Computers & Operations Research 37 (2010) 325 – 333

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

The best-fit heuristic for the rectangular strip packing problem: An efficient implementation and the worst-case approximation ratio

Shinji Imahori[a,*], Mutsunori Yagiura[b]

[a]The University of Tokyo, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
[b]Nagoya University, Furocho, Chikusa-ku, Nagoya 464-8603, Japan

ARTICLE INFO

Available online 22 May 2009

Keywords:
Cutting and packing
Rectangular strip packing
Best-fit heuristic
Time complexity
Approximation ratio

ABSTRACT

We investigate the best-fit heuristic algorithm by Burke et al. [2004. A new placement heuristic for the orthogonal stock-cutting problem. Operations Research 52, 655–671] for the rectangular strip packing problem. For its simplicity and good performance, the best-fit heuristic has become one of the most significant algorithms for the rectangular strip packing. In this paper, we propose an efficient implementation of the best-fit heuristic that requires $O(n)$ space and $O(n \log n)$ time, where n is the number of rectangles. We prove that this complexity is optimal, and we also show the practical usefulness of our implementation via computational experiments. Furthermore, we give the worst-case approximation ratio of the best-fit heuristic.

1. Introduction

Cutting and packing problems are representative combinatorial optimization problems with many applications in various areas such as steel and garment industry and VLSI design. For several decades, the field of cutting and packing has attracted many researchers. Depending on applications, different types of cutting and packing problems need to be solved, and hence many variants of cutting and packing problems have been studied in the literature. Dyckhoff [8] presented a typology of cutting and packing problems and categorized existing studies in this field. Wäscher et al. [22] presented an improved typology of cutting and packing problems. This paper addresses the problem of placing rectangles in a larger rectangular object with a fixed width in order to minimize its height. This problem is widely called the *rectangular strip packing problem*. According to the improved typology of Wäscher et al. [22], the rectangular strip packing problem is categorized into the two-dimensional regular open dimension problem (2D regular ODP) with a single variable dimension.

Almost all cutting and packing problems (including the rectangular strip packing) are known to be NP-hard [9], and hence it is impossible to solve them exactly in polynomial time unless P = NP. Therefore heuristics and metaheuristics are important in designing practical algorithms for cutting and packing problems (e.g., [2,10,18]). Burke et al. [4] proposed a heuristic algorithm (called the *best-fit heuristic*) that places rectangles in a large rectangular object one by one, where the algorithm dynamically selects the next rectangle to be placed. Because of its simplicity and good performance, the best-fit heuristic has become one of the most significant algorithms for the rectangular strip packing problem [1,5].

In this paper, we propose an efficient implementation of the best-fit heuristic that requires $O(n)$ space and $O(n \log n)$ time, where n is the number of rectangles, and we show that this complexity is optimal. In addition to such theoretical advantage, our implementation has also practical merits, it is easy to implement and it runs very fast for any value of n. Computational experiments are conducted to confirm the efficiency in practical sense. We also show that the best-fit heuristic is a $\Theta(\alpha)$-approximation algorithm for the α satisfying $\alpha^{\alpha} = n$. To the best of our knowledge, this is the first theoretical guarantee on the approximation ratio of the best-fit heuristic.

The remaining part of this paper is organized as follows: Section 2 gives a formal definition of the rectangular strip packing problem and literature review. Section 3 describes the best-fit heuristic proposed by Burke et al. [4]. Section 4 presents an efficient implementation of the best-fit heuristic and gives a proof for the optimality of our implementation. Computational results to confirm the practical usefulness are also presented in this section. Section 5 discusses the approximation ratio of the best-fit heuristic. Finally Section 6 gives concluding remarks.

* Corresponding author. Tel.: +81 3 58416907; fax: +81 3 58416908.
E-mail addresses: imahori@mist.i.u-tokyo.ac.jp (S. Imahori), yagiura@nagoya-u.jp (M. Yagiura).

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.05.008

2. Rectangular strip packing problem

2.1. Problem description

The rectangular strip packing problem can be defined as follows: We are given a set of n rectangles $I=\{1,2,\ldots,n\}$. Each rectangle $i \in I$ has its width w_i and height h_i, and the size of rectangle i is denoted by (w_i, h_i). We are also given a rectangular object (called *strip*), which has a fixed width W and a variable height H. The objective is to place all the given rectangles into the strip without overlap so as to minimize the height of the strip.

The rectangular strip packing problem may have additional constraints concerning orientation of rectangles and guillotine cut restriction. As for the rotation of rectangles, the following two cases are considered in this paper: (1) each rectangle has a fixed orientation and (2) rotations of 90° are allowed. When rectangles can be rotated by 90°, the size of each rectangle i becomes (w_i, h_i) or (h_i, w_i). Among various industrial applications of this problem, rotation of rectangles is not allowed in newspaper paging or when the rectangles to be cut are decorated or corrugated, whereas rotation is allowed in the case of plain materials. The guillotine cut constraint signifies that the rectangles must be obtained through a sequence of edge-to-edge cuts parallel to the edges of the strip, which is usually imposed because of technical limitations of the automated cutting machines. We allow non-guillotine placements, i.e., placements are not restricted to those obtainable by guillotine cuts only.

2.2. Literature review

Practical algorithms for the rectangular strip packing problem have been studied actively in these decades. In early stages, Coffman Jr. et al. [7] presented some level-oriented algorithms and Baker et al. [2] proposed a *bottom left algorithm*. Many papers related to these heuristic algorithms have appeared, e.g., Chazelle [6] gave an efficient implementation of the Baker's algorithm, Jakobs [13] and Liu and Teng [17] presented variants of the bottom left algorithm, and Lodi et al. [18] proposed new level-oriented heuristics. These algorithms have a common characteristic: Each algorithm first decides a sequence of rectangles to place, and then it places rectangles one by one in this order at an appropriate position in the strip. Such heuristic algorithms are often incorporated in metaheuristics in order to improve the quality of solutions [10,13,17,18], where metaheuristics are usually utilized to find good sequences (i.e., packing orders) of rectangles.

In these years, different type heuristics and metaheuristics have been proposed (e.g., [1,3–5,11,12]), which produce high quality placements. A heuristic algorithm proposed by Burke et al. [4] (called the *best-fit heuristic*)[1] does not specify a sequence of rectangles to place. Instead of using a sequence of rectangles, the best-fit heuristic dynamically selects the next rectangle to place.

