ELSEVIER

Discrete Applied Mathematics 123 (2002) 379–396

DISCRETE APPLIED MATHEMATICS

Recent advances on two-dimensional bin packing problems

Andrea Lodi, Silvano Martello*, Daniele Vigo

Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna, Viale Risorgimento 2, 40136 - Bologna, Italy

Received 4 October 1999; received in revised form 1 June 2001; accepted 25 June 2001

Abstract

We survey recent advances obtained for the two-dimensional bin packing problem, with special emphasis on exact algorithms and effective heuristic and metaheuristic approaches.

1. Introduction

In the *two-dimensional bin packing problem* (2BP) we are given a set of n rectangular *items* $j \in J = \{1,\dots,n\}$, each having *width* w_j and *height* h_j, and an unlimited number of finite identical rectangular *bins*, having width W and height H. The problem is to allocate, without overlapping, all the items to the minimum number of bins, with their edges parallel to those of the bins. It is assumed that the items have fixed orientation, i.e., they cannot be rotated.

Problem 2BP has many industrial applications, especially in cutting (wood and glass industries) and packing (transportation and warehousing). Certain applications may require additional constraints and/or assumptions, some of which are discussed in the final section of this paper.

The special case where $w_j = W$ $(j = 1,\dots,n)$ is the famous *one-dimensional bin packing problem* (1BP): partition n elements, each having an associated size h_j, into the minimum number of subsets so that the sum of the sizes in each subset does not exceed a given capacity H. Since 1BP is known to be strongly NP-hard, the same holds for 2BP.

* Corresponding author. Tel.: +39 051 209 3022; fax: +39 051 209 3073.

E-mail addresses: alodi@deis.unibo.it (A. Lodi), smartello@deis.unibo.it (S. Martello), dvigo@deis.unibo.it (D. Vigo).

0166-218X/02/$ - see front matter
PII: S0166-218X(01)00347-X

Gilmore and Gomory [26] proposed the first model for two-dimensional packing problems, by extending their column generation approach for 1BP (see [24,25]). Beasley [5] considered a variant of 2BP (the *cutting stock problem*), and gave an Integer Linear Programming formulation based on the use of discrete coordinates at which items may be allocated. A similar model has been introduced by Hadjiconstantinou and Christofides [30]. Very recently, Fekete and Schepers [19] proposed a new model based on a graph-theoretical characterization of the problem, while Lodi et al. [40] presented, for the special case where the items have to be packed "by levels" (see below) ILP models involving a polynomial number of variables and constraints.

In this paper we survey recent advances obtained for the two-dimensional bin packing problem, with special emphasis on exact algorithms and effective heuristic and metaheuristic approaches. Concerning heuristics, we will only consider *off-line* algorithms, for which it is assumed that the algorithm has full knowledge of the whole input. (The reader is referred to [15], for a survey on *on-line* algorithms, which pack each item as soon as it is encountered, without knowledge of the next items.)

Up to the mid-Nineties, almost all results in the literature concerned heuristic algorithms. In the next section, we first review some basic algorithms having relevant implications on the topic of the present survey, and discuss more recent results. The following sections are devoted to other results obtained in the last few years: lower bounds (Section 3), exact algorithms (Section 4) and metaheuristic approaches (Section 5). We conclude by discussing some variants of the problem (Section 6) and future directions of research (Section 7). For many of the above techniques we summarize the results of computational experiments. For some upper and lower bounds, worst-case results are also discussed.

Without loss of generality, we will assume throughout the paper that all input data are positive integers, and that $w_j \leqslant W$ and $h_j \leqslant H$ $(j = 1, \ldots, n)$.

2. Upper bounds

Most of the off-line algorithms from the literature are of greedy type, and can be classified in two families:

- *one-phase algorithms* directly pack the items into the finite bins;
- *two-phase algorithms* start by packing the items into a single *strip*, i.e., a bin having width W and infinite height. In the second phase, the strip solution is used to construct a packing into finite bins.

In addition, most of the approaches are *level algorithms*, i.e., the bin/strip packing is obtained by placing the items, from left to right, in rows forming *levels*. The first level is the bottom of the bin/strip, and subsequent levels are produced by the horizontal line coinciding with the top of the tallest item packed on the level below. Three classical strategies for the level packing have been derived from famous algorithms for the one-dimensional case. In each case, the items are initially sorted by non-decreasing height and packed in the corresponding sequence. Let j denote the current item, and s

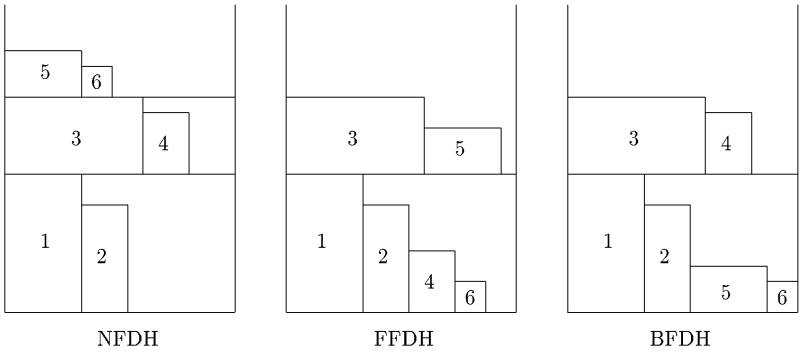

Fig. 1. Level packing strategies.

the last created level:

- *Next-Fit Decreasing Height* (NFDH) strategy: item j is packed left justified on level s, if it fits. Otherwise, a new level ($s:=s+1$) is created, and j is packed left justified into it;
- *First-Fit Decreasing Height* (FFDH) strategy: item j is packed left justified on the first level where it fits, if any. If no level can accommodate j, a new level is initialized as in NFDH;
- *Best-Fit Decreasing Height* (BFDH) strategy: item j is packed left justified on that level, among those where it fits, for which the unused horizontal space is a minimum. If no level can accommodate j, a new level is initialized as in NFDH.

The above strategies are illustrated through an example in Fig. 1.

In what follows we assume, unless otherwise specified, that the items are initially sorted by non-increasing height.

