Discrete Applied Mathematics 118 (2002) 13–24

DISCRETE
APPLIED
MATHEMATICS

A lower bound for the non-oriented two-dimensional bin packing problem

Mauro Dell'Amico[a, *], Silvano Martello[b], Daniele Vigo[b]

[a]*DISMI, Università di Modena e Reggio Emilia, via Allegri 15, I-42100 Reggio Emilia, Italy*
[b]*DEIS, Università di Bologna, Viale Risorgimento 2, I-40136 Bologna, Italy*

Received 1 February 2000; received in revised form 1 January 2001; accepted 4 June 2001

Abstract

Given a set of rectangular items, and an unlimited number of identical rectangular bins, we consider the problem of allocating, without overlapping, all the items to the minimum number of bins. We assume that the items may be rotated by 90°. The problem is strongly NP-hard, and has several industrial applications. No specific lower bound is known for it. We present a lower bound which explicitly takes into account the possible item rotation. The bound is embedded into an exact branch-and-bound algorithm. The average performance is evaluated through computational experiments. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Cutting and packing; Lower bound; Branch-and-bound

1. Introduction

Given a set of n rectangular *items*, each characterized by width w_j and height h_j $(j = 1, \dots, n)$, and an unlimited number of identical rectangular *bins*, having width W and height H, the general two-dimensional bin packing problem is to allocate, without overlapping, all the n items to the minimum number of bins. Variants arise in practical contexts, according to specific technological requirements. In this paper, we consider the case where rotation of the items by 90° is allowed, known as *Non-Oriented Two-Dimensional Bin Packing Problem* and denoted as 2BP|R|F in the typology introduced by Lodi et al. [9]. The problem arises in several practical contexts: cutting of wood, metal or glass from standardized stock pieces when the items to be cut have no texture or decoration, packing of box bases on shelves or truck beds, and so on.

* Corresponding author.
E-mail addresses: dellamico@unimo.it (M. Dell'Amico), smartello@deis.unibo.it (S. Martello), dvigo@deis.unibo.it (D. Vigo).

The problem is NP-hard in the strong sense, as it generalizes the well-known one-dimensional bin packing problem. Heuristic algorithms for 2BP|R|F have been presented by Bengtsson [2], El-Bouri et al. [6] and Lodi et al. [9]. Surveys on cutting and packing problems have been given by Dyckhoff and Finke [4] and Dowsland and Dowsland [3], and an annotated bibliography by Dyckhoff et al. [5].

Apart from the trivial continuous lower bound, which does not take into account the possibility of rotating the items, to our knowledge no specific lower bound is known for 2BP|R|F (while lower bounds for the case where rotation is not allowed have been given by Martello and Vigo [11] and by Fekete and Schepers [7]). The main results that we present here are a specifically tailored relaxation and a new lower bound on the corresponding solution value. Since our relaxation is based on decomposition of the items into squares, these results are also useful in the special cases where the problem is to pack squares (see, e.g., [1,8]). In the next section, we describe the relaxation, discuss some implementation details and introduce the lower bound. In Section 3, we show how a branch-and-bound algorithm from the literature can be adapted to our problem. The performances of the bound and of the exact algorithm are evaluated in Section 4 through computational experiments.

In the following, we will assume, without loss of generality, that all input data are positive integers and that bins and items are given in "horizontal" orientation, i.e., that $W \geqslant H$ and $w_j \geqslant h_j$ for $j = 1, \ldots, n$. We further assume, in order to ensure feasibility, that $w_j \leqslant W$ and $h_j \leqslant H$ for $j = 1, \ldots, n$.

2. Lower bounds

The only known lower bound for 2BP|R|F is the obvious *continuous lower bound*:

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

It has been proved by Martello and Vigo [11] that the absolute worst-case performance of LB_C is $\frac{1}{4}$, both in the case where rotation of the items is allowed and in the case it is not. In the same paper, several lower bounds are given for the case where no rotation is allowed (denoted as 2BP|O|F, where 'O' stands for 'oriented'). Such results, however, are not valid for our less constrained problem.