As for the worst-case approximation ratio on the rectangular strip packing problem, some simple heuristic algorithms are known to attain good approximation guarantees, e.g., the next-fit heuristic with decreasing height attains 3 [7], the bottom left heuristic with decreasing width also attains 3 [2]. To our knowledge, the best-known approximation algorithms with respect to the worst-case approximation ratio for the rectangular strip packing problem without rotations are: (1) an approximation algorithm by Steinberg [20] whose absolute bound on approximation ratio is 2 and (2) an asymptotic FPTAS by Kenyon and Remila [15]. For the case with rotations of 90°, similar results are known [14,20].

3. The best-fit heuristic

In this section, we describe the best-fit heuristic for the rectangular strip packing problem. This heuristic algorithm was proposed by Burke et al. [4], and has been widely known with its simplicity and good performance [1,5]. The best-fit heuristic is a greedy algorithm that attempts to produce a good-quality placement by examining an available space as low as possible in the strip and then placing the rectangle that best fits the space. Unlike most heuristic algorithms that have a sequence of rectangles to place, the best-fit heuristic dynamically selects the next rectangle to place. This enables the algorithm to make informed decisions about which rectangle should be placed next.

During the computation, the algorithm keeps a *skyline*, which consists of a sequence of line segments satisfying the following properties: (1) each line segment is parallel to the x-axis, (2) two adjacent line segments have different y-coordinates and have exactly one common x-coordinate (i.e., the x-coordinate of the right end point of a line segment is the same as that of the left end point of its right neighbor), (3) viewed from an infinitely high position, no point in the line segments is hidden by already placed rectangles, and (4) each line segment touches the top edge of an already placed rectangle or the bottom edge of the strip. See examples of skylines shown in thick lines in Fig. 1(a)–(c). Among all the line segments in a skyline, the *lowest available segment* is the one that has the smallest y-coordinate.

The best-fit heuristic repeats the following two operations until all the rectangles are placed: (1) find the lowest available segment of the current skyline and (2) place a rectangle on the segment. At the beginning of the packing process, no rectangles are placed in the strip, and the skyline consists of the bottom edge of the strip only (see Fig. 1(a)). Whenever a rectangle is placed, a part of the

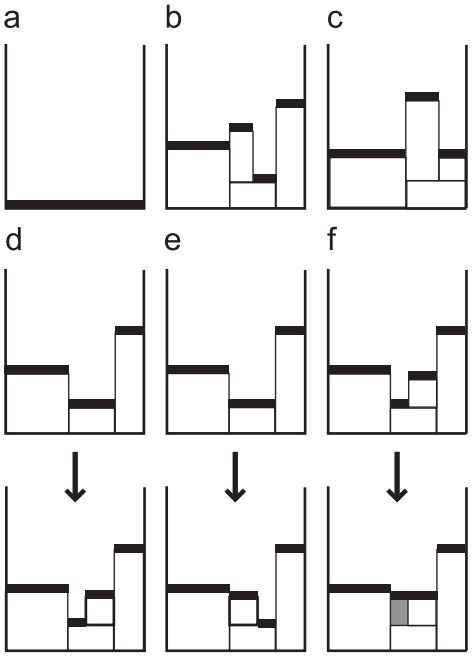

Fig. 1. Description of the best-fit heuristic: (a) initial state of the skyline, (b) line segments in the skyline, (c) two segments on the lowest level, (d) placing a rectangle with the high strategy, (e) placing a rectangle with the low strategy, and (f) raising the lowest available segment without putting a rectangle (discarding the shaded area).

[1] A level-oriented algorithm based on the best-fit heuristic for the (one-dimensional) bin packing problem is sometimes referred to by the same name. However, in this paper, we use this name to denote the algorithm proposed by Burke et al. [4].

lowest available segment moves upward in such a way that the part of the segment hidden by the bottom edge of the placed rectangle is replaced by the top edge of the rectangle (see Fig. 1(d) and (e)). If there are several segments on the lowest level (i.e., they have the same y-coordinate) as in Fig. 1(c), the algorithm selects the leftmost one as the lowest available segment. For the lowest available segment, the *best fit rectangle*, which is to be placed on the segment, is defined to be the widest rectangle (resolving equal widths by the largest height) that has not been placed yet and can be placed on the segment without overlap (i.e., its width is not larger than that of the segment). If the width of the lowest available segment is larger than that of the best fit rectangle, the algorithm places the rectangle at the left-most position on the segment (called the left strategy). We note that the following two strategies are also utilized in [4]: (1) place the rectangle next to the higher segment adjacent to the current segment (called the high strategy) as shown in Fig. 1(d) and (2) place the rectangle next to the lower segment adjacent to the current segment (called the low strategy) as shown in Fig. 1(e); however, for simplicity, the left strategy is assumed throughout the remainder of this paper unless otherwise stated. If there are no rectangles that can be placed on the lowest available segment, the segment is raised to the level of the lower segment adjacent to it, and the two segments are merged (see Fig. 1(f)). In this case, the space below the raised segment becomes waste. When all the rectangles are placed in the strip, the main part of the best-fit heuristic algorithm ends.

As mentioned in the original paper by Burke et al. [4], a drawback of using the above greedy algorithm is that it may create a poor quality placement due to *towers*, where towers are produced when long thin rectangles are placed in portrait orientation near to the top of the strip. In solving an instance in which each rectangle can be rotated by 90°, the following operations can be applied to improve the placement after all the rectangles have been placed in the strip. The algorithm finds a rectangle which touches the top edge of the strip (i.e., the y-coordinate of the top edge of the rectangle is H). If the rectangle is oriented in such a way that its height is greater than its width, the algorithm removes it from the strip and updates the information of the skyline related to this rectangle. The removed rectangle is then rotated by 90° and placed in the strip as low as possible. If the quality of the solution is improved by this operation, the algorithm looks for the next rectangle which touches the top of the strip and performs the same operation again. This operation is repeated until there is no improvement in solution quality. This additional postprocessing stage can be invoked only if rotations of rectangles are allowed.

4. Time complexity of the best-fit heuristic

4.1. Efficient implementation

The best-fit heuristic repeatedly searches for the lowest available segment and the best fit rectangle until all the rectangles are placed in the strip. In order to implement this algorithm, Burke et al. [4] used a vector of size W to store the skyline and a sorted list of rectangles by decreasing width to find the best fit rectangle. Their implementation requires $O(n+W)$ space and $O(n^2+nW)$ computation time.

