods allocate items at the lowest (bottom) and leftmost (BL) possible position, basically differing as to the way the BL position gets found and the sequence under which items are considered.

Heuristics may also be classified as (i) one phase or (ii) two phase methods. In a one phase heuristic items are allocated directly on an object, whilst in a two phase method, a strip packing problem is firstly solved to allocate items within a level, and finally a bin-packing is solved to organize these levels in finite-dimensional objects (e.g., FBS and HFF heuristics). The proposed heuristics belong to the first group, i.e., they are one phase methods.

C&P problems may be named as offline, when the set of items to be packed is known *a priori*, before the packaging start, or online, when the next item to consider is only known after the current item gets allocated and can no longer change its position. The proposed heuristics are suitable for offline problems (Ntene & Vuuren, 2007).

This paper is organized as follows. Firstly the basic terminology needed to the understanding of the study. Secondly the two new heuristics are briefly described and illustrated with a small instance created for that specific purpose. These methods are evaluated over two sets of instances: real instances from the wardrobes study case, and benchmark instances. Results comparing the proposed methods with well-known heuristics from the literature are also presented and analysed, before conclusions get drawn.

2. THEORY/CALCULATION

2.1. Basic Terminology

This section introduces the basic terminology used throughout the article, and the instance of items created and exhibited (Figure 1) for the respective example. Items are sorted and numbered, into a list N, by non-increasing order of height, with tie break decided by non-increasing order of width. Such sorting method is used throughout the heuristics discussed in this paper.

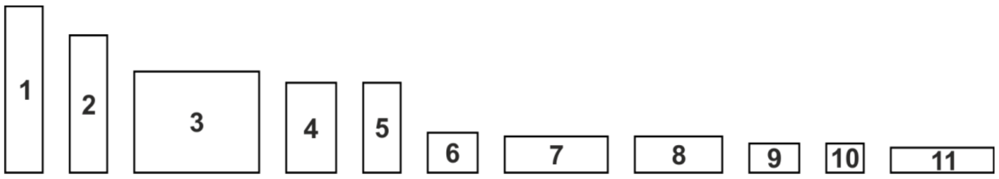

Figure 1-Instance used through definitions and the heuristics.

When cutting certain types of materials, such as glass or wood, the small items should be cut from the bin in a sequence of ***guillotine cuts***, meaning straight cuts all along the bin width or height, referred as edge-to-edge cuts, as shown in Figure 2(b) (Amossen & Pisinger, 2010). Whilst in Figure 2(a) items cannot be cut with straight cuts from a bin without cutting other items as in Figure 2(b) were the first three guillotine cuts are highlighted.

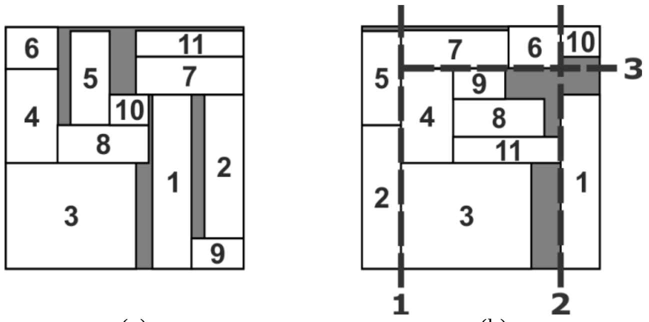

Figure 2-Non guillotine cuttable pattern (a) and guillotine cuttable pattern (b).

C&P level-oriented heuristics use the notion of level to more easily consider its several iterations. A ***level*** is therefore defined as a division of an object (or a sub-object, if the case), with the same object (sub-object) width, and a height equal to the height of its first allocated item. Two subsequent levels, l and $l+1$, are separated by an imaginary horizontal line along all the object (sub-object) width, the ***level border***, drawn coincidently to the top of the first item placed in level l, as illustrated in Figure 3.

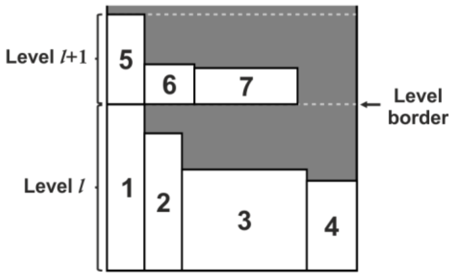

Figure 3-Definition of two subsequent levels and their respective border.

Another essential concept inherent to some methods is the ***reference point*** (RP). Represented by the coordinates (x, y), and referring to the level or object in analysis, the RP indicates the lowest leftmost point where an item may be allocated with no overlaps.

Therefore, when the first item is allocated to an object, the lower left corner of the item matches the RP (the left corner of the object). The RP then immediately becomes the lower right corner of the allocated item. If no new item can be allocated to this new point, the RP is once more update, first and whenever possible, by incrementing its abscissa before its ordinate. Figure 4 shows various RP over a possible packaging.

