Exact Solution of the Two-Dimensional Finite Bin Packing Problem

Silvano Martello • Daniele Vigo
Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna, Bologna, Italy

Given a set of rectangular pieces to be cut from an unlimited number of standardized stock pieces (bins), the Two-Dimensional Finite Bin Packing Problem is to determine the minimum number of stock pieces that provide all the pieces. The problem is NP-hard in the strong sense and finds many practical applications in the cutting and packing area. We analyze a well-known lower bound and determine its worst-case performance. We propose new lower bounds which are used within a branch-and-bound algorithm for the exact solution of the problem. Extensive computational testing on problem instances from the literature involving up to 120 pieces shows the effectiveness of the proposed approach.
(*Cutting and Packing; Branch-and-bound; Worst-case Analysis*)

1. Introduction

We are given a set of n rectangular *pieces* which must be cut from a large set of standardized stock pieces, also called (*finite*) *bins*, having height H and width W. Each piece $j \in J = \{1, \ldots, n\}$ is characterized by its height $h_j \leq H$ and width $w_j \leq W$, and may not be rotated. A piece will alternatively be denoted by $h_j \times w_j$ (or $H \times W$). The *Two-Dimensional Finite Bin Packing Problem* (2D-BPP) calls for the determination of cutting patterns which provide all the n pieces and such that the total number of required stock pieces is minimized. There is no restriction on the cutting patterns and, in particular, the guillotine-cut constraint (see, e.g., Christofides and Whitlock 1977) is not imposed.

The problem is a generalization of the well-known (one-dimensional) *Bin Packing Problem* (BPP): here we have an unlimited number of bins with capacity W and n items, each characterized by a weight w_j, and we want to pack all the items into the minimum number of bins. It is evident that BPP is the special case of 2D-BPP arising when $h_j = H$ for all $j \in J$. Hence 2D-BPP is NP-hard in the strong sense, since BPP is such.

Previous papers on the problem have been devoted to the development of heuristic algorithms (see, e.g., Berkey and Wang 1987 and Frenk and Galambos 1987, or, for the case where 90° rotation of the pieces is allowed, Bengtsson 1982 and El-Bouri et al. 1994). A considerable amount of work has also been done on a variant of 2D-BPP, known as the *Two-Dimensional Strip Packing Problem*, in which there is a single bin of given height and infinite width, and the problem is to determine a cutting pattern providing all the pieces and minimizing the width to which the bin is filled (see, e.g., Baker et al. 1980, Coffman et al. 1980, Bartholdi et al. 1989 and Coffman and Shor 1990). Finally, extensive surveys on cutting and packing problems can be found in Dyckhoff and Finke (1992) and Dowsland and Dowsland (1992), while an annotated bibliography has been presented by Dyckhoff et al. (1997).

The main results presented in this paper concern lower bounds for 2D-BPP and their use for the optimal solution of the problem. In the next section we consider the well-known continuous lower bound and determine its worst-case performance ratio: our result holds even if the pieces may be rotated by any angle. In §3 we present a lower bound that can be computed in linear time and is obtained by generalizing the bounds proposed by Martello and Toth (1990b) for BPP and by Dell'Amico and Martello (1995) for $P\|C_{max}$ (a scheduling problem directly connected to BPP). More accurate lower bounds, which explicitly take into account the two dimensions of the pieces, are introduced in §4.

0025-1909/98/4403/0388$05.00

Dominances between the bounds are discussed. In §5 we present a branch-and-bound algorithm based on the previous results, and in §6 we analyze its performance through extensive computational testing on problem instances from the literature involving up to 120 pieces.

In the following we will denote by z the optimal solution value of 2D-BPP and will assume, without loss of generality, that all input data are positive integers.

2. The Continuous Lower Bound

To our knowledge the only known lower bound for 2D-BPP is obtained by allowing each piece to be cut into any number of smaller pieces. This leads to the so-called *continuous lower bound*, L_0, which can be computed in $O(n)$ time:

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

In this section we determine the worst-case performance of L_0. Given any instance I of a minimization problem P, let $z(I)$ be the value of the optimal solution to I and $L(I)$ be the value provided by a lower bound L. The *worst-case performance ratio* of L is defined as the largest real number ρ such that

$$\frac{L(I)}{z(I)} \geq \rho \quad \text{for all instances } I \text{ of } P.$$

THEOREM 1. *The worst-case performance ratio of lower bound L_0 is $\frac{1}{4}$.*

PROOF. We show that $z \leq 4L_0$ by describing an algorithm that determines a feasible solution using no more than $4L_0$ bins. The algorithm consists of two phases: first we pack all the pieces into a strip of height H, in such a way that the total occupied area before a certain width $\bar{x}$ is no less than $\bar{x}H/2$; then we derive a feasible finite bin solution.

Phase 1. Define an (x, y) coordinate system with the origin in the bottom left corner of the strip as in Figure 1. Initialize $\bar{x}$ to 0. Sort the pieces according to decreasing h_j values, and associate with each piece j a *class*, defined as follows: $class(j) = r$ (with r integer in the range $[0, \lceil \log_2 H - 1 \rceil]$) implies that

$$\frac{1}{2^r} H > h_j \geq \frac{1}{2^{r+1}} H, \tag{2}$$

but assign $class(j) = 0$ also for the pieces with $h_j = H$. A piece j having $class(j) = r$ will be placed only with its bottom edge at one of 2^r *admissible* vertical coordinates, namely: $(t/2^r)H$ $(t = 0, 1, \ldots, 2^r - 1)$.

At each iteration the next piece is placed with its bottom left corner in $(\bar{x}, \bar{y})$, where $\bar{y}$ is the lowest unoccupied admissible position, at coordinate $\bar{x}$, for piece j. Whenever no admissible position is available for the current piece, $\bar{x}$ is advanced to the leftmost coordinate $\bar{x}' \geq \bar{x}$ corresponding to the right edge of an already packed piece. In other words, as illustrated in Figure 1, whenever a new $\bar{x}$ is determined, we know that there is a piece j^*, previously packed, that has its bottom right corner at a coordinate $(\bar{x}, \bar{y})$. The next piece j (assume $class(j) = r$) is packed with its bottom left corner in $(\bar{x}, \bar{y})$ (this is always possible since $class(j^*) \leq r$): if $class(j^*) = r$, a new $\bar{x}$ must be determined; otherwise, the next piece(s) are placed at the current $\bar{x}$ in admissible positions in the interval $]\bar{y}, \bar{y} + \frac{1}{2^{r^*}}H[$, where $r^* = class(j^*)$.