A valid lower bound for 2BP|R|F comes from the following relaxation. Given an instance of the problem, we replace each item by a number of square items obtained by appropriately cutting it: for the resulting instance, there is no difference between allowing 90° rotation or not. The cutting is obtained through the following procedure which, at each inner iteration, cuts from the current (horizontal) $w_j \times h_j$ rectangle the maximum number of $h_j \times h_j$ squares, and rotates by 90° the residual (vertical) rectangle, if any. Squares of size one are not produced, as they are of no use in the subsequent lower bound computations.

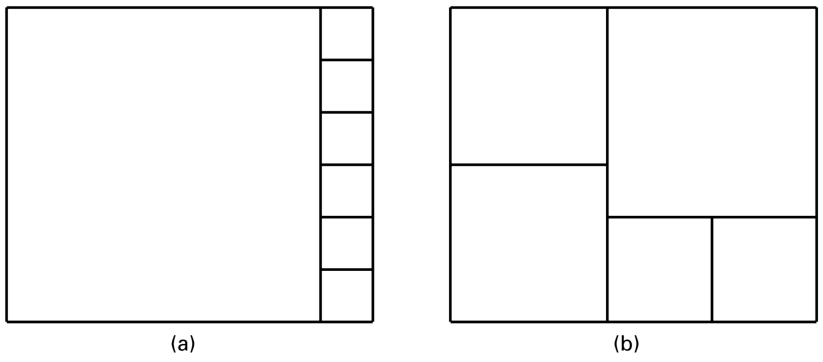

Fig. 1. (a) cutting produced by CUTSQ; (b) cutting with minimum number of squares.

```
procedure $CUTSQ$:
    $J_{SQ} := \emptyset$;
    for $j := 1$ to $n$ do
        $S := \emptyset$;
        while $h_j > 1$ do
            $k := \lfloor w_j/h_j \rfloor$;
            add $k$ squares of size $h_j$ to $S$;
            $w_j := w_j - kh_j$;
            swap $w_j$ and $h_j$
        end while;
        $J_{SQ} := J_{SQ} \cup S$
    end for
end.
```

The number of squares produced by CUTSQ is pseudo-polynomial in the worst case, although, in practical applications, it results to be reasonably small (see the computational experiments of Section 4). We also observe that the procedure produces, at each iteration, the largest possible square(s). This is useful, since the lower bound introduced below has, in general, a better performance when large items have to be packed. A different possible objective could be to produce squares with high average area, i.e., to minimize the number of resulting squares. The two objectives determine different solutions, as shown by the example in Fig. 1, where the rectangle to be cut has size 70×60.

Let $m = |J_{SQ}|$ be the number of resulting squares, define $M = \{1, \ldots, m\}$, and let l_j $(j \in M)$ be the resulting edge sizes.

Definition 1. Given an integer value q, $0 \leqslant q \leqslant \frac{1}{2}H$, let (see Fig. 2 for an illustration)

$$S_1 = \{j \in M\colon\ l_j > W - q\}, \tag{2}$$

$$S_2 = \{j \in M\colon\ W - q \geqslant l_j > \tfrac{1}{2}W\}, \tag{3}$$

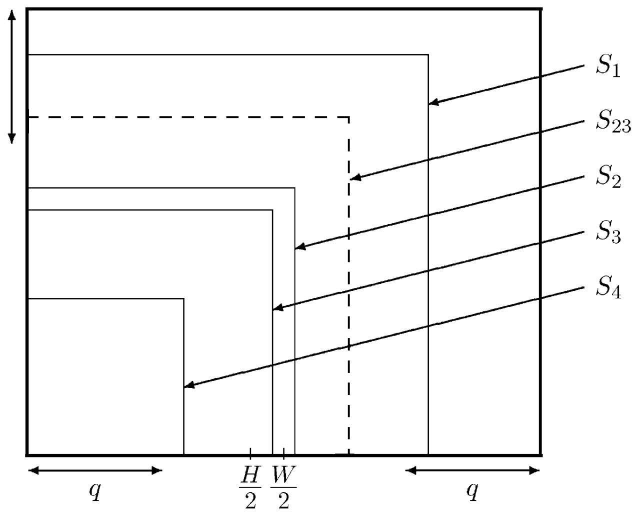

Fig. 2. The item subsets of Definition 1.