We give the following efficient implementation of the best-fit heuristic. Our implementation stores the skyline (i.e., line segments) using a heap and a doubly linked list connected by bi-directional pointers. It also uses a balanced binary search tree to find the best fit rectangle, which stores the remaining (i.e., unpacked) rectangles. We first describe how we initialize and maintain these data structures.

In order to store the remaining rectangles and efficiently find the best fit rectangle among them, we use a balanced tree [16,19] in which rectangles are compared with their widths. A typical balanced tree requires $O(n)$ space and its construction takes $O(n \log n)$ time. We note that, when rectangles can be rotated by 90°, two candidates (i.e., (w_i, h_i) and (h_i, w_i)) of a rectangle i are stored in the tree. It is possible to find the best fit rectangle and delete it (if rotation is allowed, delete both candidates of the rectangle) from the balanced tree in $O(\log n)$ time.

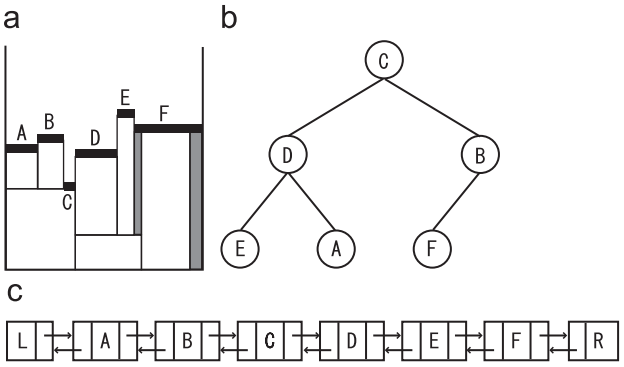

Fig. 2. A skyline of the strip: (a) a skyline consisting of six line segments, (b) a heap storing the segments using their y-coordinates as keys, and (c) a doubly linked list storing the segments sorted by their x-coordinates.

In order to store the skyline (i.e., a set of line segments), our implementation uses a heap and a doubly linked list connected by bi-directional pointers. The heap stores the line segments using their y-coordinates as keys, where a segment with smaller y-coordinate has more priority (resolving equal y-coordinates by smaller x-coordinates). As a result, the lowest available segment can be found at the root node of the heap. The doubly linked list also stores the segments sorted by their x-coordinates. Fig. 2 shows examples of the heap and the doubly linked list. Cells with labels L and R in Fig. 2(c) correspond to the left and right boundaries of the strip, respectively. Each segment appears in both of the heap and the linked list, and each of such corresponding pair (i.e., the element in the heap and the cell in the linked list representing the same segment) is connected by a bi-directional pointer. At the beginning of the execution of the algorithm, no rectangles are placed in the strip and the skyline consists of only one line segment. Hence, the heap and the doubly linked list contain only one element in the preprocessing stage, that is, their construction is done in constant time.

Let us analyze an execution of the algorithm. The lowest available segment can be found in constant time by looking at the root node of the heap, and the best fit rectangle (found in $O(\log n)$ time by using the balanced tree) is placed on the segment. After placing the best fit rectangle on the lowest available segment, the data structures (heap, doubly linked list and balanced tree) should be updated. According to the changes in the skyline, the heap and the doubly linked list are updated. The top edge of the best fit rectangle is inserted as a new line segment of the skyline, and it is merged with its adjacent segment(s) if their y-coordinates are the same. The width of the lowest available segment is shrunk, where the lowest available segment is deleted when its width becomes 0 (this happens if the original width of the lowest available segment is the same as that of the best fit rectangle). As such changes are necessary only to a constant number of segments around the lowest available segment, updating the doubly linked list takes constant time, and updating the heap takes time proportional to the height of the heap, that is, $O(\log n)$ time.

If the best fit rectangle is not found (in other words, the lowest available segment is too narrow to place a remaining rectangle), the segment is raised to the level of the lower segment adjacent to it, and the two segments are merged (if the two adjacent segments have the same height, then three segments are merged). We call this a

segment-raising operation. According to such changes in the skyline, the doubly linked list is updated in constant time, and the heap is updated in O(log n) time. We note that a pointer from the linked list to the heap is needed in the case where the two adjacent segments to the current segment have the same y-coordinate and three segments are merged to one. The segment-raising operation is applied at most $n-1$ times throughout an execution of the algorithm for the following reason. There is one line segment at the beginning of the execution. The number of line segments in the skyline is increased by at most one when a new rectangle is placed in the strip, while number of line segments is decreased by at least one when a segment-raising operation is applied.

Postprocessing stage: If the rotation of rectangles by 90° is allowed, the procedure to remove towers can also be applied. In order to remove towers, the following operations are repeated until there is no improvement in solution quality: (1) find a rectangle that touches the top of the strip and check its orientation and (2) rotate it by 90° and place it on the lowest available segment. At the beginning of the postprocessing stage, the rectangles are sorted by decreasing order of y-coordinates of those top edges, this is done in O(n log n) time. By using this sorted list, a rectangle that touches the top of the strip can be found in constant time throughout the postprocessing stage. In order to place a removed rectangle on the lowest available segment, the heap and the linked list are also utilized. Hence, O(n log n) time is required for the postprocessing stage. We note that the postprocessing stage is an optional part of the algorithm. For some pathological instances, we should examine $\Theta(n)$ rectangles in the postprocessing stage. However, a small number of rectangles are examined for practical instances and its execution time is negligible.

4.2. Optimality of our implementation

In order to show the optimality of our implementation, we first give the following lemma about the worst-case computation time of the best-fit heuristic.

Lemma 1. *The worst-case computation time of the best-fit heuristic is* $\Omega(n \log n)$.

Proof. We show that the best-fit heuristic can sort given numbers in the decreasing order. Let $X=\{x_1, x_2, \ldots, x_n\}$ be an input of the sorting problem, where X is a set of unsorted positive numbers. Let us define $x_{\max} = \max\{x_1, x_2, \ldots, x_n\}$ and set the width W of the strip to $2x_{\max}$. A set of n rectangles is generated as follows: For each x_i, a rectangle i has its width $w_i = x_i + x_{\max}$ and height $y_i = W+1$. It takes O(n) time to construct this instance of the strip packing problem.