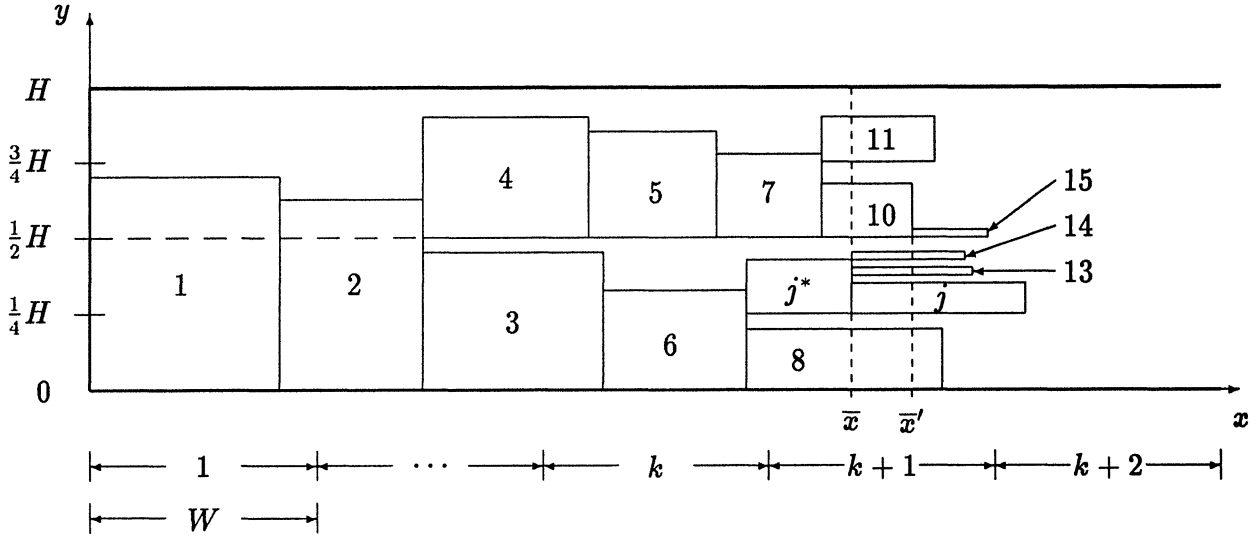
To see that the total occupied area before $\bar{x}$ is always no less than $\bar{x}H/2$, consider any iteration in which no admissible position is available at $\bar{x}$ for the current piece of class r (see Figure 1, assuming that "15" is the current piece). Consider the 2^r admissible y coordinates on the vertical segment $S = [(\bar{x}, 0), (\bar{x}, H)]$, and observe that: (i) for each admissible coordinate occupied by a piece of class r (pieces 13 and 14 in Figure 1), a segment of length at least $\frac{1}{2^{r+1}}H$ is occupied (see (2)); (ii) each piece of class $u < r$ continuing after segment S (pieces 8, j, 10, 11 in Figure 1) occupies 2^{r-u} admissible coordinates, hence occupying a segment of length at least $(2^{r-u} - 1)\frac{1}{2^r}H$, i.e., at least

$$((2^{r-u} - 1)/2^{r-u})\frac{1}{2^r} H \geq \frac{1}{2^{r+1}} H$$

per occupied coordinate; (iii) since all the 2^r admissible coordinates are occupied, the occupied portion of S is at least $2^r \frac{1}{2^{r+1}}H = \frac{1}{2}H$. Hence, whenever $\bar{x}$ is moved to $\bar{x}'$, we know that the portion of strip between $\bar{x}$ and $\bar{x}'$ is occupied by (parts of) pieces having a total area at least equal to $(\bar{x}' - \bar{x})H/2$.

Phase 2. When all the pieces have been packed into the strip, the occupied length of the strip is subdivided into numbered *slices* of width W, starting from the ori-

Figure 1 Example of the Strip Packing for the Proof of Theorem 1



gin. Let $\bar{x}'$ denote the final $\bar{x}$ value determined in Phase 1, and k the index of the rightmost slice not terminating after $\bar{x}'$ (see Figure 1): we have already proved that the occupied area within the first k slices is at least $kWH/2$. Observe that no piece can have its right edge after coordinate $\bar{x}' + W$, so the total number of slices is bounded by $k + 2$. A feasible solution for the finite bin packing that uses no more than $2k + 2$ bins is then built as follows.

For each slice i ($i = 1, \ldots, k$) we create two bins (at most): one for the pieces entirely contained in slice i, and one for the pieces placed across the boundary between slices i and $i + 1$. Finally, we add one or two bins for the remaining pieces: one for those terminating after $\bar{x}'$ and, possibly, one for the remaining pieces of slice $k + 1$. (For the example illustrated in Figure 1, where $k = 3$, we would use $2 + 1 + 2 + 2$ bins, and their contents would be, respectively: {1}, {2}, {3, 4}, {5, 6}, {7, 8, j^*}, {11, j, 13, 14, 15}, {10}.) If the feasible solution uses only $2k$ bins (i.e., none of the two additional bins is needed), by observing that the total area of the pieces is at least $kWH/2$, we have

$$L_0 \geq \left\lceil \frac{kWH/2}{WH} \right\rceil = \left\lceil \frac{k}{2} \right\rceil \geq \frac{z}{4}, \qquad (3)$$

otherwise we know that the total area of the pieces is strictly greater than $kWH/2$, hence

$$L_0 \geq \left\lceil \frac{kWH/2 + 1/2}{WH} \right\rceil \geq \frac{k}{2} + \frac{1}{2} = \frac{2k + 2}{4} \geq \frac{z}{4}. \qquad (4)$$

We conclude the proof by showing that the bound is tight. Consider indeed an instance in which n is a multiple of four and, for each $j \in J$, $w_j = \frac{W}{2} + \epsilon$, $h_j = \frac{H}{2} + \epsilon$, for some small positive ϵ. We have $z = n$. If we denote with $\frac{HW}{4} + \delta$ the area of a piece, and assume $n\delta < HW$, we obtain

$$L_0 = \left\lceil \frac{n}{4} + \frac{n\delta}{HW} \right\rceil = \frac{n}{4} + 1.$$

Hence, the ratio L_0/z is arbitrarily close to $\frac{1}{4}$ for sufficiently large n. □

Other versions of the problem considered in the literature admit rotations of the pieces. It is worth noting that the above proof that $z \leq 4L_0$ obviously holds for such cases too, and that the instance used to prove tightness of the bound remains valid if $H = W$. Hence

COROLLARY 1. *The worst-case performance ratio of lower bound L_0 is $\frac{1}{4}$ even if rotation of the pieces (by any angle) is allowed.*

3. Lower Bounds from the One-Dimensional Problem

In this section we present a lower bound for 2D-BPP, which is obtained by generalizing the bounds proposed

by Martello and Toth (1990b) for BPP and by Dell'Amico and Martello (1995) for $P\|C_{\max}$, a scheduling problem strongly related to BPP.