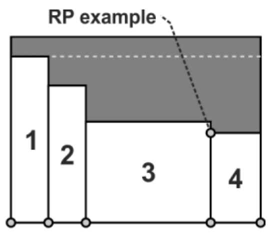

Figure 4-Example of several reference points.

In order to determine if the allocation of an item to an RP is possible, the term gap is introduced. The ***gap*** is a rectangular area, in either a level or an object, which represents the available space defined for each RP. The gap's height and width are called ***vertical gap*** and ***horizontal gap***, respectively. The vertical gap is then the length of the vertical line segment from the RP to the top of the level Figure 5(a), or to the top of the object, Figure 5(b). Its width, the horizontal gap, is the length of the horizontal line segment from the RP to the rightmost side of the object, Figure 5(a) or to the left side of the first item next allocated, Figure 5(b). Figure 5(a) illustrates a possible sequence for a level-oriented heuristic, whilst Figure 5(b) refers to non-level-oriented heuristics.

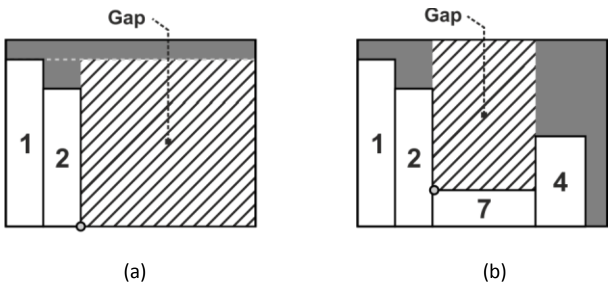

Figure 5-Gap for a given RP: (a) level-oriented algorithms, (b) non-level-oriented algorithms.

Vertical and horizontal gaps are thus essential to identify, at any time, the available space to allocate items for the RP that should be considered. Only then, a compatible item is searched and, once found, allocated over the RP. If no compatible item exists, the RP is refreshed and a new gap is calculated.

Several methods also consider small empty spaces that may be formed within a level, and named as ***sub-objects***. A sub-object has finite dimensions, here defined *a posteriori*, accordingly to the heuristic iterations. Depending on the instances and on the method itself, none to several sub-objects may be created on the empty space between the items allocated in a level and in the top of this same level, as shown in Figure 6(a).

For the methods here proposed, a sub-object, although somehow linked with the gap concept, is defined in only one level. A sub-object may be only considered when it is possible to allocate at least one item in a computed gap, and its dimensions, computed *a posteriori*, can never exceed the ones of its associated gap. Thus, its height is equal to the first item packed on it and its width is the distance between the RP and the right side of the last possible item to allocate, Figure 6(d). Figure 6 illustrates the process of setting up a sub-object. Picking a given level, and starting with the RP in (b), the gap where the sub-objet may be created is identified, and compatible items are allocated. When no more compatible items exist, in (c), the sub-object dimensions may be computed, (d). Note that meanwhile both the RP and the gap were modified.

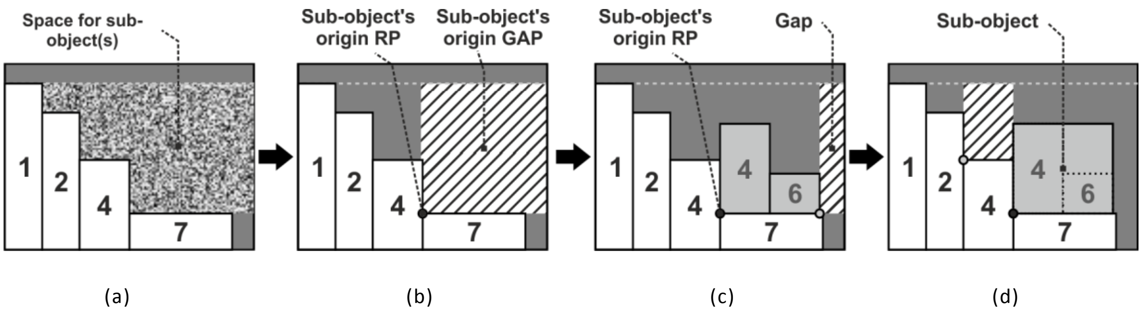

Figure 6-Design of a sub-object when the final dimensions are known *a posteriori*.

2.2. Proposed heuristics

The quality of packaging, in C&P level-oriented heuristics, may easily be improved if unused spaces between levels are considered. To this purpose several heuristics have been developed. Usually, they fill the smaller empty spaces created by applying the same techniques used for the initial levels, such as the MFFDH (Rode & Rosenberg, 1987) and AFFDH heuristics (Silva, 2003).

