TITLE: A survey on heuristics for the two-dimensional rectangular strip packing problem
AUTHORS: Jose Fernando Oliveira; Alvaro Neuenfeldt, A ; Elsa Silva; Maria Antónia Carravilla;
SOURCE: Pesquisa Operacional, VOLUME: 36, ISSUE: 2, PAGES: 197-226, PUBLISHED: 2016
DOI: http://dx.doi.org/10.1590/0101-7438.2016.036.02.0197

A Survey on Heuristics for the Two-Dimensional Rectangular Strip Packing Problem

José Fernando Oliveira*†, Alvaro Neuenfeldt Júnior*, Elsa Silva *, Maria Antónia Carravilla*

* INESC TEC and Faculty of Engineering, University of Porto

† Corresponding author: jfo@fe.up.pt

Abstract

Two-dimensional rectangular strip packing problems belong to the broader class of Cutting and Packing (C&P) problems, in which small items are required to be cut from or packed on a larger object, so that the waste (unused regions of the large object) is minimized. C&P problems differ from other combinatorial optimization problems by the intrinsic geometric constraints: items may not overlap and have to be fully contained in the large object.

This survey approaches the specific C&P problem in which all items are rectangles, therefore fully characterized by a width and a height, and the large object is a strip, i.e. a rectangle with a fixed width but an infinite height, being the problem's goal to place all rectangles on the strip so that the height is minimized.

These problems have been intensively and extensively tackled in the literature and this paper will focus on heuristic resolution methods. Both the seminal and the most recent approaches (from the last decade) will be reviewed, in a rather tutorial flavor, and classified according to their type: constructive heuristics, improvement heuristics with search over sequences and improvement heuristics with search over layouts.

Building on this review, research gaps are identified and the most interesting research directions pointed out.

Keywords: Two-Dimensional Rectangular Strip Packing Problem, Cutting and Packing, Heuristics.

1 Introduction

The Strip Packing Problem (SPP) aims to pack a set of small items inside a larger object, the container, with all dimensions but one fixed, with the objective of minimizing the free dimension of the large object. The small items cannot overlap each other and must be completely inside the large object. This description fits the definition of a Cutting and Packing (C&P) problem and indeed the SPP can be classified as an Open Dimension Problem according to Wäscher et al. (2007) typology for C&P problems.

Other well-known C&P problems are the knapsack problem, the bin-packing problem, the container loading problem, the cutting-stock problem and the pallet loading problem. The SPP can be found in many practical settings and has many real-world applications in industry and services, such as the cut of wood boards, steel plates or paper rolls and also multidimensional resource scheduling.

One of the most distinctive characteristics of C&P problems is the fundamental role of geometry in their definition and resolution. When dealing with cutting of raw-material or item packing, three dimensions (length, width and height) are always present. However, for modeling and resolution purposes, C&P problems are usually classified in one-, two- and three-dimensional problems, when taking into account the number of dimensions that are relevant for the problem. One should bear in mind that the dimensions that are common to the small items and the large object are not relevant for the optimization problem. In this review two-dimensional SPP problems will be tackled.

Geometry also plays an important role in the description of small items' and large object's shapes. Two-dimensional problems are usually divided into rectangular, circular and irregular problems. The first two categories are self-explanatory and the last one refers to problems in which the small items have polygonal shapes, at least in the most common form of these problems. In this paper, problems in which both the small items and the large object are rectangles will be approached. Conjugating the rectangular shape of the problem with the open dimension characteristic, the large object will be a rectangle with a given width and an infinite height (hereafter called strip), and the objective of the problem is to minimize the actual height used to pack all small items.

As any combinatorial optimization problem, the SPP can be solved by exact approaches (in the sense that the approach provides a guaranteed optimal solution, being usually based on mathematical programming models), approximation methods (heuristics and metaheuristics) or hybrid methods (e.g. matheuristics), resorting to elements from both worlds. Given the complexity of these problems, heuristic methods, providing good solutions in a fairly small amount of time, have been rather popular in the field, being able to solve problems with many items.

This paper will focus on heuristic methods for the two-dimensional rectangular strip packing problem. The relevant literature will be reviewed and links between the most frequently used methods and the data characteristics will be sought.

Surveys have been published in the past on the SPP. In Hopper & Turton (2001*a*) a review of metaheuristic approaches, with a special emphasis on genetic algorithms but also looking at simulated annealing, tabu search, and artificial neural networks, can be found. Lodi et al. (2002) discuss exact and approximate methods for both the rectangular Bin-Packing Problem and the rectangular SPP, giving special attention to the level (guillotinable) SPP. With the same scope, Ntene & van Vuuren (2009) discuss heuristics previously published in the literature, in order to provide a comparison basis for their own new approach. Riff et al. (2009) also review both exact and heuristic methods for the two-dimensional SPP and present some benchmark instances commonly used in the literature in the computational validation of algorithms for specific instances.

This survey will not only update the previous reviews but will also provide a classification

of both constructive and improvement heuristics, looking at characteristics they may have in common. Metaheuristic approaches will be referred and discussed along the text, but focusing on the underlying heuristics and not on the search methods that characterize and distinguish each one of them. The decomposition of the heuristics in their building blocks, together with the study of the past research effort that will be presented, will help to identify research gaps in the field.

The paper is organized as follows. Section 2 fully presents and characterizes the SPP. In section 3 the heuristic methods found in the literature are presented, while in section 4 the most relevant findings are presented and discussed, together with some future research ideas and proposals.

2 The Strip Packing Problem

The SPP is a C&P problem that is classified as an Open Dimension Problem under Wäscher et al. typology (Wäscher et al. 2007). However it can be further characterized, as the typology itself anticipates. Not only the dimensionality of the problem and the geometry can be taken into account (in this survey two-dimensional problems where the small items are rectangles will be reviewed), but further characteristics regarding the way how the rectangles are laid out on the strip can be considered for the problem categorization. The actual disposition of the rectangular small items on the strip is referred to as pattern (either cutting or packing pattern) or layout. The characteristics that may be imposed to the patterns arise from the practical constraints that the different real-world applications require. These characteristics are summarized in Figure 1.

The first level considers the rectangle list input. In the online version, rectangles are input one by one without any information regarding the following rectangle. Once the rectangle is placed on the strip there is no possibility of repositioning it (Ye et al. 2009). Not knowing which will be the next rectangle arises for instance in scheduling problems, when a sequence of tasks or jobs have to be processed in a given number of machines but the tasks are not known in advance. In fact, scheduling problems can be modeled as SPP if the height of the strip (the open dimension) is associated to time and the strip's width to the machine capacity. Each task requires a given amount of capacity and takes a given amount of time to complete, defining therefore a rectangle. The goal is to process all tasks in the minimum amount of time, i.e. to minimize the height of the strip (Hurink & Paulus 2011).

However, the most frequent in the literature (and in real-world applications) is the offline case. In this situation it is possible to define criteria for the order by which the rectangles will be placed on the strip, as length, width, area or perimeter, once all characteristics of the rectangles are known in advance. Nevertheless, this ordering possibility is not necessarily used by the solution approaches and random orders are also frequent in the literature.

The second level concerns the geometry of the small items. In practical contexts SPP problems arise with small items having irregular shapes (e.g. polygons), circular shapes or rectangular shapes, which is the focus of this survey. Directly related to the geometry

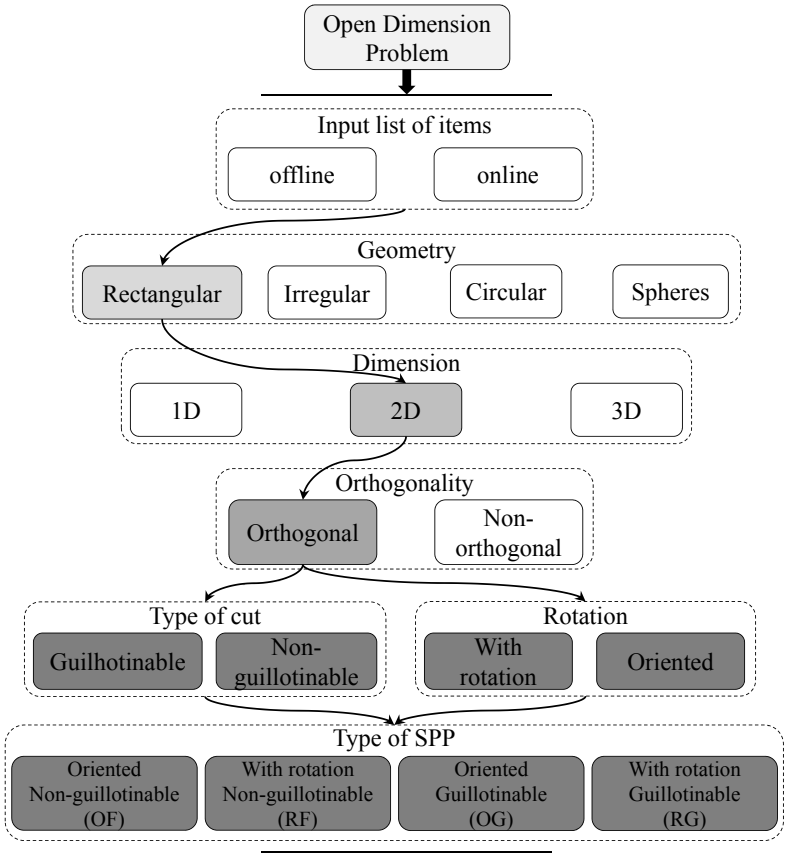


Figure 1: Classification structure for the SPP.

is orthogonality. In orthogonal patterns the edges of the rectangles are parallel to the edges of the strip. When the orthogonality constraint is not imposed, the problem is treated as an irregular SPP, even if the shapes are rectangular, and therefore it is out of the scope of this survey.

The following characteristic is the type of cut. When guillotine cuts are considered the pattern has to be formed so that it is possible to obtain the requested rectangles by cutting from edge to edge, either the strip's edges or the edges of the sub-rectangles generated by previous cuts. In fact the name arises from the applications where the cutting tool is a guillotine. This same constraint has to be imposed when patterns are to be cut by saws, such as in the wood industry. Patterns that do not have this property are designed as non-guillotinable patterns.

A further characterization of guillotinable patterns arises from the number of stages or phases required to fully cut the pattern. If, using only a succession of horizontal (or vertical) cuts, all rectangles are extracted from the strip, the pattern is called "one-stage

guillotinable". If it is necessary to orthogonally rotate the parts resulting from the first stage cuts, and apply a second series of guillotine cuts (now vertical or horizontal), the pattern is designated as a "two-stage guillotinable pattern". Following the same line of reasoning a three-stage guillotinable pattern requires a further change in the cutting direction. Patterns with no limit on the number of stages are called n-stage guillotinable patterns, where $n-1$ holds for the number of cutting direction changes (Imahori & Yagiura 2010). Guillotinable patterns may also be classified as level packing patterns if the first cuts are horizontal (across the width of the strip) and the rectangles are then extracted from the generated levels (Coffman Jr. et al. 1980, Mumford-Valenzuela et al. 2004). Each level has the height of the tallest rectangle placed there. Some examples of these patterns are represented in Figure 2.