2.1. Strip packing

Coffman et al. [12] analyzed NFDH and FFDH for the solution of the *two-dimensional strip packing problem*, in which one is required to pack all the items into a strip of minimum height, and determined their asymptotic worst-case behavior. Given a minimization problem P and an approximation algorithm A, let $A(I)$ and $OPT(I)$ denote the value produced by A and the optimal solution value, respectively, for an instance I of P. Coffman et al. [12] proved that, if the heights are normalized so that $\max_j\{h_j\}=1$, then

$$NFDH(I) \leqslant 2 \cdot OPT(I) + 1 \tag{1}$$

and

$$FFDH(I) \leqslant \tfrac{17}{10} \cdot OPT(I) + 1 \tag{2}$$

Both bounds are *tight* (meaning that the multiplicative constants are as small as possible) and, if the h_j's are not normalized, only the additive term is affected. Observe the similarity of (1) and (2) with famous results on the one-dimensional counterparts of NFDH and FFDH (algorithms *Next-Fit* and *First-Fit*, respectively, see [34]).

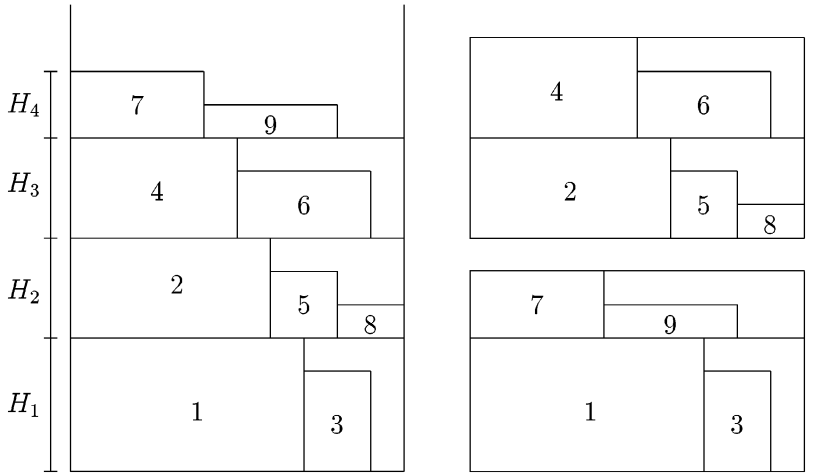

Fig. 2. First and second phase of algorithm HFF.

Any algorithm requiring item sorting is obviously $\Omega(n \log n)$. Both NFDH and FFDH can be implemented so as to require $O(n \log n)$ time, by using the appropriate data structures adopted for the one-dimensional case (see [33]).

Several other papers on the strip packing problem can be found in the literature: see, e.g., [3,45,8,28,2,4,32,46]. The algorithm of Baker et al. [3] is considered in Section 2.4, while the other results, which have not been directly used for the finite bin case, are beyond the scope of this survey, and will not be discussed here.

2.2. Bin packing: two-phase algorithms

A two-phase algorithm for the finite bin packing problem, called *Hybrid First-Fit* (HFF), was proposed by Chung et al. [11]. In the first phase, a strip packing is obtained through the FFDH strategy. Let $H_1, H_2, \ldots$ be the heights of the resulting levels, and observe that $H_1 \geqslant H_2 \geqslant \ldots$. A finite bin packing solution is then obtained by heuristically solving a one-dimensional bin packing problem (with item sizes H_i and bin capacity H) through the *First-Fit Decreasing* algorithm: initialize bin 1 to pack level 1, and, for increasing $i = 2, \ldots,$ pack the current level i into the lowest indexed bin where it fits, if any; if no bin can accommodate i, initialize a new bin. An example is shown in Fig. 2. Chung et al. [11] proved that, if the heights are normalized to one, then

$$HFF(I) \leqslant \tfrac{17}{8} \cdot OPT(I) + 5. \tag{3}$$

The bound is not proved to be tight: the worst example gives $HFF(I) = \frac{91}{45} \cdot (OPT(I) - 1)$. Both phases can be implemented so as to require $O(n \log n)$ time.

Berkey and Wang [7] proposed and experimentally evaluated a two-phase algorithm, called *Finite Best-Strip* (FBS), which is a variation of HFF. The first phase is performed by using the BFDH strategy. In the second phase, the one-dimensional bin

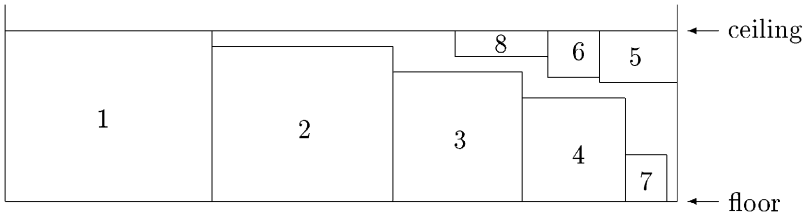

Fig. 3. Algorithm FC.

packing problem is solved through the *Best-Fit Decreasing* algorithm: pack the current level in that bin, among those where it fits (if any), for which the unused vertical space is a minimum, or by initializing a new bin. (For the sake of uniformity, *Hybrid Best-Fit* would be a more appropriate name for this algorithm.)

Let us consider now another variation of HFF, in which the NFDH strategy is adopted in the first phase, and the one-dimensional bin packing problem is solved through the *Next-Fit Decreasing* algorithm: pack the current level in the current bin if it fits, or initialize a new (current) bin otherwise. Due to the next-fit policy, this algorithm is equivalent to a one-phase algorithm in which the current item is packed on the current level of the current bin, if possible; otherwise, a new (current) level is initialized either in the current bin (if enough vertical space is available), or in a new (current) bin. Frenk and Galambos [23] analyzed the resulting algorithm, *Hybrid Next-Fit* (HNF), by characterizing its asymptotic worst-case performance as a function of $\max_j\{w_j\}$ and $\max_j\{h_j\}$. By assuming that the heights and widths are normalized to one, the worst performance occurs for $\max_j\{w_j\} > \frac{1}{2}$ and $\max_j\{h_j\} \geqslant \frac{1}{2}$, and gives:

$$HNF(I) \leqslant 3.382\ldots \cdot OPT(I) + 9 \tag{4}$$

where $3.382\ldots$ is an approximation for a tight but irrational bound. The three algorithms above can be implemented so as to require $O(n \log n)$ time. The next two algorithms have higher worst-case time complexities, although they are, in practice, very fast and effective.