Let $J^W = \{j \in J : w_j > \frac{1}{2}W\}$ and observe that no two pieces of J^W may be packed side by side into a bin. Given any integer p, with $1 \le p \le \frac{1}{2}H$, let

$$J_1 = \{j \in J^W : h_j > H - p\}, \tag{5}$$

$$J_2 = \{j \in J^W : H - p \ge h_j > \tfrac{1}{2}H\}, \tag{6}$$

and note that no two pieces of $J_1 \cup J_2$ may be packed into the same bin, so $|J_1 \cup J_2|$ is a valid lower bound (independent of p) on z. Now let

$$J_3 = \{j \in J^W : \tfrac{1}{2}H \ge h_j \ge p\}. \tag{7}$$

The lower bound can be strengthened by observing that no piece of J_3 fits into a bin used for a piece of J_1. Hence, we can prove the following.

THEOREM 2. *Given any integer p, with $1 \le p \le \frac{1}{2}H$, a valid lower bound on z is*

$$L_1^W(p) = \max\{L_\alpha^W(p), L_\beta^W(p)\}, \quad \textit{where} \tag{8}$$

$$L_\alpha^W(p) = |J_1 \cup J_2| + \max\left\{0, \left\lceil \frac{\Sigma_{j\in J_3} h_j - (|J_2|H - \Sigma_{j\in J_2} h_j)}{H} \right\rceil\right\}, \tag{9}$$

$$L_\beta^W(p) = |J_1 \cup J_2| + \max\left\{0, \left\lceil \frac{|J_3| - \Sigma_{j\in J_2} \left\lfloor \frac{H - h_j}{p} \right\rfloor}{\left\lfloor \frac{H}{p} \right\rfloor} \right\rceil\right\}, \tag{10}$$

and J_1, J_2, J_3 are given by (5)–(7).

PROOF. Both $L_\alpha^W(p)$ and $L_\beta^W(p)$ are obtained by adding to $|J_1 \cup J_2|$ the minimum number of additional bins needed for the pieces of J_3. In $L_\alpha^W(p)$ this is computed by determining the total height of such pieces and decreasing it by the total free height available in the bins used for the pieces of J_2: the resulting value is then divided by H, i.e., a continuous relaxation of the heights is considered. In $L_\beta^W(p)$ the computation is performed by considering the total number of pieces of J_3 and decreasing it by an upper bound on the number of such pieces that could be placed into the bins used for the pieces of J_2: the resulting value is then divided by an upper bound on the number of pieces of J_3 that can fit into a bin. □

COROLLARY 2. *A valid lower bound on z is*

$$L_1^W = \max_{1\le p\le (1/2)H} \{L_1^W(p)\}, \tag{11}$$

and can be computed in $O(n^2)$ time.

PROOF. The validity of the bound is immediate from Theorem 2. The time complexity for the computation of L_1^W is apparently pseudo-polynomial. However, the restriction induced by J^W implicitly defines a one-dimensional problem and it is known (see Martello and Toth 1990b and Dell'Amico and Martello 1995) that in (11) only the values of p corresponding to actual distinct piece heights must be considered (i.e., at most n distinct values). Thus the overall computation of L_1^W can be performed in $O(n^2)$ time, since $L_\alpha^W(p)$ and $L_\beta^W(p)$ can be determined in $O(n)$ time. □

Now let $J^H = \{j \in J : h_j > \frac{1}{2}H\}$. It is clear that the above results immediately produce an analogous lower bound

$$L_1^H = \max_{1\le p\le (1/2)W} \{L_\alpha^H(p), L_\beta^H(p)\}, \tag{12}$$

where $L_\alpha^H(p)$ and $L_\beta^H(p)$ are obtained from (5)–(11) by replacing J^W with J^H, h_j with w_j, and H with W. Thus, an overall lower bound can be computed in $O(n^2)$ time as

$$L_1 = \max\{L_1^W, L_1^H\}. \tag{13}$$

It is easily seen that no dominance relation exists between L_0 and L_1. Consider indeed the instance introduced in the proof of Theorem 1, for which $L_0 = \frac{n}{4} + 1$. Since $J_1 \cup J_2 = J^W = J^H = J$, we have $L_1 = n = z > L_0$. Now consider the same instance with ϵ replaced by $-\epsilon$ (with $\epsilon < \min\{H/3, W/3\}$), so that the area of a piece is $\frac{HW}{4} - \delta'$ (and $\delta' < \delta$): we have in this case $z = \frac{n}{4}$, $L_1 = 0$ (since $J^W = J^H = \varnothing$), and

$$L_0 = \left\lceil \frac{n}{4} - \frac{n\delta'}{HW} \right\rceil = \frac{n}{4} > L_1.$$

This also shows that the worst-case performance of L_1 can be arbitrarily bad.

4. Better Bounds

In this section we present lower bounds which explicitly take into account both dimensions of the pieces. These bounds dominate the previous ones, but their computation is more time-consuming.

THEOREM 3. *Given an integer value* q, $1 \le q \le \frac{1}{2}W$, *let*

$$K_1 = \{j \in J : w_j > W - q\}, \tag{14}$$

$$K_2 = \{j \in J : W - q \ge w_j > \tfrac{1}{2}W\}, \tag{15}$$

$$K_3 = \{j \in J : \tfrac{1}{2}W \ge w_j \ge q\}. \tag{16}$$

Then a valid lower bound on z *is*

$$L_2^W(q) = L_1^W + \max\left\{0, \left\lceil \frac{\sum_{j\in K_2\cup K_3} h_j w_j - (HL_1^W - \sum_{j\in K_1} h_j)W}{HW} \right\rceil\right\}. \tag{17}$$

PROOF. First observe that $K_1 \cup K_2 \equiv J^W$ (see §3) and is independent of q. Hence a valid lower bound on the number of bins needed for the pieces in $K_1 \cup K_2$ is given by L_1^W. We can tighten this value by considering the pieces in K_3 and observing that none of them can be packed beside a piece of K_1. Thus the total area of the L_1^W bins that is available for these pieces is equal to HWL_1^W decreased by the unavailable area due to the pieces of K_1 and by the area occupied by the pieces of K_2. A bound on the number of additional bins required is then

$$\max\left\{0, \left\lceil \frac{\sum_{j\in K_3} h_j w_j - (HWL_1^W - (W\sum_{j\in K_1} h_j + \sum_{j\in K_2} h_j w_j))}{HW} \right\rceil\right\}, \tag{18}$$

from which the thesis is derived. □

COROLLARY 3. *A valid lower bound on* z *is*

$$L_2^W = \max_{1\le q\le (1/2)W} \{L_2^W(q)\}, \tag{19}$$

and can be computed in $O(n^2)$ *time.*

PROOF. The validity of (19) comes from Theorem 3. As to the time complexity, observe that L_1^W can be computed once (in $O(n^2)$ time, see Corollary 2) and the remaining part of (17) requires $O(n)$ time for each value of q. We now prove that only the values of q corresponding to distinct w_j values need be considered in (19). Let q_1 and q_2 (with $q_1 < q_2$) be two values of q inducing the same set K_3 and observe that increasing q from q_1 to q_2 may cause one or more pieces to move from K_2 to K_1. For any such piece i, the value of the numerator of the fraction in (17) is increased by $h_iW - h_iw_i \ge 0$, so we have $L_2^W(q_1) \le L_2^W(q_2)$. Hence the thesis, since we know that only distinct values $q = w_j \le W/2$ induce different sets K_3 and all the remaining values of q produce dominated lower bounds. □

In this case too, a "symmetric" result can be obtained, so

COROLLARY 4. *A valid lower bound on* z *is*

$$L_2 = \max\{L_2^W, L_2^H\}, \tag{20}$$

where L_2^H *is obtained from* (14)–(19) *by replacing* W *with* H, w_j *with* h_j, *and* L_1^W *with* L_1^H.

