An Exact Algorithm for the Two-Dimensional Strip-Packing Problem

Article *in* Operations Research · December 2010

DOI: 10.1287/opre.1100.0833 · Source: DBLP

Marco Antonio Boschetti
University of Bologna
50 PUBLICATIONS **809** CITATIONS

OPERATIONS RESEARCH
Vol. 58, No. 6, November–December 2010, pp. 1774–1791
ISSN 0030-364X | EISSN 1526-5463 | 10 | 5806 | 1774

DOI 10.1287/opre.1100.0833
© 2010 INFORMS

An Exact Algorithm for the Two-Dimensional Strip-Packing Problem

Marco Antonio Boschetti, Lorenza Montaletti
Department of Mathematics, University of Bologna, 47023 Cesena, Italy
{marco.boschetti@unibo.it, montaletti.l@libero.it}

This paper considers the two-dimensional strip-packing problem (2SP) in which a set of rectangular items have to be orthogonally packed, without overlapping, into a strip of a given width and infinite height by minimizing the overall height of the packing. The 2SP is NP-hard in the strong sense and finds many practical applications. We propose reduction procedures, lower and upper bounds, and an exact algorithm for the 2SP. The new lower bounds are both combinatorial bounds and bounds derived from different relaxations of mathematical formulations of the 2SP. The new upper bounds are obtained by constructive heuristics based on different strategies to place the items into the strip. The new exact method is based on a branch-and-bound approach. Computational results on different sets of test problems derived from the literature show the effectiveness of the new lower and upper bounds and of the new exact algorithm.

Subject classifications: production/scheduling: cutting stock/trim; programming: integer; algorithms: branch and bound.
Area of review: Optimization.
History: Received November 2007; revisions received April 2009, December 2009; accepted February 2010.

1. Introduction

The *two-dimensional strip-packing problem* (2SP) consists of packing n rectangular *items* into a rectangular master surface, called a *strip*, with a given width W and infinite height. Each item i, having width w_i and height h_i, has to be packed with its edges always parallel or orthogonal to the edges of the master surface (*orthogonal cuts*). The objective is to pack all the given items minimizing the total height of the used strip. We assume that the items have a fixed orientation and integer sizes; no other restrictions are required, such as guillotine cuts.

According to the typology introduced by Wäscher et al. (2007), the 2SP can be classified as *two-dimensional open dimension problem* (2-ODP).

Martello et al. (2003) show that an instance of *one-dimensional bin-packing problem* (1BP), where n items of size w_j have to be allocated in the minimum number of bins of capacity W, can be transformed into a 2SP instance by setting the item heights $h_j = 1$. Therefore, the 1BP is strongly NP-hard, and so also is the 2SP (see Garey and Johnson 1979).

The 2SP has many real-world applications. For example, the 2SP appears in steel and paper industries, where rolls of material have to be cut into small rectangles, or in transportation, where the length of the used truck bed has to be minimized.

In the literature, many heuristic approaches have been presented to solve the 2SP so far. The first greedy heuristics were based on *Bottom-Left* (BL) and *Bottom-Left-Fill* (BLF) approaches and were proposed by Baker et al. (1980) and by Chazelle (1983). Given an ordered list of items, BL and BLF approaches place each item in turn into the strip. BL heuristics usually put each item in the top-right corner and move it as far down and left as possible. BLF heuristics generate a list of positions from the emerging solution and place each item in the lowest and leftmost feasible position contained in the list. Hence, BLF heuristics are able to fill some empty areas in the strip that BL heuristics cannot fill, but BLF heuristics are computationally more expensive. For the interested reader, a detailed description of BL and BLF approaches can be found in Hopper and Turton (2001) or Burke et al. (2004), and an interesting implementation of a BL procedure for the two-stage cutting stock problem was proposed by Alvarez-Valdes et al. (2007).

Recently, a number of metaheuristics have been proposed for the 2SP. These algorithms make use of BL and BLF approaches as local search, and at each iteration the order of the items is updated following different frameworks. Genetic algorithms have been proposed by Jakobs (1996), Gómez and de la Fuente (2000), Liu and Teng (1999), and Yeungm and Tang (2004); whereas Iori et al. (2003) present a genetic algorithm, a tabu search, and a hybrid algorithm. Lesh et al. (2005) and Lesh and Mitzenmacher (2006) randomly update the order of the items for their BL local search according to a given probability distribution or using a *BubbleSearch* approach, respectively.

Two completely different algorithms are proposed by Martello et al. (2003) and are called *BUILD* and *JOIN*. The algorithm BUILD starts from the solution provided by a new relaxation of the 2SP, called *one-dimensional contiguous bin-packing problem* (1CBP), and tries to recover a feasible layout. JOIN applies a BL heuristic and three level-oriented packing algorithms to a modified instance obtained from the original by joining pairs of items differing in height by no more than a given threshold.

Burke et al. (2004) propose a greedy heuristic based on a *Best-Fit* (BF) approach. At each iteration the BF heuristic adds to the emerging solution the item that best fits the lowest available space within the strip. Burke et al. (2009) improve the BF heuristic making use of BLF and of metaheuristics. In particular, their improved algorithm applies the BF heuristic until $n-m$ items are packed, where m is a parameter, and then applies for the remaining m items the BLF heuristic in a metaheuristic framework that generates different item orderings for the BLF. The metaheuristic used is a simulated annealing algorithm. Burke et al. (2004, 2009) allow rotations of 90 degrees. A complexity analysis of the BF heuristic has been recently proposed by Imahori and Yagiura (2010). They propose an efficient implementation of the BF heuristic that requires $O(n)$ space and $O(n \log n)$ time, where n is the number of rectangles, and they show the optimality of their implementation.

Bortfeldt (2006) presents a genetic algorithm where the solutions are not encoded, but the corresponding layouts of the items into the strip are fully defined and directly manipulated by means of specific genetic operators. His genetic algorithm can take into account the orientation and the guillotine constraints.

Alvarez-Valdes et al. (2008) propose a *greedy randomized adaptive search* (GRASP) algorithm that does not allow rotations. Similarly to the BF, at each iteration, their algorithm adds to the emerging solution the item that best fits the available spaces, but avoiding the rigidity of static list by randomizing the selection process. At the end, an improving phase tries to correct some wrong decisions inherent to the randomized construction process.

Belov et al. (2006) adapt heuristics for one-dimensional packing problems to the 2SP and propose two iterative heuristics. The first one is based on a single-pass heuristic that fills every most bottom-left free space in a greedy fashion by solving a one-dimensional knapsack problem that considers only item widths and assigns suitable pseudo profits to the items using the *sequential value correction* (SVC) method already used by Mukhacheva et al. (2000) and Belov and Scheithauer (2007). The second heuristic is based on the randomized framework BubbleSearch of Lesh and Mitzenmacher (2006). It generates different item sequences and applies a *bottom-left-right* (BLR) algorithm that is a simple modification of the BL heuristic.

Lower bounds for the 2SP have been proposed and experimentally evaluated by Martello et al. (2003), Belov et al. (2006), and Alvarez-Valdes et al. (2009); moreover, two exact methods have been proposed by Martello et al. (2003) and by Alvarez-Valdes et al. (2009).

Martello et al. (2003) describe simple lower bounds derived from geometric considerations and adapting results presented in the literature for the 1BP and a lower bound based on the 1CBP relaxation of the problem. The lower bounds of Martello et al. (2003) are also used by Iori et al. (2003) and Bortfeldt (2006).

Belov et al. (2006) propose a lower bound based on the *bar relaxation* of Scheithauer (1999), where the basic idea is to generate a set of one-dimensional packing patterns (*bar patterns*) in horizontal and vertical directions and then to model the problem by means of the generated bar patterns. Belov et al. (2006) compute both the horizontal and vertical bar relaxations, and the overall lower bound is given by the maximum of the two values obtained.

The lower bounds proposed by Alvarez-Valdes et al. (2009) are based on geometric considerations and on relaxations of an integer formulation of the 2SP.

The exact method of Martello et al. (2003) is based on a branch-and-bound approach that makes use of the upper and lower bounds proposed in the same paper and is able to solve test problem instances from the literature involving up to 200 items.

Alvarez-Valdes et al. (2009) describe a branch-and-bound algorithm where the structure of the search tree is the same as that of Martello et al. (2003), but the performance is improved by using the aforementioned new lower bounds and new dominance conditions.

1.1. Contributions

In our view, the main contributions of this paper can be summarized as follows:

(1) We present an effective reduction procedure that can fix a large number of items in the solution for several instances and can provide the optimal solution of some instances. Two other well-known reduction procedures are derived from other packing problems.

(2) Three completely new lower bounds are proposed for the 2SP. The first is a combinatorial bound based on considerations involving item heights. The other two lower bounds are based on relaxations of two different mathematical formulations of the 2SP. The two latter lower bounds are time consuming, but they are very useful for the hardest instances. A fourth lower bound is a combinatorial bound based on the *dual feasible functions* already proposed in the literature.

(3) A new postprocessing is presented to improve the lower bounds that makes use of the *normal pattern principle* (see §2.1), too.

(4) Three new heuristic algorithms and a simple postprocessing procedure to improve the upper bound are described. The first heuristic is an improvement of the *best-fit* approach proposed by Burke et al. (2004). The remaining two heuristics are completely new.

(5) We propose a branch and bound that follows the same classical depth-search strategy as Martello et al. (2003) and Alvarez-Valdes et al. (2009) but uses a different rule to generate nodes and a simple *dynamic* strategy to select the node to expand.

(6) Computational results on a large number of test instance sets are presented. Our exact algorithm was also tested on instances not considered by the exact algorithms

proposed in the literature (e.g., instances *gcut5*–*gcut13*, where the strip width ranges between 500 and 3,000). Furthermore, our exact algorithm can solve to optimality several instances not solved so far.

1.2. Outline

The remainder of this paper is organized as follows: In §2 we give the notation and the definitions used throughout the paper and some rules for reducing the problem complexity. New lower bounds and new heuristic algorithms are presented in §§3 and 4, respectively, and the exact method is described in §5. In §6 we analyze the computational performance of the procedures proposed in this paper on test problems derived from the literature. Conclusions are given in §7.

2. Notation, Definitions, and Reductions

A set of n rectangular items of size (w_i, h_i), $i \in P = \{1, \dots, n\}$ and a strip of width W and infinite height are given. Each item i has $w_i \leqslant W$ and a fixed orientation, and it must be packed with its edges always parallel or orthogonal to the edges of the strip, without overlapping other items or the region outside the strip. We assume that the size of the items and the width of the strip are integers. The objective is to cut all the items of P into the strip minimizing the overall height H of the packing pattern.

The strip is located in the positive quadrant of the Cartesian coordinate system with its bottom left-hand corner placed in position $(0, 0)$ and its bottom and left-hand edges parallel to the x-axis and the y-axis, respectively. The position of each item i is defined by the coordinates (x_i, y_i) of its bottom left-hand corner, referred to as the *origin* of the item, into the Cartesian system.

2.1. Principle of Normal Patterns

The origin of an item $i \in P$ can be located at every integer point (x_i, y_i) of the strip, such that $0 \leqslant x_i \leqslant W - w_i$. However, this set of points (x_i, y_i) can be reduced by applying the *principle of normal patterns* proposed by Herz (1972), who speaks of *canonical dissections*, and also used by Christofides and Whitlock (1977).

The principle of normal patterns is based on the observation that, in a given feasible cutting pattern, any cut item can be moved to the left and/or downward as much as possible until its left-hand edge and its bottom edge are both adjacent to other cut items or to the edges of the strip. Let X_i and Y_i denote the subsets of all x-positions and y-positions, respectively, where piece i can be positioned. Using the principle of normal patterns, the sets X_i and Y_i for each piece $i \in P$ can be computed as follows:

$$X_i = \left\{ x = \sum_{k \in P \setminus \{i\}} w_k \xi_k \colon 0 \leqslant x \leqslant W - w_i, \quad \xi_k \in \{0, 1\},\ k \in P \setminus \{i\} \right\} \qquad (1)$$

and

$$Y_i = \left\{ y = \sum_{k \in P \setminus \{i\}} h_k \xi_k \colon 0 \leqslant y \leqslant H - h_i, \quad \xi_k \in \{0, 1\},\ k \in P \setminus \{i\} \right\} \qquad (2)$$

where H is a suitable upper bound to the optimal height of the strip. A simple dynamic programming recursion for computing X_i and Y_i, for every $i \in P$, is described in Christofides and Whitlock (1977).