In the resulting strip packing instance, any rectangle cannot be rotated by 90°, and any two rectangles cannot be placed at the same height. When the best-fit heuristic is applied to this instance, it places rectangles in the strip from the bottom to the top with decreasing width of the rectangles. It is known that any sorting algorithm without particular assumptions requires $\Omega(n \log n)$ time in the worst case, and the result follows. □

We then analyze the space and time complexities of our implementation. It uses a balanced tree of size O(n) to find the best fit rectangle and a combination of heap and doubly linked list of size O(n) to store all the line segments and to find the lowest available segment. Some information for each rectangle (size, coordinates, pointers and so on) is also stored in a vector of size O(n). Hence, O(n) space is required in total. As for the time complexity, in the initialization of the data structures, our implementation requires O(n log n) time to construct the balanced tree of rectangles and O(1) time to initialize the heap and the doubly linked list. In the main part of the algorithm, the number of iterations is O(n) and each iteration requires O(log n) time, i.e., O(n log n) time is spent. The postprocessing operations also require O(n log n) time. In total, our implementation of the best-fit heuristic requires O(n log n) time.

Putting together this result and Lemma 1, we have the following theorem. We note that any implementation of the best-fit heuristic requires $\Omega(n)$ space.

Theorem 1. *The optimal implementation of the best-fit heuristic needs* $\Theta(n)$ *space and runs in* $\Theta(n \log n)$ *time.*

4.3. Computational results

We evaluate the execution time of the proposed implementation via computational experiments. The algorithm was coded in C and run on a PC with an Intel Pentium 4 3.2 GHz CPU and 1 GB RAM. Test instances were randomly generated with a method proposed by Wang and Valenzela [21]. The whole set of instances comprises 17 classes of instances, classified depending on the number of rectangles, each class consisting of 10 instances. The smallest instance has $2^4 = 16$, the next one has $2^5 = 32$, and the largest one has $2^{20} = 1,048,576$ rectangles. All the test instances are electronically available from our web site (http://www.simplex.t.u-tokyo.ac.jp/~imahori/packing/). In our computational experiments, we assume that each rectangle can be rotated by 90°. The results are shown in Fig. 3.

In this figure, horizontal axis shows the number of rectangles and vertical axis shows the computation time (average of 10 different instances) in seconds. We note that this is a double logarithmic chart. From Fig. 3, we can observe that our implementation runs very fast for every instance. It can place 256 rectangles within 0.001 s (1 ms), it spends <0.1 s for instances with up to 16,384 rectangles, and it takes <10 s to place about one million rectangles.

5. Approximation ratio of the best-fit heuristic

5.1. Worst-case approximation ratio

This section discusses the worst-case approximation ratio of the best-fit heuristic. Let h_{OPT} be the optimal height of the strip for a given instance and h_{BF} be the height computed by the best-fit heuristic. We first show a negative aspect of the best-fit heuristic: It cannot guarantee any approximation ratio in the worst case if the rectangles can be rotated by 90°.

Theorem 2. *Consider the rectangular strip packing problem in which each rectangle can be rotated by* 90°. *Then, for any constant* $M>0$, *there exist instances such that* $h_{\mathrm{BF}}/h_{\mathrm{OPT}} > M$ *holds.*

Proof. We consider the following instance with 4 rectangles: The sizes of rectangles are $(1-2\varepsilon, 2\varepsilon)$, $(1-2\varepsilon, 2\varepsilon)$, $(1-5\varepsilon, 3\varepsilon)$ and $(1-5\varepsilon, 3\varepsilon)$, where $0<\varepsilon \ll 1$ holds, and the strip width W is 1. When the best-fit heuristic is applied to this instance, the resulting placement has height $1-2\varepsilon$ as shown in Fig. 4(a). (If the best-fit heuristic with the high or low strategies [4] is applied, the resulting placement also has height $1-2\varepsilon$ as shown in Fig. 4(a) and (b), respectively.) We note that the postprocessing operation (i.e., the procedure to remove towers) cannot improve the above placements. The optimal height of this instance is 10ε as shown in Fig. 4(c). By setting ε to $0<\varepsilon<1/(10M+2)$, the result follows. □

We then consider the case without rotations. Let n be the number of rectangles to be placed and α be the positive value satisfying $\alpha^{\alpha}=n$. We show the following two lemmas.

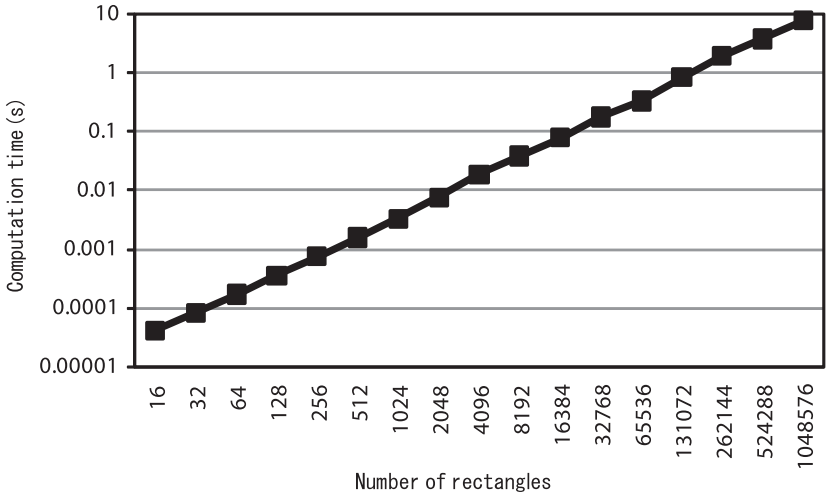

Fig. 3. Computation time (s) for instances with various sizes.