It is worth noting that lower bounds L_1 and L_2 have the same worst-case time complexity, but the computation of L_2^W (resp. L_2^H) requires the value of L_1^W (resp. L_1^H). Thus, the average computing time required for L_2 is about twice the time required for L_1. However, L_2 clearly dominates L_1 (see (17)), and it is easily shown that also L_0 is dominated by the new bound:

THEOREM 4. *Lower bound* L_2 *dominates both* L_0 *and* L_1.

PROOF. If we consider $q = 1$ we have $K_1 = \{j \in J : w_j = W\}$ and $K_1 \cup K_2 \cup K_3 = J$. Hence, from (17)

$$L_2^W(1) = L_1^W + \max\left\{0, \left\lceil \frac{\sum_{j\in J} h_j w_j - HWL_1^W}{HW} \right\rceil\right\}$$

$$= \max\left\{L_1^W, \left\lceil \frac{\sum_{j\in J} h_j w_j}{HW} \right\rceil\right\}, \tag{21}$$

so $L_2^W \ge L_2^W(1) = \max(L_1^W, L_0)$. In an analogous way we obtain $L_2^H \ge \max(L_1^H, L_0)$. The thesis follows. □

We finally describe a lower bound which is computationally more expensive, but can in some cases improve the previous one. Given any pair of integers (p, q), with $1 \le p \le \frac{1}{2}H$ and $1 \le q \le \frac{1}{2}W$, define:

$$I_1 = \{j \in J : h_j > H - p \quad \text{and} \quad w_j > W - q\}, \tag{22}$$

$$I_2 = \{j \in J \backslash I_1 : h_j > \tfrac{1}{2}H \quad \text{and} \quad w_j > \tfrac{1}{2}W\}, \tag{23}$$

$$I_3 = \{j \in J : \tfrac{1}{2}H \geq h_j \geq p \quad \text{and} \quad \tfrac{1}{2}W \geq w_j \geq q\}. \tag{24}$$

Observe that $I_1 \cup I_2$ is independent of (p, q), that no two pieces of $I_1 \cup I_2$ may be packed into the same bin, and that no piece of I_3 fits into a bin containing a piece of I_1 (see Figure 2). A valid lower bound can thus be computed by adding to $|I_1 \cup I_2|$ the minimum number of bins needed for those pieces of I_3 that cannot be packed into the bins used for the pieces of I_2. We will determine a lower bound on the latter value by considering a relaxed instance where each piece $i \in I_3$ has the minimum size, i.e., $h_i = p$ and $w_i = q$.

LEMMA 1. *Given a bin $H \times W$ containing a piece $h_j \times w_j$, the maximum number of $p \times q$ pieces that can be packed into the bin is*

$$m(j, p, q) = \left\lfloor \frac{H}{p} \right\rfloor \left\lfloor \frac{W - w_j}{q} \right\rfloor + \left\lfloor \frac{W}{q} \right\rfloor \left\lfloor \frac{H - h_j}{p} \right\rfloor - \left\lfloor \frac{H - h_j}{p} \right\rfloor \left\lfloor \frac{W - w_j}{q} \right\rfloor. \tag{25}$$

PROOF. Consider an optimal pattern containing piece $h_j \times w_j$ and the maximum number of $p \times q$ pieces. Christofides and Whitlock (1977) noted that an equivalent pattern (called *normal pattern*) exists in which all pieces have their left and bottom edges both adjacent to another piece (or to the bin), as in Figure 3(a). We observe that, given a normal pattern, we can build an equivalent pattern in which piece j is placed at the bottom left corner of the bin. Indeed, this can be obtained by: (i) removing all the rows of $p \times q$ pieces that are below piece j (marked with an asterisk in Figure 3); (ii) shifting down the residual pattern until it is adjacent to the bottom of the bin; (iii) inserting the removed rows at the top of the bin, and by performing similar operations for the columns of $p \times q$ pieces which are to the left of piece j (marked with a dot in Figure 3). The resulting pattern can be further modified by shifting all the $p \times q$ pieces up and right as much as possible (see Figure 3(b)). The number of $p \times q$ pieces in the optimal pattern can then be easily computed as

$$\left\lfloor \frac{W}{q} \right\rfloor \left\lfloor \frac{H - h_j}{p} \right\rfloor + \left\lfloor \frac{W - w_j}{q} \right\rfloor \left\lfloor \frac{H - \left\lfloor \frac{H - h_j}{p} \right\rfloor p}{p} \right\rfloor, \tag{26}$$

where the first term indicates the number of $p \times q$ pieces that are packed in the rows above piece j, and the second term considers the pieces that are packed in the rows beside piece j. From (26) one easily obtains (25). □

THEOREM 5. *Given a pair of integers (p, q), with $1 \leq p \leq \frac{1}{2}H$ and $1 \leq q \leq \frac{1}{2}W$, a valid lower bound on z is*

$$L_3(p, q) = |I_1 \cup I_2| + \max \left\{ 0, \left\lceil \frac{|I_3| - \Sigma_{j \in I_2} m(j, p, q)}{\left\lfloor \frac{H}{p} \right\rfloor \left\lfloor \frac{W}{q} \right\rfloor} \right\rceil \right\}, \tag{27}$$

where I_1, I_2, I_3 and $m(j, p, q)$ are defined by (22)–(25).

PROOF. Immediate from the properties of I_1, I_2, I_3 and Lemma 1. □

COROLLARY 5. *A valid lower bound on z is*

$$L_3 = \max_{1 \leq p \leq (1/2)H, 1 \leq q \leq (1/2)W} \{L_3(p, q)\}, \tag{28}$$

and can be computed in $O(n^3)$ time.