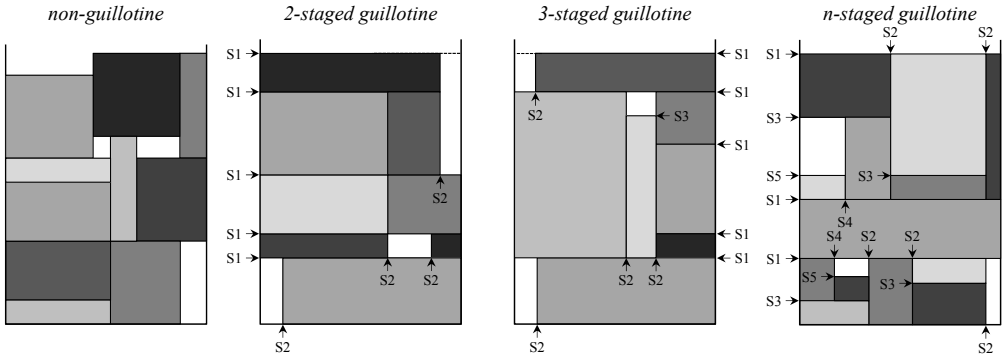


Figure 2: Examples of cutting patterns.

Regarding the orientation of the rectangles, when the materials have vein or other physical properties that make the width and height directions different from each other (e.g. when cutting striped fabrics in the garment industry), the rectangles have to be cut as they are defined, i.e. width along the strip's width and height along the strip's height. However, there are materials that are isomorphic along the two directions and therefore the rectangles can be rotated, i.e. it is indifferent to cut the rectangle's width along the strip's width to the strip's height, and vice-versa. In this case it is said that the rectangles may be 90° rotated or more simply, rotated.

Combining the rectangle orientation constraints and the types of cuts, four sub-problems can be defined (last level of the characterization structure). The less constrained problem is the RF where the rectangles may be rotated and no constraints on the type of cuts are imposed. Fixing the rectangles' orientation defines the sub-problem OF. When imposing the guillotine cut constraint the sub-problems RG and OG are defined, respectively allowing for the rotation of the rectangles or fixing their orientation, (Lodi et al. 1999). These two sub-problems may be further refined if the number of stages is considered, as previously explained.

Given the real-world origin of these problems, additional constraints and characteristics may appear in the literature, motivated by different cutting technologies and material properties, but the most commonly tackled are the ones described above.

3 Heuristics

Heuristic algorithms do not find guaranteed optimal solutions and are not able to recognize an optimal solution if one is found. Additionally, no distance to the optimal solution can be computed or guaranteed. However, given the low computational times, when compared to exact methods, they are considered to be an efficient resolution strategy for large instances of hard combinatorial optimization problems. Heuristic algorithms are usually divided into constructive heuristics and improvement heuristics. The most representative rectangular two-dimensional SPP heuristics will be described in detail under this division.

3.1 Constructive heuristics

Constructive heuristics build solutions by adding elements one by one. Only in the end of the construction process a complete solution is obtained and if the process is interrupted no feasible solution is generated. In the SPP context, the rectangles are successively placed on the strip's free/empty spaces or regions.

In general, the older constructive heuristics resort to simple placement mechanisms, with a low computational complexity, and therefore are less able to check for empty spaces in the strip. The improvement of this capacity became the most important trend in the development of the more recent heuristics.

When analyzing the solution construction methodologies used in the literature, four strategies were found: "positioning', "fitness", "level" and "profile". This classification is based on the conceptual idea behind the actual placement of the rectangles on the the strip and further details are presented in the following sections.

3.1.1 Positioning-based heuristics

Positioning-based heuristics are the oldest in the SPP literature, and most common in the early works. They are rather flexible, allowing to incorporate the most usual constraints of the problem. The basic mechanism behind positioning-based heuristics is the identification of the free space on the strip that is most suitable for a given piece, according to an also given criterion.

The "Bottom-Left (BL)" heuristic, proposed by Baker et al. (1980), was not only the first positioning-based heuristic for the SPP but is also the most widely and well-known approach to this problem. The goal is to place each rectangle the lowest and to the left as possible, as illustrated in Figure 3. An initial, feasible position is assigned to the rectangle, and then it is alternately moved down and to the left. A direction change only occurs when moving along the current direction becomes unfeasible.

With a complexity $O(n^2)$ (where n is the number of rectangles), the main advantage of the bottom-left strategy is being very fast. However, one of its main drawbacks is that it is unable to fill "holes", i.e. empty spaces that are surrounded by rectangles that were previously placed. To overcome this limitation several modified versions of

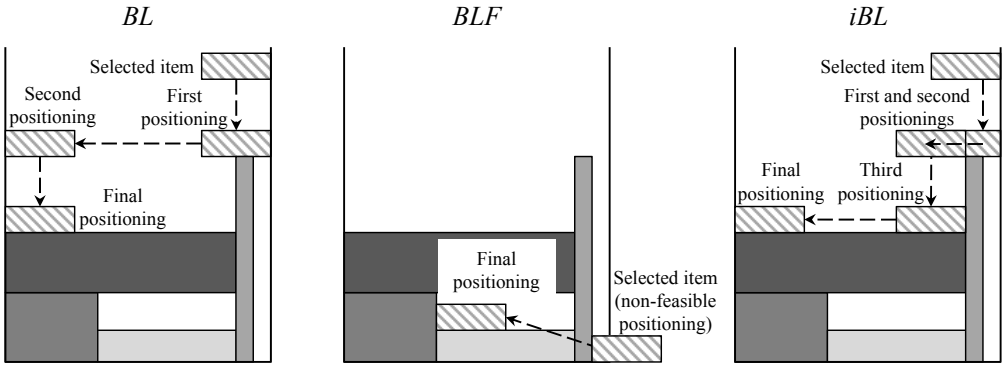


Figure 3: Examples of application of the bottom-left (BL), bottom-left-fill (BLF) and improved bottom-left (iBL) placement strategies.

the bottom-left heuristic were proposed along the years. The most relevant one is the "Bottom-Left-Fill (BLF)", developed by Chazelle (1983).

As in the bottom-left heuristic, the bottom-left-fill heuristic aims to place the rectangles the lowest and to the left as possible. However, as illustrated in Figure 3, spaces between already placed pieces are considered to be admissible. The search for empty feasible (in the sense that the rectangle fits the space) spaces is much more complex and therefore the bottom-left-fill heuristic has a complexity of $O(n^3)$, but for the same sequence is always equal or better than the bottom-left heuristic. The implementations of the bottom-left-fill heuristic places the rectangle on a position that is not feasible, because it overlaps other rectangles that had been already placed (e.g. at the bottom-left corner of the strip), and then moves the rectangle right and up until a feasible position is found (Figure 3).

The "Improved Bottom-Left (iBL)" heuristic, proposed by Liu & Teng (1999), gives always priority to the down movement, i.e. when it is not possible to move the rectangle downwards it moves it to the left, but just the distance necessary to make again a downward movement (Figure 3).

Common to these and other positioning-based heuristics is their dependence of the order by which the rectangles are placed, as the focus of these approaches is to find the best placement for a given rectangle.

In order to decrease this dependence from the rectangle input order Zhang, Shi, Leung & Wu (2016) have very recently further developed a recursive heuristic previously proposed by the same authors (Zhang et al. 2006), the "Priority Heuristic (PH)", and dynamically assign a priority to each rectangle. This recursive algorithm successively divides the strip into rectangles and sub-rectangles and when choosing which rectangle to place in a given empty rectangle, this choice does not depend just on the rectangle input sequence but on how well the rectangle fits the space (the "priority"). In Zhang et al. (2007) the authors has already resorted to the recursive algorithm proposed in (Zhang et al. 2006), but decomposing the strip into layers and filling each layer by a simple positioning-based constructive heuristic.

In the nineties Dowsland et al. (1998) have proposed a rather innovative heuristic for the irregular two-dimensional strip packing problem, named "Jostle". In this heuristic a bottom-left-fill and a up-right-fill (an artificial top edge is defined for the strip) heuristic are alternately applied, "throwing" the pieces up and down. But the piece sequence is defined by the position of each piece in layout generated in the the previous iteration. When applying the bottom-left-fill heuristic, the pieces are ordered by decreasing y-coordinates, regarding the previous layout, while when applying the top-right-fill heuristic, the pieces are ordered by increasing y-coordinates, also regarding the previous layout. The general idea is that the pieces that are farthest away in the previous layout should now be placed first. The possibility of filling holes in the middle of the already placed pieces has revealed to be critical for the efficacy of this approach. Wauters et al. (2013) test this idea on the rectangular two-dimensional SPP and name the algorithm "Improved Deepest Bottom Left Fill (iDBLF)". Alternative ways of generating sequences from the actual layouts are proposed and tested, as well as alternatives to the bottom-left-fill/top-right-fill heuristics.

3.1.2 Fitness-based heuristics

Fitness-based heuristics are focused on the free spaces, i.e. the best-fit between the rectangle to place and the empty spaces is sought. It is common to give some priority to the spaces closer to the edges of the strip (left, right or bottom).

The first constructive heuristic that has explored this mechanism was the "Best-Fit (BF)" heuristic by Burke et al. (2004). The selection of the rectangle to place is dynamic and depends on the space to be filled. The lowest free space is selected and the first choice will be a rectangle that perfectly fits this space. If there is no rectangle with this characteristic, the largest rectangle that fits the space is selected and placed according to one of three different policies: leftmost policy; next-to-the-tallest policy; and next-to-the-shortest policy (Figure 4). In the leftmost policy the rectangle is always placed at the left of the space, while in the other two policies the placement inside the space depends on the previously placed rectangles. In the next-to-the-tallest policy the side of the empty space where the tallest rectangle is, is chosen, while in the next-to-the-shortest policy it is the other way around.