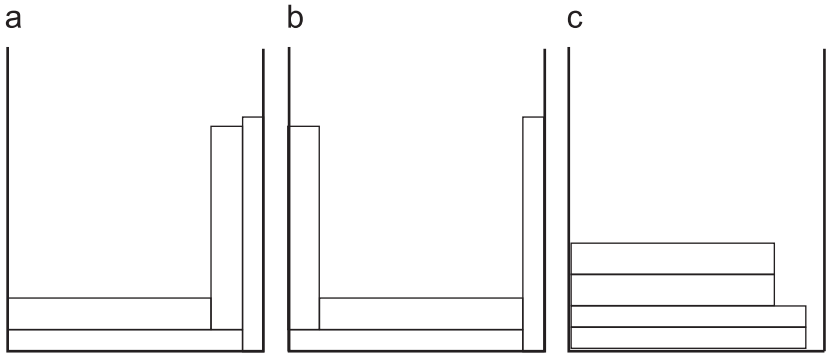

Fig. 4. An adverse instance to the best-fit heuristic for the case with rotations of 90°: (a) a placement by the best-fit heuristic with the left or high strategies, (b) a placement by the best-fit heuristic with the low strategy, and (c) an optimal placement.

Lemma 2. *For the rectangular strip packing problem with each rectangle having a fixed orientation, there exist instances for which* $h_{BF}/h_{OPT} > \alpha/2 - 1$ *holds.*

Proof. We design adversary instances of the rectangular strip packing problem for which the best-fit heuristic cannot work effectively. Let $k(\geq 4)$ be an even number and ε be a small positive. A strip with width $W = k^k$ and a set of rectangles $I = I_1 \cup I_2 \cup \cdots \cup I_{k/2}$ are given, where each subset of rectangles I_j for $j = 1, 2, \ldots, k/2$ is defined as follows: Let $p_j = k^{k-j} - k^{k-j-1}$. For each $i = 1, 2, \ldots, p_j - 1$, there are $2k^{j-1} + 2$ rectangles with four different sizes $(k^{k-j+1} - ik + 1, \varepsilon), (k^j - k^{j-1}, 2(p_j - i)\varepsilon), (k^{k-j+1} - ik, \varepsilon)$ and $(k^{j-1}, 1 - (2i - 1)\varepsilon)$, which consist of k^{j-1} congruent rectangles with the first and third sizes, respectively, and one rectangle with each of the second and fourth sizes. We call these rectangles Types 1–4, respectively. There are also k^{j-1} congruent rectangles whose size is $(k^{k-j} + 1, 1 - 2(p_j - 1)\varepsilon)$, which is called Type 5. In total, the set I_j consists of $2k^{k-1} - 2k^{k-2} + 2k^{k-j} - 2k^{k-j-1} - k^{j-1} - 2$ rectangles. Each rectangle in I_j has width w that satisfies $k^{j-1} \leq w < k^j$ (Type 2 or 4) or $k^{k-j} < w < k^{k-j+1}$ (Type 1, 3 or 5). See < ?MCtwidthtextwidth? > Table 1 for the details of this instance. (Note that for $j = k/2$, rectangles of Type 5 and those of Types 1–4 with $i \geq 2$ are not shown because they are not necessary to show that h_{BF} is large, as will be discussed later.) In this table, each column "#" shows the number of rectangles and each column "size" shows their sizes.

Let us estimate h_{BF} on this instance. Fig. 5 and Table 1 bring a better understanding for the following explanation. When the best-fit heuristic is applied to this instance, the widths of the lowest available segments in the first $|I_1| = 4k^{k-1} - 4k^{k-2} - 3$ steps are sufficiently large for all the remaining rectangles (i.e., the widest rectangle is selected to place) or less than k (i.e., too narrow to place rectangles belonging to $I_2 \cup I_3 \cup \cdots \cup I_{k/2}$). Hence, rectangles belonging to the set I_1 are placed in the strip first. (The rectangles of Types 1, 2, 3 and 4 are placed in this order for $i = 1$, and then the same happens for $i = 2, 3, \ldots, p_1 - 1$. The Type 5 rectangle is placed last among those in I_1.) After $4k^{k-1} - 4k^{k-2} - 3$ steps, all the rectangles in I_1 have already been placed and no other rectangles have been placed yet. Fig. 5(b) shows this situation. In the skyline at this moment, line segments whose y-coordinates are 1 and those whose y-coordinates are < 1 appear alternately, and any line segment whose y-coordinate is < 1 cannot accommodate any remaining rectangle. Thus, all these segments are merged with their adjacent segments, and the resulting skyline consists of a line segment whose width is W and y-coordinate is 1. Rectangles belonging to I_2 are placed in the strip next, and a similar situation happens. The segments whose y-coordinates are < 2 are narrower than the width of any remaining rectangle, and the skyline becomes a line segment with width W at y=2. The remaining rectangles are placed similarly for j=3, 4, …, k/2. Finally, the height of the strip h_{BF} becomes $k/2$. Furthermore, the most part of rectangles belonging to $I_{k/2}$ can be deleted without changing h_{BF} (e.g., for $j = k/2$, we can remove all the rectangles of Types 1 to 4 for $i \geq 2$ and all the

Table 1
The detailed information of an adversary instance for the best-fit heuristic.

j	i	Type 1		Type 2		Type 3	
		#	size	#	size	#	size
	1	1	(k^k-k+1,ε)	1	$(k-1,2(p_1-1)\varepsilon)$	1	(k^k-k,ε)
	⋮		⋮		⋮		⋮
1	p_1-1	1	$(k^{k-1}+k+1,\varepsilon)$	1	$(k-1,2\varepsilon)$	1	$(k^{k-1}+k,\varepsilon)$
	1	k	$(k^{k-1}-k+1,\varepsilon)$	1	$(k^2-k,2(p_2-1)\varepsilon)$	k	$(k^{k-1}-k,\varepsilon)$
	⋮		⋮		⋮		⋮
2	p_2-1	k	$(k^{k-2}+k+1,\varepsilon)$	1	$(k^2-k,2\varepsilon)$	k	$(k^{k-2}-k,\varepsilon)$
⋮				⋮			
	⋮		⋮		⋮		⋮
j	i	k^{j-1}	$(k^{k-j+1}-ik+1,\varepsilon)$	1	$(k^j-k^{j-1},2(p_j-i)\varepsilon)$	k^{j-1}	$(k^{k-j+1}-ik,\varepsilon)$
	⋮		⋮		⋮		⋮
⋮				⋮			
	1	$k^{k/2-2}$	$(k^{k/2+2}-k+1,\varepsilon)$	1	$(k^{k/2-1}-k^{k/2-2},2(p_{k/2-1}-1)\varepsilon)$	$k^{k/2-2}$	$(k^{k/2+2}-k,\varepsilon)$
$\frac{k}{2}-1$	⋮		⋮		⋮		⋮
	$p_{k/2-1}-1$	$k^{k/2-2}$	$(k^{k/2+1}+k+1,\varepsilon)$	1	$(k^{k/2-1}-k^{k/2-2},2\varepsilon)$	$k^{k/2-2}$	$(k^{k/2+1}+k,\varepsilon)$
$\frac{k}{2}$	1	$k^{k/2-1}$	$(k^{k/2+1}-k+1,\varepsilon)$	1	$(k^{k/2}-k^{k/2-1},2(p_{k/2}-1)\varepsilon)$	$k^{k/2-1}$	$(k^{k/2+1}-k,\varepsilon)$