PROOF. Theorem 5 proves the validity of the bound. Observe that the value of $L_3(p, q)$ can be computed in $O(n)$ time for each pair (p, q). Now let (p_1, q_1) and (p_2, q_2) (with $p_1 \leq p_2$ and $q_1 \leq q_2$) be two (p, q) pairs inducing the same set I_3 and note that $L_3(p_1, q_1)$ and $L_3(p_2, q_2)$ may differ only if one or more pieces which belong to I_2 in $L_3(p_1, q_1)$, belong to I_1 in $L_3(p_2, q_2)$. For any such piece i, the value of the numerator of the fraction in (27) is increased by $m(i, p, q)$, so we have $L_3(p_1, q_1) \leq L_3(p_2, q_2)$. Thus, only the values of p (resp. q) corresponding to distinct h_j (resp. w_j) values need be considered in (28), since these are the only values which induce different sets I_3, and all the remaining values of (p, q) produce dominated lower bounds. The thesis follows. □

No dominance relation exists between L_2 and L_3. Consider the instance with $n = 5$, $H = 10$, $W = 20$, $h_1 = 8$,

Figure 2 **Example of Pieces Belonging to Sets I_1, I_2, and I_3**

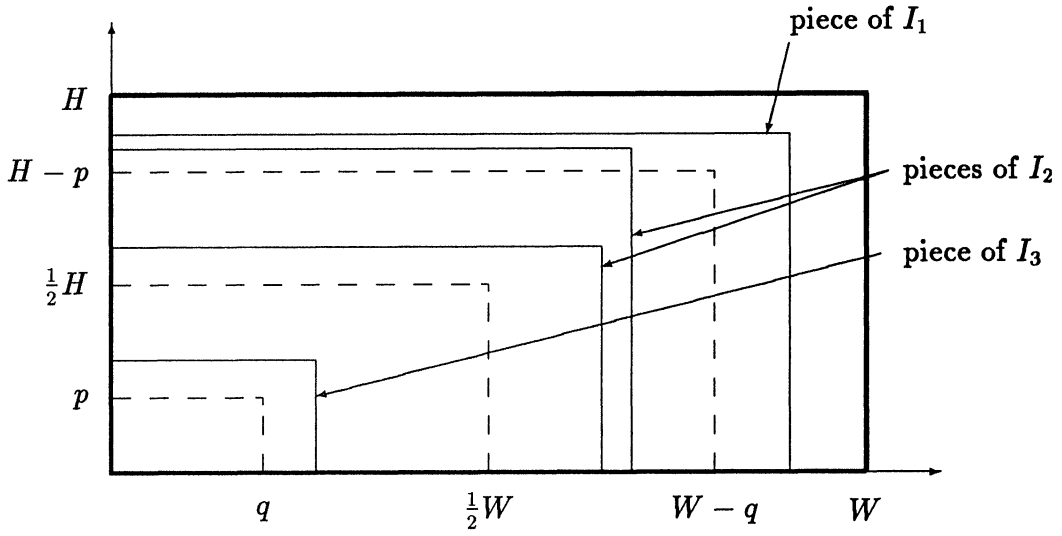
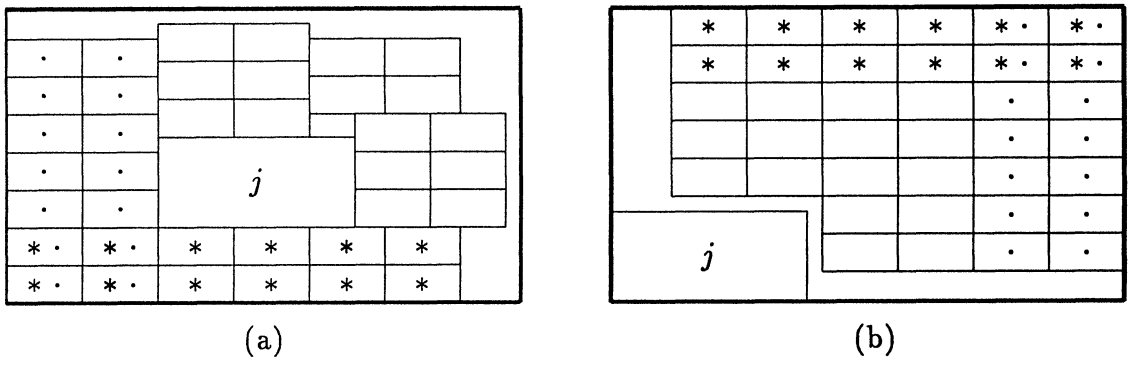


Figure 3 **An Example of Normal Pattern (a) and Equivalent Pattern (b) used in the Proof of Lemma 1**



$w_1 = 16$, and $h_j = w_j = 3$ for $j = 2, \ldots, 5$: we have $L_2 = 1$ and $L_3 = z = 2$. If, instead, we consider the instance with $n = 16$, $H = 10$, $W = 20$, $h_1 = 8$, $w_1 = 16$, $h_j = w_j = 3$ for $j = 2, 3, 4$, and $h_j = w_j = 2$ for $j = 5, \ldots, 16$, we have $L_2 = z = 2$ and $L_3 = 1$.

The overall lower bound is then:

$$L_4 = \max\{L_2, L_3\}. \tag{29}$$

5. Exact Algorithm

The lower bounds of the previous sections have been embedded into a branch-and-bound algorithm for the exact solution of 2D-BPP. The algorithm adopts a nested branching scheme. At each decision-node of an *outer branch-decision tree*, a piece k is assigned to a bin. If a feasible packing of k and all the pieces currently assigned to such a bin can be determined heuristically, the search continues. Otherwise, the problem of establishing whether such a feasible packing exists is solved through an *inner branch-decision tree*: if the answer is yes, the search (in the outer tree) continues; otherwise, a backtracking occurs. Let z^* denote the best incumbent solution value. The algorithm starts by initializing z^* through heuristic algorithms FFF, FBS, and FBS' described in §5.3.

5.1. Outer Tree

The outer branch-decision tree is based on a depth-first branching scheme. The pieces are initially sorted in non-

increasing order of their area and renumbered accordingly. The bins are numbered according to the order in which they are initialized. Whenever it is possible to establish that no more unassigned pieces can be assigned to a given initialized bin, such a bin is *closed*: an initialized and not closed bin is called *active*. At each level k ($k = 1, \ldots, n$), piece k is considered and assigned, in turn, to all active bins and, possibly, to a new bin.

Let us consider a decision node in which piece k is assigned to an active bin i: let I denote the instance defined by piece k and all the pieces currently assigned to bin i. The node is explored as follows.

Step 1. The feasibility of the packing is first heuristically checked in the following way. Lower bound L_4 is computed for I: if $L_4 > 1$, then the insertion is not possible, so a backtracking is performed. Otherwise, the heuristic algorithms FFF and FBS in §5.3 are applied to I: if a feasible packing of the pieces of I into a single bin is not found, the inner branching scheme (see §5.4) determines whether such a packing exists. If the answer is "no," a backtracking is executed; otherwise.