Lodi et al. [37,39] presented an approach (*Floor-Ceiling*, FC) which extends the way items are packed on the levels. Denote the horizontal line defined by the top (resp. bottom) edge of the tallest item packed on a level as the *ceiling* (resp. *floor*) of the level. The previous algorithms pack the items, from left to right, with their bottom edge on the level floor. Algorithm FC may, in addition, pack them, from right to left, with their top edge on the level ceiling. The first item packed on a ceiling can only be one which cannot be packed on the floor below. A possible floor-ceiling packing is shown in Fig. 3. In the first phase, the current item is packed, in order of preference: (i) on a ceiling (provided that the requirement above is satisfied), according to a best-fit strategy; (ii) on a floor, according to a best-fit strategy; (iii) on the floor of a new level. In the second phase, the levels are packed into finite bins, either through the Best-Fit Decreasing algorithm or by using an exact algorithm for the one-dimensional

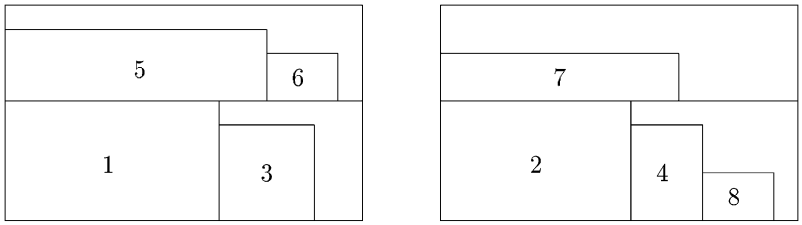

Fig. 4. Algorithm FFF.

bin packing problem, halted after a prefixed number of iterations. The implementation of the first phase given in [37] requires $O(n^3)$ time, while the complexity of the second one obviously depends on the selected algorithm.

Another level packing strategy based on the exact solution of induced subproblems is adopted in the *Knapsack Packing* (KP) algorithm proposed by Lodi et al. [39]. In the (binary) *knapsack problem* one has to select a subset of n elements, each having an associated profit and weight, so that the total weight does not exceed a given capacity and the total profit is a maximum. The first phase of algorithm KP packs one level at a time as follows. The first (tallest) unpacked item, say j^*, initializes the level, which is then completed by solving an associated knapsack problem instance over all the unpacked items, where: (i) the knapsack capacity is $W - w_{j^*}$; (ii) the weight of an item j is w_j; (iii) the profit of an item j is its area $w_j\, h_j$. Finite bins are finally obtained as in algorithm FC. Algorithm KP (as well as algorithm FC above) may require the solution of NP-hard subproblems, producing a non-polynomial time complexity. In practice, however, the execution of the codes for the NP-hard problems is always halted after a prefixed (small) number of iterations, and in almost all cases, the optimal solution is obtained before the limit is reached (see the computational experiments in [39] and in Section 2.5).

2.3. Bin packing: one-phase algorithms

Two one-phase algorithms were presented and experimentally evaluated by Berkey and Wang [7].

Algorithm *Finite Next-Fit* (FNF) directly packs the items into finite bins exactly in the way algorithm HNF of the previous section does. (Papers [7,23] appeared in the same year.)

Algorithm *Finite First-Fit* (FFF) adopts instead the FFDH strategy. The current item is packed on the lowest level of the first bin where it fits; if no level can accommodate it, a new level is created either in the first suitable bin, or by initializing a new bin (if no bin has enough vertical space available). An example of application of FFF is given in Fig. 4.

Both algorithms can be implemented so as to require $O(n \log n)$ time.

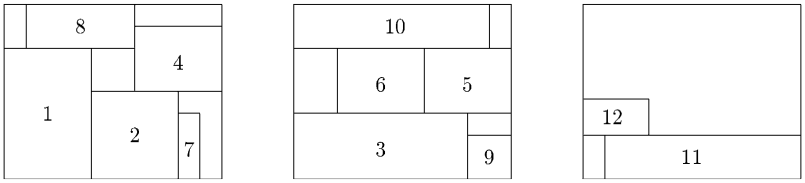

Fig. 5. Algorithm AD.

2.4. Bin packing: non-level algorithms

We finally consider algorithms which do not pack the items by levels. All the algorithms discussed in the following are one-phase.

The main non-level strategy is known as *Bottom-Left* (BL), and consists in packing the current item in the lowest possible position, left justified. Baker et al. [3] analyzed the worst-case performance of the resulting algorithm for the strip packing problem, and proved that: (i) if no item ordering is used, BL may be arbitrarily bad; (ii) if the items are ordered by non-increasing width then $BL(I) \leqslant 3 \cdot OPT(I)$, and the bound is tight.

Berkey and Wang [7] proposed the BL approach for the finite bin case. Their *Finite Bottom-Left* (FBL) algorithm initially sorts the items by non-increasing width. The current item is then packed in the lowest position of any initialized bin, left justified; if no bin can allocate it, a new one is initialized. The computer implementation of algorithm BL was studied by Chazelle [9], who gave a method for producing a packing in $O(n^2)$ time. The same approach was adopted by Berkey and Wang [7].

Lodi et al. [39] proposed a different non-level approach, called *alternate directions* (AD). The method is illustrated in Fig. 5. The algorithm initializes L bins (L being a lower bound on the optimal solution value, see Section 3) by packing on their bottoms a subset of the items, following a best-fit decreasing policy (items 1, 2, 3, 7 and 9 in Fig. 5, where it is assumed that $L = 2$). The remaining items are packed, one bin at a time, into *bands*, alternatively from left to right and from right to left. As soon as no item can be packed in either direction in the current bin, the next initialized bin or a new empty bin (the third one in Fig. 5, when item 11 is considered) becomes the current one. The algorithm has $O(n^3)$ time complexity.