j	i	Type 4		Type 5		Total # of rectangles
		#	size	#	size	
	1	1	$(1,1-\varepsilon)$			
	⋮		⋮			
1	p_1-1	1	$(1,1-(2p_1-3)\varepsilon)$	1	$(k^{k-1}+1,1-(2p_1-1)\varepsilon)$	$4k^{k-1}-4k^{k-2}-3$
	1	1	$(k,1-\varepsilon)$			
	⋮		⋮			
2	p_2-1	1	$(k,1-(2p_2-3)\varepsilon)$	k	$(k^{k-2}+1,1-(2p_2-1)\varepsilon)$	$2k^{k-1}-2k^{k-3}-k-2$
⋮				⋮		
	⋮		⋮			$2k^{k-1}-2k^{k-2}$
j	i	1	$(k^{j-1},1-(2i-1)\varepsilon)$	k^{j-1}	$(k^{k-j}+1,1-(2p_j-1)\varepsilon)$	$+2k^{k-j}-2k^{k-j-1}$
	⋮		⋮			$-k^{j-1}-2$
⋮				⋮		
	1	1	$(k^{k/2-2},1-\varepsilon)$			$2k^{k-1}-2k^{k-2}$
$k/2-1$	⋮		⋮	$k^{k/2-2}$	$(k^{k/2+1}+1,1-(2p_{k/2-1}-1)\varepsilon)$	$+2k^{k/2+1}-2k^{k/2}$
	$p_{k/2-1}-1$	1	$(k^{k/2-2},1-(2p_{k/2-1}-3)\varepsilon)$			$-k^{k/2-2}-2$
$\frac{k}{2}$	1	1	$(k^{k/2-1},1-\varepsilon)$			$2k^{k/2-1}+2$

rectangles of Type 5). With such deletion of rectangles, our instance has

$$\sum_{j=1}^{k/2-1}(2k^{k-1}-2k^{k-2}+2k^{k-j}-2k^{k-j-1}-k^{j-1}-2)+2k^{k/2-1}+2$$
$$=\left(k^k-2k^{k-1}-(k-2)k^{k-2}+2k^{k-1}-2k^{k/2}-\sum_{j=1}^{k/2-1}k^{j-1}-k+2\right)+2k^{k/2-1}+2$$
$$=k^k-(k-2)k^{k-2}-2(k-1)k^{k/2-1}-\sum_{j=1}^{k/2-1}k^{j-1}-(k-4)<k^k$$

rectangles in total and $h_{\mathrm{BF}}=k/2$ holds.

We then consider a feasible placement for this instance, which is obtained by modifying the above placement. All the rectangles whose heights are more than $k^k\varepsilon$ (i.e., rectangles of Types 4 and 5) are removed from the placement generated by the best-fit heuristic, and the rectangles left in the strip are moved as downward as possible. At this moment, the height of the strip becomes

$$\sum_{j=1}^{k/2}2(p_j-1)\varepsilon=\sum_{j=1}^{k/2}(2k^{k-j}-2k^{k-j-1}-2)\varepsilon=(2k^{k-1}-2k^{k/2-1}-k)\varepsilon.$$

The removed rectangles (i.e., rectangles whose heights are more than $k^k\varepsilon$) are then placed in the strip. Since the total width of such rectangles is less than the width of the strip, all of them can be placed on

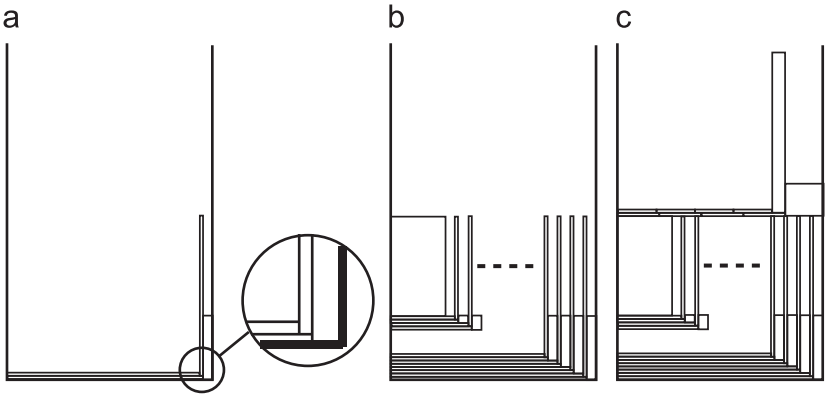

Fig. 5. An adverse instance ($k=4$) to the best-fit heuristic without rotations: (a) first four rectangles are placed in the strip, (b) $4k^{k-1}-4k^{k-2}-3$ rectangles that belong to I_1 are placed in the strip, and (c) next $2k+2$ rectangles are placed in the strip.

a horizontal line (in other words, on the same level). The maximum height of the rectangles is $1-\varepsilon$. We now have a feasible placement whose strip height is $(2k^{k-1}-2k^{k/2-1}-k)\varepsilon+1-\varepsilon$, and this implies that $h_{\mathrm{OPT}}<1+2k^{k-1}\varepsilon$ holds.

We summarize the above results. For an even number k, our instance has less than k^k rectangles and $h_{\mathrm{BF}}/h_{\mathrm{OPT}}>k/2(1+2k^{k-1}\varepsilon)$ holds. By setting k to be the maximum even number satisfying $k^k\le n$ and ε to be an appropriate small positive value, the result follows. □

Lemma 3. *For the rectangular strip packing problem with each rectangle having a fixed orientation, the best-fit heuristic always outputs a solution with* $h_{\mathrm{BF}}\,h_{\mathrm{OPT}}<2\alpha+3$.