We propose two new level-oriented heuristics, the Bottom-Left First-Fit Decreasing Height (BLFFDH), for instances that do not require guillotine cuts, and the Guillotinable Bottom-Left First-Fit Decreasing Height (BLFFDHg), for instances with mandatory guillotinable patterns. In both heuristics, item rotation is not allowed.

These two new algorithms arise from the combination of two other heuristics, the FFDH and the BL. They include two fundamental concepts: the items, although ordered, may be allocated without respecting the sequence, i.e. item $i+1$ can be packaged before item i; it is not possible to return to previous levels, as in FFDH. Items are then chosen from the ordered list of items, N, starting with the highest one compatible with the available space. In the heuristic BLFFDHg, as in the BL, the sub-objects are positioned to maintain bottom-left stability, with some new rules, as will be explained.

2.2.1. Bottom-Left First-Fit Decreasing Height Heuristic (BLFFDH)

BLFFDH begins with the initialization of an object, called the current object. The RP is defined coincidently to the lowest leftmost point of that object ($x=0,\ y=0$), and the dimensions of the vertical and the horizontal gap are set equal to the height and the width of the object, respectively.

The packaging process first resorts to the FFDH for item allocation. The first level is defined after the allocation of the first item, and the next item to be allocated is the next in list N, sorted by decreasing item heights, and compatible with the vertical and the horizontal gaps, which represent the available space.

Every time no more items can be allocated in a given level, from an RP with the same abscissa as the one of the first item of that level, an attempt is made to fill the empty space in the current level. As next explained, to try to fill these empty spaces, the RP is updated using the BL heuristic.

The area available for allocation is analysed in order to position the RP in the lowest leftmost point with some empty space on its right, Figure 7(a). Then the gaps (vertical and horizontal) are updated, and the first unused item from list N, when existing and compatible with the gaps, gets allocated. The RP is updated once more to correspond to the lower right corner of the allocated item, Figure 7(b). If no such item exists, the RP is successively updated (considering the abscissa before ordinate) in order to either allocate a new item or to conclude that its actualization is no longer possible. In this latter case, the level is closed, a new one opened, Figure 7(c), and the process repeats itself until all items have been allocated, Figure 7(d).

The current object is closed when it cannot hold more levels. In this case, and should some items remain unpacked, a new object to accommodate these items is initialized and the process is repeated.

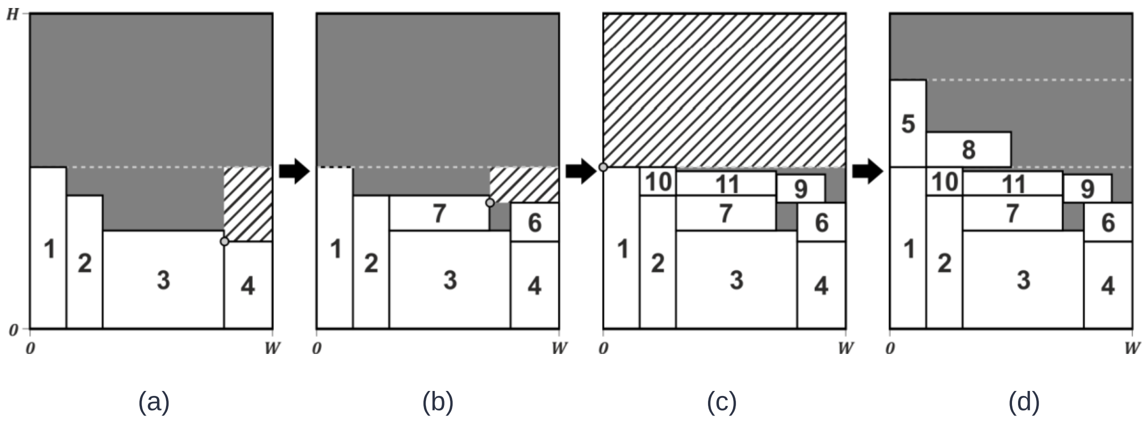

Figure 7-Packaging with BLFFDH heuristic.

2.2.2. Guillotinable Bottom-Left First-Fit Decreasing Height (BLFFDHg) Heuristic

As in the previous method, the BLFFDHg starts with the initialization of an object, called the current object. The RP is also defined as the lowest leftmost point of that object $(x = 0, y = 0)$, as well as the dimensions of the vertical and the horizontal gaps that are equal to the height and width of the object, respectively.

As before, the packaging process begins by using the FFDH heuristic, as illustrated in Figure 8. The first level is defined by allocating the first item (item 1 in Figure 8(a)). Remainder items get sequentially picked from list N according to the computed vertical and horizontal gaps (Figure 8(a)). The packing process remains in the same level as long as the empty space at the right of the last allocated item is sufficient to accommodate an item. When the first level cannot hold more items, as in after the allocation of item 4 in Figure 8(a), the RP is updated, as are both the vertical and horizontal gaps, and the empty space between the two consecutive levels is analysed.