2.5. Computational experiments

Probabilistic analysis and experimental tests are two classical methods for evaluating the expected behavior of approximation algorithms. The former technique is fully illustrated in the book by Coffman and Lueker [13], where specific results on two-dimensional bin packing and strip packing algorithms can be found in Chapter 7. More recent results are in [14]. In this section we summarize the outcome of a series of computational experiments aimed at analyzing the typical behavior of the

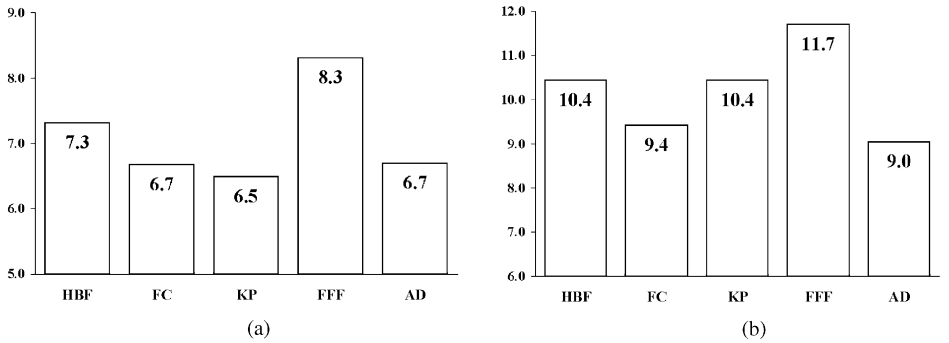

Fig. 6. Average percentage deviations from lower bound: (a) Martello and Vigo instance, Classes 1–4; (b) Berkey and Wang instances, Classes 5–10.

main heuristic algorithms on some classes of instances. The benchmark consists of 500 random instances, with $n \in \{20, 40, 60, 80, 100\}$. Ten different classes of instances were used.

The first four classes were proposed by Martello and Vigo [43], and are based on the generation of items of four different types:

Type 1: w_j uniformly random in $[\frac{2}{3}W, W]$, h_j uniformly random in $[1, \frac{1}{2}H]$.

Type 2: w_j uniformly random in $[1, \frac{1}{2}W]$, h_j uniformly random in $[\frac{2}{3}H, H]$.

Type 3: w_j uniformly random in $[\frac{1}{2}W, W]$, h_j uniformly random in $[\frac{1}{2}H, H]$.

Type 4: w_j uniformly random in $[1, \frac{1}{2}W]$, h_j uniformly random in $[1, \frac{1}{2}H]$.

Class k ($k \in \{1,2,3,4\}$) is then obtained by generating an item of type k with probability 70%, and of the remaining types with probability 10% each. The bin size is always $W = H = 100$.

The next six classes have been proposed by Berkey and Wang [7]:

Class 5: $W = H = 10$, w_j and h_j uniformly random in $[1, 10]$.

Class 6: $W = H = 30$, w_j and h_j uniformly random in $[1, 10]$.

Class 7: $W = H = 40$, w_j and h_j uniformly random in $[1, 35]$.

Class 8: $W = H = 100$, w_j and h_j uniformly random in $[1, 35]$.

Class 9: $W = H = 100$, w_j and h_j uniformly random in $[1, 100]$.

Class 10: $W = H = 300$, w_j and h_j uniformly random in $[1, 100]$.

For each class and value of n, ten instances have been generated. The 500 instances, as well as the generator code, are available on the internet at http://www.or.deis.unibo.it/ORinstances/2BP/.

Fig. 6 summarizes the results, by giving, for each algorithm, the average percentage deviation of the heuristic solution value from a lower bound value, computed as $\max\{L_2, L_3\}$ (see Section 3), with respect to the 200 instances of Classes 1–4 (Fig. 6 (a)) and to the 300 instances of Classes 5–10 (Fig. 6 (b)). The results show that algorithms FC, KP and AD have the best behavior, clearly superior to the classical approaches.

3. Lower bounds

Good lower bounds on the optimal solution value are important both in the implementation of exact enumerative approaches and in the empirical evaluation of approximate solutions. The simplest bound for 2BP is the *Continuous Lower Bound*

$$L_0 = \left\lceil \frac{\sum_{j=1}^{n} w_j h_j}{WH} \right\rceil$$

computable in linear time. Martello and Vigo [43] determined the absolute worst-case behavior of L_0:

$$L_0(I) \geqslant \tfrac{1}{4} \cdot OPT(I)$$

where $L_0(I)$ and $OPT(I)$ denote the value produced by L_0 and the optimal solution value, respectively, for an instance I of problem P. The bound is tight, as shown by the example in Fig. 7. The result holds even if rotation of the items (by any angle) is allowed.

In many cases, the value provided by L_0 can be inadequate (too small) for an effective use within an exact algorithm. A better (tighter) bound was proposed by Martello and Vigo [43]. Given any integer value q, $1 \leqslant q \leqslant \frac{1}{2}W$, let

$$K_1 = \{j \in J : w_j > W - q\}, \tag{5}$$

$$K_2 = \{j \in J : W - q \geqslant w_j > \tfrac{1}{2}W\}, \tag{6}$$

$$K_3 = \{j \in J : \tfrac{1}{2}W \geqslant w_j \geqslant q\}. \tag{7}$$

and observe that no two items of $K_1 \cup K_2$ may be packed side by side into a bin. Hence, a lower bound L_1^W for the sub-instance given by the items in $K_1 \cup K_2$ can be obtained by using any lower bound for the 1BP instance defined by element sizes h_j ($j \in K_1 \cup K_2$) and capacity H (see [42,17]). A lower bound for the complete instance is then obtained by taking into account the items in K_3, since none of them may be packed besides an item of K_1:

$$L_2^W(q) = L_1^W + \max\left\{0, \left\lceil \frac{\sum_{j \in K_2 \cup K_3} w_j h_j - (HL_1^W - \sum_{j \in K_1} h_j)W}{WH} \right\rceil \right\}. \tag{8}$$

A symmetric bound $L_2^H(q)$ is clearly obtained by interchanging widths and heights. By observing that both bounds are valid for any q, we have an overall lower bound:

$$L_2 = \max\left(\max_{1 \leqslant q \leqslant \frac{1}{2}W} \{L_2^W(q)\}, \max_{1 \leqslant q \leqslant \frac{1}{2}H} \{L_2^H(q)\} \right) \tag{9}$$

It is shown in [43] that, for any instance of 2BP, the value produced by L_2 is no less than that produced by L_0, and that L_2 can be computed in $O(n^2)$ time.