Proof. We analyze the placement obtained by the best-fit heuristic. (Note that this analysis can also be applied to placements obtained by the best-fit heuristic with high or low placement strategies [4].) Let us consider a horizontal line $y=h(0\le h\le h_{\mathrm{BF}})$, which has intersections with some rectangles in the strip. We assume that a line intersects the top edges of rectangles but does not intersect the bottom edges of rectangles, i.e., a line $y=h$ intersects a rectangle i if and only if $y_i<h\le y_i+h_i$. We define $W_{\mathrm{int}}(h)=\sum_{i:y_i<h\le y_i+h_i}w_i$ and

$$f(h)=\begin{cases}0 & \text{if } W_{\mathrm{int}}(h)\le W/(\alpha+2),\\ 1 & \text{if } W_{\mathrm{int}}(h)>W/(\alpha+2).\end{cases}$$

Then, $h_{\mathrm{BF}}=\int_0^{h_{\mathrm{BF}}}f(h)\,\mathrm{d}h+\int_0^{h_{\mathrm{BF}}}(1-f(h))\,\mathrm{d}h$ holds. Let us analyze each term of this equation independently. For the first term, the following equality and inequalities hold:

$$\int_0^{h_{\mathrm{BF}}}f(h)\,\mathrm{d}h<\frac{\alpha+2}{W}\int_0^{h_{\mathrm{BF}}}W_{\mathrm{int}}(h)\,\mathrm{d}h=\frac{\alpha+2}{W}\sum_{i=1}^{n}w_ih_i\le(\alpha+2)h_{\mathrm{OPT}},$$

where the *continuous lower bound* (i.e., $h_{\mathrm{OPT}}\ge\sum_{i=1}^{n}w_ih_i/W$) is used.

In order to analyze the second term, we consider a set of horizontal lines $y=H_j(j=0,1,2,\ldots)$:

$$H_0=\begin{cases}0 & \text{if } \{h|f(h)=0,0<h\le h_{\mathrm{BF}}\}=\emptyset,\\ \max\{h|f(h)=0,0<h\le h_{\mathrm{BF}}\} & \text{otherwise,}\end{cases}$$

$$H_j=\begin{cases}0 & \text{if } \{h|f(h)=0,0<h\le H_{j-1}-h_{\max}\}=\emptyset\ (j\ge1),\\ \max\{h|f(h)=0,0<h\le H_{j-1}-h_{\max}\} & \text{otherwise,}\end{cases}$$

where $h_{\max}=\max\{h_1,h_2,\ldots,h_n\}$. For each $j=0,1,2,\ldots$, we now show that $H_j=0$ holds or the number of rectangles that intersect line $y=H_j$ is at least α^j.

A key to proving this claim is the following observation. Recall that the best-fit heuristic creates waste space (i.e., space unoccupied by any rectangle) only when it executes the operation of raising the lowest available segment. When this operation is executed, all the remaining rectangles have width greater than that of the raised segment. Now consider a horizontal line $y=h$ and any waste interval on this line, which is the intersection of a waste space and the line $y=h$. This waste interval must have been created by an operation of raising a lowest available segment of height less than h. The above observation indicates that the width of this waste interval must be less than that of any rectangle placed above it (i.e., rectangle i with $y_i\ge h$) if such a rectangle exists.

Suppose $H_0>0$. Then at least one $(=\alpha^0)$ rectangle intersects line $y=H_0$, and hence the above claim is satisfied for $j=0$. Moreover, $H_0>0$ implies $f(H_0)=0$, and hence $W_{\mathrm{int}}(H_0)=\sum_{i:y_i<H_0\le y_i+h_i}w_i\le W/(\alpha+2)$ holds. A rectangle i that intersects $y=H_0$ has width $w_i\le W/(\alpha+2)$ and y-coordinate of the bottom edge $y_i\ge H_0-h_{\max}$. This means that any waste interval whose y-coordinate is at most $H_0-h_{\max}$ has width less than $W/(\alpha+2)$ from the observation in the previous paragraph.

Let us consider the next line $y=H_1$ and assume $H_1>0$. This assumption implies $f(H_1)=0$, and hence we have $W_{\mathrm{int}}(H_1)\le W/(\alpha+2)$. The sum of the widths of waste intervals on this line (i.e., $W-W_{\mathrm{int}}(H_1)$) is at least $W-W/(\alpha+2)=(\alpha+1)W/(\alpha+2)$. Because $H_1\le H_0-h_{\max}$, the width of each waste interval on $y=H_1$ is less than $W/(\alpha+2)$. Hence there are more than $\alpha+1$ waste intervals on line $y=H_1$. Consequently, more than α rectangles intersect line $y=H_1$, and the minimum width of these rectangles is less than $W/(\alpha(\alpha+2))$, because the sum of these rectangles' widths is at most $W/(\alpha+2)$. Moreover, any waste interval whose y-coordinate is at most $H_1-h_{\max}$ has width less than $W/(\alpha(\alpha+2))$.

By continuing similar analysis on lines $y=H_j(j=2,3,\ldots)$, we can confirm that the number of rectangles intersecting line $y=H_j$ is at least α^j unless $H_j=0$. We defined α as $\alpha^\alpha=n$, and hence $H_j=0$ if $j>\alpha$. We now have the following equality and inequalities:

$$\int_0^{h_{\mathrm{BF}}}(1-f(h))\,\mathrm{d}h=\sum_{j=0}^{\lfloor\alpha\rfloor}\left(\int_{H_{j+1}}^{H_j}(1-f(h))\,\mathrm{d}h\right)$$
$$\le(\lfloor\alpha\rfloor+1)h_{\max}\le(\alpha+1)h_{\mathrm{OPT}}.$$

Putting together these estimations on $\int_0^{h_{\mathrm{BF}}}f(h)\,\mathrm{d}h$ and $\int_0^{h_{\mathrm{BF}}}(1-f(h))\,\mathrm{d}h$, the result follows. □

By Lemmas 2 and 3, we have the following theorem on the approximation guarantee of the best-fit heuristic.