$$S_3 = \{j \in M\colon \tfrac{1}{2}W \geqslant l_j > \tfrac{1}{2}H\}, \tag{4}$$

$$S_4 = \{j \in M\colon \tfrac{1}{2}H \geqslant l_j \geqslant q\}. \tag{5}$$

Lemma 2. *Let*

$$\tilde{L} = |S_2| + \max\left\{\left\lceil \frac{\sum_{j \in S_3 \setminus \bar{S}_3} l_j}{W} \right\rceil, \left\lceil \frac{|S_3 \setminus \bar{S}_3|}{\left\lfloor \frac{W}{\lfloor H/2+1 \rfloor} \right\rfloor} \right\rceil \right\}, \tag{6}$$

where $\bar{S}_3$ is the set of the largest items of S_3 that can be packed into the bins that pack the items of S_2. A valid lower bound on the number of bins needed for packing the items of $S_1 \cup S_2 \cup S_3$ is then $|S_1| + \tilde{L}$.

Proof. Remind that $W \geqslant H$, and observe that, by definition: (i) each item of $S_1 \cup S_2$ requires a separate bin; (ii) no item of S_3 can be packed into a bin containing an item of S_1; (iii) no item of S_3 can be packed over an item of S_2. Moreover, at most one item of S_3 can be packed besides an item of S_2, since for an item j of S_2 we have $H \geqslant l_j > W/2$; hence, $H > W/2$ (if S_2 is not empty). Set $\bar{S}_3$ can thus be easily determined as follows. Assume that the bins packing the items of S_2 are sorted by a non-increasing value of the residual horizontal space $W - l_j$: assign to the next bin the largest item of S_3 that fits, stopping as soon as no such item exists. Observe that the resulting set $\bar{S}_3$ also has the largest possible cardinality. Both values in the 'max' term of (6) are thus valid lower bounds on the number of bins needed for the items of $S_3 \setminus \bar{S}_3$. The thesis follows from observation (i) above. □

We illustrate Lemma 2 through a numerical example. Let $n = 15$, $l_1 = l_2 = 18$, $l_3 = l_4 = 12$, $l_5 = \cdots = l_9 = 11$, $l_{10} = \cdots = l_{15} = 7$, $W = 23$, $H = 20$, and $q = 6$. We have $S_1 = \{1,2\}$, $S_2 = \{3,4\}$, $S_3 = \{5,\ldots,9\}$, $S_4 = \{10,\ldots,15\}$, and $\bar{S}_3 = \{5,6\}$. Hence, $\tilde{L} = 2 + \max\{\lceil 33/23 \rceil, \lceil 3/2 \rceil\} = 4$. Observe that the value $|S_1| + \tilde{L}$ is independent of the value of q. Note also that, if the items are sorted by non-increasing size, set $\bar{S}_3$ can be easily determined in $O(m)$ time.

Theorem 3. *Given an integer value q, $0 \leqslant q \leqslant \frac{1}{2}H$, consider sets $S_1,\ldots,S_4$ defined by* (2)–(5). *A valid lower bound on the optimal solution value is*

$$L(q) = |S_1| + \tilde{L} + \max\left\{0, \left\lceil \frac{\sum_{j \in S_2 \cup S_3 \cup S_4} l_j^2 - (WH\tilde{L} - \sum_{j \in S_{23}} l_j(H - l_j))}{WH} \right\rceil \right\}, \tag{7}$$

where $S_{23} = \{j \in S_2 \cup S_3 \colon l_j > H - q\}$ (see again Fig. 2).

Proof. Consider the relaxed instance that includes only the items in $S_1 \cup S_2 \cup S_3 \cup S_4$. We know from Lemma 2 that $|S_1| + \tilde{L}$ is a lower bound on the number of bins needed for the items in $S_1 \cup S_2 \cup S_3$. Let $|S_1| + \tilde{L} + \Delta$ (where Δ is a non-negative integer) be the optimal number of bins needed for the items in $S_1 \cup S_2 \cup S_3$. Consider now an item of S_4, and observe that it cannot be packed in a bin containing an item of S_1, nor above an item of S_{23}. The total area of the $\tilde{L} + \Delta$ bins that is available for these items is thus $WH(\tilde{L} + \Delta)$, decreased by the unavailable area due to the items in S_{23} (including the items area) and by the area of the items in $(S_2 \cup S_3) \setminus S_{23}$. A lower bound on the number of additional bins needed for the items in S_4 is thus

$$B = \max\left\{0, \left\lceil \frac{\sum_{j \in S_4} l_j^2 - (WH(\tilde{L} + \Delta) - H \sum_{j \in S_{23}} l_j - \sum_{j \in (S_2 \cup S_3) \setminus S_{23}} l_j^2)}{WH} \right\rceil \right\},$$

so an overall lower bound for the instance is $|S_1| + \tilde{L} + \Delta + B$. The thesis follows by algebraic manipulation. □