Step 2. The possibility of closing bin i is checked by computing lower bound L_4 for the instances defined by $I \cup \{h\}$ ($h = k + 1, \ldots, n$): if all these bounds are greater than 1 we conclude that no more pieces can be added to the bin, hence we close it. If the bin is not closed, the node exploration ends; otherwise.

Step 3. Lower bound L_4 is computed for the instance I' defined by the unassigned pieces and all the pieces currently assigned to active bins: if $L_4 + c \geq z^*$ (where c denotes the number of currently closed bins) we backtrack. (No bound is computed when bin i is not closed, as it would result equal to that computed for the father node.) The node exploration is concluded in this case by executing the heuristics FFF and FBS in §5.3, for instance I', in order to possibly improve the incumbent solution.

When, at the current level k, the total number of active and closed bins is less than $z^* - 1$, an additional decision node is generated by assigning piece k to a new bin. The exploration of this type of nodes is simpler, since we know that the packing is always feasible, so only Steps 2 and (possibly) 3 are executed.

5.2. Reduction

At the root node of the outer tree a procedure, derived from that presented by Martello and Toth (1990b) for BPP, tries to determine the optimal packing of some bins, thus reducing the size of the instance.

It should be remembered that the pieces are sorted according to decreasing area. For increasing $j \in J$ the following steps are executed:

1. Let $Q = \{i \in J : h_i + h_j \leq H \text{ or } w_i + w_j \leq W\}$ be the set of pieces *compatible* with piece j.
2. If $Q = \emptyset$ then pack j alone into a bin, set $J = J \setminus \{j\}$, and iterate. Otherwise.
3. Determine an upper bound on the number of pieces that can be packed with j as the maximum value l such that the sum of the areas of the l last (smallest) pieces of Q does not exceed $HW - h_j w_j$.
4. Let i^* be the first (largest) item in Q. Reduction occurs in two cases:
   (a) $l = 1$ and, for all $i \in Q \setminus \{i^*\}$, $h_{i^*} \geq h_i$ and $w_{i^*} \geq w_i$;
   (b) $l = 2$ and all pairs of pieces in $Q \setminus \{i^*\}$ that can be packed with j can also be packed into a bin of size $h_{i^*} \times w_{i^*}$.

In both cases, in fact, it is easily seen that i^* is the best possible piece to be packed with j. Hence pack j and i^* into a bin, set $J = J \setminus \{j, i^*\}$, and iterate.

The criterion used in Step 4 could obviously be extended by considering, for $l = 3$, all the triplets of pieces in Q, and so on. This, however, would be computationally too heavy, since the current procedure already requires $O(n^3)$ time (see Step 4(b)).

5.3. Heuristic Algorithms

At each node of the outer tree we have used two fast and effective approximation algorithms from the literature (see Berkey and Wang 1987). The first heuristic, *Finite First-Fit* (FFF), initially sorts the pieces by nonincreasing width. The algorithm then iteratively considers each piece in turn, and packs it into the first initialized bin into which it fits, if any, or into a new bin. The worst-case time complexity of algorithm FFF is $O(n^2)$. The second heuristic, *Finite Best Strip* (FBS), initially sorts the pieces by nonincreasing height and consists of two phases. In the first phase the pieces are packed into an infinite height strip using a best-fit algorithm to select the level for each piece: the resulting strip packing is

made up of blocks, each corresponding to a different level, having width equal to the strip width and a different height. In the second phase the blocks are packed into finite bins using a best-fit heuristic for the BPP. The worst-case time complexity of FBS is $O(n \log n)$.

We have also considered a variant of algorithm FBS, called FBS', in which the packing of the second phase is obtained by using FORTRAN code MTP (included in the diskette accompanying the book by Martello and Toth 1990a) for the exact solution of BPP, by limiting to 5000 the number of backtrackings allowed: within this limit indeed, most BPP instances are exactly solved. Computational experiments (see §6) have shown that this variant often determines better solutions with respect to FFF and FBS; however, due to the higher computing time needed, the best choice was to use FBS' only at the root node of the branch-decision tree.

Due to the heuristic nature of these algorithms it turned out to be convenient to always execute them twice: once for the given instance and once for the equivalent instance obtained by interchanging H with W and h_j with w_j ($j = 1, \ldots, n$).

5.4. Inner Tree

At each node of the outer tree, whenever no feasible packing of the pieces of set I is found by the heuristics, the inner branching procedure is executed. The procedure generates all the possible patterns to pack I into a bin through the "left-most downward" strategy used by Hadjiconstantinou and Christofides (1995) for the two-dimensional knapsack problem.

The pieces are packed according to nonincreasing area: at each level of the branch-decision tree the next piece is packed. Let π denote the number of pieces packed in the current pattern: the feasible locations considered for the next piece j are those in which j has its left edge adjacent either to the right edge of one of the π packed pieces or to the left edge of the bin, and its bottom edge adjacent either to the top edge of one of the π packed pieces or to the bottom edge of the bin (i.e., no further vertical-downwards or horizontal-leftwards movement of j is possible).

If no feasible position exists for the current piece, a backtracking is performed. As soon as a feasible packing is found for the last piece, the procedure terminates.

6. Computational Experiments

The exact algorithm of the previous section was coded in FORTRAN 77 and run on a DIGITAL DECstation 5000/240 (5.3 Mflops) on test instances from the literature and on new instances.

We first deal with the results obtained on instances from the literature with up to 120 pieces, reported in Table 1. Problems `beng1-beng8` are the instances

Table 1 Results on Problem Instances from the Literature

Name	n	L_0	L_4	UB	z	Nodes	Time
beng1	20	3	4	4	4	0	0.02
beng2	40	6	6	7	–	–	–
beng3	60	9	9	9	9	0	0.01
beng4	80	11	11	12	11	103	7,245.07
beng5	100	14	14	14	14	0	0.02
beng6	40	2	2	2	2	0	0.01
beng7	80	3	3	3	3	0	0.02
beng8	120	5	5	5	5	0	0.02
cgcut1	16	2	2	2	2	0	0.01
cgcut2	23	2	2	2	2	0	0.01
cgcut3	62	16	23	23	23	0	0.04
gcut1	10	3	4	5	5	4	0.02
gcut2	20	5	6	7	6	5	0.02
gcut3	30	7	8	8	8	0	0.01
gcut4	50	12	13	14	14	481	3.73
gcut5	10	3	3	4	3	1	0.01
gcut6	20	5	6	7	7	305	0.90
gcut7	30	9	10	12	11	18	0.28
gcut8	50	12	12	14	–	–	–
gcut9	10	3	3	3	3	0	0.02
gcut10	20	6	7	8	7	293	1.22
gcut11	30	7	8	9	9	110,116	909.70
gcut12	50	13	16	17	16	2	0.17
gcut13	32	2	2	2	2	0	0.01
ngcut1	10	2	2	3	3	34	0.13
ngcut2	17	3	3	4	4	113	0.85
ngcut3	21	3	3	4	3	26	2.38
ngcut4	7	2	2	2	2	0	0.01
ngcut5	14	3	3	4	3	0	0.01
ngcut6	15	2	2	3	3	518	68.98
ngcut7	8	1	1	1	1	0	0.01
ngcut8	13	2	2	2	2	0	0.01
ngcut9	18	3	3	4	3	63	0.67
ngcut10	13	2	3	3	3	0	0.01
ngcut11	15	2	2	3	2	2	3.53
ngcut12	22	3	3	4	3	18	2.07