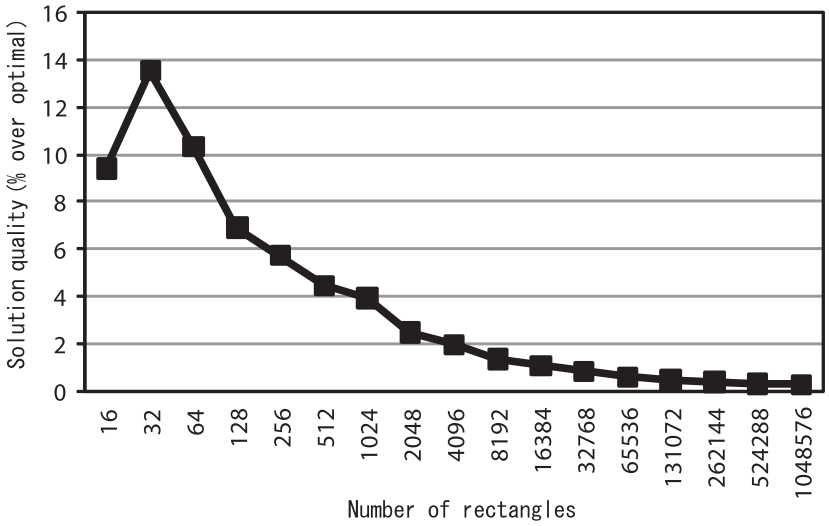

Fig. 6. Solution quality (% over optimal) for instances with various sizes.

Theorem 3. *The best-fit heuristic is a $\Theta(\alpha)$-approximation algorithm for the rectangular strip packing problem with each rectangle having a fixed orientation, where α is the positive value satisfying $\alpha^\alpha = n$ and n is the number of rectangles to be placed.*

5.2. Computational results

As confirmed in the paper by Burke et al. [4], the best-fit heuristic works very effectively for many benchmark instances. In this section, we investigate the relationship between the number of rectangles and solution quality of the best-fit heuristic. The results in Section 5.1 indicate that in the worst case, the best-fit heuristic can output solutions with deviation from the optimal height increasing with the number of rectangles. This tendency, however, was proved based on a pathological instance including rectangles with very large aspect ratio. The objective of this section is to observe the behavior of the best-fit heuristic for more standard instances. We use the test instances explained in Section 4.3 whose numbers of rectangles are from $n = 16$ to 1,048,576. Note that the optimal height for each instance is known (more precisely, each instance has a placement without any waste space). The computational results are shown in Fig. 6.

In this figure, horizontal axis shows the number of rectangles and vertical axis shows the solution quality (average of 10 instances), where the quality is measured by the deviation in % from the optimal height H^*, i.e., $100(H - H^*)/H^*$, and hence the smaller value means a better solution. From Fig. 6, we can observe that the number of rectangles has a substantial influence on the solution quality of the best-fit heuristic. When the number of rectangles becomes larger, solution quality becomes better.

6. Conclusions

In this paper we proposed an efficient implementation of the best-fit heuristic for the rectangular strip packing problem. The proposed implementation requires $O(n)$ space and $O(n \log n)$ time, which is optimal for the best-fit heuristic. The proposed implementation is simple, and hence it is easy to implement. We confirmed through computational experiments that our implementation required very small execution time: It spent about 0.001 s to place hundreds of rectangles, and 10 s for test instances with about one million rectangles.

As for the solution quality of the best-fit heuristic, we proved that the best-fit heuristic is a $\Theta(\alpha)$-approximation algorithm for the rectangular strip packing problem for the α satisfying $\alpha^\alpha = n$. We also showed via computational experiments that in practice, the solution quality becomes better as the number of rectangles increases.

Acknowledgments

The authors thank the anonymous referees for their helpful comments. This work was partially supported by Grants-in-Aid for Scientific Research, by the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[1] Alvarez-Valdes R, Parreno F, Tamarit JM. Reactive GRASP for the strip-packing problem. Computers and Operations Research 2008;35:1065–83.
[2] Baker BS, Coffman Jr EG, Rivest RL. Orthogonal packings in two dimensions. SIAM Journal on Computing 1980;9:846–55.
[3] Bortfeldt A. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. European Journal of Operational Research 2006;172:814–37.
[4] Burke EK, Kendall G, Whitwell G. A new placement heuristic for the orthogonal stock-cutting problem. Operations Research 2004;52:655–71.
[5] Burke EK, Kendall G, Whitwell G. A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem. INFORMS Journal on Computing 2009, in press, doi:10.1287/ijoc.1080.0306.
[6] Chazelle B. The bottom-left bin-packing heuristic: an efficient implementation. IEEE Transactions on Computers 1983;32:697–707.
[7] Coffman Jr EG, Garey MR, Johnson DS, Tarjan RE. Performance bounds for level-oriented two-dimensional packing algorithms. SIAM Journal on Computing 1980;9:808–26.
[8] Dyckhoff H. A typology of cutting and packing problems. European Journal of Operational Research 1990;44:145–59.
[9] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. New York: Freeman; 1979.
[10] Hopper E, Turton BCH. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. European Journal of Operational Research 2001;128:34–57.
[11] Huang W, Chen D. An efficient heuristic algorithm for rectangle-packing problem. Simulation Modelling Practice and Theory 2007;15:1356–65.
[12] Imahori S, Yagiura M, Ibaraki T. Improved local search algorithms for the rectangle packing problem with general spatial costs. European Journal of Operational Research 2005;167:48–67.
[13] Jakobs S. On genetic algorithms for the packing of polygons. European Journal of Operational Research 1996;88:165–81.
[14] Jansen K, van Stee R. On strip packing with rotations. In: Proceedings of the 37th ACM symposium on theory of computing, 2005. p. 755–61.
[15] Kenyon C, Remila E. A near-optimal solution to a two-dimensional cutting stock problem. Mathematics of Operations Research 2000;25:645–56.

[16] Knuth DE. The art of computer programming, vol. 3, 2nd ed. Reading, MA: Addison-Wesley; 1998.
[17] Liu D, Teng H. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. European Journal of Operational Research 1999;112: 413–20.
[18] Lodi A, Martello S, Vigo D. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. INFORMS Journal on Computing 1999;11:345–57.
[19] Sedgewick R. Algorithms in C. 3rd ed., Reading, MA: Addison-Wesley; 1997.
[20] Steinberg A. A strip-packing algorithm with absolute performance bound 2. SIAM Journal on Computing 1997;26:401–9.
[21] Wang PY, Valenzela CL. Data set generation for rectangular placement problems. European Journal of Operational Research 2001;134:378–91.
[22] Wäscher G, Haußner H, Schumann H. An improved typology of cutting and packing problems. European Journal of Operational Research 2007;183: 1109–30.