For the previous numerical example we obtain $L(6) = 2 + 4 + 0 = 6$ (while the continuous lower bound gives $LB_C = \lceil 1835/460 \rceil = 4$).

Corollary 4. *A valid lower bound for* 2BP|R|F *is*

$$LB = \max_{0 \leqslant q \leqslant \frac{1}{2}H} \{L(q)\} \tag{8}$$

and can be computed in $O(m)$ *time* (*plus* $O(m \log m)$ *for the item sorting*).

Proof. The validity of the bound is immediate from Theorem 3. In order to analyze the time complexity, we first show that only values of q equal to distinct l_j values need be considered. Given a value $q_1 < H/2$ not equal to any l_j value, let l_a be the smallest l_j value such that $q_1 < l_a$. Let us consider what happens when the threshold q increases from q_1 to l_a. Any q value satisfying $q_1 \leqslant q \leqslant l_a$ induces the same sets S_3 and S_4 induced by q_1, and may only cause some items to move from S_2 to S_1. Let q_2 be the smallest value satisfying $q_1 < q_2 \leqslant l_a$ and causing one item, say i, to move from S_2 to S_1, i.e., $l_i = W - q_2 + 1$ (we assume, for the sake of simplicity, that this is the unique square of size l_i). Set $\bar{S}_3$ induced by q_2 (see Lemma 2) is the same induced by q_1, since any item of S_3 has size greater than the residual horizontal space $(q_2 - 1)$ in the bin that packs item i. It follows that, on increasing the threshold from q_1 to q_2, the 'max' term in (6) does not change, hence $\tilde{L}$ decreases by 1. Compare now $L(q_1)$ and $L(q_2)$: the quantity $|S_1| + \tilde{L}$ is the same, while the variation in the third term of (7) is non-negative. Indeed: (a) if item i was not in set S_{23} induced by q_1 then S_{23} does not change and the variation in the '$\lceil\,\rceil$' term is $-l_i^2/(WH) + 1 \geqslant 0$; (b) if item i was in set S_{23} induced by q_1 then the variation is $-l_i^2/(WH) + 1 - l_i(H - l_i)/(WH) \geqslant 0$. We have thus proved that $L(q_2) \geqslant L(q)$ for any q satisfying $q_1 \leqslant q \leqslant q_2$: by iterating for higher values $q_3, q_4, \ldots \leqslant l_a$, one can conclude that the highest bound is produced by l_a. Hence the claim, since only distinct l_j values produce different sets S_4.

Once the items are sorted by non-increasing size, the computation of $L(l_{\bar{m}})$ from scratch (where $\bar{m}$ is the largest index value such that $l_{\bar{m}} \leqslant H/2$) needs $O(m)$ time. For each subsequent l_j value (by decreasing j), one only needs to update the sets and the summations needed to compute $\tilde{L}$ and $L(l_j)$. Whenever the threshold increases from l_{j-1} to l_j: (a) some item may leave S_4; (b) some (large) item may move from S_2 to S_1 and, possibly, leave S_{23}; (c) concerning $\bar{S}_3$, for each item that moves from S_2 to S_1, the corresponding (small) residual is no longer available, hence the item of S_3 currently matched with such residual (if any) leaves $\bar{S}_3$. It follows that the computation for all l_j values can be performed in overall $O(m)$ time. □

Lower bound LB dominates the continuous lower bound, see (1), computed over the relaxed instance. Indeed,

Corollary 5. $L(0) \geqslant \lceil \sum_{j \in M} l_j^2/(WH) \rceil$.