The implementation proposed by Burke et al. has a time complexity of $O(n^2 + nW)$. However, Imahori & Yagiura (2010) propose an implementation based on balanced binary search trees with a time complexity of $O(n\ log\ n)$. This efficient implementation of the best-fit heuristic decomposes the "skyline" formed by the already placed rectangles into line segments and stores them in a heap structure using their y-coordinate as the key to access the memory structure. At the same time these same segments are stored in a doubly linked list ordered by their x-coordinates. The authors not only prove that the time and space complexity achieved are optimal, but also provide a worst-case approximation ratio (W-C appr.) for the best-fit heuristic. Computational experiments demonstrate the capacity of efficiently solving extra-large instances ($n > 1000000$). Verstichel et al. (2013) use these data structures in their implementation of the best-fit algorithm, which is named "Three-way Best-Fit (T-w BF)" heuristic, and add three new placement criteria: place the fitting rectangle at the right-hand side of the space; place the fitting rectangle so that the difference of the top edge with its neighbour is maximal; place the fitting

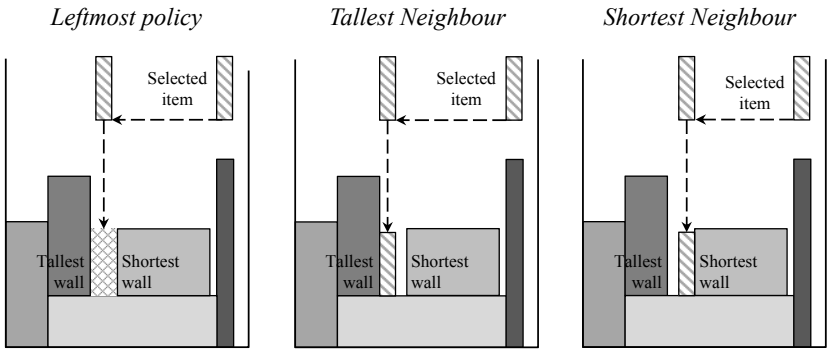


Figure 4: Example of application of the best-fit (BF) heuristic.

rectangle so that the difference of the top edge with its neighbour is minimal.

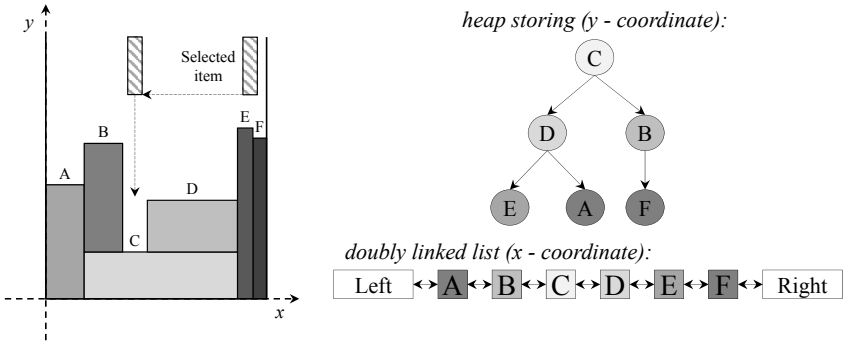


Figure 5: Memory structures to efficiently implement the best-fit heuristic.

Another improvement to the best-fit heuristic was proposed by Aşık & Özcan (2009): the "Bidirectional Best-Fit (BBF)" heuristic. In this approach not only the spaces among already placed rectangles are considered as feasible placement places for the following rectangles, but also what the author calls vertical spaces, i.e. spaces on the top of previously placed rectangles and with no side neighbour rectangles, are considered. The space is closed at the top by what is designated as the "expected best height" (Figure 6). The "expected best height" is the lower bound on the height and is obtained by dividing the sum of the areas of all the rectangles by the width of the strip. Vertical niches are also identified giving to the heuristic a spatial two-dimensional awareness that the original proposal did not have, once it is focused on the x-coordinate. In Özcan et al. (2013) the authors extend this work to the "Modified Bidirectional Best-Fit (BBFM)" heuristic, by placing the rectangles on the strip in groups, which created by looking at the geometric similarities of the rectangles, instead of placing them one by one.

As usually rectangles do not perfectly fit the available empty spaces, the approaches described until now tend to generate many small spaces that, for large instances, deteriorate

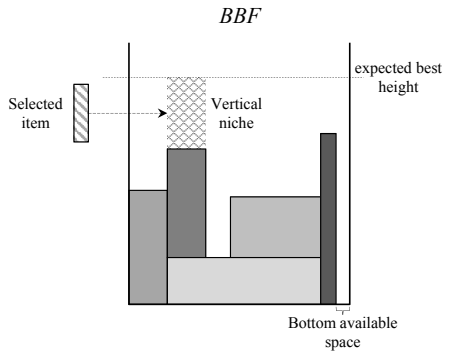


Figure 6: Example of application of the bidirectional best-fit (BBF) heuristic.

significantly the efficiency when seeking for the best placement for a given rectangle. To overcome this drawback Leung & Zhang (2011) developed the "Fast Layer-Based Heuristic (FH)" that aims to keep the layout skyline as flat as possible. To achieve this, the placement strategy favors placements that eliminate (or least increase) the number of edges of the skyline.

To achieve the flat skyline goal, the quality of the relationship between a free space and a rectangle to be placed is assessed and is designated as the *fitness value*. This measure takes values between a maximum of 4, when there is a perfect match between the space and the rectangle, and a minimum of 0. In Figure 7 examples corresponding to the fitness values of 0 to 3 are presented. In each layout construction step the fast layer-based heuristic selects the pair rectangle/space that maximizes the fitness value.

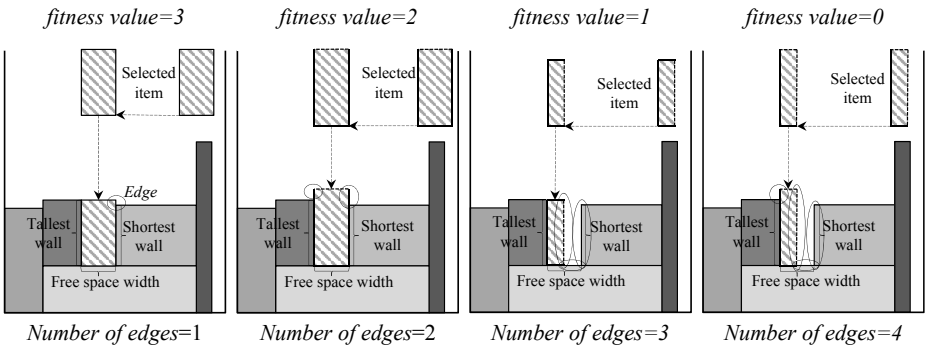


Figure 7: Examples of fitness values computed by the fast layer-based heuristic (FH).

Several authors have worked on improving this scoring methodology. Leung et al. (2011) in the "Intelligent Search Algorithm (ISA)" resort to 5 scoring values, between 0 and 4, as does Chen et al. (2015) ("Hybrid Demon Algorithm (HDA)"), while Yang et al. (2013) in the "Simple Randomized Algorithm (SRA)" considers 8 scoring values, between -1 and 6, as does Wei, Qin, Cheang & Xu (2014) ("Efficient Intelligent Search Algorithm (IA)")

that also uses a 8 value scoring system. It is worthwhile to notice that these different scoring values correspond to different valuations of different configurations of the skyline.

A similar idea, of flattening the skyline, is followed by Kotov & Cao (2011) that restrict the potential placement points to the concave vertices of the skyline, aiming its flatness.

3.1.3 Level-based heuristics

The central idea of these heuristics is the placement of the rectangles on levels, i.e. parallel guillotine cuts across the width of the strip. The height of each level i is defined by the tallest rectangle h_i placed on that level. The layouts generated by these heuristics are not comparable with the ones generated by the heuristics described in the previous sections. They address specific real-world problems in which the level orientation of the layouts is a constraint, as for instance when planning the display of goods on supermarket shelves.

Proposed by Coffman Jr. et al. (1980) the "Next-Fit Decreasing Height (NFDH)" heuristic sorts the rectangles by non-increasing height and places them one by one on the currently open level and at the leftmost admissible position. The currently open level is closed when it does not have enough space to accommodate the candidate rectangle, a new level is opened and the candidate rectangle is placed on this level, next to the left edge of the strip (Figure 8).

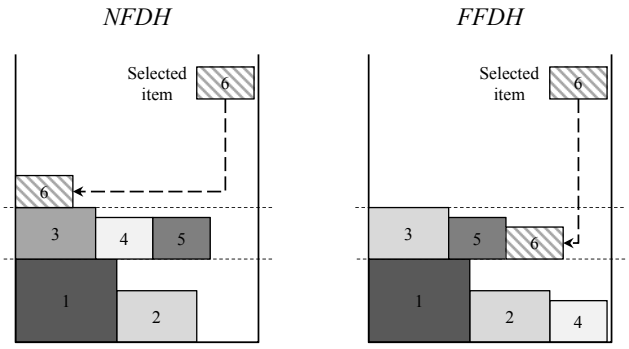


Figure 8: An example of application of the next-fit decreasing height (NFDH) and the first-fit decreasing height (FFDH) heuristics (the numbers represent the order by which the rectangles were placed).

A clear limitation of this heuristic, and a source of space waste, is that free spaces of closed levels cannot be later on used by smaller rectangles. The "First-Fit Decreasing Height (FFDH)" heuristic, proposed by the same authors (Coffman Jr. et al. 1980), addresses this drawback by not closing levels, i.e. each rectangle is placed at the lowest possible level as long as it fits. As depicted in Figure 8, a new level is created only if the rectangle does not fit any previous level. A variation of this heuristic was proposed by Berkey & Wang (1987): the "Best-Fit Decreasing Height (BFDH)" heuristic. In this approach the rectangle is not placed at the lowest level where it fits, but at the level,

among those where it fits, for which the unused horizontal space is minimum.

The previous heuristics sort the rectangles by non-increasing height, as their names suggest. Therefore, instances where consecutive rectangles, which most probably will be placed on the same level, have very different heights lead to a very poor performance of these approaches as the taller rectangles will define a height for the level that can not be afterwards met by the shorter ones. It should be kept in mind that these heuristics are intrinsically uni-dimensional, trying to make good choices along the width, and only look at the height to determine if the rectangles fit or not in the level.

To address this poor behavior Ntene (2007) have more recently developed the "Size Alternating Stack (SAS)" heuristic. The rectangles are divided in two lists (L_1 and L_2). List L_1 consists of all rectangles that are strictly taller than wider (*narrow* rectangles), and list L_2 has the rectangles that have a width equal or greater than the height (*wide* rectangles). The first list is ordered by decreasing height while the second list is ordered by decreasing width. The main idea behind the size alternating stack heuristic is to alternate between *narrow* and *wide* rectangles. Each level is initiated by comparing the heights of the first rectangles of each list and the tallest one is selected for placement, hence defining the level height. The following rectangle is taken from the list that did not open the level and additionally rectangles from this list are placed on the top of each other, forming a column, until the level height is reached. Only afterwards a new rectangle is taken from the alternate list. The resulting layout is a set of stacks of rectangles organized by levels. In Ortmann et al. (2010) this work is further extended and combined with ideas from Ntene & van Vuuren (2009), originating the "Modified Size-Alternating Stack Algorithm (SASm)", and with the floor ceiling heuristic from Lodi et al. (1999), resulting in the "Stack Ceiling with Re-sorting algorithm (SCR)".