BLFFDHg differs from BLFFDH in the way empty spaces on a level get filled. Every time that no more items can be packed in the bottom of a level, an attempt is made to identify sub-objects in the empty space of the current level. To achieve such a result, an heuristic which preserves the bottom-left stability is used, and a new RP computed if and when a corresponding new sub-object may be identified. As mentioned, several sub-objects within the same level may be generated.

Note also that a sub-object can be created over an item, or over other sub-object of the current level, using the FFDH algorithm. However, and diverging from the BLFFDH heuristic, here the RP must have the same abscissa of the referred item or sub-object. This difference between both methods is better illustrated in the comparison between Figure 8(b) and Figure 7(b).

Remember that a sub-object is defined with only one level and it is filled with the FFDH logic. Its height is given by the height of the first item packed and its width is equal to the distance from the initial RP to the lower right corner of the last item packed, as in Figure 6(d).

A sub-object is only created if it may accommodate at least one item. For this, the RP is successively updated until a new item can be packed, or until the RP cannot be updated. In the example of Figure 8(b), and after the allocation of items 6 and 7 the empty space on the right is insufficient to accommodate other items. Then, two new sub-objects are defined, one above item 6, with its same width, and another above item 2. After placing item 7, the RP is updated to the upper left corner of the item 6. Note that it is only allowed to create sub-objects in the upper left corner of other sub-objects or items that are on the floor of their respective levels (see Figure 8).

Whenever unallocated items still exist and a new RP may not be defined to accommodate a new item, the level is closed and a new one should be opened, as Figure 8(c) illustrates. The process continues until all items have been packed, Figure 8(d).

As before, the current object is closed when it cannot hold more levels or if all items have been considered. If needed, a new object is initialized and the process gets repeated until all items are packed.

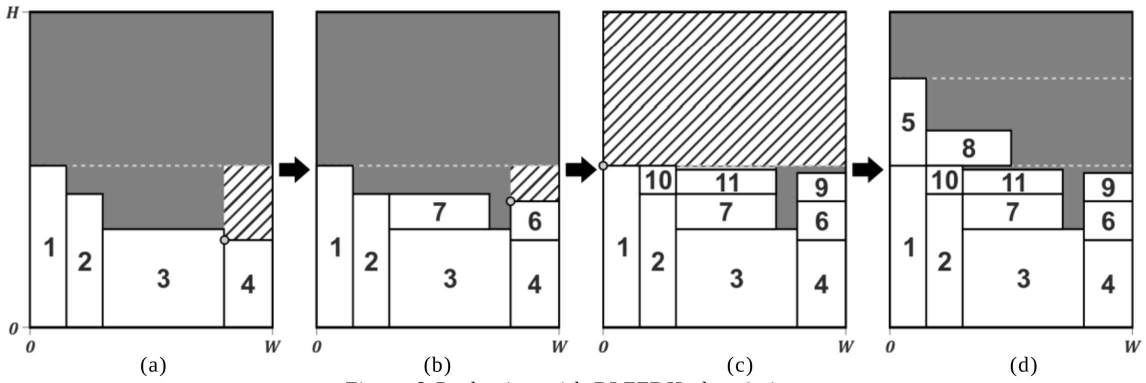

Figure 8-Packaging with BLFFDHg heuristic.

2.3. Future work

Heuristics here presented share some limitations which may well come to be the focus of future work. In some real applications it makes sense to allow the orthogonal rotation of some items. Thus, the heuristics may be adapted to accommodate orthogonal rotation of either all or a subset of items.

The guillotinable method, BLFFDHg, may be more flexible and permit sub-objects to be justified as much to the left as possible, while maintaining a guillotinable pattern. It seems that this feature would promote a better use of empty space, in certain instances.

A new challenge presents itself in finding out the heuristics limitations, by means of identifying the best instance type to apply to each method. Quality of the final solutions nevertheless depends on the type of problem being addressed, hence becoming crucial to classify methods accordingly.

2.4. Computational experiments

The quality of the solutions provided by the heuristics is evaluated from some computational tests. Usually, level-oriented heuristics performance is measured from the distance between the bottom of the object and the top of the last level, Figure 9(a), referred to as the ***maximum height*** for that object. Performance of non-level-oriented heuristics is given by the maximum of the distances between the bottom of the object and the upper edge of each item, Figure 9(b) (item 1).