Time limit of 10,000 CPU seconds on a Digital DECstation 5000/240 (5.3 Mflops).

given in Bengtsson (1982), while the remaining problems are instances proposed in the literature for other two-dimensional cutting problems and transformed into 2D-BPP instances by using the relative bin and piece sizes. In particular problems `cgcut1-cgcut3` are described in Christofides and Whitlock (1977), while problems `gcut1-gcut13` and `ngcut1-ngcut12` are described in Beasley (1985a and 1985b, respectively). All problem data (except those in Bengtsson 1982) can be obtained via e-mail through the OR-Library (see Beasley 1990). For each problem the table gives the problem name, the number of pieces (n), the value of the continuous lower bound L_0 and that of lower bound L_4 computed at the root node of the branch-decision tree. The table also reports the value of the best heuristic solution obtained by algorithms FFF and FBS of §5.3 (UB), the value of the optimal solution (z), the number of explored nodes of the outer tree (nodes) and the computing time (time) expressed in seconds. Only 2 problems out of 36 were not solved to optimality within the imposed time limit of 10,000 seconds. We note that for the `ngcut` instances we obtained $L_4 > L_0$ only in one case; all of these instances, however, were easily solved through branch-and-bound. For the remaining instances L_4 was better than L_0 in half of the cases, thus allowing optimal solution of the problems.

Table 2 reports the results obtained on six classes of random instances generated as described in Berkey and

Table 2 Results on the Random Problem Instances Proposed by Berkey and Wang

Pieces	Bins	n	L_0/z	L_4/z	UB/z	Nodes	#Opt	Time
[1, 10] × [1, 10]	10 × 10	20	84.1	92.2	102.5	10.2	10	0.06
		40	88.5	98.6	102.4	708.3	10	2.10
		60	91.3	98.4	102.1	4.9	7	0.61
		80	92.6	97.4	101.5	8.7	3	0.22
		100	91.9	96.2	101.5	17.0	1	0.40
[1, 10] × [1, 10]	30 × 30	20	100.0	100.0	100.0	0.0	10	0.01
		40	100.0	100.0	100.0	0.0	10	0.01
		60	80.0	80.0	100.0	0.0	4	0.01
		80	100.0	100.0	100.0	0.0	10	0.01
		100	100.0	100.0	100.0	0.0	10	0.01
[1, 35] × [1, 35]	40 × 40	20	79.8	95.0	103.3	1.0	9	10.90
		40	80.0	95.8	100.0	368.7	9	1.01
		60	83.4	92.1	102.2	17,651.6	5	111.20
		80	86.1	90.7	102.6	–	–	–
		100	86.0	89.7	100.4	–	–	–
[1, 35] × [1, 35]	100 × 100	20	100.0	100.0	100.0	0.0	10	0.01
		40	100.0	100.0	100.0	0.0	10	0.01
		60	90.0	90.0	115.0	0.0	7	0.01
		80	100.0	100.0	100.0	0.0	10	0.01
		100	100.0	100.0	100.0	0.0	10	0.01
[1, 100] × [1, 100]	100 × 100	20	81.0	96.6	100.0	1.6	10	0.01
		40	81.9	96.6	101.7	640.7	10	4.63
		60	82.9	94.3	101.2	15,004.3	8	111.45
		80	85.6	93.0	102.1	–	–	–
		100	86.6	94.4	101.4	53.0	1	2.38
[1, 100] × [1, 100]	300 × 300	20	100.0	100.0	100.0	0.0	10	0.01
		40	75.0	75.0	100.0	0.0	5	0.01
		60	100.0	100.0	100.0	0.0	10	0.01
		80	100.0	100.0	100.0	0.0	10	0.01
		100	80.0	80.0	106.7	0.0	2	0.01

Time limit of 300 CPU seconds on a Digital DECstation 5000/240 (5.3 Mflops).

Wang (1987). Each class is characterized by a different (fixed) size of the bins and by the ranges in which the piece dimensions were uniformly randomly generated. For each class and for each value of n (n = 20, 40, 60, 80, 100), 10 instances were generated, and the entries give the average values obtained. A time limit of 300 seconds was imposed for the solution of each instance. The table reports the average percentage ratio of the lower bounds and of the upper bound with respect to the optimal solution value (or to the best known solution for the instances not solved to optimality within the imposed time limit), the average number of nodes of the outer branch-decision tree, the number of instances solved to optimality (#opt), and the average computing time expressed in seconds. Statistics for nodes and times were computed with respect to the solved instances. The branch and bound algorithm optimally solved 59 out of 60 instances with n = 20, 113 instances out of 120 for $n \leq 40$, 154 out of 180 for $n \leq 60$, 187 out of 240 for $n \leq 80$, and 211 out of 300 in total. For the first, third, and fifth class the difficulty grows quite regularly with n, while for the other classes the instances are either very easy or very hard, according to different values of n. Such behaviour is explained by the fact that L_4 never improves L_0 (as the maximum piece size is always less than half the bin size) and by the relation between bin and piece sizes. Consider, for example, the second class and observe that the average area of a piece is 30.25 so each bin can accommodate, on average, 29.75 pieces. Hence, for n = 20 we always obtained $L_0 = 1$ and the heuristic algorithms easily found a solution using one bin. The same happened for n = 40, 80, 100 (with $L_0 = 2 = UB$, $L_0 = 3 = UB$, $L_0 = 4 = UB$, respectively). For n = 60, instead, we always had $L_0 = 2$, and in six cases the heuristics could not find a solution using only two bins, i.e., filling each bin almost completely.

We finally examine new classes of randomly generated problem instances having a different mix of pieces. The bin size was always 100 × 100, and four piece types were considered:

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

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

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

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

Table 3 presents the computational results for seven classes of test problems obtained from the above types. Also in this case, for each class and for each value of n, ten instances were generated and a time limit of 300 CPU seconds was imposed for the solution of each instance. The information reported in the table is the same as in Table 2.