A similar extension of the idea of level is used by Cui et al. (2008). The strip is divided into levels (sections, according to the terminology of these authors), the height of each level is equal to either the height or the width (rectangles may be rotated) of the rectangle placed at the bottom-left corner of the level, but the remaining region of the level may be freely occupied by any set of rectangles, without any additional constraint. A recursive function is proposed to find the "best" occupancy of the this region. This approach is further developed in Cui et al. (2013) in the "Sequential Grouping and Value Correction Procedure (SGVCP)" by assigning to each rectangle a value not proportional to its area, trying to guide the recursive function towards the use of arrangements that led to good partial solutions in previous levels.

In Bortfeldt & Jungmann (2012) the authors propose the "Strip Packing by Tree Search (SPTRS)", an adaptation of the tree search algorithm developed by Fanslau & Bortfeldt (2010) for the three-dimensional container loading problem, also based on the idea of sectioning the strip in layers.

Concluding, although rooted in the eighties, research on level-oriented heuristics is still active, as shows the recent paper Buchwald & Scheithauer (2016) where an improved version of the FFDH heuristic is proposed, the "Modified First-Fit Decreasing Height (FFDH*)".

3.1.4 Profile-based heuristics

Scheithauer (1997) introduced for the first time the concept of profile (the "contour" in these authors' terminology) to describe a partial solution. The contour is a polygonal line starting on the left side of the strip and ending on the bottom or on the right side of the strip, composed by vertical and horizontal edges. These edges are either edges of rectangles already placed or an extension of those edges. Globally the contour looks like a staircase. The orthogonal polygon defined by this line and the sides and bottom of the strip fully contains all placed rectangles (Figure 9). This representation is used by Scheithauer to build exact methods, which rely on the implicit enumeration of the placement sequences and of the contour's corner points (the feasible placement points). It is revisited by Martello et al. (2003), in the same context of the exact resolution of the SPP, but later on its potential for the development of heuristics for this same problem was recognized by other authors and used to develop new solution approaches.

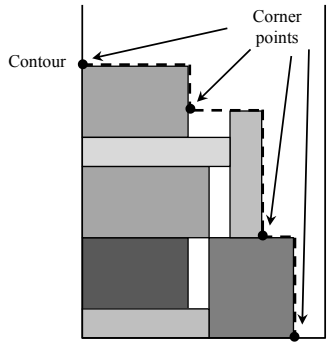


Figure 9: Example of a contour line, used to describe partial solutions in profile-based heuristics.

Pisinger (2007) use sequence pairs to represent the solutions (the placed rectangles), and the partial solution profile and the corner points to decide where to place the following rectangle. The use of sequence pairs to represent two-dimensional packing solutions was proposed by Murata et al. (1996), and further used by other researchers. As the name suggests sequence pairs are two sequences of rectangles, each of it representing the order by which the rectangles appear to a fictitious observer located at two different points of the space, namely at the upper-left corner of the layout and at the bottom-left corner of the layout. These sequences are named positive and negative sequences and each of them defines a step-line (layout profiles), named as positive and negative step-lines. In Figure 10 the positive and negative sequence that describe a layout are represented, together with the correspondent sets of positive and negative step-lines, the former starting at the bottom-left corner and ending at the upper-right corner, and the latter starting at the upper-left corner and ending at the bottom-right corner. In the figure lines are artificially spaced from each other to improve drawing legibility. Pisinger (2007) propose an innovative and computationally efficient transformation of sequences pairs into layouts. Two different algorithms are described, a "Sequence Pair

(SP)" algorithm and a "Semi-normalized Sequence Pair (Seminorm-SP)" algorithm. The first one transforms sequence pairs into normalized layouts, i.e. layouts in which all pieces are bottom-left placed, and the second one applies the same bottom-left property but to the corners of the partial solution profile.

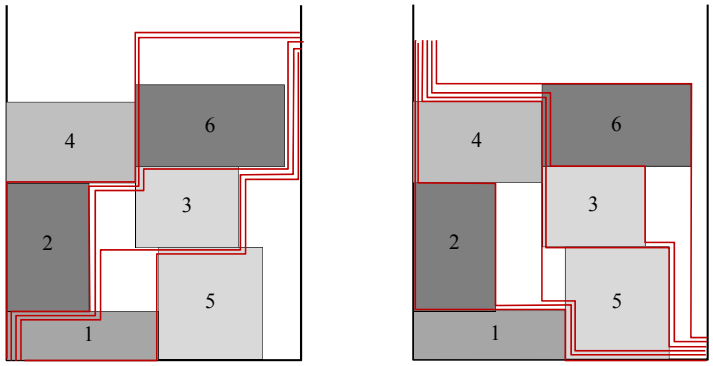


Figure 10: Negative and positive step-lines and their relationship with sequence pairs.

Wei et al. (2011) also uses a profile to describe the partial solution in the "Iterative Doubling Binary Search (IDBS)" heuristic, but in a conceptually different way from the previous authors. Wei's profile does not hold the staircase property of Scheithauer's contour. However, as in the contours, the edges of the profile (called skyline) are either edges or extensions of edges of already placed rectangles (Figure 11). The feasible placement points correspond to place the bottom-left corner of the rectangles at the left endpoint of the horizontal lines that belong to the skyline, and to place the bottom right corner of the rectangles at the right endpoint of these same lines. When choosing the placement point for a concrete rectangle, Wei et al. (2011) resort to several criteria based on the waste that is induced by the new rectangle in the position under evaluation and on a scoring method, close to the strategies used in fitness-based heuristics.

Finally, the "Binary Search Heuristic Algorithm (BSHA)" was proposed by Zhang et al. (2013) introducing some improvements in the representation of Scheithauer's contour, by discarding regions that, when considered the shape of the rectangles that remain to be placed, can not be used anymore. The selection of the placement point is based on the waste generated by each tentative placement.

3.2 Improvement heuristics

Improvement heuristics start with an initial solution, i.e. a complete solution obtained either by a constructive heuristic or randomly generated. This initial solution is then improved by applying small consecutive changes until a stopping criterion is met. The goal is to keep the computational times fairly low while improving the quality of the

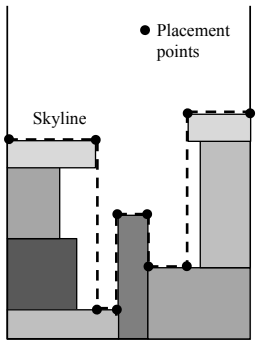


Figure 11: Example of a skyline line, used to describe partial solutions, and its feasible placement points.

solutions, when compared to constructive heuristics (Hopper & Turton 2001*a*, Burke et al. 2004).

Depending on how solutions are modified, two different types of improvement heuristics may be defined. When modifications are directly applied to the layouts, by moving rectangles around, the search is performed over the actual layout. These heuristics are usually slower because the geometric feasibility has to be enforced. However in these heuristics there is a very direct link between the modification applied to the solution and its effects on the layout. Another option is to apply the modifications on the order (or sequence) by which the rectangles are inserted on the strip, relying on an additional constructive heuristic to transform this input order in an actual layout. These heuristics search over sequences. The generation of modified solutions is faster but there is a weaker link between the modification applied and its effects on the layout.

3.2.1 Search over sequences

Heuristics for the SPP that search over sequences resort to modification operators that are common to other problems that rely on sequences to codify their solutions (e.g. traveling salesperson problem, scheduling problems, vehicle routing problems, etc.): the insert and the exchange or swap operators. The insert operator takes a rectangle out of the sequence and re-inserts it in a different position in the sequence. The exchange operator takes two rectangles and exchange their position in the sequence. Notice that, as in the SPP all rectangles have to be placed on the strip, the also common delete operator cannot be applied, and as only one strip is being considered the exchange between partial sequences (e.g. inter-route swap in vehicle routing problems) does not also make sense. Figure 12 illustrates the overall structure of an improvement heuristic that searches over sequences. An initial input sequence 1 is decoded by a constructive heuristic to produce an initial layout that has a height h_1, a swap operator is applied to rectangles 1 and 3, resulting in sequence 2. This sequence is decoded into a layout with height h_2, which is better than h_1. Then an insert operator is applied to rectangle 2 and sequence 3 is

generated. The correspondent layout has a height h_3, which is equal to h_2. The several heuristics of this type will mainly defer on:

- the type of modification operators used, i.e on how an incumbent sequence is generated from a previous one (many times designated as neighbourhood structure);
- the decoding (constructive) heuristic, that transforms a sequence in a layout;
- when the incumbent sequence is accepted to be the base for further modifications, i.e. the search strategy.

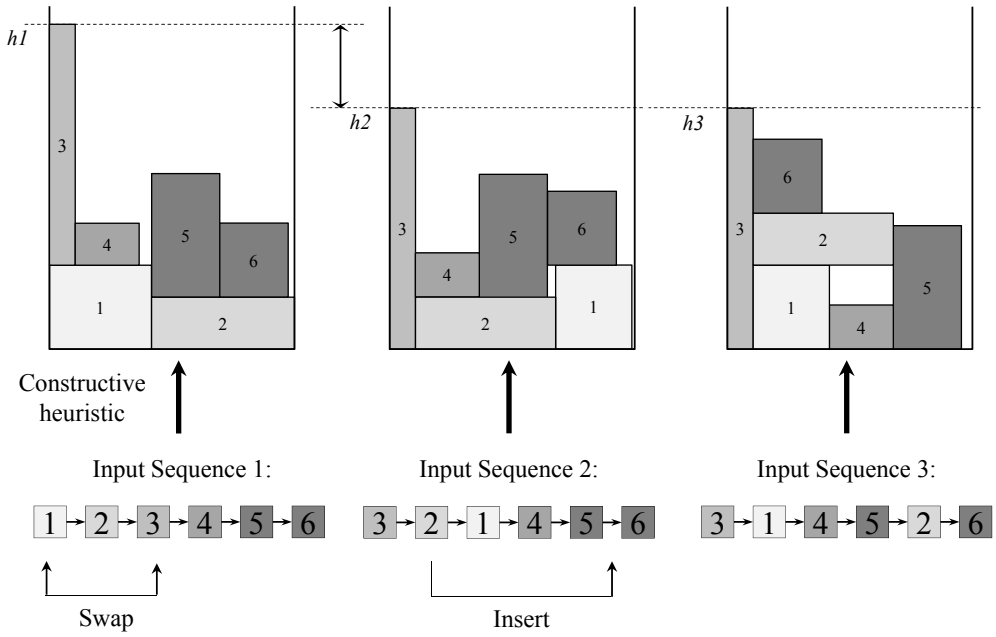


Figure 12: Overall structure of an improvement heuristic that searches over sequences.