Martello and Vigo [43] also proposed a computationally more expensive lower bound, which in some cases improves on L_2. Given any pair of integers (p, q), with

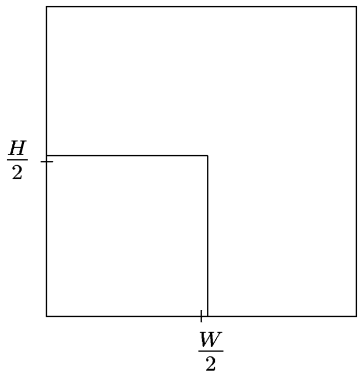

Fig. 7. Worst-case of the continuous lower bound.

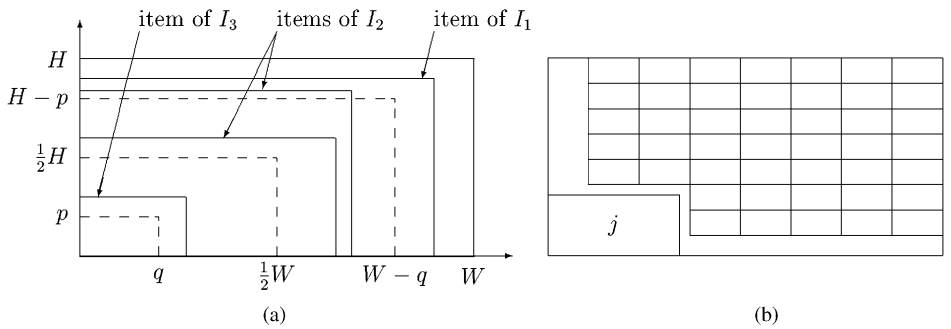

Fig. 8. (a) Items in I_1, I_2 and I_3; (b) relaxed instance with reduced items.

$1 \leqslant p \leqslant \frac{1}{2}H$ and $1 \leqslant q \leqslant \frac{1}{2}W$, define:

$$I_1 = \{j \in J : h_j > H - p \text{ and } w_j > W - q\}, \tag{10}$$

$$I_2 = \{j \in J \setminus I_1 : h_j > \tfrac{1}{2}H \text{ and } w_j > \tfrac{1}{2}W\}, \tag{11}$$

$$I_3 = \{j \in J : \tfrac{1}{2}H \geqslant h_j \geqslant p \text{ and } \tfrac{1}{2}W \geqslant w_j \geqslant q\} \tag{12}$$

(see Fig. 8(a)), and observe that: (i) $I_1 \cup I_2$ is independent of (p,q); (ii) no two items of $I_1 \cup I_2$ may be packed into the same bin; (iii) no item of I_3 fits into a bin containing an item of I_1. A valid lower bound can thus be computed by adding to $|I_1 \cup I_2|$ the minimum number of bins needed for those items of I_3 that cannot be packed into the bins used for the items of I_2. Such a bound can be determined by considering a relaxed instance where each item $i \in I_3$ has the minimum size, i.e., $h_i = p$ and $w_i = q$. Given a bin containing an item j, the maximum number of $p \times q$ items that can be packed

into the bin is (see Fig. 8(b)):

$$m(j,p,q) = \left\lfloor \frac{H}{p} \right\rfloor \left\lfloor \frac{W-w_j}{q} \right\rfloor + \left\lfloor \frac{W}{q} \right\rfloor \left\lfloor \frac{H-h_j}{p} \right\rfloor - \left\lfloor \frac{H-h_j}{p} \right\rfloor \left\lfloor \frac{W-w_j}{q} \right\rfloor \quad (13)$$

Hence, for any pair (p,q) a valid lower bound is

$$L_3(p,q) = |I_1 \cup I_2| + \max\left\{0, \left\lceil \frac{|I_3| - \sum_{j\in I_2} m(j,p,q)}{\lfloor \frac{H}{p} \rfloor \lfloor \frac{W}{q} \rfloor} \right\rceil \right\} \quad (14)$$

so an overall bound is

$$L_3 = \max_{1\leqslant p\leqslant \frac{1}{2}H,\ 1\leqslant q\leqslant \frac{1}{2}W} \{L_3(p,q)\}. \quad (15)$$

Lower bound L_3 can be computed in $O(n^3)$ time. No dominance relation exists between L_2 and L_3.

A general bounding technique for bin and strip packing problems in one or more dimensions, based on the use of dual feasible functions, was recently proposed by Fekete and Schepers [22,20].

4. Exact algorithms

An enumerative approach for the exact solution of 2BP was presented by Martello and Vigo [43]. The items are initially sorted in non-increasing order of their area. A reduction procedure tries to determine the optimal packing of some bins, thus reducing the size of the instance. A first incumbent solution, of value z^*, is then heuristically obtained.

The algorithm is based on a two-level branching scheme:

- *outer branch-decision tree*: at each decision node, an item is assigned to a bin without specifying its actual position;
- *inner branch-decision tree*: a feasible packing (if any) for the items currently assigned to a bin is determined, possibly through enumeration of all the possible patterns.

The outer branch-decision tree is searched in a depth-first way, making use of the lower bounds described in the previous section. Whenever it is possible to establish that no more unassigned items can be assigned to a given initialized bin, such a bin is *closed*: an initialized and not closed bin is called *active*. At level k ($k = 1,\ldots,n$), item k is assigned, in turn, to all the active bins and, possibly, to a new one (if the total number of active and closed bins is less than $z^* - 1$).

The feasibility of the assignment of an item to a bin is first heuristically checked. A lower bound $L(I)$ is computed for the instance I defined by the items currently assigned to the bin: if $L(I) > 1$, a backtracking follows. Otherwise, heuristic algorithms are applied to I: if a feasible single-bin packing is found, the outer enumeration is resumed. If not, the inner branching scheme enumerates all the possible ways to pack I into a bin through the *left-most downward* strategy (see [29]): at each level, the next item is placed, in turn, into all positions where it has its left edge adjacent either to the

Table 1
Number of instances, out of ten, solved to proved optimality

	Class										
n	1	2	3	4	5	6	7	8	9	10	Total
20	10	10	10	10	10	10	9	10	10	10	99
40	8	7	10	9	10	10	10	10	10	5	89
60	8	7	10	2	7	4	7	7	8	10	70
80	7	3	10	—	3	10	—	10	—	10	53
100	7	6	8	—	1	10	—	10	1	2	45
Total	40	33	48	21	31	44	26	47	29	37	356

right edge of another item or to the left edge of the bin, and its bottom edge adjacent either to the top edge of another item or to the bottom edge of the bin. As soon as a feasible packing is found for all the items of I, the outer enumeration is resumed. If no such packing exists, an outer backtracking is performed.