Table 3 Results on the New Random Problem Instances

Class	n	L_0/z	L_4/z	UB/z	Nodes	#Opt	Time
I	20	81.0	87.5	100.0	2,397.2	10	44.34
	40	88.0	98.0	102.0	0.0	8	0.01
	60	87.5	98.8	100.0	0.0	8	0.01
	80	86.8	98.6	101.4	12.7	7	0.11
	100	87.1	98.9	102.6	50.0	7	1.22
II	20	88.0	96.0	107.0	1,216.1	10	3.95
	40	90.3	97.1	103.0	157.9	7	12.05
	60	86.5	98.2	101.2	3.7	7	0.05
	80	85.6	97.3	101.7	9.0	3	0.38
	100	87.0	97.6	102.1	25.0	6	54.19
III	20	72.0	96.4	100.0	1.4	10	0.01
	40	68.9	100.0	100.0	0.0	10	0.01
	60	67.3	100.0	101.1	0.6	10	0.14
	80	65.1	100.0	100.0	0.0	10	0.01
	100	63.4	99.0	100.0	1,811.9	8	40.75
IV	20	97.5	97.5	130.0	136.9	10	7.55
	40	95.0	95.0	114.0	62.0	7	62.39
	60	90.6	90.6	102.3	44.0	2	111.33
	80	89.0	89.0	100.0	–	–	–
	100	91.0	91.0	101.7	–	–	–
V	20	80.6	100.0	103.3	0.0	10	0.01
	40	67.7	100.0	101.5	1.0	10	0.04
	60	73.6	97.7	101.7	0.4	7	0.04
	80	67.1	98.3	101.0	4.4	9	0.14
	100	71.6	98.2	100.3	2.9	9	0.07
VI	20	71.3	100.0	100.0	0.0	10	0.00
	40	70.8	100.0	100.0	0.0	10	0.01
	60	72.9	96.7	101.1	3.3	7	7.01
	80	71.6	98.7	100.0	0.0	8	0.01
	100	74.7	97.6	100.6	2.0	8	0.28
VII	20	79.0	100.0	104.0	2.3	10	0.16
	40	72.3	98.3	100.0	3,133.8	9	31.81
	60	71.1	100.0	100.0	0.0	10	0.03
	80	76.2	99.5	101.0	17.9	9	4.60
	100	71.3	98.1	100.4	0.4	7	0.10

Time limit of 300 CPU seconds on a Digital DECstation 5000/240 (5.3 Mflops).

Each piece of an instance of Class k ($k \in \{I, II, III, IV\}$) was of type k with probability 70%, and of the remaining types with probability 10% each. For classes I, II, and III, in which there is a high percentage of "large" pieces, L_4 was considerably better than L_0 and many instances were optimally solved within small computing times. Class IV is much harder: this is not surprising since these instances have a large number of small pieces, so L_4 hardly improves L_0.

Classes V–VII simulate situations in which there are several copies of identical small pieces which have to be combined with large pieces. In all cases, a piece was of type 3 with probability 30%. For classes V, VI, and VII, we generated, respectively, two, three, and four type 4 pieces and we used them, with equal probability, for the remaining pieces. The results show that L_4 was always much better than L_0: this allowed exact solution of most of these difficult instances.

7. Conclusions

The two-dimensional finite bin packing problem considered in this paper has several important applications in the cutting and packing area. It is NP-hard and, in practice, very difficult to solve: no exact approach had been attempted so far. We have developed new lower bounds for this problem. Their combination within an enumerative algorithm permits the exact solution of problems involving up to 120 pieces. In addition, this paper contains a new result on the worst-case performance ratio of the well-known continuous lower bound for the problem.[1]

[1] This work was supported by Ministero dell'Università e della Ricerca Scientifica e Tecnologica, and by Consiglio Nazionale delle Ricerche, Italy. Thanks are also due to Valentina Serafini and Massimiliano Zappoli for their assistance with programming.

References

Baker, B. S., E. G. Coffman, Jr., and R. L. Rivest, "Orthogonal Packing in Two Dimensions," *SIAM J. Computing*, 9, 4 (1980), 846–855.

Bartholdi, J. J. III, J. H. Vande Vate, and J. Zhang, "Expected Performance of the Shelf Heuristic for the 2-dimensional Packing," *Operations Res. Letters*, 8 (1989), 11–16.

Beasley, J. E., "Algorithms for Unconstrained Two-dimensional Guillotine Cutting," *J. Operational Res. Society*, 36 (1985a), 297–306.

——, "An Exact Two-dimensional Non-guillotine Cutting Tree Search Procedure," *Operations Res.*, 33 (1985b), 49–64.

——, "Or-library: Distributing Test Problems by Electronic Mail," *J. Operational Res. Society*, 41 (1990), 1069–1072.

Bengtsson, B. E., "Packing Rectangular Pieces—A Heuristic Approach," *The Computer J.*, 25 (1982), 353–357.

Berkey, J. O. and P. Y. Wang, "Two Dimensional Finite Bin Packing Algorithms," *J. of Operational Res. Society*, 38 (1987), 423–429.

Christofides, N. and C. Whitlock, "An Algorithm for Two-dimensional Cutting Problems," *Operations Res.*, 25 (1977), 30–44.

Coffman, E. G., Jr., M. R. Garey, D. S. Johnson, and R. E. Tarjan, "Performance Bounds for Level-oriented Two-dimensional Packing Algorithms," *SIAM J. Computing*, 9, 4 (1980), 801–826.

—— and P. W. Shor, "Average-case Analysis of Cutting and Packing in Two Dimensions," *European J. Operational Res.*, 44 (1990), 134–144.

Dell'Amico, M. and S. Martello, "Optimal Scheduling of Tasks on Identical Parallel Processors," *ORSA J. Computing*, 7, 2 (1995), 191–200.

Dowsland, K. A. and W. B. Dowsland, "Packing Problems," *European J. Operational Res.*, 56 (1992), 2–14.

Dyckhoff, H. and U. Finke, *Cutting and Packing in Production and Distribution*, Physica Verlag, Heidelberg, 1992.

——, G. Scheithauer, and J. Terno, "Cutting and Packing (C&P)," in M. Dell'Amico, F. Maffioli, and S. Martello (Eds.), *Annotated Bibliographies in Combinatorial Optimization*, John Wiley & Sons, Chichester, 1997.

El-Bouri, A., N. Popplewell, S. Balakrishnan, and A. Alfa, "A Search Based Heuristic for the Two-dimensional Bin-packing Problem," *INFOR*, 32, 4 (1994), 265–274.

Frenk, J. B. and G. G. Galambos, "Hybrid Next-fit Algorithm for the Two-dimensional Rectangle Bin-packing Problem," *Computing*, 39 (1987), 201–217.

Hadjiconstantinou, E. and N. Christofides, "An Exact Algorithm for General, Orthogonal, Two-dimensional Knapsack Problems," *European J. Operational Res.*, 83 (1995), 39–56.

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

—— and ——, "Lower Bounds and Reduction Procedures for the Bin Packing Problem," *Discrete Applied Math.*, 28 (1990b), 59–70.

Accepted by T. M. Liebling; received July 1996. This paper was with the authors 1 month for 1 revision.