The most basic search strategy is Local Search (LS). An incumbent solution is accepted if is better than the previous solution. If not, a new modification is applied to the sequence. The search stops after a previously defined number of sequences have been generated (iterations of the search procedure) without an improvement of the solution. This strategy is used in Hopper & Turton (2001*b*) but slightly different versions can be found in the literature. In Leung & Zhang (2011) the Fast Layer-Based Heuristic is transformed in a local search heuristic, using the swap operator between two random rectangles to generate different sequences.

Pure local search methods, as the ones just described, are known for getting stuck in local minima, i.e. solutions that are better than any other solution that can be generated from it, with the chosen modification operator, but with no guarantee of global optimality.

To overcome this drawback it is common in combinatorial optimization problems to resort to metaheuristics instead of pure local search. What differentiates metaheuristics is the controlled acceptance of worse incumbent solutions, hoping that the temporary degradation of the solution may lead the search to better regions of the search space, and eventually to the global optimal solution.

One of the seminal works using metaheuristics in the SPP resolution is Jakobs (1996), with genetic algorithms being used as a search strategy and each sequence being transformed into a layout by the bottom-left heuristic. The initial population is composed by random sequences and the crossover operator uses q consecutive rectangles of one parent, starting at position p, and completes the sequence with the missing rectangles by the order given by the other parent. Three mutation operators are used: the sequence of a random block is inverted; some elements of the sequence are exchanged; rectangles are rotated. A similar approach was developed by Hopper & Turton (1999). More recent applications of genetic algorithms to search over sequences can be found in Hadjiconstantinou & Iori (2007), Salto et al. (2008), Burke et al. (2010), Thomas & Chaudhari (2014), resorting to different constructive heuristics to decode the sequences into layouts.

In a hybrid constructive/improvement heuristic, Burke et al. (2009) used simulated annealing to improve a partial initial solution generated by the best-fit heuristic. A first set of rectangles is placed by the best-fit heuristic and these rectangles are not moved during the search procedure. The remaining rectangles are placed by the bottom-left-fill heuristic, according to several sequences that are generated during the search, trying to use the holes between the initially placed rectangles. Resorting to the semi-normalized profile-based constructive heuristic, Pisinger (2007) also developed a simulated annealing approach. Leung et al. (2011) implemented also a simulated annealing algorithm over a constructive heuristic, obtaining improved results over the pure local search algorithm.

Adapting algorithms developed for three-dimensional packing problems to the two-dimensional SPP is a natural path explored by several authors (e.g. Bortfeldt (2006)). A different approach was followed by Belov et al. (2008) that have adapted the uni-dimensional "Sequential Value Correction" algorithm (Belov & Scheithauer 2007) to the two-dimensional SPP. Behind this proposal is the observation that solving a two-dimensional rectangular problem is equivalent to solve a set of one-dimensional bin-packing problem, in which the size of the bins is the width of the strip and the items to pack are horizontal slices of the rectangles. In other words, the rectangles to place are horizontally sliced according to a given grid and minimizing the height of the strip becomes minimizing the number of bins used to pack all slices. Of course that contiguity constraints (slices of the same rectangle have to be assigned to contiguous bins) and location constraints (slices of the same rectangle have to be paced at the same x-coordinate) have to be additionally imposed. As the problem is heuristically solved as a succession of knapsack problems (it always selects the left-most free space in the last slice and fills it by a one-dimensional greedy heuristic, considering only item widths) it is possible to impose these additional constraints. In the one-dimensional knapsack problems pseudo-profits are assigned to the slices. These pseudo-profits are related to the area of the rectangle from here each slice comes, and they aim to give priority to the largest rectangles, harder to efficiently place. The overall approach is based on a local search strategy where different sequences are generated by an insert operator.

Wei et al. (2009) use a mix of profile-based heuristic and fitness-based heuristic, as each rectangle is placed on one of the corner points of the profile but the latter is chosen based on how well the rectangle fits the space. To guide the search a local search strategy is used. Later on Wei et al. (2011) developed a taboo search algorithm over a profile-based constructive heuristic. In Zhang et al. (2013) the profile-based heuristic is also randomized in a pure local search algorithm. Chen et al. (2012) resort to the algorithm by Wei et al. (2009) but apply it by regions, in which the strip is previously divided.

An original and different approach to the concept of searching over sequences was proposed by Burke et al. (2011): the "Squeaky Wheel Heuristic". This improvement heuristic, as all the others, changes the current sequence to generate the incumbent sequence, but incorporates what could be called a learning factor. It uses the best-fit heuristic as a decoder of the rectangle sequence, but the sequence is generated taking into account penalties assigned to each rectangle. A target for the height is defined (in this work it is the so called area lower bound – the sum of the areas of the rectangles divided by the width of the strip), and when a rectangle protrudes the target on the height a penalty is assigned to it. The more penalized rectangles are the first ones to place in the next iteration, i.e. the rectangles are ordered by non-increasing value of penalty. This methodology is illustrated in Figure 13.

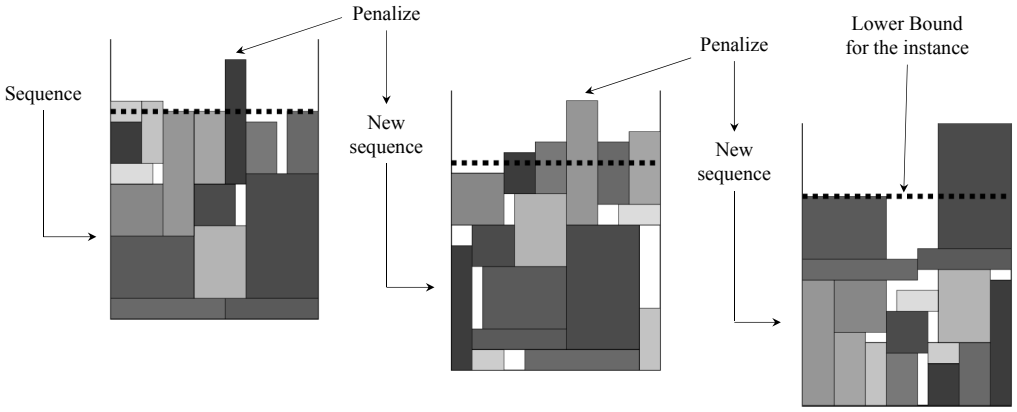


Figure 13: An example of the application of the squeaky wheel heuristic.

Worthwhile of reference are also the works by Borgulya (2014), Zhang, Che, Ye, Si & Leung (2016). If from the point of view of the constructive decoding heuristics they build on previous work, in what concerns the search process these are the first applications of hyper-heuristics and variable neighbourhood search.

3.2.2 Search over the layout

When searching over layouts the modifications are operated directly over the layout. An example is presented in Figure 14, where the placement of rectangle 1 is exchanged with the placement of rectangles 2, 3 and 4, originating a better solution ($h2 < h1$).

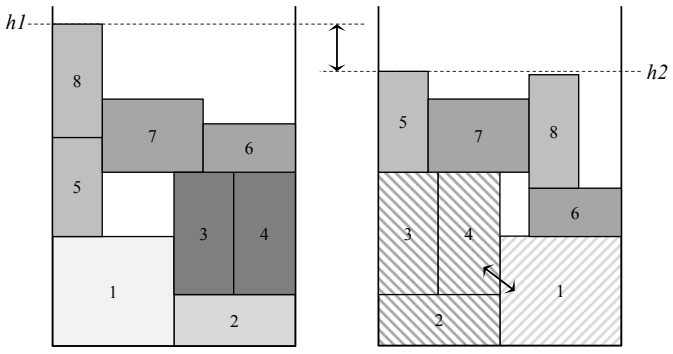


Figure 14: Search over the layout.

One of the first improvement heuristics searching over the layout is proposed by Alvarez-Valdes et al. (2005). Although developed for the two-dimensional cutting problem (a "Placement Problem", according to Wäscher et al. typology, in which a sub-set of rectangles is to be chosen to be cut from a limited sized stock rectangle), the ideas presented in this paper were at the basis of a later approach to the SPP resolution. The algorithm is based on the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic and therefore has two phases: a constructive phase and an improvement phase. The constructive heuristic is a fitness-based heuristic that gives priority to the use of the smallest empty spaces. A fitness function, which assigns to each piece a fitness value based on how well the piece occupies the space, is defined and, according to the GRASP paradigm, one of the pieces in the restricted candidate list is chosen to be the following one to be placed. The improvement phase is run over the actual layouts, and two different operators are defined:

1. To transform several small unused spaces into larger empty spaces, one or more pieces adjacent to an empty space are removed, all pieces are pushed to the corners of the stock rectangle and the remaining pieces are tentatively placed by the constructive heuristic. This is a pretty good example why the SPP requires specific resolution algorithms: in the SPP all pieces are placed on the strip, there are never remaining pieces except the ones we may explicitly remove from the layout; and there are no corners, or at least there are only two corners, which hinders the idea of separating the already placed rectangles to create room for the incoming ones.

2. To improve the arrangement of the placed rectangles, the last $k\%$ rectangles of the solution are removed (they were placed using the randomized version of the constructive algorithm) and the empty space filled again with (some of) the remaining rectangles by the deterministic version of the constructive heuristic.

In Alvarez-Valdés et al. (2008) this GRASP algorithm is evolved to deal with the SPP. For the improvement phase, four different operators are proposed:

1. From a solution with a height H the last $k\%$ rectangles of the solution are removed,

an artificial height of $H - 1$ is imposed to the strip and the solution completed with the deterministic constructive heuristic. If all rectangles are placed in this $W \times (H - 1)$ stock rectangle, then the SPP solution has improved.

2. All rectangles that define the value H of the current solution (i.e. the rectangles whose top edge is at height H – the "guilty" rectangles) are removed from the solution and placed on some of the empty spaces, lower on the strip. If the rectangle exceeds the dimensions of the empty space, the overlapping rectangles are deleted and placed again using the deterministic constructive heuristic (Figure 15).

3. The last k% rectangles that have been placed in the constructive phase are removed and placed again with the deterministic constructive heuristic (Figure 16). There is no guarantee that the chosen percentage will lead to a layout, with the remaining rectangles, that has a height strictly lower than the original one. If this is the case, additional rectangles are removed until this condition is satisfied.

4. Similar to the previous operator, but in this case all pieces with their top edge exceeding a height λH are removed, with $0 < \lambda < 1$ (Figure 17).