Whenever the current assignment is feasible, the possibility of closing the bin is checked through lower bound computations.

Table 1 gives the results of computational experiments performed, with a Fortran 77 implementation, on the 500 instances described in Section 2.5. The entries give, for each class and value of n, the number of instances (out of ten) solved to proved optimality within a time limit of 300 CPU seconds on a PC Pentium 200 MHz.

Fekete and Schepers [21] recently derived from their graph-theoretical model [19] an alternative enumerative approach.

5. Metaheuristics

In recent years, metaheuristic techniques have become a popular tool for the approximate solution of hard combinatorial optimization problems. (See [1,27] for general introductions to the field.) Lodi et al. [37–39] developed effective tabu search algorithms for 2BP and for some of the variants discussed in the next section. We briefly describe here the unified tabu search framework given in [39], whose main characteristic is the adoption of a search scheme and a neighborhood which are independent of the specific packing problem to be solved. The framework can thus be used for virtually any variant of 2BP, by simply changing the specific deterministic algorithm used for evaluating the moves within the neighborhood search.

Given a current solution, the moves modify it by changing the packing of a subset S of items, trying to empty a specified *target bin*. Let S_i be the set of items currently packed into bin i: the target bin t is the one minimizing, over all bins i, the function

$$\varphi(S_i) = \alpha \frac{\sum_{j \in S_i} w_j h_j}{WH} - \frac{|S_i|}{n} \tag{16}$$

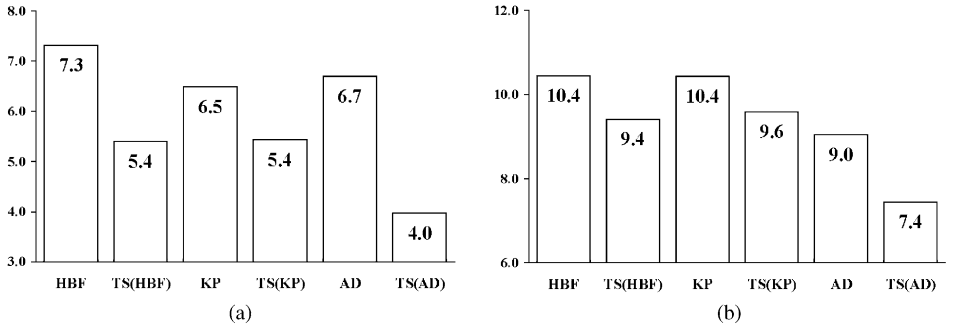

Fig. 9. Average percentage deviations from lower bound: (a) Martello and Vigo instances, Classes 1–4; (b) Berkey and Wang instances, Classes 5–10.

(α is a pre-specified positive weight), which gives a measure of the easiness of emptying the bin. It favors, indeed, target bins packing a small area and a relatively large number of items.

Once the target bin has been selected, subset S is defined so as to include one item, j, from the target bin and the current contents of k other bins. The new packing for S is obtained by executing an appropriate heuristic algorithm A on S. The value of parameter k, which defines the size and the structure of the current neighborhood, is automatically updated during the search.

If the move packs the items of S into k (or less) bins, i.e., item j has been removed from the target bin, a new item is selected, a new set S is defined accordingly, and a new move is performed. Otherwise S is changed by selecting a different set of k bins, or a different item j from the target bin (if all possible configurations of k bins have been attempted for the current j).

If the algorithm gets stuck, i.e., the target bin is not emptied, the neighborhood is enlarged by increasing the value of k, up to a prefixed upper limit. There are a tabu list and a tabu tenure for each value of k.

An initial incumbent solution is obtained by executing algorithm A on the complete instance, while the initial tabu search solution consists of packing one item per bin. In special situations, a move is followed by a diversification action. The execution is halted as soon as a proven optimal solution is found, or a time limit is reached.

Fig. 9 shows the impact of tabu search for three of the heuristics described in Section 2: notation TS(A) indicates that algorithm A is used within the tabu search. The figure gives the average percentage deviations of the heuristic solution value (without and with tabu search) from the best known lower bound value, with respect to the 200 instances of Classes 1–4 (Fig. 9(a)) and to the 300 instances of Classes 5–10 (Fig. 9(b)), as described in Section 2.5. The tabu search was performed with a time limit of 60 s on a Silicon Graphics INDY R10000sc (195 MHz). The results show that the use of tabu search considerably improves the performance of all algorithms.

6. Variants

Two-dimensional bin packing problems occur in several real-world contexts, especially in cutting and packing industries. As a consequence, a number of variants arises, according to specific applications. In most cases the additional requirements concern *orientation* and/or *guillotine cutting*.

In the bin and strip packing problems considered so far we have assumed that the items have a fixed orientation (i.e., they cannot be rotated), and that no restriction is imposed on the cutting patterns. In certain real-world contexts, item rotation (usually by 90°) may be allowed in order to produce better packings. In addition, many practical cutting contexts may impose that the items are obtained through a sequence of guillotine cuts, i.e., edge-to-edge cuts parallel to the edges of the bin. For example, rotation is not allowed when the items are articles to be paged in newspapers or are pieces to be cut from decorated or corrugated stock units, whereas it is allowed in the cutting of plain materials and in most packing contexts. The guillotine constraint is usually imposed by technological characteristics of the automated cutting machines, whereas it is generally not present in packing applications.

Lodi et al. [39] proposed the following typology for the four possible cases produced by the above two characterizations:

2BP|O|G: the items are oriented (O) and guillotine cutting (G) is required;
2BP|R|G: the items may be rotated by 90° (R) and guillotine cutting is required;
2BP|O|F: the items are oriented and cutting is free (F);
2BP|R|F: the items may be rotated by 90° and cutting is free.

(The problem considered so far is thus 2BP|O|F.) The following references are examples of industrial applications involving the above variants. A problem of trim-loss minimization in a crepe-rubber mill, studied by Schneider [44], induces subproblems of 2BP|O|G type; fuzzy two-dimensional cutting stock problems arising in the steel industry, discussed by Vasko et al. [47], are related to 2BP|R|G; the problem of optimally placing articles and advertisements in newspapers and yellow pages, studied by Lagus et al. [36], falls into the 2BP|O|F case; finally, several applications of 2BP|R|F are considered by Bengtsson [6].