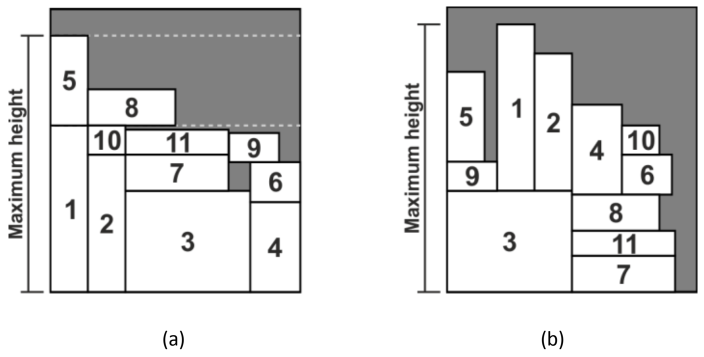

Figure 9-Maximum height in level-oriented heuristic (a) and in non-level-oriented heuristic (b).

Two sets of instances were used: wardrobe real instances and benchmark instances from literature. In all instances the items have orientation, i.e., orthogonal rotation is not allowed. For some benchmark instances the optimum is known and has a guillotinable pattern.

2.4.1. Wardrobe Instances

This database contains the historic of six jobs undertaken by a Portuguese company, where it is assumed that all pieces have similar thickness. Identified as Wardrobe Series 1 (S1R), the number of items needed to assemble the wardrobes is $|N| = 76, 245, 310, 615, 642, 1293$, respectively.

In the real instances the objects, to which the items must be allocated, have a dimension of 2740mm x 1830mm. However, and to make possible the comparison against other methods found in literature, it is considered but a single bin with sufficient height to allocate all the items.

2.4.2. Literature Instances

Five sets of benchmark instances from literature are used in the tests and summarized in Table 1. For all this instances the optimal solution is known.

Class of instances	Reference	Nr. of instances
Nice	(Wang & Valenzuela, 2001)	21
Path	(Wang & Valenzuela, 2001)	21
T-series	(Hopper, 2000)	35
T2-Kendall	(Burke & Kendall, 1999)	1
T2-J	(Jakobs, 1996)	2

Table 1-Group of literature instances used in computational tests.

The Nice and Path sets consist of 42 instances, coupled in groups of three instances each, with $|N| = 25, 50, 100, 200, 500, 1000, 2000$ items. Average results are presented for instances belonging to the same group and having the same number of items to allocate ($|N|$). An object having 100 units width and an optimal maximum height of 100 units is considered.

The set T-series consists of 35 instances, grouped in classes of five and having the same number of items, respectively, $|N| = 17, 25, 29, 49, 73, 97, 199$. For these instances the object has 200 units wide and an optimum maximum height of 200 units.

T2-Kendall instance has 13 items to pack and the object has 80 units of width and an optimum height of 140 units. T2-J has two instances (T2-J1 and T2-J2) with $|N| =$ 25, 50 items to allocate in an object with 40 units of width and an optimum maximum height of 15 units.

2.5. Results

The computational experiments were implemented under Microsoft Windows ® 7 32-bit, using Microsoft ® Visual Basic ® for Applications (VBA) version 6.5.1053 retail, implemented in Microsoft ® Excel 2010. The computer used had the following specifications: Intel ® Core ™ i3 M370 to 2.40 GHz, 4GB of DDR3 RAM and 500GB hard drive.

The proposed heuristics are compared against methods from literature, namely: BL with items ordered according the descend width (DW) and height (DH), FFDH, MFFDH, AFFDH and LGFi$_{OF}$.

2.5.1. Results for Wardrobe Instances

For these real instances, the optimum maximum height is unknown. The value used for the comparisons is thus the maximum height obtained by BLFFDHg heuristic. Values in the tables of results compare the maximum height obtained with the heuristic being tested ($H_{heuristic}$) against the maximum height obtained with the BLFFDHg heuristic ($H_{BLFFDHg}$). These comparisons are then made accordingly to the following expression:

$$Rate\ of\ Change\ (\%) = \left(\frac{H_{heuristic}}{H_{BLFFDHg}} - 1\right) \times 100, \tag{1}$$

Results are summarized in Table 2, with the average value of the best solution being highlighted in bold. This table also contains the average (μ) and the standard deviation (σ) values for the width (μ_w and σ_w) and the height (μ_h and σ_h) of the instances.

						Heuristics						
Instances	**|N|**	μ_w	μ_h	σ_w	σ_h	**BLFFDH**	**FFDH**	**MFFDH**	**AFFDH**	**LGFi$_{OF}$**	**BL**(DW)	**BL**(DH)
S1R.76	76	817.1	378.6	669.0	212.5	**0.0**	1.4	1.3	1.4	0.8	0.8	0.1
S1R.245	245	747.8	329.0	620.8	216.0	**0.0**	0.8	0.3	**0.0**	0.8	0.9	6.9
S1R.310	310	692.4	345.6	587.0	213.0	0.0	0.9	0.5	0.5	**-1.0**	0.5	6.0
S1R.618	618	686.1	345.5	590.5	198.8	**0.0**	0.9	0.8	0.8	0.1	0.4	10.7
S1R.642	642	697.7	342.6	604.4	203.0	0.0	1.0	0.9	1.0	**-0.3**	0.3	8.4
S1R.1293	1293	660.5	364.1	597.5	206.0	0.0	2.1	2.0	2.0	**-1.4**	0.3	6.6

Table 2-Rate of change in relation to maximum height of BLFFDHg heuristic.