2.2. Problem Reductions

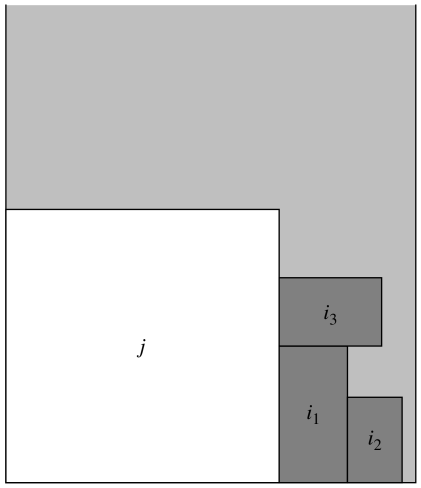
In this paper, we apply two types of reductions: we fix some items in the solution, and we modify the item and strip sizes. These reductions can improve the value of the lower bounds and help the heuristic and the exact algorithms in finding a good-quality feasible solution.

2.2.1. Fixing Some Items in Solution. There are some instances where there is a subset of items $R \subseteq P$ that, without changing the optimal solution value, can be permanently allocated into the bottom of the strip before starting to solve the problem. The subset R contains all items that cannot be placed side by side with any other item of P, i.e., $R = \{j \in P\colon W - w_j < w_i,\ \forall i \in P \setminus \{j\}\}$. All items belonging to the set R can be allocated into a stack in the bottom of the strip occupying a height equal to $H_R = \sum_{j \in R} h_j$. The exact algorithm can solve the problem using the remaining items, $P = P \setminus R$. To recover the optimal solution of the original instance, it is sufficient to join the solution, of value H^*, of the reduced instance with the stack of the items in R and to set the optimal strip height equal to $H^* = H^* + H_R$.

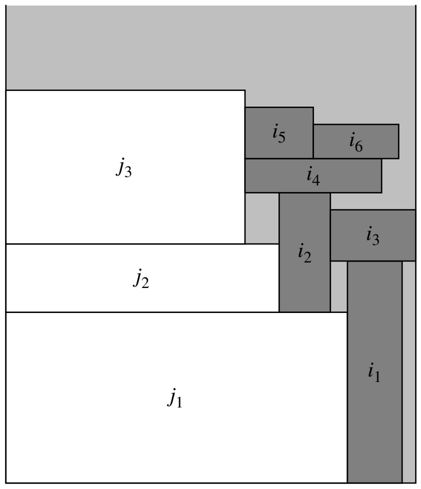
This reduction can be further improved by generalizing the idea proposed by Alvarez-Valdes et al. (2009). They consider the items of $R' = \{j \in P\colon w_j > W/2\}$ in nonincreasing order of width and observe that if all items belonging to set $P_j = \{k \in P\colon w_j + w_k \leqslant W\}$, $j \in R'$, can be allocated into a rectangle of size $(W - w_j, h_j)$, then all items in the set $P_j \cup \{j\}$ can be fixed in the solution in the bottom of the strip (see Figure 1(a)). This idea can be generalized by extending to the 2SP the approach proposed by Clautiaux et al. (2007c) for the two-dimensional orthogonal packing problem. We consider a subset $\bar{R} \subseteq R'$ instead of a single item $j \in R'$; and we allocate into a stack the items of $\bar{R}$ for nonincreasing widths and in normal pattern positions (i.e., left justified). Then we check if all the items of set $P_{\bar{R}} = \{k \in P\colon \exists j \in \bar{R} \text{ s.t. } w_j + w_k \leqslant W\}$ can be allocated in the remaining area in the right side of the stack of height $\sum_{j \in \bar{R}} h_j$ (see Figure 1(b)). Because the items of $P_{\bar{R}}$ are the only items that can be packed side by side with items of $\bar{R}$, if a feasible packing exists, we can permanently allocate it at the bottom of the strip without changing the optimal solution value. To check whether the items in $P_{\bar{R}}$ can be allocated in the remaining area in the right side of the stack, we use the heuristic algorithms presented in §4. Obviously,

Figure 1. Two examples where some items are fixed in the solution.

(a) Reduction involving a single item j ∈ R': an example where $P_j = \{i_1, i_2, i_3\}$.



(b) Reduction involving a subset $\bar{R} \subseteq R'$: an example where $\bar{R} = \{j_1, j_2, j_3\}$ and $P_{\bar{R}} = \{i_1, i_2, i_3, i_4, i_5, i_6\}$.



before running the heuristic algorithms, we use some simple lower bounds, described in §3, to check in advance whether a feasible packing cannot exist. Figure 1(b) reports an example where no items can be fixed in the solution if only one item of R' is considered at a time.

This new reduction procedure considers subsets $\bar{R}$ not considered in Clautiaux et al. (2007c), which can be useful in our heuristic setting, and contains the procedure of Alvarez-Valdes et al. (2009) as a special case by considering only the subsets $\bar{R}$ of cardinality one, i.e., $|\bar{R}| = 1$. This procedure usually works well when there are many items in R' with respect to the "smallest" items in $P \backslash R'$.

2.2.2. Modifying the Strip Width. If no combination of items exactly filling the strip width W exists, then there is a useless space that can be removed from the strip without modifying the optimal solution value. The strip width can be updated by solving the following subset sum problem SSP_W:

$$W_P = \max\left\{\sum_{i\in P} w_i\xi_i \colon \sum_{i\in P} w_i\xi_i \leqslant W, \quad \xi_i \in \{0,1\}, i\in P\right\}. \quad (3)$$

SSP_W is solved using the algorithm proposed in Pisinger (1995), and the new strip width is W_P, i.e., $W = W_P$. This reduction has also been proposed by Alvarez-Valdes et al. (2009).

2.2.3. Modifying the Item Widths. It is straightforward to observe that a packing layout containing item j remains feasible if the width of j is increased as $w_j = w_j + (W - W_j^*)$, where W_j^* is the optimal solution cost of the following subset sum problem $SSP_w(j)$:

$$W_j^* = \max\left\{w = \sum_{k\in P\backslash\{j\}} w_k\xi_k + w_j \colon w \leqslant W, \right.$$
$$\left. \xi_k \in \{0,1\},\ k \in P\backslash\{j\}\right\}. \quad (4)$$

Because we would like to maximize the number of updated widths, we heuristically consider the items ordered by nonincreasing width, i.e., $w_1 \geqslant w_2 \geqslant \cdots \geqslant w_n$. More details on this reduction procedure can be found in Boschetti et al. (2002) and Boschetti and Mingozzi (2003a).

3. Lower Bounds

In this section, we present some lower bounds for the 2SP. First, we propose some extensions of well-known lower bounds, then we introduce new approaches derived from other similar packing problems. We complete this section proposing a postprocessing procedure to further improve the lower bounds.

3.1. Simple Lower Bounds

Two simple lower bounds are proposed by Martello et al. (2003). The first one, called *continuous lower bound*, is computed in linear time by considering the total area of the items to allocate into the strip of width W as follows: $L_c = \lceil \sum_{i\in P} w_i h_i / W \rceil$. The second trivial lower bound is the height of the tallest item, i.e., $L_h = \max_{i\in P}\{h_i\}$.

3.2. Lower Bounds Based on Dual Feasible Functions

The concept of *dual feasible function* was originally introduced by Johnson (1973) and was used for computing lower bounds for bin-packing problems for the first time by Lueker (1983).

In recent years, dual feasible functions have been used to compute lower bounds for different packing problems by Fekete and Schepers (1997, 1998, 2000), Fekete et al. (2007), Boschetti and Mingozzi (2003a, b), Boschetti

(2004), Carlier et al. (2007), Clautiaux et al. (2008, 2007b, c, d), and Alvarez-Valdes et al. (2009). Furthermore, dual feasible functions have been used to generate valid inequalities in Alves (2005) and Baldacci and Boschetti (2007). Recently, Clautiaux et al. (2007a) surveyed and analyzed several dual feasible functions and reported a computational comparison, too.

The basic idea is to transform the widths of the items and of the strip by a function f, called *dual feasible function*, in such a way that if a set of items of the original instance can be packed side by side, then they can still be packed even with the new widths. A dual feasible function can be defined as follows.

Definition 1. A function f: $[0, W] \to [0, W']$ is called dual feasible if, for any finite set S of nonnegative real numbers, we have $\sum_{w\in S} w \leqslant W \Rightarrow \sum_{w\in S} f(w) \leqslant f(W) = W'$.

In general, the values of a dual feasible function do not depend on the instance data. However, there exist dual feasible functions, called *data-dependent dual feasible functions* by Carlier et al. (2007), whose values cannot be computed in advance but depend on the size of the items of the instance considered. A data-dependent dual feasible function can be defined as follows.

Definition 2. A function f: $[0, W] \to [0, W']$ is called data-dependent dual feasible if for any subset of items $S \subseteq P$, we have: $\sum_{i\in S} w_i \leqslant W \Rightarrow \sum_{i\in S} f(w_i) \leqslant f(W) = W'$.

Given a dual feasible function f, a valid lower bound for the 2SP can be obtained by solving the continuous lower bound as follows:

$$L_f = \left\lceil \frac{\sum_{i\in P} f(w_i)h_i}{f(W)} \right\rceil. \tag{5}$$

The dual feasible functions used in this paper are described in the following.

Dual Feasible Function 1. Let $\alpha \in \Omega_1 \subseteq \mathbb{N}$. A dual feasible function defined by Fekete and Schepers (1997, 1998, 2000) is:

$$\begin{cases} f^1_\alpha(w) = w, & \text{if } (\alpha+1)\frac{w}{W} \in \mathbb{Z} \\ \left\lfloor (\alpha+1)\dfrac{w}{W} \right\rfloor \dfrac{W}{\alpha}, & \text{otherwise.} \end{cases} \tag{6}$$

Dual Feasible Function 2. Let $\alpha \in \Omega_2 = [1, W/2]$, $\Omega_2 \subseteq \mathbb{N}$. A dual feasible function defined by Fekete and Schepers (1997, 1998, 2000), but used indirectly for the first time by Martello and Toth (1990) and proposed for the 2SP in Martello et al. (2003), is:

$$f^2_\alpha(w) = \begin{cases} W, & \text{if } w > W-\alpha \\ w, & \text{if } \alpha \leqslant w \leqslant W-\alpha \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Dual Feasible Function 3. Let $\alpha \in \Omega_3 = [1, W/2]$, $\Omega_3 \subseteq \mathbb{N}$. A dual feasible function proposed by Carlier et al. (2007) and derived from the one presented in Boschetti (2004) for the three-dimensional bin-packing problem is:

$$f^3_\alpha(w) = \begin{cases} 2\left(\left\lfloor \dfrac{W}{\alpha} \right\rfloor - \left\lfloor \dfrac{W-w}{\alpha} \right\rfloor\right), & \text{if } w > \frac{W}{2} \\ \left\lfloor \dfrac{W}{\alpha} \right\rfloor, & \text{if } w = \frac{W}{2} \\ 2\left\lfloor \dfrac{w}{\alpha} \right\rfloor, & \text{if } w < \frac{W}{2}. \end{cases} \tag{8}$$

Dual Feasible Function 4. Let $\alpha \in \Omega_4 = [1, W/2]$, $\Omega_4 \subseteq \mathbb{N}$, and $I_\alpha = \{i \in P\colon\ w_i \geqslant \alpha\}$. A data-dependent dual feasible function proposed for the two-dimensional bin-packing problem in Boschetti and Mingozzi (2003a) is:

$$f^4_\alpha(w) = \begin{cases} M(W, I_\alpha) - M(W-w, I_\alpha), & \text{if } \dfrac{W}{2} < w \leqslant W \\ 1, & \text{if } \alpha \leqslant w \leqslant \dfrac{W}{2} \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where $M(w, I)$ is the maximum number of items of the set I that can be packed side by side in a bin of width w. $M(w, I)$ can be computed by solving the following knapsack problem:

$$M(w, I) = \max\left\{ \sum_{k\in I} \xi_k\colon\ \sum_{k\in I} w_k \xi_k \leqslant w,\quad \xi_k \in \{0,1\},\ k \in I \right\}. \tag{10}$$

$M(w, I)$ can be solved in $O(n)$ if items of the set I are sorted by nondecreasing width.