An algorithm for one of the variants may obviously guarantee solutions which are feasible for others. The complete set of compatibilities between algorithms and problems is shown in Fig. 10, where A_{XY} is an algorithm for 2BP$|X|Y$ and an edge $(A_{XY}, 2\mathrm{BP}|Q|T)$ indicates that A_{XY} produces solutions feasible for 2BP$|Q|T$.

It is easily seen that all the level algorithms described in Section 2 directly produce guillotine packings, the only exception being FC. Adaptations of FC which modify the way items are packed on the ceilings so as to preserve the guillotine constraint (with or without rotation) were presented by Lodi et al. [37,39].

Most of the other heuristics of Section 2 can be modified so as to handle rotation and/or guillotine cutting. Lodi et al. [39] proposed the following new algorithms for the considered variants.

For 2BP|O|G, algorithm KP of Section 2.2 (denoted as $\mathrm{KP_{OG}}$ in [39]) directly produces guillotine packings.

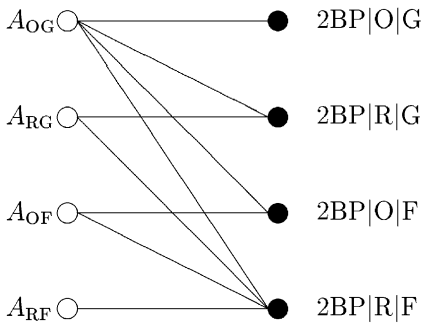

Fig. 10. Compatibilities between algorithms and problems.

For 2BP|R|G, algorithm KP_{RG} modifies KP as follows. The items are initially sorted according to non-increasing value of their shortest edge, and *horizontally oriented* (i.e., with their longest edge as the base): this orientation is always used for the level initializations. For each level, say of height h^*, the knapsack instance includes each unpacked item, either in vertical orientation, if its size does not exceed h^*, or in horizontal orientation, otherwise. Once a feasible finite bin solution has been obtained from the resulting levels, an alternative solution is derived by considering the set of the items currently packed in each level as a pseudo-item, rotating it whenever possible, and applying algorithm KP to the resulting instance.

For 2BP|R|F, algorithm TP_{RF} sorts the items according to non-increasing area, horizontally orients them and reserves L empty bins (L a lower bound on the optimal solution value, see [16]). The algorithm packs one item at a time (either in an existing bin, or by initializing a new one) in a so-called *normal position* (see Christofides and Whitlock [10]), i.e., with its bottom edge touching either the bottom of the bin or the top edge of another item, and with its left edge touching either the left edge of the bin or the right edge of another item. The choice of the bin and of the packing position is done by evaluating a *score*, defined as the percentage of the item perimeter which touches either the bin or other already packed items.

We finally mention that other variants of the two-dimensional bin packing problem can be found in the literature. For example, in guillotine cutting, an upper bound (usually two or three) may be imposed on the number of *stages* (rounds of cuts having the same direction) that are needed to obtain all the items: see, e.g., [31,40,41]. Note that all the level algorithms of Section 2 but FC produce two-stage packings (with trimming). In certain practical applications a secondary objective can also be of interest, namely the maximization of the unused area in one bin, so as to produce a possibly large trim to be used later: see, e.g., [6,18] for 2BP|R|F.

Note that the metaheuristic approach of Section 5 solves all of the above variants, by appropriately changing the deterministic algorithm used for evaluating the moves within the neighborhood search.

7. Conclusions and directions of research

We surveyed recent advances on exact algorithms and effective off-line heuristic and metaheuristic approaches to the two-dimensional bin packing problem, and to some variants typically arising in practical applications. The computational experiments show that many instances of small to moderate size can be solved to optimality, while larger instances can only be handled by approximation algorithms.

In order to increase the effectiveness of the exact approaches, especially those based on implicit enumeration, tighter lower bounds should be devised, capable of fathoming a larger number of branch-decision nodes. In addition, other exact methods, such as, for example, column generation techniques, may lead to interesting results. As far as approximate solutions are concerned, an important open problem is to find polynomial-time approximation schemes. A fully polynomial-time approximation scheme was recently developed, for the two-dimensional strip packing problem, by Kenyon and Rémila [35]. For 2BP, instead, to our knowledge, no polynomial-time approximation scheme is known.

Acknowledgements

We thank the Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST) and the Consiglio Nazionale delle Ricerche (CNR), Italy, for the support given to this project. The computational experiments have been executed at the Laboratory of Operations Research of the University of Bologna (Lab.O.R.). We are indebted to two anonymous referees for useful comments.

References

[1] E. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, John Wiley & Sons, Chichester, 1997.
[2] B.S. Baker, D.J. Brown, H.P. Katseff, A 5/4 algorithm for two-dimensional packing, J. Algorithms 2 (1981) 348–368.
[3] B.S. Baker, E.G. Coffman Jr., R.L. Rivest, Orthogonal packing in two dimensions, SIAM J. Comput. 9 (1980) 846–855.
[4] B.S. Baker, J.S. Schwarz, Shelf algorithms for two-dimensional packing problems, SIAM J. Comput. 12 (1983) 508–525.
[5] J.E. Beasley, An exact two-dimensional non-guillotine cutting tree search procedure, Oper. Res. 33 (1985) 49–64.
[6] B.E. Bengtsson, Packing rectangular pieces—a heuristic approach, Comput. J. 25 (1982) 353–357.
[7] J.O. Berkey, P.Y. Wang, Two dimensional finite bin packing algorithms, J. Oper. Res. Soc. 38 (1987) 423–429.
[8] D.J. Brown, An improved BL lower bound, Inform. Process. Lett. 11 (1980) 37–39.
[9] B. Chazelle, The bottom-left bin packing heuristic: An efficient implementation, IEEE Trans. on Comput. 32 (1983) 697–707.
[10] N. Christofides, C. Whitlock, An algorithm for two-dimensional cutting problems, Oper. Res. 25 (1977) 30–44.
[11] F.K.R. Chung, M.R. Garey, D.S. Johnson, On packing two-dimensional bins, SIAM J. Algebraic Discrete Meth. 3 (1982) 66–76.