2.5.2. Results for Literature Instances

For all instances of this group the optimal maximum height is known and used to evaluate the methods. Results reflect the quality of the method, measured by the disparity between the maximum heights achieved with the heuristic and the optimum ones. As instances are grouped, average values are considered. The following expression is therefore used:

$$Rate\ of\ Change\ (\%) = \left(\frac{\overline{H}_{heur}}{H_{opt}} - 1\right) \times 100, \tag{2}$$

with,

$$\overline{H}_{heur} = \frac{1}{m}\sum_{k=1}^{m}(H_{heur})_k \tag{3}$$

and where m is the number of instances in the same class and with the same number of items; $(H_{heur})_k$ is the maximum height achieved by heuristic *heur* and for instance k; $\overline{H}_{heur}$ is the maximum height average of the m grouped instances if heuristic *heur* is applied; H_{opt} is the optimum height for the used instance.

These values are summarized in Table 3, with the average of the best solution highlighted in bold. The average and the standard deviation of width and height of the instances are depicted in columns 3 to 6.

						Heuristics							
Instances	**|N|**	μ_w	μ_h	σ_w	σ_h	**BLFFDH**	**BLFFDHg**	**FFDH**	**MFFDH**	**AFFDH**	**LGFi$_{OF}$**	**BL**(DW)	**BL**(DH)
Nice 25	25	22.5	18.8	58.7	15.7	22.0	22.0	22.0	22.0	22.0	**16.7**	21.5	24.2
Nice 50	50	15.4	13.8	22.8	10.9	**12.6**	**12.6**	15.5	12.8	12.8	13.4	18.0	18.2
Nice 100	100	10.7	10.1	11.3	9.1	10.9	10.9	14.0	12.5	12.5	**9.5**	16.5	13.9
Nice 200	200	7.6	7.2	5.2	4.1	**7.7**	**7.7**	11.4	9.8	9.7	8.4	13.5	13.1
Nice 500	500	4.8	4.6	1.9	1.9	**5.6**	**5.6**	8.8	7.4	7.4	6.7	10.4	8.3
Nice 1000	1000	3.4	3.3	0.8	0.8	**4.0**	**4.0**	6.6	4.5	4.5	4.5	9.0	7.1
Nice 2000	2000	2.4	2.3	0.4	0.4	**3.5**	**3.5**	5.1	4.0	4.0	5.1	7.5	6.1
Path 25	25	32.9	13.9	305.7	115.4	16.5	19.3	39.2	20.9	16.0	**11.1**	23.5	16.9
Path 50	50	24.6	10.7	215.3	67.0	16.4	16.4	30.9	18.2	19.3	**11.2**	17.0	16.9
Path 100	100	16.7	9.3	133.8	58.3	10.3	11.1	31.0	17.0	15.8	**6.9**	18.3	19.2
Path 200	200	12.2	7.3	87.1	36.1	**6.5**	7.2	30.4	18.4	14.0	7.5	24.0	11.6
Path 500	500	8.1	5.5	38.1	23.4	5.2	5.7	29.7	15.1	11.2	**5.1**	19.7	7.7
Path 1000	1000	6.0	4.2	21.1	15.0	4.6	5.1	30.4	15.3	10.6	**4.5**	17.5	7.3
Path 2000	2000	4.3	3.2	11.6	8.2	3.9	4.3	29.1	15.6	10.0	**3.8**	19.6	6.5
T series T1	17	51.7	45.9	866.0	686.8	32.0	32.1	46.5	35.9	30.7	**24.5**	38.7	25.6
T series T2	25	40.7	39.7	418.9	424.2	19.6	22.9	40.8	24.2	21.0	**15.0**	31.9	24.6
T series T3	29	38.5	37.8	517.3	377.6	20.0	23.1	40.8	26.6	24.6	**12.6**	31.5	19.9
T series T4	49	28.8	27.6	237.7	273.0	**12.0**	**12.0**	27.6	18.1	16.6	14.0	27.2	14.7
T series T5	73	23.8	22.9	195.2	193.0	10.4	11.8	27.2	18.0	16.1	**9.6**	21.0	12.8
T series T6	97	19.7	19.9	175.3	193.2	9.5	**8.7**	31.7	17.1	14.6	9.3	24.3	11.0
T series T7	199	14.1	14.1	71.0	72.1	5.6	5.7	27.3	15.6	13.0	**5.1**	10.0	6.0
T2 Kendall	13	32.7	32.0	138.7	153.8	20.0	20.0	50.0	38.6	38.6	22.9	**11.4**	31.4
Jakobs T2-J1	25	5.2	4.7	3.5	1.4	26.7	26.7	40.0	26.7	26.7	20.0	**13.3**	**13.3**
Jakobs T2-J2	50	3.4	3.5	0.8	0.7	20.0	20.0	20.0	20.0	20.0	13.3	13.3	**6.7**