Other dual feasible functions have been proposed in Clautiaux et al. (2007a) and Alvarez-Valdes et al. (2009) but are not used in this paper.

Fekete and Schepers (1997, 1998, 2000) show that dual feasible functions can also be obtained by composing two or more dual feasible functions. Hence, a dual feasible function $f^{ij}_{\alpha\beta}$ can be obtained as follows: $f^{i,j}_{\alpha,\beta}(w) = f^i_\alpha(f^j_\beta(w))$. In our computational results, we compose the four dual feasible functions described above only with f^2_α. Therefore, the overall lower bound is given by:

$$L^{BM}_{dff} = \max_{k=1,2,3,4} \left\{ \max_{\substack{\alpha\in\Omega_k \\ \beta\in\Omega_2}} \left\{ \left\lceil \frac{\sum_{i\in P} f^{k,2}_{\alpha,\beta}(w_i)h_i}{f^{k,2}_{\alpha,\beta}(W)} \right\rceil \right\} \right\}. \tag{11}$$

Notice that the dual feasible function f^2_α for $\alpha = 1$ is an identity function, i.e., $f^2_1(w) = w$. Furthermore, in our implementation we use $\Omega_1 = \{1, \ldots, W\}$ and $\Omega_k = \{w_i\colon\ w_i \leqslant W/2, i \in P\} \cup \{W - w_i\colon\ w_i > W/2, i \in P\}$, for $k = 2, 3, 4$.

3.3. Lower Bounds Based on Item Heights

In §3.1 we introduced a simple lower bound proposed by Martello et al. (2003) that is given by the height of the tallest item. This lower bound can be improved, but not dominated, by considering the item heights in a new, completely different way.

First, we have to compute a lower bound on the number of *layers* required to pack all items. In layer 1 there are items placed on the strip base (i.e., with a y-position equal to zero), whereas in layer k there are items that in every x-positions are placed on at most $k-1$ items, and in at least one x-position are placed on exactly $k-1$ items. To compute the lower bound on the number of layers, Lay, we can use the continuous lower bound for the 1BP:

$$Lay = \left\lceil \frac{\sum_{i\in P} w_i}{W} \right\rceil. \tag{12}$$

Because each layer has to be at least as tall as the minimum height of the items contained in the layer, sorting items by nondecreasing height, a lower bound for the 2SP is equal to the sum of the first Lay minimum heights:

$$L_h^1 = \sum_{k=1}^{Lay} h_k. \tag{13}$$

Lower bound L_h^1 can be further improved by the following considerations. If the first $Lay-1$ layers are completely filled for the entire width W, then the *top layer*, i.e., the layer Lay, contains items covering a width equal to $w_{Top} = (\sum_{i\in P} w_i)\%W$, where % is the modulo operator that is the remainder of division. If the first $Lay-1$ layers are not completely filled, then the remaining layers contain items covering a width greater than w_{Top}.

We define the set $I_{Full} \subseteq P$ as the set of $Lay-1$ items of minimum height, i.e., $|I_{Full}| = Lay-1$ and $h_i \leqslant h_j$, for each $i \in I_{Full}$ and $j \in P\backslash I_{Full}$. Furthermore, we define the set I_{Top} of k items of minimum height of $P\backslash I_{Full}$ such that $\sum_{i\in I_{Top}} w_i \geqslant w_{Top}$, but $\sum_{i\in I_{Top}\backslash\{j\}} w_i < w_{Top}$, where j is the tallest item of I_{Top}. A valid lower bound for the 2SP is computed as follows:

$$L_h^2 = \sum_{k\in I_{Full}} h_k + \max_{k\in I_{top}}\{h_k\}. \tag{14}$$

Similarly, we define the set $I'_{Top} \subseteq P$ of k items of minimum height such that $\sum_{i\in I'_{Top}} \geqslant w_{Top}$, but $\sum_{i\in I'_{Top}\backslash\{j\}} < w_{Top}$, where j is the tallest item of I'_{Top}. Then, we define the set $I'_{Full} \subseteq P\backslash I'_{Top}$ as the set of the remaining $Lay-1$ items of minimum height, i.e., $|I'_{Full}| = Lay-1$ and $h_i \leqslant h_j$, for each $i \in I'_{Full}$ and $j \in P\backslash(I'_{Full} \cup I'_{Top})$. Another valid lower bound for the 2SP is computed as follows:

$$L_h^3 = \sum_{k\in I'_{Full}} h_k + \max_{k\in I'_{top}}\{h_k\}. \tag{15}$$

There are no dominance relations among lower bounds L_h, L_h^2 and L_h^3, whereas L_h^2 and L_h^3 dominate L_h^1. Hence, the overall lower bounds for the 2SP are:

$$L_h^{BM} = \max\{L_h, L_h^2, L_h^3\}. \tag{16}$$

3.4. Lower Bounds Based on Mathematical Formulations

In this section we propose two completely new lower bounds based on the mathematical formulations used by Baldacci and Boschetti (2007) and Boschetti et al. (2002) for the *two-dimensional nonguillotine cutting problem* (NGCP).

3.4.1. Lower Bound L_{F1}^{BM}. An electronic companion to this paper is available as part of the online version that can be found at http://or.journal.informs.org/. It reports the details of how to derive the relaxation proposed in this section. Here we give only the details of the relaxed problem that are sufficient to understand the basic idea.

Recall that X_i is the subset of all x-positions where item i can be placed following the principle of normal patterns (see §2.1). Furthermore, we define $X' = \bigcup_{i\in P} X_i$.

Let t_{ix} be (0–1) binary variables equal to 1 if and only if the bottom-left corner of the item i is placed in position (x, y). The relaxed problem *F1* is the following:

$$(F1) \quad z_{F1} = \min \quad h \tag{17}$$

$$\text{s.t.} \quad \sum_{x\in X_i} t_{ix} = 1, \quad i \in P \tag{18}$$

$$\sum_{i\in P} h_i \sum_{\bar{x}\in\alpha(i,x)} t_{i\bar{x}} \leqslant h, \quad x \in X' \tag{19}$$

$$t_{ix} \in \{0,1\}, \quad x \in X_i,\ i \in P \tag{20}$$

$$h \geqslant 0, \tag{21}$$

where $\alpha(i,x) = \{\bar{x} \in X_i\colon \max[0, x-w_i+1] \leqslant \bar{x} \leqslant x\}$. Constraints (18) impose that every item is packed exactly once into the strip. Constraints (19) impose that the height of the strip h is sufficient to contain every item covering the x-positions specified. The objective function (17) minimizes the height h of the strip. Notice that problem *F1* considers only normal pattern positions.

The optimal solution value of problem *F1* is a valid lower bound, denoted by L_{F1}^{BM}, and it can be computed by an MIP solver such as CPLEX. In order to improve the performance of the MIP solver avoiding equivalent solutions, we can add to *F1* the following constraints:

$$\sum_{j\in P(i,x)} t_{j,x-w_j} \geqslant t_{ix}, \quad x \in X_i,\ i \in P, \tag{22}$$

where $P(i,x) = \{j \in P\backslash\{i\}\colon x-w_j \in X_j\}$. Constraints (22) ensure that only solutions satisfying the normal pattern principle are considered. In our computational results, we have added constraints (22) only if they are fewer than $\max\{500, 1/2nW\}$.

If the optimal solution of problem *F1* is not reached within a given time limit, we set the lower bound L_{F1}^{BM} equal to the minimum lower bound associated with the remaining unexplored nodes gotten from the MIP solver. In our computational results, the time limit for the MIP solver is 600 seconds.

3.4.2. Lower Bound L_{F2}^{BM}. The relaxed problem proposed in this section is based on the observation that any feasible 2SP solution can be represented by a sequence $(S_1, \ldots, S_h)$, where each $S_{y'} \in \mathbb{Y}$ is the subset of items covering the y-position y'.

Definition 3. A subset of items $S \subseteq P$ is a *y-feasible subset* if $\sum_{i\in S} w_i \leqslant W$.

Let $\mathbb{Y} = \{1, \ldots, m\}$ be the index set of all y-feasible subsets, and let $\mathbb{Y}_i \subseteq \mathbb{Y}$ be the index set of all y-feasible subsets containing item $i \in P$. However, it is not necessary to consider all y-feasible subsets, as shown by Boschetti et al. (2002) for the NGCP, but only the *undominated* y-feasible subsets, which are defined as follows.

Definition 4. A y-feasible subset S_j, $j \in \mathbb{Y}$, is undominated if $S_j \nsubseteq S_k$, $\forall k \in \mathbb{Y}\backslash\{j\}$.

We denote with $\overline{\mathbb{Y}}$ the index set of undominated y-feasible subsets, i.e., $\overline{\mathbb{Y}} = \{j \in \mathbb{Y}: S_j \nsubseteq S_k,\ \forall k \in \mathbb{Y}\backslash\{j\}\}$. Moreover, $\overline{\mathbb{Y}}_i = \overline{\mathbb{Y}} \cap \mathbb{Y}_i$. Let v_j be the number of times that the undominated y-feasible subset $j \in \overline{\mathbb{Y}}$ is in the optimal solution. A valid lower bound for the 2SP can be computed by solving the following problem:

$$(F2) \quad z_{F2} = \min \sum_{j\in\overline{\mathbb{Y}}} v_j \tag{23}$$

$$\text{s.t.} \quad \sum_{j\in\overline{\mathbb{Y}}_i} v_j \geqslant h_i, \quad i \in P \tag{24}$$

$$v_j \geqslant 0, \quad j \in \overline{\mathbb{Y}}. \tag{25}$$

Constraints (24) ensure that in the optimal solution every item i is contained in at least h_i undominated y-feasible subsets. The objective function (23) minimizes the number of undominated y-feasible subsets used.

Problem $F2$ can be solved by an LP solver, and its optimal solution value is the valid lower bound denoted by L_{F2}^{BM}; however, if the number of y-feasible subsets (i.e., $|\overline{\mathbb{Y}}|$) is huge, we can generate only a subset of $\overline{\mathbb{Y}}$ by a column generation procedure. Such a column generation procedure needs to solve a knapsack problem at each iteration. Instances where $|\overline{\mathbb{Y}}|$ is huge could require many iterations, and we might not be able to generate the entire set $\overline{\mathbb{Y}}$. In these cases, we set $L_{F2}^{BM} = 0$. In our computational results, $\overline{\mathbb{Y}}$ cannot exceed one million subsets, and the time limit for the LP solver is set to 600 seconds.

3.5. A Postprocessing to Improve the Lower Bound

In this section we propose a simple procedure to improve the current best lower bound LB.

The first improvement of LB can be obtained by noticing that the required strip height must be a normal pattern position computed by considering all items P. Therefore, we can update the current lower bound as follows:

$$LB = \min\left\{ y = \sum_{k\in P} h_k \xi_k \colon LB \leqslant y \leqslant UB,\right.$$
$$\left. \xi_k \in \{0,1\},\ k \in P \right\}, \tag{26}$$

where UB is the current best upper bound that is certainly a normal pattern position.

A more expensive procedure to improve the lower bound LB fixes the strip height at a value $\hat{h} \in [LB, UB[$ and computes a lower bound on the strip width required to allocate all items P. To compute such a lower bound, we rotate the instance exchanging widths with heights, and we apply lower bounds L_h^{BM}, L_{dff}^{BM}, and L_{F2}^{BM}, described in the previous sections, to the resulting problem. We do not use L_{F1}^{BM} because it is expensive, and for L_{F2}^{BM} we set a time limit of only 60 seconds. We generate at most half a million undominated y-feasible subsets, i.e., $|\overline{\mathbb{Y}}| \leqslant 500{,}000$. If the computed lower bound on the strip width is less than or equal to W, it means that all items can be packed in a strip of size $(W, \hat{h})$. Otherwise, height $\hat{h}$ is not sufficient to contain all items of the problem instance, and LB can be increased to the smallest normal pattern height greater than LB by setting $LB = \hat{h} + 1$ and applying expression (26).

This search procedure starts setting $\hat{h} = UB - 1$, and at each iteration $\hat{h}$ is decreased by one unit. The procedure terminates only if items P cannot be packed in a strip of size $(W, \hat{h})$; hence, in the worst case, this procedure is iterated $UB - LB$ times.