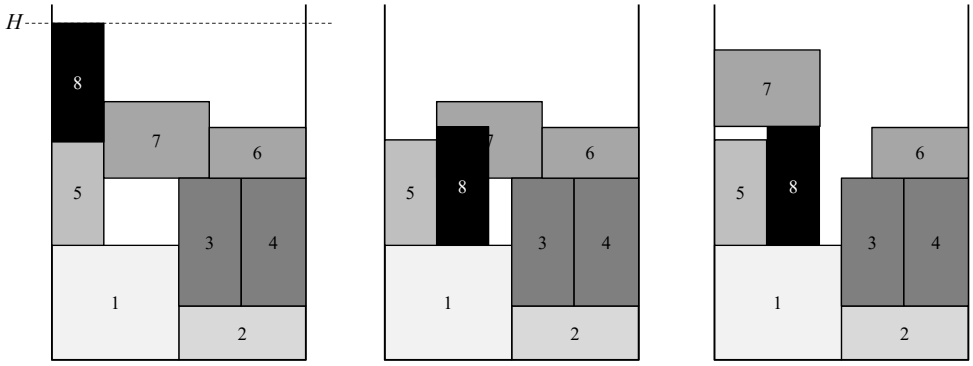


Figure 15: Example of removal and reinsertion of a "Guilty" rectangle.

Bortfeldt (2006) adapts a genetic algorithm developed for the three-dimensional Container Loading Problem (Bortfeldt & Gehring 2001) to the two-dimensional SPP. The algorithm builds on the concept of layer, rather similar to the levels of the level-oriented heuristics, but within each layer rectangles may be placed on the top of other rectangles, as in the more sophisticated level-oriented heuristics. Therefore, a solution is a set of layers and the operators of the genetic algorithm exchange layers between solutions (crossover operator) or simply transfer a certain number of layers (mutation). A repair heuristic is proposed to enforce solutions' feasibility and improve their quality. As partial solutions (the layers) are moved around among different solutions, this approach is rather innovative once usually genetic algorithms for the SPP resort to searching over sequences. Also Zhang et al. (2007) developed a genetic algorithm based on layers, which are generated by the recursive algorithm previously proposed by the same authors (Zhang et al. 2006), the "Improved Heuristic Recursive algorithm (IHR)". Wei, Tian, Zhu & Lim (2014)

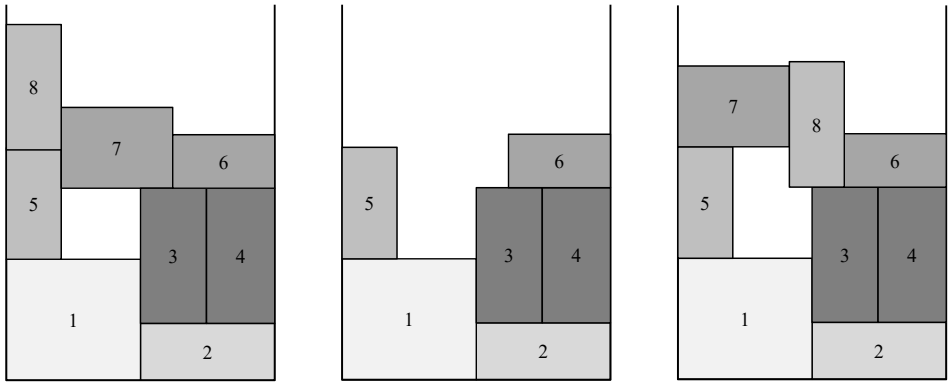


Figure 16: Example of removal and reinsertion of the last 25% rectangles.

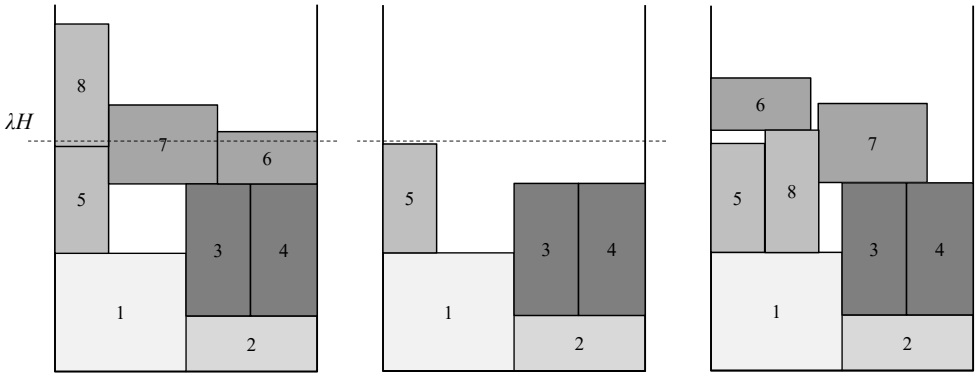


Figure 17: Example of the removal and reinsertion of all rectangles above λH.

also used layers during the search. The layers are initially created in a level-oriented approach, resorting to the best-fit heuristic to fill each level. However, during the search blocks of rectangles are formed and then placed in each layer instead of the individual rectangles, i.e. the layers' height is dynamically adjusted aiming a tighter placement of the rectangles and, indirectly, a lower layout's height.

Neveu et al. (2008) use the same modification operator as Alvarez-Valdés et al. (2008), picking the rectangle that defines the height of the layout, placing it on an empty space and removing eventually overlapping rectangles, and finally placing them again with a constructive heuristic. The maximal-space representation of the empty regions of the strip is used (here under the name of maximal holes) and a local search strategy, similar to the one proposed by Wei et al. (2009), guides the search. Several constructive heuristics are used to generate different initial solutions.

4 Discussion

This work aims to present a structured view over heuristic approaches for the two-dimensional rectangular SPP. To achieve this, a systematic literature survey was run and 36 papers published in the last decade (from 2007 onward), in international journals with peer review, were reviewed and categorized. Also some previous papers that can be considered seminal in the use of a relevant method, concept or idea, were discussed. This led to the analysis of a total of 48 papers. In Table 1 a list of the references discussed in the previous sections, classified by type of heuristic, is presented. Quite often, in a single paper more than one method or algorithm is proposed and in these papers it is frequent to find both constructive and improvement heuristics, mainly when using metaheuristics. Therefore, their classification is based on the authors' opinion about where the main contribution was made, but it is naturally arguable. Research gaps and future research directions will now be discussed.

The chart from Figure 18 shows that the last decade has been rich in terms of contributions for the SPP problem. All types of strategies have been used, from the more simple constructive heuristics to the more complex improvement heuristics with search over the layout. However, most probably due to the complexity of dealing with the geometric feasibility of the layouts, the latter strategy has been less frequently used. It should be kept in mind that, in improvement heuristics that search over sequences, all modifications to a solution originate a feasible layout, as it is the decoder responsibility to produce the actual layouts, given the sequences of rectangles. Additionally, these heuristics can use as decoders the many and efficient constructive heuristics available. This may be the main reason for the ratio of 1 to 4 that we find when comparing the number of improvement heuristics that search over sequences versus improvement heuristics that search over layouts. From the search control algorithm point of view, pure local search, simulated annealing and genetic algorithms are the most frequent algorithms, with the most recent metaheuristics being absent. With just one approach based on variable neighbourhood search, and another one based on hyper heuristics, there is plenty of room for research in the use of more sophisticated search control mechanisms, in particular in improvement heuristics with search over layouts.

Another relevant factor for the heuristics' design choices is the size of the instances. Since 2007, very large instances became the main challenge for new heuristics, and by very large instances it is meant instances with more than 5000 rectangles. These instances require very fast solution generation and evaluation and special care has to be put on the computational implementation of the methods, with important and relevant contributions from the Computer Science field. Some of the methods proposed in the past are specialized in solving zero-waste instances, but these are not only rather artificial instances, that do not arise in practice, but these instances can also significantly simplify the SPP resolution, when this characteristic is explored by the resolution method.

When looking in more detail at the constructive heuristics (Figure 19), it can be seen that more than 40% of them are fitness-based heuristics. It is surely due to their good performance, when compared to positioning-based heuristics. In fact, fitness-based heuristics

Table 1: Heuristics for the SPP.

Constructive	Improvement			
	Search over sequences	Search strategy[a]	Search over layouts	Search strategy[a]
Baker et al. (1980)	Jakobs (1996)	GA	Alvarez-Valdes et al. (2005)	GRASP
Coffman Jr. et al. (1980)	Hopper & Turton (1999)	GA	Bortfeldt (2006)	GA
Chazelle (1983)	Liu & Teng (1999)	GA	Zhang et al. (2007)	GA
Berkey & Wang (1987)	Hopper & Turton (2001*b*)	GA; SA; NE; LS	Alvarez-Valdés et al. (2008)	GRASP
Burke et al. (2004)	Hadjiconstantinou & Iori (2007)	GA	Neveu et al. (2008)	LS
Zhang et al. (2006)	Pisinger (2007)	SA		
Ntene (2007)	Belov et al. (2008)	SVC		
Cui et al. (2008)	Salto et al. (2008)	GA		
Aşık & Özcan (2009)	Burke et al. (2009)	SA		
Imahori & Yagiura (2010)	Wei et al. (2009)	LS		
Ortmann et al. (2010)	Burke et al. (2010)	HH; GA		
Kotov & Cao (2011)	Burke et al. (2011)	LS		
Bortfeldt & Jungmann (2012)	Leung & Zhang (2011)	SA; LS		
Cui et al. (2013)	Leung et al. (2011)	LS		
Özcan et al. (2013)	Wei et al. (2011)	TS		
Verstichel et al. (2013)	Chen et al. (2012)	LS		
Buchwald & Scheithauer (2016)	Yang et al. (2013)	SA		
Wei, Tian, Zhu & Lim (2014)	Wauters et al. (2013)	ShP		
Zhang, Shi, Leung & Wu (2016)	Zhang et al. (2013)	LS		
	Borgulya (2014)	HH; GA; LS		
	Thomas & Chaudhari (2014)	GA		
	Wei, Qin, Cheang & Xu (2014)	LS		
	Chen et al. (2015)	LS		
	Zhang, Che, Ye, Si & Leung (2016)	VNS; LS		

[a] GA: Genetic Algorithm; GRASP: Greedy Randomized Adaptive Search Procedure; HH: Hyper-Heuristic; LS: Local Search; NE: Naive evolution; SA: Simulated Annealing; ShP: Shaking procedure; SVC: Sequential Value Correction; TS: Tabu Search; VNS: Variable Neighbourhood Search.

have been a trend in the research in this field. In particular scoring heuristics, which are able to measure the quality of the placement of a rectangle in a space, as initially proposed by Leung & Zhang (2011), seems to deliver good results and several other scoring heuristics were proposed (IA, ISA, SRA and HDA).

Level-based heuristics are basically extensions of the FFDH constructive heuristic. They received a lot of attention in the eighties but are currently less studied. Probably this is because only very specific applications require this type of layouts, not being the extra waste that levels impose worthwhile, when levels are not a concrete constraint of the application.