Table 3-Rate of change compared to the average of the optimal maximum height for each instance.

2.5.3. Execution times

Since the heuristics consume very little cpu time (in the order of milliseconds), a larger simulation cycle was created to make a comparison possible. Heuristics were set to run for a predetermined amount of time (30 minutes), the respective number of executed loops noted and used to calculate the heuristics running time.

Path and Nice instances were chosen to calculate execution times since they have greater dimensions, and differ from each other in terms of size and standard deviation. Furthermore Nice instances are more homogeneous.

	BLFFDH		**BLFFDHg**	
|N|	**Nice**	**Path**	**Nice**	**Path**
25	0.2	0.4	0.2	0.3
50	0.6	1.1	0.5	0.8
100	1.3	3.4	1.2	2.2
200	3.8	10.3	3.2	5.5
500	19.3	40.2	12.3	23.3
1000	53.7	126.5	41.7	70.7
2000	147.9	338.9	122.9	210.5

Table 4-Simulation cycle in milisseconds.

3. *DISCUSSION*

3.1. Wardrobe Instances

Wardrobe instances share similar performance throughout all the heuristics. In fact, Table 2 depicts no difference between the maximum height gathered with either BLFFDH or BLFFDHg heuristics, despite the extra guillotinable constraint. Moreover, these two methods yield better results than FFDH or BL, the base heuristics. Note also, that even if the analysis is limited to the level-oriented heuristics (i.e., BLFFDH, BLFFDHg, FFDH, MFFDH and AFFDH), it becomes evident that the two new heuristics always lead to better solutions. Actually, AFFDH – for the S1R.245 instance only – yields an equal maximum height, whilst $LGFi_{OF}$ performs better in half of the instances.

3.2. Literature Instances

For instances derived from literature, Table 3 shows that no difference was found in maximum height, between BLFFDH and BLFFDHg, for the Nice, T2-Kendall, T2-J1 and T2-J2 instances. This is not the case with other instances though where the BLFFDH outperforms BLFFDHg. Indeed, BLFFDH yields the best solution values in about 46% of the total (11 in 24 sets of instances), ties on 50% (12 in 24) and comes up with a worst solution on only 4% (1 in 24).

Table 5 depicts the results presenting them under three groups, so as to highlight its main characteristics. The first group (first line in Table 5) compares guillotinable heuristics; the second is used to level-oriented ones; and in the third group all the methods get to be considered. Values shown indicate the percentage of times each heuristic (in the column) yields the best solution, with the best mark in bold.

Best solution	**BLFFDH**	**BLFFDHg**	**FFDH**	**MFFDH**	**AFFDH**	**LGFi$_{OF}$**	**BL**(DW)	**BL**(DH)
Guilhotinable pattern	N/A	87.5	8.3	12.5	25.0	N/A	N/A	N/A
Level-oriented	87.5	54.2	8.3	12.5	20.8	N/A	N/A	N/A
All heuristics	29.2	29.2	0.0	0.0	0.0	45.8	8.3	8.3

Table 5-Best solutions percentage for each heuristic (over the total instances).

Amongst the guillotinable pattern heuristics (BLFFDHg, FFDH, MFFDH and AFFDH), BLFFDHg present the best solution in 87.5% of the cases (which corresponds to 21 over 24 sets of instances). Along comes a distant second best method, AFFDH, with 25% of best values.

In the level-oriented heuristics group, BLFFDH yields the largest percentage of best solutions, with 21 over 24, the BLFFDHg being the second best with 13 best values.

Only when all heuristics are considered do BLFFDH and BLFFDHg fail to provide the best solutions, being the best performance triggered by LGFi$_{OF}$. This provides 14 over 24 best solutions, which corresponds to about 58.3% of the cases.

In Table 3 it is visible that the average quality of the solutions, within the same group of instances, improves as the number of items increases. This characteristic is transversal to all the heuristics and probably comes from the fact that the dimensions of the object, fixed and optimal, are unchanged, while the dimensions of the items decrease with the increasing number of items.