4. Heuristics

In this section, we propose three new heuristic algorithms and a simple postprocessing procedure to improve the current best upper bound. The first one is completely new and is based on original ideas and the principle of normal patterns. The second one is a modified version of algorithms already proposed in the literature. The third heuristic algorithm is completely new and makes use of the lower bound L_{F1}^{BM}, described in §3.4.1.

The first two heuristic procedures add at each iteration an item to the emerging solution E. At the beginning, the emerging solution is set empty, i.e., $E = \varnothing$. When at each iteration an item i is placed into the strip in position (x_i, y_i), the procedures add to the emerging solution E the triplet (i, x_i, y_i), i.e., $E = E \cup \{(i, x_i, y_i)\}$. The third procedure fixes the x-positions computed for all items by lower bound L_{F1}^{BM} and tries to find the y-positions by a truncated branch and bound.

4.1. Normal Pattern Shifting (NPS)

The *normal pattern shifting* (NPS) heuristic places items into the strip in turn, following a given order. In our computational results we have ordered items using different criteria (e.g., nonincreasing value of area, height, etc.), and we have also updated the order of items as shown in §4.1.1.

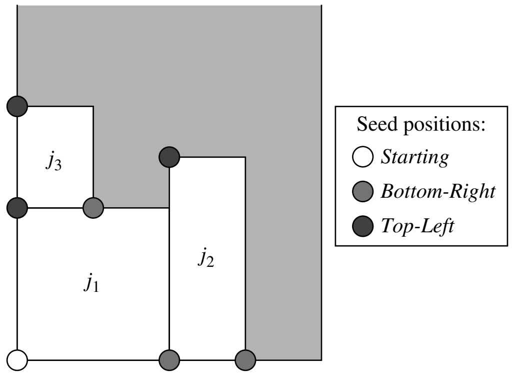
To place each item into the strip, the procedure NPS starts from the *seed positions*, which are defined by the items already allocated into the strip and are of three different types: *starting*, *top-left*, and *bottom-right*. The starting seed position is only one and corresponds to position $(0, 0)$. Top-left and bottom-right seed positions correspond to the positions of top-left corners and bottom-right corners of the items already allocated into the strip, respectively (see Figure 2).

At the beginning the strip is empty (i.e., $E = \varnothing$), and only the starting seed position $(0, 0)$ is available. When, at each iteration, a new item i is placed into the strip in position (x_i, y_i), its top-left and bottom-right corners are added to the list of the available seed positions, called *ListSPos*. Notice that at each iteration the seed positions to consider are $O(n)$.

The NPS heuristic finds at each iteration the normal pattern positions for the current item i starting from the seed positions contained in *ListSPos*. If the search starts from a bottom-right seed position, the procedure tries to move down the item i as much as possible; if it starts from a top-left seed position the procedure tries to move the item i left as much as possible (see Figure 3). No other movements are required because if the position reached is feasible but is not a normal pattern position, the available normal pattern positions can always be reached from other seed positions (see Figure 3(b)). If a feasible normal pattern position (x_i, y_i) is found for the item i (i.e., the item is inside the strip, does not overlap other items already allocated, and its position satisfies the normal pattern principle), the procedure evaluates its quality $q(i, x_i, y_i, E)$ by computing the *density* of the corresponding emerging solution as $q(i, x_i, y_i, E) = (\sum_{(j, x_j, y_j)\in E} w_j h_j)/W\bar{h}$, where $\bar{h} = \max\{y_j + h_j\colon\ (j, x_j, y_j) \in E\}$. The NPS heuristic chooses the lowest and leftmost position that maximizes the density of the emerging solution.

Procedure NPS can be modified to consider at each iteration *all* items not already allocated and to choose the item and the position that maximize the density of the emerging solution. If more items and/or positions maximize the density, the procedure chooses the item with the lowest and leftmost position. However, our computational results, not reported in this paper, show that the upper bounds are not improved significantly in spite of a considerable increase of the computing time.

Figure 2. Example of seed positions for an emerging solution.



4.1.1. Normal Pattern Shifting with Pricing. Procedure NPS can be repeated with different item orders generated using the framework proposed by Boschetti and Mingozzi (2003b) for the two-dimensional bin-packing problem (2BP).

At the beginning, for every item $i \in P$ we set an initial *price* p_i, which can be equal to its area, height, etc., or to a fixed value (e.g., $p_i = 100$). We execute NPS with items sorted by nonincreasing price values. Let $\bar{H}$ be the height of the strip used. We use the position (x_i, y_i) of each item $i \in P$ in the solution to increase or decrease its price. If the item i is placed in the *bottom half* of the strip (i.e., $y_i \leqslant \bar{H}/2$), its price is decreased: $p_i = \alpha p_i$, where $\alpha < 1$. If the item i is placed in the *top half* of the strip (i.e., $\bar{H}/2 < y_i \leqslant \bar{H}$), its price is increased: $p_i = \beta p_i$, where $\beta > 1$. Values α and β are randomly generated as follows: $\alpha = 1 - r$ and $\beta = 1 + r$, where r is a random real number in the range $[0, 1)$.

Procedure NPS with pricing requires several iterations to give some improvements; thus, it is time consuming. Therefore, in our computational results we have used the pricing only for small instances, i.e., $n \leqslant 20$.

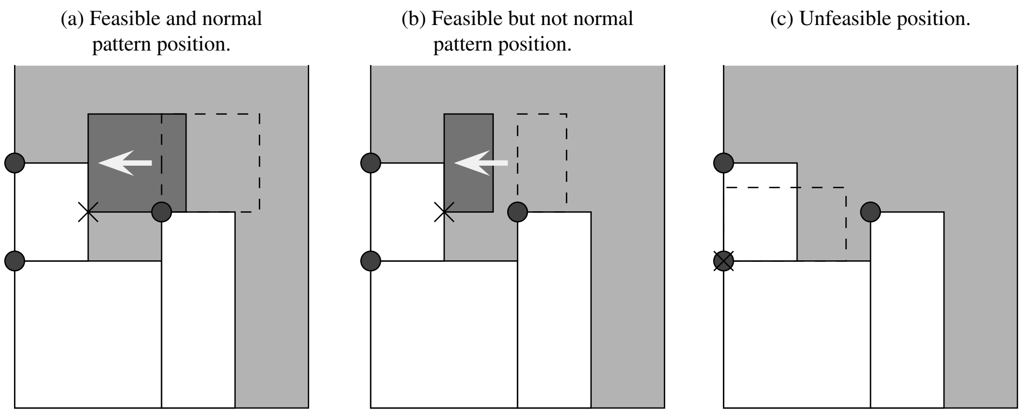
4.2. Priority Best-Fit (PBF)

The *Priority Best-Fit* (PBF) heuristic procedure is based on the BF algorithm of Burke et al. (2004) and makes use of some improvements proposed by Alvarez-Valdes et al. (2008), but it does not make use of any metaheuristic framework because a fast procedure is needed. We add some new *placement criteria*, described in §4.2.1; we modify the *look-ahead* procedure, described in §4.2.2; and we propose a completely new *warm-start* procedure, described in §4.2.3. These new features permit some improved results with respect to Burke et al. (2004) and Alvarez-Valdes et al. (2008) that make use of metaheuristic frameworks, as shown in the computational results reported in §6.

Given an emerging solution E, the PBF heuristic identifies the available positions using a *skyline* data structure (see Burke et al. 2004 for more details). In Figure 4 is shown an example where in the emerging solution E seven items have been already allocated into the strip. Figure 4(a) shows the skyline generated by E, where the free dark areas below the skyline cannot be used anymore. The available positions are the gaps of the skyline, as shown in Figure 4(b).

Procedure PBF selects the lowest available gap, also called *niche*, and tries to place there the item not already allocated that best fits the available space. If there are more gaps at the lowest height, we select the leftmost. The gaps at the left and at the right of the niche are called *left gap*

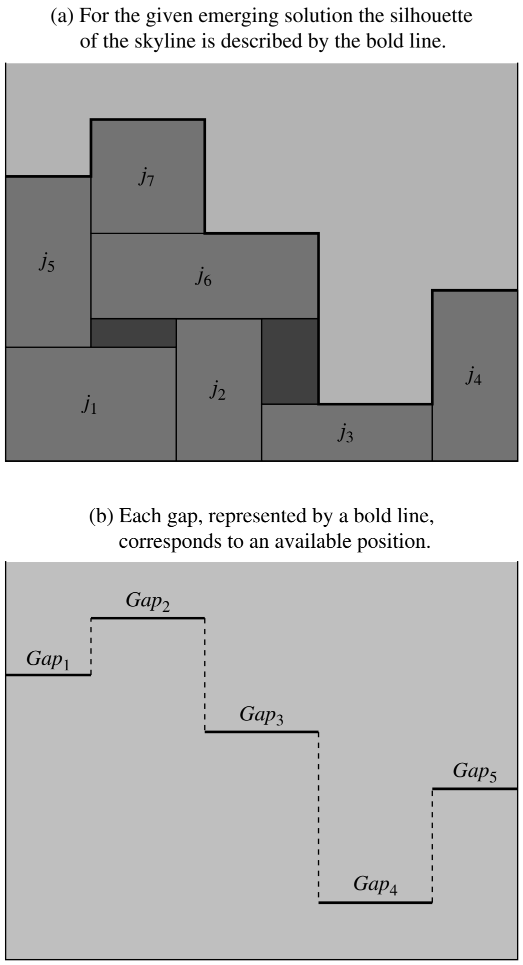
Figure 3. Examples of positions (denoted by a cross) generated from top-left seed positions (denoted by circle).


(a) Feasible and normal pattern position.
(b) Feasible but not normal pattern position.
(c) Unfeasible position.

and *right gap*, respectively; both gaps are called *neighbor gaps*. In the example of Figure 4 the lowest gap is Gap_4, and its neighbors are Gap_3 and Gap_5.

4.2.1. Placement Criteria. To identify the best-fitting item, procedure PBF applies different placement criteria

Figure 4. An example of *skyline* data structure: the lowest gap (*niche*) is Gap_4.


(a) For the given emerging solution the silhouette of the skyline is described by the bold line.
j_7
j_5
j_6
j_4
j_2
j_1
j_3
(b) Each gap, represented by a bold line, corresponds to an available position.
Gap_2
Gap_1
Gap_3
Gap_5
Gap_4

that can be classified in two categories: *hard selective* and *weakly selective*.

The hard selective criteria used are the following: (h.1) the item has the same width of the niche; (h.2) the item has the same height of the niche, i.e., its top edge reaches one of the neighbor gaps; (h.3) the item reaches the same height of the left gap or, if the niche touches the left edge of the strip, it is the tallest; (h.4) the item fills the whole niche width together with another of the remaining items, even of different height.

The weakly selective criteria are the following: (w.1) the item has the largest height; (w.2) the item has the largest width; (w.3) the item has the largest area; (w.4) the item maximizes the niche width filled together with another of the remaining items of the same height; (w.5) the item maximizes the niche width filled together with another of the remaining items, even of different height; (w.6) the item maximizes the density of the emerging solution.

If none of the remaining items can be allocated because the width of the lowest available gap is too small, the space in the niche is considered waste. Therefore, the lowest gap is deleted, and its width is added to the lowest neighbor gap.

Procedure PBF can be used with different ordered subsets of the criteria previously listed. Not all combinations of criteria are necessary, because a lot of them are meaningless. Table 1 reports the combinations of criteria used in procedure PBF, specifying the order in which they are applied. Each column corresponds to a different combination and indicates, for each criterion included, the priority assigned (priority 1 means it is the first criterion applied, etc.).

When a criterion selects an item of width smaller than the niche width, we have to choose between the left or right alignment of the item inside the niche. We have used the same procedure proposed by Alvarez-Valdes et al. (2008). Let S_l and S_r be the heights of the left and of the right gaps, respectively, and let h^* be the y-position of the niche and $h' = h^* + h_i$. The item is aligned on the left if the niche touches the left edge of the strip or $S_l = h'$; otherwise, if $S_r = h'$ the item is aligned on the right. If none of the

Table 1. Combinations of criteria used at each execution of the PBF heuristic. The number reported in each column specifies the order in which they are applied.