[12] E.G. Coffman Jr., M.R. Garey, D.S. Johnson, R.E. Tarjan, Performance bounds for level-oriented two-dimensional packing algorithms, SIAM J. Comput. 9 (1980) 801–826.
[13] E.G. Coffman Jr., G.S. Lueker, Probabilistic Analysis of Packing and Partitioning Algorithms, John Wiley & Sons, Chichester, 1992.
[14] E.G. Coffman Jr., P.W. Shor, Packings in two dimensions: Asymptotic average-case analysis of algorithms, Algorithmica 9 (1993) 253–277.
[15] J. Csirik, G. Woeginger, On-line packing and covering problems, in: Online algorithms, Lecture Notes in Computer Science, eds. F. Amos and G. Woeginger, Vol. 144, Springer, Berlin, 1998, pp. 147–177.
[16] M. Dell'Amico, S. Martello, D. Vigo, A lower bound for the non-oriented two-dimensional bin packing problem, Discrete Appl. Math. 2002, to appear.
[17] M. Dell'Amico, S. Martello, Optimal scheduling of tasks on identical parallel processors, ORSA J. Comput. 7 (1995) 191–200.
[18] A. El-Bouri, N. Popplewell, S. Balakrishnan, A. Alfa, A search based heuristic for the two-dimensional bin-packing problem, INFOR 32 (1994) 265–274.
[19] S.P. Fekete, J. Schepers, On more-dimensional packing I: Modeling, Technical Report ZPR97-288, Mathematisches Institut, Universität zu Köln, 1997.
[20] S.P. Fekete, J. Schepers, On more-dimensional packing II: Bounds, Technical Report ZPR97-289, Mathematisches Institut, Universität zu Köln, 1997.
[21] S.P. Fekete, J. Schepers, On more-dimensional packing III: Exact algorithms, Technical Report ZPR97-290, Mathematisches Institut, Universität zu Köln, 1997.
[22] S.P. Fekete, J. Schepers, New classes of lower bounds for bin packing problems, in: Integer Programming and Combinatorial Optimization (IPCO 98), Lecture Notes in Computer Science, eds. R.E. Bixby, E.A. Boyd and R.Z. Ríos-Mercado, Vol. 1412, Springer, Berlin, 1998, pp. 257–270.
[23] J.B. Frenk, G.G. Galambos, Hybrid next-fit algorithm for the two-dimensional rectangle bin-packing problem, Computing 39 (1987) 201–217.
[24] P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting stock problem, Oper. Res. 9 (1961) 849–859.
[25] P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting stock problem—part II, Oper. Res. 11 (1963) 863–888.
[26] P.C. Gilmore, R.E. Gomory, Multistage cutting problems of two and more dimensions, Oper. Res. 13 (1965) 94–119.
[27] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Boston, 1997.
[28] I. Golan, Performance bounds for orthogonal oriented two-dimensional packing algorithms, SIAM J. Comput. 10 (1981) 571–582.
[29] E. Hadjiconstantinou, N. Christofides, An exact algorithm for general, orthogonal, two-dimensional knapsack problems, Eur. J. Oper. Res. 83 (1995) 39–56.
[30] E. Hadjiconstantinou, N. Christofides, An exact algorithm for the orthogonal, 2-D cutting problems using guillotine cuts, Eur. J. Oper. Res. 83 (1995) 21–38.
[31] M. Hifi, Exact algorithms for large-scale unconstrained two and three staged cutting problems, In Contribution à la Résolution de Quelques Problèmes Difficiles de l'Optimisation Combinatoire, Thèse d'Habilitation à Diriger des Recherches en Informatique, Université de Versailles–Saint Quentin en Yvelines, 1998.
[32] S. Høyland, Bin-packing in 1.5 dimension, In Proceedings of the Scandinavian Workshop on Algorithm Theory, Lecture Notes in Computer Science, Vol. 318, Springer, Berlin, 1988, pp. 129–137.
[33] D.S. Johnson, Near-optimal bin packing algorithms, PhD Thesis, MIT, Cambridge, MA, 1973.
[34] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, R.L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM J. Comput. 3 (1974) 299–325.
[35] C. Kenyon, E. Rémila, A near-optimal solution to a two-dimensional cutting stock problem, Math. Oper. Res. 25 (2000) 645–656.
[36] K. Lagus, I. Karanta, J. Ylä-Jääski, Paginating the generalized newspaper: A comparison of simulated annealing and a heuristic method. Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature, Berlin, 1996, pp. 549–603.
[37] A. Lodi, S. Martello, D. Vigo, Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem, in: S. Voss, S. Martello, I.H. Osman, C. Roucairol (Eds.),

Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers, Boston, 1998, pp. 125–139.
[38] A. Lodi, S. Martello, D. Vigo, Approximation algorithms for the oriented two-dimensional bin packing problem, Eur. J. Oper. Res. 112 (1999) 158–166.
[39] A. Lodi, S. Martello, D. Vigo, Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems, INFORMS J. Comput. 11 (1999) 345–357.
[40] A. Lodi, S. Martello, D. Vigo, Heuristic algorithms for the three-dimensional bin packing problem, Eur. J. Oper. Res. 2002, to appear.
[41] A. Lodi, M. Monaci, Integer linear programming models for two-staged cutting stock problems, Technical Report OR/00/12, DEIS-Università di Bologna, 2000, to appear in Mathematical Programming.
[42] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Chichester, 1990.
[43] S. Martello, D. Vigo, Exact solution of the two-dimensional finite bin packing problem, Manage. Sci. 44 (1998) 388–399.
[44] W. Schneider, Trim-loss minimization in a crepe-rubber mill; optimal solution versus heuristic in the 2 (3)-dimensional case, Eur. J. Oper. Res. 34 (1988) 273–281.
[45] D. Sleator, A 2.5 times optimal algorithm for packing in two dimensions, Inform. Process. Lett. 10 (1980) 37–40.
[46] A. Steinberg, A strip-packing algorithm with absolute performance bound 2, SIAM J. Comput. 26 (1980) 401–409.
[47] F.J. Vasko, F.E. Wolf, K.L. Stott, A practical solution to a fuzzy two-dimensional cutting stock problem, Fuzzy Sets and Systems 29 (1989) 259–275.