It appears that amongst the level-oriented heuristics, BLFFDH and BLFFDHg provide better packaging for instances with bigger standard deviations (σ_w and σ_h), taking into account the average size of the items (μ_w and μ_h), particularly in Path instances.

3.2.1. Execution Time

From Table 4 we can see that BLFFDHg heuristic runs faster than the BLFFDH. This is due to the use of sub-objects. Allowing the allocation of several items in a single sub-object leads to smaller computational times. This is the case, for instance, with the search of RPs. Both heuristics have faster execution times with more homogeneous instances, the Nice.

4. CONCLUSIONS

Two new heuristics were proposed, the Guillotinable Bottom-Left First-Fit Decreasing Height (BLFFDHg) and the *Bottom*-Left First-Fit Decreasing Height (BLFFDH) to allocate rectangular items on to rectangular objects. None of the items can be rotated or overlapped. Both heuristics are bin packing type, level-oriented and suited to offline problems. The difference between them is that BLFFDHg imposes a guillotinable pattern. Both methods combine FFDH and BL heuristics, and in almost every test, provide better solutions. In terms of execution time, the BLFFDHg is faster than BLFFDH.

ACKNOWLEDGEMENTS

We want to thank National Funding from FCT - Fundação para a Ciência e a Tecnologia for supporting this article.

REFERENCES

Amossen, R., & Pisinger, D. (2010, Fevereiro). *Multi-dimensional bin packing problems with guillotine constrains.* Retrieved Abril 1, 2011, from Rasmus Resen Amossen: http://rasmus.resen.org/pub/ap10_opp_csp.pdf

Baker, B., Coffman, J. E., & Rivest, R. (1980). Orthogonal packings in two dimensions. *SIAM J. Comput., 9 (4)*, 846-855.

Berkey, J., & Wang, P. (1987, Maio). Two-dimensional finite bin-packing algorithms. *The Journal of the Operational Reserch Society, 38 (5)*, 423-429.

Bortfeldt, A. (2006). A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research, 172*, 814-837.

Burke, E., & Kendall, G. (1999). *Listing Gallery: Data Sets 2D - Rectangular.* Retrieved August 3, 2011, from EURO Special Interest Group on Cutting and Packing: http://paginas.fe.up.pt/~esicup/tiki-download_file.php?fileId=137

Chazelle, B. (1983). The bottom-left bin-packing heuristic: an efficient implementation. *IEEE Transactions on Computers, 32 (8)*, 697-707.

Coffman, J. E., & Shor, P. W. (1990). Average-case analysis of cutting and packing in two dimensions. *European Journal of Operational Research, 44*, 134-144.

Coffman, J. E., Garey, D., Johnson, D. S., & Tarjan, R. (1980). Performance bounds for level oriented two-dimensional packing algorithms. *SIAM Journal on Computing, 9 (4)*, 808-826.

Hopper, E. (2000). *Listing Gallery: Data Sets 2D - Rectangular.* Retrieved August 3, 2011, from EURO Special Interest Group on Cutting and Packing: http://paginas.fe.up.pt/~esicup/tiki-download_file.php?fileId=131

Hopper, E. (2000). *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods.* A Thesis submitted to the University of Wales for the Degree of Doctor of Philosophy, University of Wales, Cardiff, School of Engineering.

Jakobs, S. (1996). *Listing Gallery: Data Sets 2D - Rectangular.* Retrieved August 3, 2011, from EURO Special Interest Group on Cutting and Packing: http://paginas.fe.up.pt/~esicup/tiki-download_file.php?fileId=138

Ntene, N., & Vuuren, J. H. (2007, Janeiro 27). *A survey and comparison of heuristics for the 2D oriented on-line strip packing problem.* Retrieved Março 1, 2011, from Jan H van Vuuren Research Publications: http://dip.sun.ac.za/~vuuren/papers/onlinestrip.pdf

Ortmann, F. G., Ntene, N., & Vuuren, J. H. (2010). New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems. *European Journal of Operational Research, 203*, 306-315.

Rode, M., & Rosenberg, O. (1987). An analysis of heuristic trim-loss algorithms. *Engineering Costs and Production Economics, 12*, 71-78.

Silva, H. (2003). *Heurísticas orientadas à camada para problemas de empacotamento rectangular.* Master Thesis, Escola de Gestão do Porto - Universidade do Porto.

Wang, P., & Valenzuela, C. (2001). Data set generation for rectangular placement problems. *European Journal of Operational Research, 134(2)*, 378-391.

Wäscher, G., Haussner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research, 183*, 1109-1130.

Wong, L., & Lee, L. (2009). Heuristic placement routines for two-dimensional bin packing problem. *Journal of Mathematics and Statistics 5 (4)*, Páginas 334-341.

APPENDICES

Wardrobe instances can be obtained from http://sites.google.com/site/orheuristic/