	Combinations																			
Criteria	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}
h.1	2	1			1	1		1	1	1				2			1			2
h.2					2	2							1		1				1	
h.3	1	2	1	1			1				2			1		1				
h.4																2	2	2	2	
w.1	3	3		3				2				2							3	
w.2					3					3	3				3					
w.3						3	2		2				2			3	3	3		3
w.4			2	2						2	1	1			2			1		
w.5																				1
w.6			3											3						

previous conditions occurs and $S_l = S_r$, the item is placed in the position nearest the strip edge; otherwise, the item is aligned to the tallest neighbor gap.

4.2.2. Look-Ahead. Unfortunately, tallest items can be placed into the strip only at the last iterations because the placement criteria try to build a smooth solution. Thus, to avoid bad packing we apply the following *look-ahead* procedure, similar to the one proposed by Alvarez-Valdes et al. (2008). Let i be the item selected by the placement criteria, and let j be the tallest of remaining items for which the selected niche is feasible. Let A_E be an estimation of the empty area after placing the tallest item j, and let A_M be the sum of the areas of remaining items. Given the height $\bar{H}$ of the current packing and the y-position $\bar{y}$ of the niche, A_E is computed by subtracting from $W(\bar{H} - \bar{y})$ the area of the items in the emerging solution E covering the area of the strip from $\bar{y}$ to $\bar{H}$, i.e., $A_E = W(\bar{H} - \bar{y}) - \sum_{(k, x_k, y_k) \in E} w_k \Delta h'_k(\bar{y})$, where $\Delta h'_k(\bar{y}) = \max\{\bar{y}, y_k + h_k\} - \max\{\bar{y}, y_k\}$. If $A_E > A_M$, we place j into the strip; otherwise, we perform a further look-ahead estimation to verify if packing the item i into the niche implies that at the next iteration $A_E > A_M$. In this case we place the item j, otherwise, we place i.

4.2.3. PBF with a Warm Start. The items allocated into the strip at the beginning of procedure PBF are very important, but when the strip is empty the different placement criteria may be too weak. Therefore, we initialize the emerging solution by placing in the bottom of the empty strip a y-feasible subset that maximizes the width covered. Then, we apply the procedure PBF as described in the previous section.

We repeat procedure PBF for k different starting y-feasible subsets that maximize the width covered. In particular, in our computational results we have chosen the k y-feasible subsets of width equal to W, and we limited k to $\lceil MS/n^2 \rceil$, where $MS = 2 \times 10^6$ in order to have $k = 5{,}000$, for instances with $n = 20$, and $k = 200$, for $n = 100$.

4.3. Heuristic Based on Lower Bound L_{F1}^{BM} (HF1)

The third procedure HF1 fixes the x-positions computed for all items by lower bound L_{F1}^{BM} and tries to find the y-positions by a truncated branch and bound.

Lower bound L_{F1}^{BM} is computed by solving problem *F1*. Hence, if the MIP solver is not able to solve *F1* within the given time limit, in order to improve the heuristic solution of *F1* we apply the *polishing* procedure available in CPLEX, setting a time limit of 60 seconds.

Given the best solution found solving *F1*, hopefully the optimal one, we fix the x-positions of the items in accordance, and we apply a truncated version of the branch and bound described in §5, where only the y-positions of each item must be fixed. It is always possible to find a feasible solution if some requirements are relaxed, such as normal pattern principle, or others. There is no proof that a feasible 2SP solution of height equal to L_{F1}^{BM} exists; otherwise L_{F1}^{BM} should be the optimal solution.

The truncated branch and bound, compared to the exact method described in §5, makes use only of the continuous lower bound and, when it generates the child nodes for each item, it considers only the positions having the x-coordinate fixed by lower bound L_{F1}^{BM}. Moreover, in our computational results, we set a small time limit of 60 seconds.

4.4. A Postprocessing to Improve the Upper Bound

The current best upper bound UB can be further improved by applying a postprocessing described in this section.

Let LB be the best lower bound computed so far, and let $\hat{h} \in [LB, UB[$. We rotate the problem instance by 90 degrees by exchanging for each item $i \in P$ the width w_i and the height h_i and by setting the strip width equal to $\hat{h}$. If there exists a heuristic solution where every item can be packed in a strip of height less than or equal to W, then UB can be updated by setting $UB = \hat{h}$.

Because we use heuristic algorithms, if we fail in proving that $\hat{h}$ is a valid upper bound, it does not mean that a valid upper bound $h' < \hat{h}$ does not exist. For this reason we do not use a bisection method, but we propose a sequential search in the range $[LB, UB[$. The search procedure starts by setting $\hat{h} = LB$, and at each iteration, $\hat{h}$ is increased by one unit. The procedure terminates as soon as items P can be packed in a strip of size $(\hat{h}, W)$; thus, in the worst case, this procedure is iterated $UB - LB - 1$ times.

5. Exact Method

The new exact algorithm, called BB, is a branch and bound that makes use of reduction procedures, described in §2.2, and of the lower and upper bounds, described in §§3 and 4, respectively.

Let LB and UB be the current best lower and upper bounds, respectively. We initialize $LB = 0$ and $UB = \infty$. Every time a new lower bound or a new upper bound is computed, LB and UB are updated accordingly. As soon as $LB = UB$, an optimal solution is found and the algorithm stops.

Algorithm BB, at the root node after reductions, applies the lower bounds L_{dff}^{BM} and L_{h}^{BM} and then the heuristic procedures NPS and PBF. If the optimal solution is not found (i.e., $LB < UB$), algorithm BB first computes L_{F1}^{BM} and then L_{F2}^{BM}. If within the given time limit the MIP solver has found at least a feasible solution for *F1*, BB applies the heuristic procedure HF1. Algorithm BB ends the root node with the two postprocessing to improve the current best lower and upper bounds. If at the root node the optimal solution is not found, algorithm BB starts branching.

At the beginning, no item has been placed and the strip is empty. Therefore, according to the normal pattern principle, items can be placed only in position $(0, 0)$. Thus, at the root node BB generates exactly n child nodes, one for each item placed in $(0, 0)$.

At the other nodes of the search tree, the remaining items can be placed only in particular positions, called *corners*. A corner is computed using the skyline data structure described in §4.2. Remember that the skyline is a set of consecutive segments positioned at different y-positions, and each segment represents a gap. Therefore, each gap k is represented by the triplet (x_k^l, x_k^r, y_k), where x_k^l and x_k^r are the x-positions of the left and right ends of the gap, respectively, and y_k is its y-position. The left end x_k^l of the gap k generates a corner (x_k^l, y_k) if it is next to the left-hand edge of the strip (i.e., $x_k^l = 0$) or if the previous gap has a greater y-position. The algorithm selects the corner that minimizes the y-position, and among corners with the same minimum y-position, the corner that minimizes the x-position. Algorithm BB generates at most $n + 1$ child nodes, one for each remaining item placed in the selected corner, and one for the case when the selected corner is not used. However, algorithm BB generates a child node only if it is feasible to place the item into the corner and the normal pattern principle is satisfied. Furthermore, if more items have the same size, only one child node is generated.

For each generated child node, BB updates the skyline data structure and computes a lower bound on the remaining problem, using L_{dff}^{BM}, L_{h}^{BM}, and L_{F2}^{BM}, provided that $\overline{\mathbb{Y}}$ does not exceed 200,000 subsets and with a time limit of one second. The postprocessing is applied only if the current lower bound is less than W. If the overall lower bound is greater than or equal to the current best upper bound, the node is fathomed.

Algorithm BB follows a *depth-first* approach. When it has expanded a node, if no child node has been generated, BB backtracks. Otherwise, it selects one of its child nodes to continue the tree search. In our computational experiments, we have observed that the order of the child nodes is very important for the overall performance of BB. Therefore, we propose the following dynamic strategy. If 40% of the remaining items have a width larger than $\bar{w}$, we sort the child nodes by nonincreasing heights of the corresponding items. Otherwise, we sort the child nodes by nonincreasing widths. In our computational results we have set $\bar{w} = (W - w_{\min}/2)/2$, where $w_{\min}$ is the smallest width of the remaining items.

6. Computational Results

The algorithms presented in this paper have been coded in ANSI C using Microsoft Visual Studio 6 and run on a laptop equipped with an Intel Pentium M 725 1.60 GHz. ILOG CPLEX 10.1 was used as the MIP solver for computing lower bound L_{F1}^{BM}, described in §3.4.1, and as the LP solver for computing lower bound L_{F2}^{BM} described in §3.4.2.

We have considered the following sets of test problems: *ngcut*, 12 instances from Beasley (1985b); *cgcut*, 3 instances from Christofides and Whitlock (1977); *gcut*, 13 instances from Beasley (1985a); *beng*, 10 instances from Bengtsson (1982); *ht*, 9 instances from Hopper and Turton (2001); *bkw*, 13 instances from Burke et al. (2004); *class*, 500 instances subdivided in 10 classes, where each class contains five groups that differ for the different number of items, and each group contains 10 instances. The first six classes were proposed by Berkey and Wang (1987), the other four classes were proposed by Martello and Vigo (1998). The two sets *ht* and *bkw* are obtained by cutting a master surface of size (W, H) into smaller rectangles, so the optimal solution value is always equal to H. The remaining sets are well-known test instances for other two-dimensional packing problems that have been transformed into strip-packing instances by setting the strip width W equal to the width of the master surface or of the bin.

Detailed tables are collected in the electronic companion. In Tables 2, 3, 5, and 9 we report in each row the overall results (i.e., averages or sums) of a set of test instances whose name is reported in column *Name*. For test instances *class*, we report the results of each class in a different row. When in our tables a cell is empty, it means that the value is not available. The character "–" is used when the corresponding value cannot be computed (e.g., if no instance has been solved, the average computing time for the solved instances is not available). In Tables 4, 6, 7, and 8 we give some details useful to analyze the computational performance of the proposed lower and upper bounds.

Table 2. Reduction test procedures described in §2.2.

Problem			APT			Our reductions				
Name	n	W	$\Delta n\%$	$\Delta H\%$	Opt	$\Delta n\%$	$\Delta H\%$	$\Delta W\%$	$\Delta A\%$	Opt
ngcut	10–22	10–30	7.19	4.12	0	7.19	4.12	0.00	1.30	0
cgcut	16–62	10–70	0.00	0.00	0	0.00	0.00	0.00	0.00	0
gcut	10–50	250–3,000	17.13	21.45	1	31.69	35.74	0.21	5.05	3
beng	20–200	25–40	0.00	0.00	0	0.00	0.00	0.00	0.00	0
ht	16–29	20–60	0.00	0.00	0	0.00	0.00	0.00	0.00	0
bkw	10–3,152	40–640	8.15	9.27	1	8.15	9.27	0.00	0.00	1
Class 1	20–100	10	10.51	17.88	0	53.43	59.33	0.00	0.03	5
Class 2	20–100	30	0.00	0.00	0	0.00	0.00	0.00	0.00	0
Class 3	20–100	40	1.20	1.35	0	44.87	45.37	0.00	0.33	4
Class 4	20–100	100	0.00	0.00	0	0.00	0.00	0.00	0.00	0
Class 5	20–100	100	4.92	8.29	0	63.89	66.82	0.00	0.59	13
Class 6	20–100	300	0.00	0.00	0	0.00	0.00	0.00	0.00	0
Class 7	20–100	100	18.07	19.39	1	91.20	90.67	0.06	3.10	18
Class 8	20–100	100	1.19	1.42	0	1.44	1.79	0.00	0.03	0
Class 9	20–100	100	39.45	41.21	7	99.17	99.33	0.06	3.62	46
Class 10	20–100	100	6.88	11.03	0	16.50	22.26	0.00	0.25	0

6.1. Reductions

Table 2 shows the results of the reduction procedures and the details of the test instances, reporting for each set the minimum and maximum number of items in column n and the minimum and maximum width of the strip in column W. The computing times are not reported because they are negligible.

The reduction procedure that fixes items in the solution, described in §2.2.1, can be evaluated considering the percentage reduction of the number of items $\Delta n\%$ (i.e., $\Delta n\% = (n-n')/n \times 100$, where n' is the number of items to allocate after reductions), the percentage of height of the strip containing items fixed in the solution $\Delta H\%$ (i.e., $\Delta H\% = (\bar{H} - H_R)/\bar{H} \times 100$, where $\bar{H}$ is the best height found by the new exact algorithm), and the number of instances solved to optimality, *Opt*. In several sets of test instances, many items are fixed in the solution, and for 90 out of 560 instances the optimal solution is reached, as reported in column *Opt*. In columns APT we have reported the results obtained by our implementation of the reduction procedure that fixes items in the solution proposed by Alvarez-Valdes et al. (2009). These results have been obtained using our procedure considering only one "wide" item (i.e., of width larger than $W/2$) at a time. Table 2 shows the effectiveness of our procedure, which considers the subset of wide items (see §2.2.1).