Profile-based heuristics have been less explored and represent a clear research opportunity in this field. It is well-known since the eighties (Baker et al. 1980) that there are optimal solutions for the SPP that can not be achieved by the bottom-left heuristic, whatever

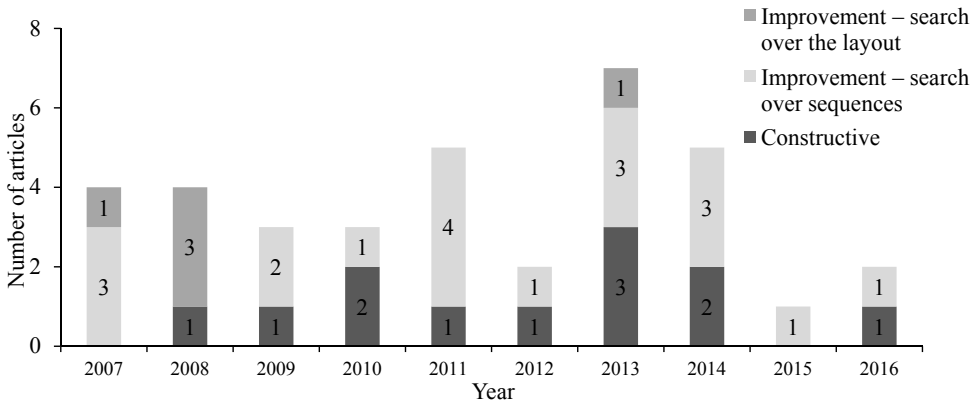


Figure 18: Number of publications distributed by type of improvement heuristic.

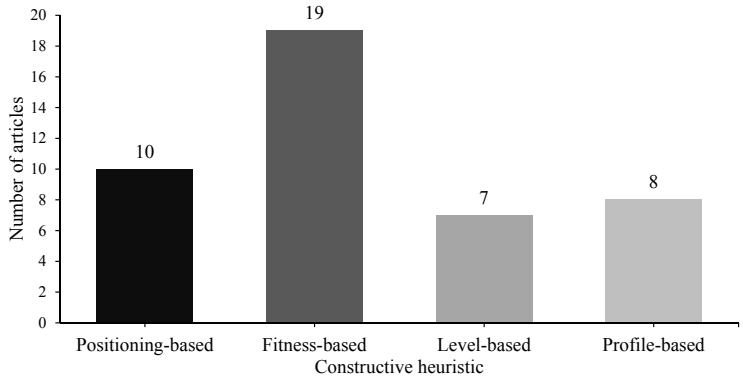


Figure 19: Number of papers by type of constructive heuristics.

rectangle sequence is considered. However, this is not true when profile-based approaches are used. Martello et al. (2003) proposed an exact tree-search algorithm in which, in each node of the branch-decision tree, child nodes are generated combining all rectangles remaining to place with all corner points of the profile describing the partial solution. Therefore, for each optimal layout there is a sequence of rectangles that, properly choosing the placement points on the profile, generates it. Profile-based solution representations are, therefore, rather promising for the the development of effective heuristics for the SPP, as they seem to gather the advantages of positioning-based and fitness-based methods, while keeping under control time complexity.

The chart of Figure 20 relates the improvement heuristics based on search over sequences with the constructive heuristics used to decode the sequences (the list of acronyms used in the x-axis is presented in Table 2). The acronym *WIH*, in the y-axis, stand for "without improvement heuristic" and refers to the algorithms that do not have an improvement phase, i.e. are just constructive. The first four shades of gray stand for the four types

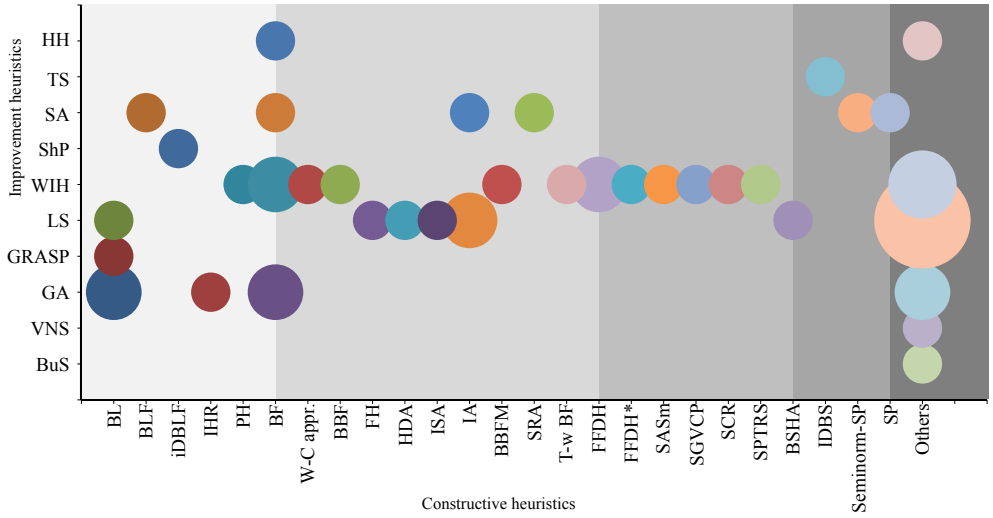
of constructive heuristics. From the lightest gray to the darkest gray: positioning-based, fitness-based, level-based and profile-based. This chart confirms the findings already mentioned. Clearly, combining genetic algorithms with the bottom-left heuristic, and its variants, is the most frequent approach and the most recent metaheuristics have not yet been explored.

Table 2: Constructive heuristics for the SPP: list of acronyms and descriptions.

Acronym	Constructive Heuristic	Authors
BF	Best-Fit	Burke et al. (2004)
W-C appr.	Best-Fit Worst-Case Approximation ratio	Imahori & Yagiura (2010)
BBF	Bidirectional Best-Fit	Aşık & Özcan (2009)
BSHA	Binary Search Heuristic Algorithm	Zhang et al. (2013)
BL	Bottom-Left	Baker et al. (1980)
BLF	Bottom-Left-Fill	Chazelle (1983)
IA	Efficient Improved Algorithm (IA)	Wei, Qin, Cheang & Xu (2014)
FH	Fast layer-based Heuristic	Leung & Zhang (2011)
FFDH	First-Fit Decreasing Height	Coffman Jr. et al. (1980)
HDA	Hybrid Demon Algorithm	Chen et al. (2015)
iDBLF	Improved Deepest Bottom Left Fill	Wauters et al. (2013)
IHR	Improved Heuristic Recursive algorithm	Zhang et al. (2007)
ISA	Intelligent Search Algorithm	Leung et al. (2011)
IDBS	Iterative Doubling Binary Search	Wei et al. (2011)
BBFM	Modified Bidirectional Best-Fit	Özcan et al. (2013)
FFDH*	Modified First-Fit Decreasing Height	Buchwald & Scheithauer (2016)
SASm	Modified Size-Alternating Stack algorithm	Ortmann et al. (2010)
PH	Priority Heuristic	Zhang, Shi, Leung & Wu (2016)
Seminorm-SP	Seminormalized-Sequence Pair	Pisinger (2007)
SP	Sequence Pair	Pisinger (2007)
SGVCP	Sequential Grouping and Value Correction Procedure	Cui et al. (2013)
SRA	Simple Randomized Algorithm	Yang et al. (2013)
SCR	Stack Ceiling with Re-sorting algorithm	Ortmann et al. (2010)
T-w BF	Three-way Best-Fit	Verstichel et al. (2013)
SPTRS	Strip Packing by Tree Search	Bortfeldt & Jungmann (2012)

A final word about benchmark instances. There is a well-established set of instances that is consistently used by the research community. However, many of these instances have been proposed decades ago and it is reasonable to question if they are still challenging for now-a-days algorithms, given the hardware where they are currently run. Moreover, some of these instances have characteristics that are not typical from real-world applications and allow that less general approaches outperform more solid and robust algorithms, by taking advantage of the special characteristics of these instances. Computational experiments are not league tables. Computational experiments are meant to understand the proposed algorithm, in which situations it goes well, in which it does not so well, which are its strengths and its limitations.

The limitations of the benchmark instances can only be overcome by the use of problem generators that generate instances to systematically explore the space of input data configurations: rectangles area, aspect ratio, heterogeneity, number, etc. Silva et al. (2014) proposed a problem generator for two-dimensional (and three-dimensional) rectangular cutting and packing problems, in all the variants, that could also be used for strip packing



Improvement heuristics: BuS: Bubble Search; GA: Genetic Algorithm; GRASP: Greedy Randomized Adaptive Search Procedure; HH: Hyper Heuristic; LS – Local Search; SA: Simulated Annealing; ShP: Shaking Procedure; TS: Tabu Search; VNS: Variable Neighbourhood Search; WIH: Without Improvement Heuristic.

Figure 20: Relation between the improvement heuristics based on search over sequences and the constructive heuristics used to decode the sequences.

problems, to overcome the limitations of the existing benchmark instances.

5 Conclusion

In this review, the last decade of publications related to heuristics for the two-dimensional rectangular strip packing has been surveyed. Each contribution was classified according to a well-known conceptual framework (constructive heuristics and improvement heuristics, both with search over sequences and search over layouts), with an additional division of constructive heuristics in positioning-based, fitness-based, level-based and profile-based heuristics.

From the literature review emerges the big dynamism of the field. Despite having its roots in the very early eighties, strip packing problems are still intensively researched and more complex approaches are being proposed by researchers from all over the world. Research gaps are identified, mainly associated to the use of more sophisticated search algorithms, specially metaheuristics, to the use of profile representations of solutions and to the development of improvement heuristics with search over the layout.

A necessary step in the field is related to the computational validation of the new algorithms, that have to make progresses in the direction of having as goal the understanding of why and when the methods work better and worse.

Acknowledgments

The third author was supported by FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within the grant SFRH/BPD/98981/2013. The research was partially supported by Project "TEC4Growth – Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020", financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

References

Alvarez-Valdes, R., Parreño, F. & Tamarit, J. M. (2005), 'A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems', *Journal of the Operational Research Society* **56**(4), 414–425.

Alvarez-Valdés, R., Parreño, F. & Tamarit, J. M. (2008), 'Reactive grasp for the strip-packing problem', *Computers & Operations Research* **35**(4), 1065–1083.

Aşık, Ö. B. & Özcan, E. (2009), 'Bidirectional best-fit heuristic for orthogonal rectangular strip packing', *Annals of Operations Research* **172**(1), 405–427.

Baker, B. S., Coffman, Jr, E. G. & Rivest, R. L. (1980), 'Orthogonal packings in two dimensions', *SIAM Journal on Computing* **9**(4), 846–855.

Belov, G. & Scheithauer, G. (2007), 'Setup and open-stacks minimization in one-dimensional stock cutting', *INFORMS Journal on Computing* **19**(1), 27–35.