To evaluate the reductions that modify the width of the strip and of the items, described in §§2.2.2 and 2.2.3, column $\Delta W\%$ shows the percentage reduction of the strip (i.e., $\Delta W\% = (W - W')/W \times 100$, where W' is the strip width after reductions), and column $\Delta A\%$ shows the percentage increasing of the item areas (i.e., $\Delta A\% = (A'_I - A_I)/A_I \times 100$, where A_I and A'_I are the total area of items before and after reductions), respectively. Even though the procedures that modify the strip and item sizes give a small contribution, it is convenient to use them because they are computationally cheap.

6.2. Lower Bounds

Table 3 compares the new lower bounds, described in §3, with the lower bounds proposed in the literature.

Columns G_x^{BM} report the percentage gap between the new lower bounds L_x^{BM}, described in §3, and the best solution $\bar{H}$ provided by our new exact algorithm, i.e., $G_x^{BM} = (\bar{H} - L_x^{BM})/\bar{H} \times 100$. For $F1$ and $F2$, the average percentage gaps only consider the instances where a valid lower bound is computed (e.g., for *gcut13* and *bkw13* $F1$ is not solved). L_{BM} represents the value of the new overall lower bound after the postprocessing, described in §3, and G_{BM} is the corresponding percentage gap with respect to $\bar{H}$. The computing times of L_{dff}^{BM} and L_h^{BM} are not reported because they are negligible; column T reports the computing time in seconds for the remaining lower bounds. Column S shows the number of lower bounds solved within the given time limit for L_{F1}^{BM}, and computed because all the y-feasible subsets were generated for L_{F2}^{BM}.

Our lower bounds can be compared with the lower bound values L_{MMV} and L_{BSM} reported by Martello et al. (2003) (and also provided by Iori et al. 2003 for set *class*) and Belov et al. (2006), respectively, and the percentage deviation G_{APT} of the lower bound from the best solution known reported in Alvarez-Valdes et al. (2009). In order to allow a complete comparison, for set *gcut* we also report the averages computed on instances 1–4 and 1–12.

The results show that our new overall lower bound L_{BM} outperforms the lower bounds proposed in the literature, but it often takes a long computing time. However, as the exact algorithm performance demonstrates, in many cases the less expensive lower and upper bounds are sufficient to prove optimality. Only for the hardest instances are the computation of G_{F1}^{BM} and G_{F2}^{BM} required, and the computing time spent is often repayed.

In Table 4 we show for each set the number of times each lower bound leads the best value, also reporting the

Table 3. Comparison among new lower bounds, described in §3, and lower bounds proposed in the literature.

	New lower bounds											Literature		
Problem name	G^{BM}_{dff}	G^{BM}_{h}	G^{BM}_{F1}	T	S	G^{BM}_{F2}	T	S	G_{BM}	L_{BM}	T	L_{MMV}	L_{BSM}	G_{APT}
ngcut	8.14	12.05	0.00	34.63	12	5.68	0.02	12	0.00	40.8	34.64	40.7		0.17
cgcut	2.28	17.99	1.73	441.56	1	1.78	1.02	3	1.21	246.7	445.36	240.7		1.38
gcut 1–4	0.86	4.58	0.08	166.71	3	0.16	0.01	4	0.08	1,749.7	166.92	1,721.5		
gcut 1–12	1.59	7.30	0.21	165.99	9	1.03	0.01	12	0.21	4,266.1	166.21			0.31
gcut 1–13	1.74	7.36	0.21	165.99	9	1.22	1.72	13	0.46	4,305.3	192.60			
beng	0.00	58.32	0.00	82.44	9	0.00	0.49	1	0.00	89.8	82.92	89.8		0.00
ht	0.00	32.96	0.00	314.85	5	0.00	28.60	9	0.00	21.7	343.44	21.7		0.00
bkw	0.53	34.31	0.56	488.10	4	0.00	63.13	3	0.53	177.7	470.97			
Class 1	0.27	5.18	0.00	0.08	50	0.15	8.12	49	0.00	187.8	8.21	187.2	187.7	0.07
Class 2	0.00	51.71	0.00	41.03	49	0.00	7.56	11	0.00	60.5	48.60	60.5	60.5	0.41
Class 3	0.97	5.06	0.47	150.51	39	0.57	10.26	46	0.47	508.1	161.52	504.1	507.6	0.86
Class 4	2.01	53.93	2.01	615.83	0	2.50	2.26	11	1.86	193.6	618.59	193.5	193.6	2.47
Class 5	0.79	3.57	0.34	159.65	39	0.49	0.60	49	0.34	1,632.3	161.54	1,613.2	1,630.7	0.64
Class 6	3.28	54.02	3.28	600.43	0	5.01	13.64	8	2.89	507.0	602.79	506.4	507.0	3.37
Class 7	0.25	0.71	0.00	0.01	50	0.21	0.00	50	0.00	1,591.3	0.01	1,577.8	1,588.8	0.03
Class 8	3.22	18.04	3.07	640.23	1	3.31	44.12	26	2.95	1,400.5	678.36	1,397.9	1,399.8	3.81
Class 9	0.03	0.03	0.00	0.00	50	0.03	0.00	50	0.00	3,346.2	0.00	3,343.1	3,344.6	0.00
Class 10	2.56	18.87	1.80	501.02	13	1.44	3.44	33	1.78	917.9	506.09	909.2	917.6	2.10

number of times it is the only one to reach the best value. Table 4 also reports the number of times the postprocessing improves the lower bound and the number of instances where L^{BM}_{h} is greater than L^{BM}_{dff}.

Each lower bound finds the best value for a large number of times, and for at least one instance it is the only one to reach the best value (also the lower bounds less expensive, i.e., L^{BM}_{dff} and L^{BM}_{h}). It is interesting to notice that L^{BM}_{h} is greater than L^{BM}_{dff} for 25 out of 560 instances, and the more expensive lower bound L^{BM}_{F1} dominates the other lower bounds for 75 out of 560 instances.

Table 4. Comparison among new lower bounds described in §3: number of times each lower bound is the best and is the only one to lead the best value; number of times the postprocessing improves the lower bounds; number of times $L^{BM}_{h} > L^{BM}_{dff}$.

Problem name	Best lower bound				Unique best lower bound				Post proc.	L^{BM}_{h} $>L^{BM}_{dff}$
	L^{BM}_{dff}	L^{BM}_{h}	L^{BM}_{F1}	L^{BM}_{F2}	L^{BM}_{dff}	L^{BM}_{h}	L^{BM}_{F1}	L^{BM}_{F2}		
ngcut	3	3	12	3	0	0	7	0	0	3
cgcut	2	0	3	2	0	0	1	0	1	1
gcut	4	3	12	6	0	0	7	1	0	0
beng	10	0	10	1	0	0	0	0	0	0
ht	9	0	9	9	0	0	0	0	0	0
bkw	13	1	12	3	1	0	0	0	0	0
Class 1	42	9	50	44	0	0	4	0	0	2
Class 2	50	0	50	11	0	0	0	0	0	0
Class 3	28	9	50	32	0	0	13	0	0	3
Class 4	50	0	50	11	0	0	0	0	4	0
Class 5	22	15	49	32	0	0	16	1	0	2
Class 6	50	0	50	10	0	0	0	0	8	0
Class 7	30	31	50	32	0	0	11	0	0	8
Class 8	37	2	46	24	0	1	1	3	2	2
Class 9	46	48	50	46	0	0	2	0	0	2
Class 10	14	3	48	29	0	0	13	2	0	2
Total	409	124	551	295	1	1	75	7	15	25

6.3. Upper Bounds

Table 5 compares the new upper bounds, described in §4, with the upper bounds proposed in the literature.

Columns G_x report the percentage gap between the new upper bounds H_x, described in §4, and our best lower bound $\bar{L}$ provided by our exact method, i.e., $G_x = (H_x - \bar{L})/\bar{L} \times 100$. The value of the new overall upper bound after the postprocessing, described in §4, is shown in column H_{BM}. The computing times of H_{NPS} and H_{PBF} are on average under 10 seconds, whereas the computing times in seconds of H_{HF1} and H_{BM} are reported in column T. For H_{HF1} we also report in column S the number of instances where procedure *HF1* finds a feasible solution. In order to allow a complete comparison, for sets *gcut* and *beng* we also report the averages computed on instances 1–8 and 1–7, respectively.

The new upper bound H_{BM} can be compared with the value of the upper bound H_{IMM}, H_{LMMM}, H_{BKW}, H_{APT}, H_B, and H_{BSM} reported by Iori et al. (2003), Burke et al. (2004), Lesh et al. (2005), Alvarez-Valdes et al. (2008), Bortfeldt (2006), and Belov et al. (2006), respectively. The overall upper bound H_{BM} is competitive with respect to H_{APT} and H_{BSM}, and it outperforms the other heuristic algorithms proposed in the literature.

In Table 6 we show for each set the number of times each upper bound finds the best value, also reporting the number of times it is the only one to reach the best value. Table 6 also reports the number of times the postprocessing improves the upper bound and the number of instances where H_{NPS} is smaller than H_{PBF}.

Table 5. Comparison among the upper bounds proposed in this paper and in the literature.

	New heuristics								Literature				
Problem name	G_{NPS}	G_{PBF}	G_{HF1}	T	S	G_{BM}	H_{BM}	T	H_{IMM}	H_{LMMM}	H_{BKW}	H_{APT}	H_{BSM}
ngcut	0.33	0.70	0.52	0.00	11	0.00	40.8	0.46	40.8			40.8	
cgcut	5.34	1.81	0.00	20.02	1	1.29	252.0	23.88	254.7			249.7	
gcut 1–8	2.86	0.37	0.00	15.01	6	0.25	2,688.0	36.70	2,762.7	2,729.0		2,689.1	
gcut 1–13	3.28	0.81	0.44	15.01	10	0.48	4,330.8	81.77				4,335.5	
beng 1–7	2.30	0.00	7.78	50.05	2	0.00	73.6	49.79	75.3			73.6	
beng 1–10	1.71	0.00	7.78	53.37	2	0.00	89.8	55.37				89.8	
ht	0.00	0.93	0.00	1.88	5	0.00	21.7	6.56					
bkw	3.07	0.69	1.67	46.68	3	0.53	178.4	98.27			180.1	178.9	
	G_{NPS}	G_{PBF}	G_{HF1}	T	S	G_{BM}	H_{BM}	T	H_{IMM}	H_{LMMM}	H_B	H_{APT}	H_{BSM}
Class 1	0.72	0.14	0.36	22.88	39	0.01	187.8	25.54	188.1	188.1	188.1	188.0	187.9
Class 2	2.36	0.12	3.19	46.02	14	0.00	60.5	53.40	61.6	60.8	60.7	60.5	60.5
Class 3	2.86	1.00	1.07	28.65	31	0.60	510.5	34.12	516.2	513.2	513.4	510.6	510.8
Class 4	7.13	1.96	4.00	36.02	1	1.90	196.7	22.99	202.0	200.1	197.9	196.9	196.2
Class 5	1.88	0.73	0.46	23.76	35	0.44	1,638.6	34.99	1,656.6	1,645.9	1,647.0	1,638.4	1,640.0
Class 6	8.64	3.20	—	0.00	0	3.06	521.5	54.76	531.8	531.0	524.7	520.3	518.4
Class 7	0.02	0.02	0.00	0.23	50	0.00	1,591.3	0.75	1,592.3	1,591.9	1,592.7	1,591.3	1,591.4
Class 8	7.53	3.92	1.47	48.62	2	3.07	1,441.8	130.63	1,473.6	1,462.6	1,442.1	1,442.9	1,441.2
Class 9	0.00	0.01	0.00	0.00	50	0.00	3,346.2	0.06	3,346.2	3,346.2	3,347.7	3,346.2	3,346.2
Class 10	4.37	2.03	1.04	46.39	11	1.84	934.9	89.07	949.4	940.0	933.3	933.8	935.4

Procedure PBF is able to reach the best value for 516 out of 560 instances, and for 297 of them it is the only one that can reach it. However, sometimes the other two heuristics are able to find better upper bounds than PBF, because NPS and HF1 are the only heuristics that reach the best value for 7 and 25 instances, respectively. Furthermore, H_{NPS} is smaller than H_{PBF} for 27 out of 560 instances. Notice that the postprocessing is very effective in improving the upper bound for 70 out of 560 instances.

Table 6. Comparison among the new upper bounds described in §4: number of times each upper bound is the best and is the only one to lead the best value; number of times the postprocessing improves the upper bounds; number of times $H_{NPS} < H_{PBF}$.

Problem	Best lower bound			Unique best upper bound			Post	H_{NPS}
name	H_{NPS}	H_{PBF}	H_{HF1}	H_{NPS}	H_{PBF}	H_{HF1}	proc.	$<H_{PBF}$
ngcut	11	9	10	0	0	1	0	2
cgcut	1	3	1	0	2	0	1	0
gcut	5	9	9	0	4	4	3	0
beng	2	9	1	0	7	1	0	0
ht	9	7	5	1	0	0	0	2
bkw	2	13	2	0	10	0	1	0
Class 1	26	47	33	0	16	1	1	2
Class 2	10	49	4	0	38	1	0	0
Class 3	20	44	26	0	23	6	3	4
Class 4	2	50	1	0	47	0	2	0
Class 5	26	44	29	1	19	5	4	4
Class 6	1	49	0	1	49	0	6	1
Class 7	48	44	50	0	0	1	0	5
Class 8	4	46	2	2	44	2	42	2
Class 9	50	49	50	0	0	0	0	1
Class 10	9	44	8	2	38	3	7	4
Total	226	516	231	7	297	25	70	27

Because procedure PBF is based on the best-fit approach proposed by Burke et al. (2004, 2009) and improved by Alvarez-Valdes et al. (2008), in Tables 7 and 8 we give some details to prove computationally the contribution of the new features introduced in this paper. Both Burke et al. (2004, 2009) and Alvarez-Valdes et al. (2008) make use of metaheuristic frameworks and run their algorithms until a time limit of 60 seconds (of a Pentium 4 2GHz) is reached, whereas we do not use any metaheuristics in order to have a fast procedure.

In Tables 7 and 8 we report for procedure PBF the gap G_{PBF}, the upper bound value H_{PBF}, and the computing time T. For Burke et al. (2004, 2009), we report the upper-bound value obtained making use of tabu search, H_{BKW}^{BF+TS}, simulating annealing, H_{BKW}^{BF+SA}, and genetic algorithm, H_{BKW}^{BF+GA}. Because Alvarez-Valdes et al. (2008) run their algorithm 10 times, we report the average upper bound H_{APT}^{mean} and the best upper bound H_{APT}^{Best}.

Although the new procedure PBF does not use metaheuristic frameworks, sometimes it improves the upper bound provided by the algorithms of Burke et al. (2004, 2009) and Alvarez-Valdes et al. (2008). However, when procedure PBF is not able to obtain better upper bounds, it is fast enough to leave computing time to procedures NPS and HF1 and to the postprocessing to reach a competitive upper bound (see Table 5).

Table 7. Comparison among the new procedure PBF proposed in §4.2 and the algorithms proposed by Burke et al. (2004, 2009) and Alvarez-Valdes et al. (2008), based on a best-fit approach.

	PBF heuristic			Burke et al. (2004, 2009)			Alvarez-Valdes et al. (2008)	
Problem name	G_{PBF}	H_{PBF}	T	H_{BKW}^{BF+TS}	H_{BKW}^{BF+SA}	H_{BKW}^{BF+GA}	H_{APT}^{mean}	H_{APT}^{Best}
bkw1	0.00	40	0.00	40	40	40	40.0	40
bkw2	0.00	50	0.41	50	50	50	50.0	50
bkw3	2.00	51	8.06	51	51	52	51.0	51
bkw4	1.25	81	7.66	83	82	83	81.0	81
bkw5	1.00	101	8.67	103	103	104	102.0	102
bkw6	1.00	101	8.44	102	102	102	101.0	101
bkw7	0.00	100	6.52	105	104	104	101.0	101
bkw8	1.25	81	7.47	82	82	82	81.0	81
bkw9	0.67	151	9.42	152	152	152	151.0	151
bkw10	0.67	151	13.66	152	152	152	151.0	151
bkw11	0.67	151	19.08	153	153	153	151.0	151
bkw12	0.33	301	32.92	306	306	306	303.2	303
bkw13	0.10	961	825.95	964	964	964	963.0	963

6.4. Branch-and-Bound Algorithm

Table 9 compares the results obtained with the new branch-and-bound procedure described in §5 and the exact methods proposed in the literature by Martello et al. (2003) and Alvarez-Valdes et al. (2009). For each exact algorithm we report the average of the percentage gap G between the best lower bound L and the best feasible solution found H, i.e., $G = (H - L)/H \times 100$, the maximum percentage gap $G_{\max}$ within the set of test instances, the computing time T in seconds for the test instances solved to optimality, whose number is reported in column S. In order to allow a complete comparison, for set *gcut* we also report the averages computed on instances 1–4.

Martello et al. (2003) have used a Pentium III 800 MHz with a time limit of 3,600 seconds, and Alvarez-Valdes et al. (2009) have used a Pentium IV 2GHz with a time limit of 1,200 seconds. The new exact algorithm, within a time limit of 1,200 seconds, solves to optimality more instances than Martello et al. (2003) (8 out of 38) and Alvarez-Valdes et al. (2009) (20 out of 538). Moreover, our algorithm has lower average and maximum gaps. Only for set *Class 10* does Alvarez-Valdes et al. (2009) obtain a better result. We solve, at the root node, instances *gcut2* and *ht8*, not solved by Alvarez-Valdes et al. (2009). The first one is solved, improving both lower and upper bounds by L_{F1}^{BM} and procedure $HF1$, respectively; the second one is solved by procedure NPS, which finds an improved upper bound. Table 10 reports detailed computational results for the set *class* where we solve some new instances.

The new exact algorithm also solves to optimality 10 instances of sets *gcut* and *bkw* not included in the computational results of Martello et al. (2003) and Alvarez-Valdes et al. (2009). These instances have a large number of items and/or a large strip width.

The computing time required by the new exact algorithm is sometimes larger than the computing time reported by Alvarez-Valdes et al. (2009), but we are often repaid by solving to optimality more instances and by reducing the average gap between the lower and upper bounds. On the other hand, for many sets we solve the same number of instances or more instances in less computing time.

Table 8. Comparison between the new procedure PBF proposed in §4.2 and the GRASP algorithm proposed by Alvarez-Valdes et al. (2008), based on a best-fit approach.

	PBF heuristic			Alvarez-Valdes et al. (2008)	
Problem name	G_{PBF}	H_{PBF}	T	H_{APT}^{mean}	H_{APT}^{Best}
Class 1	0.14	187.9	2.28	188.1	188.0
Class 2	0.12	60.5	7.14	60.6	60.5
Class 3	1.00	511.3	2.96	510.8	510.6
Class 4	1.96	196.7	8.56	197.2	196.9
Class 5	0.73	1,640.4	1.66	1,639.4	1,638.4
Class 6	3.20	521.8	9.52	521.2	520.3
Class 7	0.02	1,591.8	0.01	1,591.6	1,591.3
Class 8	3.92	1,450.8	8.87	1,446.3	1,442.9
Class 9	0.01	3,346.3	0.00	3,346.2	3,346.2
Class 10	2.03	936.0	7.78	935.1	933.8

7. Conclusions

In this paper we have proposed a new branch-and-bound algorithm for the 2SP that follows the classical depth-search strategy and makes use of a new reduction procedure and new lower and upper bounds.

In our computational experiments, we have considered a large number of test instance sets that also include instances not tested for the other exact algorithms reported in the literature. The new exact algorithm solves to optimality several instances unsolved so far, and its average and maximum gaps are better than the ones reported in the literature (only for set *Class 10* does Alvarez-Valdes et al. 2009 obtain a better result). The computational results also show that all the proposed procedures give some advantage (see Tables 2, 4, and 6). The new reduction procedure is able to

Table 9. Comparison among the exact algorithms proposed in this paper and in the literature.

	New exact algorithm				Martello et al. (2003)				Alvarez-Valdes et al. (2009)			
Problem name	T	G	$G_{\max}$	S	T	G	$G_{\max}$	S	T	G	$G_{\max}$	S
ngcut	35.11	0.00	0.00	12	118.35	0.17	2.00	11	6.97	0.00	0.00	12
cgcut	0.00	1.21	2.10	1	11.48	3.68	5.97	1	0.02	1.87	3.08	1
gcut 1–4	2.47	0.08	0.33	3	0.00	2.71	6.21	2	0.12	0.14	0.50	2
gcut 1–13	1.33	0.45	3.46	9								
beng	138.29	0.00	0.00	10	318.89	0.65	2.70	6	1.00	0.00	0.00	10
ht	350.00	0.00	0.00	9	514.33	0.72	3.23	7	1.35	0.36	3.23	8
bkw	345.04	0.53	1.23	4								
Class 1	34.02	0.00	0.00	50					41.58	0.10	0.47	41
Class 2	102.00	0.00	0.00	50					2.66	0.29	2.36	43
Class 3	32.10	0.47	2.01	30					64.62	0.60	2.62	29
Class 4	608.88	1.86	2.65	1					8.00	2.19	3.01	1
Class 5	26.73	0.34	1.82	32					40.80	0.38	1.89	31
Class 6	—	2.89	3.59	0					—	3.20	3.85	0
Class 7	0.76	0.00	0.00	50					1.32	0.00	0.00	50
Class 8	37.67	2.95	3.94	1					10.55	3.12	4.25	1
Class 9	0.06	0.00	0.00	50					0.17	0.00	0.00	50
Class 10	20.73	1.78	4.14	10					42.83	1.68	3.73	10

reduce the size of many instances and solve to optimality some of them. The less expensive lower and upper bounds are able to find the best value for many instances, whereas the more time-consuming procedures are able to improve several lower- and upper-bound values. Furthermore, we show in Tables 7 and 8 that the new features introduced in our procedure PBF give rise to some improved results with respect to Burke et al. (2004, 2009) and Alvarez-Valdes et al. (2008).

Our major contributions consist of the new reduction procedure, described in §2.2.1; the three new lower bounds, described in §§3.3, 3.4.1, and 3.4.2; the new postprocessing procedure to improve the lower bound, described in §3.5; the two new heuristics; the new modified version of an

Table 10. Comparison between the exact method proposed in this paper and the one of Alvarez-Valdes et al. (2009) for test instances *Class* 1, *Class* 2, *Class* 3, and *Class* 5.

Problem			New exact algorithm						Alvarez-Valdes et al. (2009)			
Name	n	W	L	H	T	G	$G_{\max}$	S	T	G	$G_{\max}$	S
Class 1	20	10	61.2	61.2	0.67	0.00	0.00	10	2.75	0.00	0.00	10
	40	10	121.8	121.8	19.55	0.00	0.00	10	6.29	0.00	0.00	10
	60	10	188.5	188.5	39.97	0.00	0.00	10	28.71	0.28	1.20	7
	80	10	262.6	262.6	14.26	0.00	0.00	10	105.30	0.08	0.41	9
	100	10	304.8	304.8	95.68	0.00	0.00	10	93.14	0.25	0.74	5
Avg			187.8	187.8	34.02	0.00	0.00	50	41.58	0.10	0.47	41
Class 2	20	30	19.7	19.7	75.75	0.00	0.00	10	0.40	0.56	5.56	9
	40	30	39.1	39.1	103.55	0.00	0.00	10	0.48	0.21	2.13	9
	60	30	60.1	60.1	146.93	0.00	0.00	10	4.68	0.33	1.79	8
	80	30	83.2	83.2	101.84	0.00	0.00	10	2.63	0.23	1.22	8
	100	30	100.5	100.5	81.93	0.00	0.00	10	5.32	0.11	1.11	9
Avg			60.5	60.5	102.00	0.00	0.00	50	2.66	0.29	2.36	43
Class 3	20	40	162.4	162.6	77.01	0.28	2.76	9	229.49	0.41	2.76	8
	40	40	331.6	333.8	0.37	0.78	2.90	6	0.44	0.94	3.25	6
	60	40	504.0	506.3	19.23	0.51	1.80	4	2.38	0.62	3.11	4
	80	40	707.7	709.8	12.97	0.32	1.21	5	1.79	0.38	1.51	5
	100	40	835.0	838.7	20.99	0.48	1.37	6	2.84	0.66	2.49	6
Avg			508.1	510.2	32.10	0.47	2.01	30	64.62	0.60	2.62	29
Class 5	20	100	532.6	534.1	10.06	0.44	2.61	6	148.75	0.44	2.63	8
	40	100	1,072.1	1,073.8	12.50	0.18	1.53	8	1.45	0.18	0.89	7
	60	100	1,637.8	1,644.0	61.03	0.43	2.63	7	1.39	0.56	3.32	6
	80	100	2,284.8	2,288.5	10.23	0.17	0.71	6	1.83	0.20	0.81	6
	100	100	2,635.3	2,646.3	41.25	0.46	1.63	5	11.32	0.50	1.80	4
Avg			1,632.5	1,637.3	26.73	0.34	1.82	32	40.80	0.38	1.89	31

existing heuristic procedure; and the simple postprocessing procedure to improve the upper bound, described in §4.