Belov, G., Scheithauer, G. & Mukhacheva, E. a. (2008), 'One-dimensional heuristics adapted for two-dimensional rectangular strip packing', *Journal of the Operational Research Society* **59**(6), 823–832.

Berkey, J. O. & Wang, P. Y. (1987), 'Two-dimensional finite bin-packing algorithms', *Journal of the Operational Research Society* **38**(5), 423–429.

Borgulya, I. (2014), 'A parallel hyper-heuristic approach for the two-dimensional rectangular strip-packing problem', *Journal of Computing and Information Technology* **22**(4), 251–265.

Bortfeldt, A. (2006), 'A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces', *European Journal of Operational Research* **172**(3), 814–837.

Bortfeldt, A. & Gehring, H. (2001), 'A hybrid genetic algorithm for the container loading problem', *European Journal of Operational Research* **131**(1), 143–161.

Bortfeldt, A. & Jungmann, S. (2012), 'A tree search algorithm for solving the multi-dimensional strip packing problem with guillotine cutting constraint', *Annals of Operations Research* **196**(1), 53–71.

Buchwald, T. & Scheithauer, G. (2016), 'Upper bounds for heuristic approaches to the strip packing problem', *International Transactions in Operational Research* **23**(1-2), 93–119.

Burke, E. K., Hyde, M., Kendall, G. & Woodward, J. (2010), 'A genetic programming hyper-heuristic approach for evolving 2-D strip packing heuristics', *IEEE Transactions on Evolutionary Computation* **14**(6), 942–958.

Burke, E. K., Hyde, M. R. & Kendall, G. (2011), 'A squeaky wheel optimisation methodology for two-dimensional strip packing', *Computers and Operations Research* **38**(7), 1035–1044.

Burke, E. K., Kendall, G. & Whitwell, G. (2004), 'A New Placement Heuristic for the Orthogonal Stock-Cutting Problem', *Operations Research* **52**(4), 655–671.

Burke, E. K., Kendall, G. & Whitwell, G. (2009), 'A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem', *INFORMS Journal on Computing* **21**(3), 505–516.

Chazelle, B. (1983), 'The bottomn-left bin-packing heuristic: An efficient implementation', *Computers, IEEE Transactions on* **100**(8), 697–707.

Chen, B., Wang, Y. & Yang, S. (2015), 'A Hybrid Demon Algorithm for the Two-Dimensional Orthogonal Strip Packing Problem', *Mathematical Problems in Engineering* **2015**, 1–14.

Chen, J., Zhu, W. & Peng, Z. (2012), 'A heuristic algorithm for the strip packing problem', *Journal of Heuristics* **18**(4), 677–697.

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

Cui, Y., Yang, L. & Chen, Q. (2013), 'Heuristic for the rectangular strip packing problem with rotation of items', *Computers and Operations Research* **40**(4), 1094–1099.

Cui, Y., Yang, Y., Cheng, X. & Song, P. (2008), 'A recursive branch-and-bound algorithm for the rectangular guillotine strip packing problem', *Computers and Operations Research* **35**(4), 1281–1291.

Dowsland, K. A., Dowsland, W. B. & Bennell, J. A. (1998), 'Jostling for position: local improvement for irregular cutting patterns', *Journal of the Operational Research Society* **49**, 647–658.

Fanslau, T. & Bortfeldt, A. (2010), 'A tree search algorithm for solving the container loading problem', *INFORMS Journal on Computing* **22**(2), 222–235.

Hadjiconstantinou, E. & 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.

Hopper, E. & Turton, B. (1999), 'A genetic algorithm for a 2D industrial packing problem', *Computers and Industrial Engineering* **37**(1985), 375–378.

Hopper, E. & Turton, B. C. H. (2001*a*), 'A review of the application of meta-heuristic algorithms to 2D strip packing problems', *Artificial Intelligence Review* **16**(4), 257–300.

Hopper, E. & Turton, B. C. H. (2001*b*), 'An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem', **128**, 34–57.

Hurink, J. L. & Paulus, J. J. (2011), 'Improved online algorithms for parallel job scheduling and strip packing', *Theoretical Computer Science* **412**(7), 583–593.

Imahori, S. & Yagiura, M. (2010), 'The best-fit heuristic for the rectangular strip packing problem: An efficient implementation and the worst-case approximation ratio', *Computers and Operations Research* **37**(2), 325–333.

Jakobs, S. (1996), 'On genetic algorithms for the packing of polygons', *European Journal of Operational Research* **88**(1), 165–181.

Kotov, V. M. & Cao, D. (2011), 'A heuristic algorithm for the non-oriented 2d rectangular strip packing problem', *Buletinul Academiei de Ştiinţe a Republicii Moldova. Matematica* (2), 81–88.

Leung, S. C. H. & Zhang, D. (2011), 'A fast layer-based heuristic for non-guillotine strip packing', *Expert Systems with Applications* **38**(10), 13032–13042.

Leung, S. C. H., Zhang, D. & Sim, K. M. (2011), 'A two-stage intelligent search algorithm for the two-dimensional strip packing problem', *European Journal of Operational Research* **215**(1), 57–69.

Liu, D. & Teng, H. (1999), 'An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles', *European Journal of Operational Research* **112**(2), 413–420.

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

Lodi, A., Martello, S. & Vigo, D. (1999), 'Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems', *INFORMS Journal on Computing* **11**(4), 345–357.

Martello, S., Monaci, M. & Vigo, D. (2003), 'An exact approach to the strip-packing problem', *INFORMS Journal on Computing* **15**(3), 310–319.

Mumford-Valenzuela, C. L., Vick, J. & Wang, P. Y. (2004), 'Heuristics for large strip packing problems with guillotine patterns: An empirical study', *Metaheuristics: computer decision-making* **2004**, 501–522.

Murata, H., Fujiyoshi, K., Nakatake, S. & Kajitani, Y. (1996), 'Vlsi module placement based on rectangle-packing by the sequence-pair', *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* **15**(12), 1518–1524.

Neveu, B., Trombettoni, G., Araya, I. & Riff, M.-c. (2008), 'A Strip Packing Solving Method Using an Incremental Move Based on Maximal Holes', *International Journal on Artificial Intelligence Tools* **17**(5), 881–901.

Ntene, N. (2007), An algorithmic approach to the 2D oriented strip packing problem, PhD thesis, Department of Logistics, University of Stellenbosch.

Ntene, N. & van Vuuren, J. H. (2009), 'A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem', *Discrete Optimization* **6**(2), 174–188.

Ortmann, F. G., Ntene, N. & van 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**(2), 306–315.

Özcan, E., Kai, Z. & Drake, J. H. (2013), 'Bidirectional best-fit heuristic considering compound placement for two dimensional orthogonal rectangular strip packing', *Expert Systems with Applications* **40**(10), 4035–4043.

Pisinger, D. (2007), 'Denser packings obtained in o (n log log n) time', *INFORMS Journal on Computing* **19**(3), 395–405.

Riff, M. C., Bonnaire, X. & Neveu, B. (2009), 'A revision of recent approaches for two-dimensional strip-packing problems', *Engineering Applications of Artificial Intelligence* **22**(4-5), 833–837.

Salto, C., Alba, E., Molina, J. M. & Leguizamón, G. (2008), 'Greedy seeding procedure for GAs solving a strip packing problem', *Inteligencia Artificial* **12**(40), 73–85.

Scheithauer, G. (1997), 'Equivalence and dominance for problems of optimal packing of rectangles', *Ricerca Operativa* **83**, 3–34.

Silva, E., Oliveira, J. F. & Wäscher, G. (2014), '2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems', *European Journal of Operational Research* **237**(3), 846–856.

Thomas, J. & Chaudhari, N. S. (2014), 'A new metaheuristic genetic-based placement algorithm for 2D strip packing', *Journal of Industrial Engineering International* **10**(1), 1–16.

Verstichel, J., De Causmaecker, P. & Berghe, G. V. (2013), 'An improved best-fit heuristic for the orthogonal strip packing problem', *International Transactions in Operational Research* **20**(5), 711–730.

Wäscher, G., Haußner, H. & Schumann, H. (2007), 'An improved typology of cutting and packing problems', *European Journal of Operational Research* **183**(3), 1109–1130.

Wauters, T., Verstichel, J. & Vanden Berghe, G. (2013), 'An effective shaking procedure for 2D and 3D strip packing problems', *Computers and Operations Research* **40**(11), 2662–2669.

Wei, L., Oon, W. C., Zhu, W. & Lim, A. (2011), 'A skyline heuristic for the 2D rectangular packing and strip packing problems', *European Journal of Operational Research* **215**(2), 337–346.

Wei, L., Qin, H., Cheang, B. & Xu, X. (2014), 'An efficient intelligent search algorithm for the two-dimensional rectangular strip packing problem', *International Transactions in Operational Research* **23**(1-2), 65–92.

Wei, L., Tian, T., Zhu, W. & Lim, A. (2014), 'A block-based layer building approach for the 2d guillotine strip packing problem', *European Journal of Operational Research* **239**(1), 58–69.

Wei, L., Zhang, D. & Chen, Q. (2009), 'A least wasted first heuristic algorithm for the rectangular packing problem', *Computers and Operations Research* **36**(5), 1608–1614.

Yang, S., Han, S. & Ye, W. (2013), 'A simple randomized algorithm for two-dimensional strip packing', *Computers and Operations Research* **40**(1), 1–8.

Ye, D., Han, X. & Zhang, G. (2009), 'On-Line Multiple-Strip Packing', *Combinatorial Optimization and Applications* **2009**, 155–165.

Zhang, D., Che, Y., Ye, F., Si, Y.-W. & Leung, S. C. (2016), 'A hybrid algorithm based on variable neighbourhood for the strip packing problem', *Journal of Combinatorial Optimization* **32**(2), 513–530.

Zhang, D.-F., Chen, S.-D. & Liu, Y.-J. (2007), 'An Improved Heuristic Recursive Strategy Based on Genetic Algorithm for the Strip Rectangular Packing Problem', *Acta Automatica Sinica* **33**(9), 911–916.

Zhang, D., Kang, Y. & Deng, A. (2006), 'A new heuristic recursive algorithm for the strip rectangular packing problem', *Computers and Operations Research* **33**(8), 2209–2217.

Zhang, D., Shi, L., Leung, S. C. & Wu, T. (2016), 'A priority heuristic for the guillotine rectangular packing problem', *Information Processing Letters* **116**(1), 15–21.

Zhang, D., Wei, L., Chen, Q. & Leung, S. C. H. (2013), 'A Binary Search Heuristic Algorithm based on Randomized Local Search for the Rectangular Strip Packing Problem', *INFORMS Journal on Computing* **25**(2), 332–345.