This paper does not consider some important issues arising in real-world applications, such as floating point data and 90-degree rotations.

The data structures used in the current version of the code exploit our assumption that the item sizes are integer. Therefore, a possibility is to approximate the floating point sizes with integer sizes (e.g., we can approximate in centimeters, millimeters, etc.). However, to use floating point data we need to replace some data structures and rework some parts of the code to map floating point positions in the discrete set of the normal pattern positions. Using floating point data, the computing time of our procedures would probably increase. Burke et al. (2009) report that their heuristic algorithm, which makes use of floating point data, is up to five times slower than an integer representation implementation.

To consider the 90-degree rotations, we need to deeply change the reduction procedures and the lower bounds, whereas the heuristic procedures and the branch and bounds require only minor adjustments.

The reduction procedure that fixes items in the solution, described in §2.2.1, can be modified as follows. We define $\bar{w}_j = \bar{h}_j = \min\{w_j, h_j\}$ if the item $j \in P$ can be rotated, or $\bar{w}_j = w_j$ and $\bar{h}_j = h_j$ otherwise. We redefine $R' = \{j \in P\colon \bar{w}_j > W/2\}$ and $P_{\bar{R}} = \{k \in P\colon \exists j \in \bar{R} \text{ s.t. } \bar{w}_j + \bar{w}_k \leqslant W\}$, where $\bar{R} \subseteq R'$. If the item of $\bar{R} \cup P_{\bar{R}}$ can be allocated into a strip of width W and height $\sum_{j\in\bar{R}} \bar{h}_j$, then the resulting feasible packing can be fixed at the bottom of the strip without changing the optimal solution value. A similar approach can be used to extend the lower bound based on the item heights, as described in §3.3.

The reduction procedure that modifies the strip width, described in §2.2.2, can be extended by replacing each item that can be rotated with two copies, one for each rotation, and adding a side constraint that forbids using both copies, e.g., $W_P = \max\{\sum_{i\in P} w_i\xi_i^W + h_i\xi_i^H\colon \sum_{i\in P} w_i\xi_i^W + h_i\xi_i^H \leqslant W, \xi_i^W + \xi_i^H \leqslant 1, \xi_i^W, \xi_i^H \in \{0,1\}, i \in P\}$. A similar approach can be also used for lower bounds L_{F1}^{BM} and L_{F2}^{BM}, described in §§3.4.1 and 3.4.2.

The lower bound based on dual feasible functions, described in §3.2, can be extended as shown in Boschetti (2004) for the three-dimensional bin-packing problem with rotations.

However, these are simple extensions, and further research is required to develop new, effective reduction procedures and lower bounds, which consider 90-degree rotations.

8. Electronic Companion

An electronic companion to this paper is available as part of the online version that can be found at http://or.journal.informs.org/.

References

Alvarez-Valdes, R., F. Parreño, J. M. Tamarit. 2008. Reactive GRASP for the strip-packing problem. *Comput. Oper. Res.* **35**(4) 1065–1083.

Alvarez-Valdes, R., F. Parreño, J. M. Tamarit. 2009. A branch and bound algorithm for the strip packing problem. *OR Spectrum* **31**(2) 431–459.

Alvarez-Valdes, R., R. Marti, J. M. Tamarit, A. Parajon. 2007. GRASP and path relinking for the two-dimensional two-stage cutting stock problem. *INFORMS J. Comput.* **19**(2) 261–272.

Alves, C. 2005. Cutting and packing: Problems, models and exact algorithms. Ph.D. thesis, Universidade do Minho, Braga, Portugal.

Baker, B. S., Jr., G. Coffman, R. L. Rivest. 1980. Orthogonal packings in two dimensions. *SIAM J. Comput.* **9**(4) 808–826.

Baldacci, R., M. A. Boschetti. 2007. A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem. *Eur. J. Oper. Res.* **183**(3) 1136–1149.

Beasley, J. E. 1985a. Algorithms for unconstrained two-dimensional guillotine cutting. *J. Oper. Res. Soc.* **36**(4) 297–306.

Beasley, J. E. 1985b. An exact two-dimensional non-guillotine cutting tree search procedure. *Oper. Res.* **33**(1) 49–64.

Belov, G., G. Scheithauer. 2007. Setup amd open-stack minimization in one-dimensional stock cutting. *INFORMS J. Comput.* **19**(1) 27–35.

Belov, G., G. Scheithauer, E. A. Mukhacheva. 2006. One-dimensional heuristics adapted for two-dimensional rectangular strip packing. Preprint MATH-NM-02-2006, Dresden University, Dresden, Germany.

Bengtsson, B. E. 1982. Packing rectangular pieces—A heuristic approach. *Comput. J.* **25**(3) 353–357.

Berkey, J. O., P. Y. Wang. 1987. Two dimensional finite bin packing algorithms. *J. Oper. Res. Soc.* **38**(5) 423–429.

Bortfeldt, A. 2006. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *Eur. J. Oper. Res.* **172**(3) 814–837.

Boschetti, M. A. 2004. New lower bounds for the three-dimensional finite bin packing problems. *Discrete Appl. Math.* **140**(1–3) 241–258.

Boschetti, M. A., A. Mingozzi. 2003a. The two-dimensional finite bin packing problem. Part I: New lower bounds for the oriented case. *4OR* **1**(1) 27–42.

Boschetti, M. A., A. Mingozzi. 2003b. The two-dimensional finite bin packing problem. Part II: New lower and upper bounds. *4OR* **1**(2) 137–147.

Boschetti, M. A., E. Hadjinconstantinou, A. Mingozzi. 2002. New upper bounds for the finite two-dimensional orthogonal non-guillotine cutting stock problem. *IMA J. Management Math.* **13**(2) 95–119.

Burke, E. K., G. Kendall, G. Whitwell. 2004. A new placement heuristic for the orthogonal stock-cutting problem. *Oper. Res.* **54**(4) 655–671.

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

Carlier, J., F. Clautiaux, A. Moukrim. 2007. New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation. *Comput. Oper. Res.* **34**(8) 2223–2250.

Chazelle, B. 1983. The bottom-left bin packing heuristic: An efficient implementation. *IEEE Trans. Comput.* **32**(8) 697–707.

Christofides, N., C. Whitlock. 1977. An algorithm for two-dimensional cutting problems. *Oper. Res.* **25**(1) 30–44.

Clautiaux, F., C. Alves, J. Valério de Carvalho. 2007a. A comparative analysis and computational study of dual-feasible functions for bin-packing problems. Technical report, Université des Sciences et Technologies de Lille, Lille, France.

Clautiaux, F., J. Carlier, A. Moukrim. 2007b. A new exact method for the two-dimensional bin-packing problem with fixed orientation. *Oper. Res. Lett.* **35**(3) 357–364.

Clautiaux, F., J. Carlier, A. Moukrim. 2007c. A new exact method for the two-dimensional orthogonal packing problem. *Eur. J. Oper. Res.* **183**(3) 1196–1211.

Clautiaux, F., A. Jouglet, J. El Hayek. 2007d. A new lower bound for the non-oriented two-dimensional bin-packing problem. *Oper. Res. Lett.* **35**(3) 365–373.

Clautiaux, F., A. Jouglet, J. Carlier, A. Moukrim. 2008. A new constraint programming approach for the orthogonal packing problem. *Comput. Oper. Res.* **35**(3) 944–959.

Fekete, S. P., J. Schepers. 1997. A new exact algorithm for general orthogonal d-dimensional knapsack problems. *Lecture Notes in Computer Science, ESA '97—5th Annual European Symposium*. Springer, New York, 144–156.

Fekete, S. P., J. Schepers. 1998. New classes of lower bounds for bin-packing problem. *Lecture Notes in Computer Science, IPCO 98*, Springer, London, 257–270.

Fekete, S. P., J. Schepers. 2000. On more-dimensional packing II: Bounds. Technical Report 97.289, Universität zu Köln, Köln, Germany.

Fekete, S. P., J. Schepers, J. C. van der Veen. 2007. An exact algorithm for higher-dimensional orthogonal packing. *Oper. Res.* **55**(3) 569–587.

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, New York.

Gómez, A., D. de la Fuente. 2000. Resolution of strip-packing problems with genetic algorithms. *J. Oper. Res. Soc.* **51**(11) 1289–1295.

Herz, J. C. 1972. Recursive computational procedure for the two dimensional stock cutting. *IBM J. Res. Development* **16**(5) 462–469.

Hopper, E., B. C. H. Turton. 2001. An empirical investigation of meta-heuristic and heuristic algorithm for a 2D packing problem. *Eur. J. Oper. Res.* **128**(1) 34–57.

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

Iori, M., S. Martello, M. Monaci. 2003. Metaheuristic algorithms for the strip packing problem. P. M. Pardalos, V. Korotkith, eds. *Optimization and Industry: New Frontiers*. Kluwer Academic Publishers, Boston, 159–179.

Jakobs, S. 1996. On genetic algorithms for the packing of polygons. *Eur. J. Oper. Res.* **88**(1) 165–181.

Johnson, D. S. 1973. Near-optimal bin packing algorithms. Dissertation, MIT, Cambridge, MA.

Lesh, N., M. Mitzenmacher. 2006. BubbleSearch: A simple heuristic for improving priority-based greedy algorithms. *Inform. Processing Lett.* **97**(4) 161–169.

Lesh, N., J. Marks, A. McMahon, M. Mitzenmacher. 2005. New heuristic and interactive approaches to 2D rectangular strip packing. *ACM J. Experiment. Algorithmics* **10** 1–18.

Liu, D., H. Teng. 1999. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *Eur. J. Oper. Res.* **112**(2) 413–420.

Lueker, G. S. 1983. Bin packing with items uniformly distributed over intervals $[a, b]$. *Proc. 24th Annual Sympos. Foundations Comput. Sci. (FOCS 83)*. IEEE Computer Society Press, New York, 289–297.

Martello, S., P. Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, UK.

Martello, S., D. Vigo. 1998. Exact solution of the two-dimensional finite bin packing problem. *Management Sci.* **44**(3) 388–399.

Martello, S., M. Monaci, D. Vigo. 2003. An exact approach to the strip packing problem. *INFORMS J. Comput.* **15**(3) 310–319.

Mukhacheva, E. A., G. N. Belov, V. M. Kartack, A. S. Mukhacheva. 2000. Linear one-dimensional cutting-packing problems: Numerical experiments with the sequential value correction method (SVC) and a modified banch-and-bound method (MBB). *Pesquisa Operacional* **20**(2) 153–168.

Pisinger, D. 1995. An expanding-core algorithm for the exact 0–1 knapsack problem. *Eur. J. Oper. Res.* **87**(1) 175–187.

Scheithauer, G. 1999. LP-based bounds for the container and multi-container loading problem. *Internat. Trans. Oper. Res.* **6**(2) 199–213.

Wäscher, G., H. Haussner, H. Schumann. 2007. An improved typology of cutting and packing problems. *Eur. J. Oper. Res.* **183**(3) 1109–1130.

Yeungm, L. H. W., W. K. S. Tang. 2004. Strip-packing using hybrid genetic approach. *Engrg. Appl. Artificial Intelligence* **17**(2) 169–177.