Proof. For $q = 0$ we have $S_1 = \emptyset$, $S_2 \cup S_3 \cup S_4 = M$ and $S_{23} = \emptyset$. It follows that $L(0) = \tilde{L} + \max\{0, \lceil \sum_{j \in M} l_j^2/(WH) - \tilde{L} \rceil\}$. □

However, LB does not dominate the continuous lower bound LB_C computed over the original instance, since procedure CUTSQ disregards all unit squares. It is then convenient to set LB to the maximum of the two values. The computational experiments of Section 4 show that LB is considerably better than LB_C.

It is also interesting to observe that, for the specific problem of packing squares into squares (see, e.g., [8]), our lower bound simplifies considerably:

Corollary 6. *If $W = H$, Eq.* (7) *simplifies to*

$$L'(q) = |S_1 \cup S_2| + \max\left\{0, \left\lceil \frac{\sum_{j\in S_2\cup S_4} l_j^2}{W^2} - |S_2| \right\rceil \right\}.$$

Proof. It is easy to see that $W = H$ induces $S_3 = \emptyset$, $\tilde{L} = |S_2|$ and $S_{23} = \emptyset$. Eq. (9) follows by algebraic manipulation. □

Martello and Vigo [11] gave lower bounds for problem 2BP|O|F, in which rotation is not allowed: L_1 and L_2 with time complexity $O(n^2)$, and L_3 with time complexity $O(n^3)$. These bounds are obviously valid for our relaxed instance. We observe that none of them dominates lower bound LB, as shown by the following numerical example.

Let $n = 5$, $W = H = 10$, $w_1 = h_1 = 9$, $w_2 = 9$, $h_2 = 8$, $w_3 = 10$, $h_3 = 5$, $w_4 = 10$, $h_4 = 3$, $w_5 = 7$, $h_5 = 5$. By executing procedure CUTSQ we get: $m = 10$, $l_1 = 9$, $l_2 = 8$, $l_3 = l_4 = 5$, $l_5 = l_6 = l_7 = 3$, $l_8 = 5$, $l_9 = l_{10} = 2$. For $q = 3$ we have: $S_1 = \{1,2\}$, $S_2 = S_3 = \bar{S}_3 = S_{23} = \emptyset$, $S_4 = \{3,\dots,8\}$ hence $\tilde{L} = 0$ and $L(3) = 2+0+\lceil(102-0)/100\rceil = 4$. By computing the Martello and Vigo [11] lower bounds one obtains instead $L_1 = L_2 = L_3 = 3$.

We finally mention that, for instances with rectangular bins and such that some item cannot be rotated (i.e., $w_j > H$ for some j), an alternative bound can be obtained as follows. Let $T = \{j\colon w_j > H\}$, apply CUTSQ only to the items of $\{1,\dots,n\} \setminus T$ and compute, for the instance defined by T plus the resulting squares, any lower bound for 2BP|O|F. We used lower bound L_4 by Martello and Vigo [11], and improved LB (see (8)) by setting $LB = \max\{LB, L_4\}$.

3. A branch-and-bound algorithm

In order to test the effectiveness of lower bound LB of the previous section, we used a branch-and-bound algorithm for the exact solution of 2BP|R|F, obtained by adapting the nested branching scheme introduced by Martello and Vigo [11] for 2BP|O|F.

An *outer tree* assigns the items to the bins without specifying their actual position. The items are initially sorted according to non-increasing area, and the search is performed according to a depth-first strategy: at level k, item k is assigned, in turn, to all active bins and, possibly, to a new bin. At each decision-node, an inner tree may be used to determine a feasible single bin packing (if any) for item k and the set J of items currently assigned to the selected bin. Before executing the inner search, however, two heuristic attempts are performed. Lower bound LB is first computed for item set $J \cup \{k\}$: if $LB > 1$, then the decision node is fathomed. Otherwise, simple approximation algorithms are executed, trying to obtain a packing of $J \cup \{k\}$ into a single bin. If both attempts fail, the inner tree enumerates all possible patterns as follows.

At the root node of the *inner tree*, at most $2(|J|+1)$ descendant nodes are generated by placing each item, in both orientations, with a vertex in the bottom-left corner of the bin. At every decision-node, each unassigned item is allocated, in both possible orientations, to all feasible positions where it cannot be moved leftward or downward. In order to avoid the duplication of identical patterns, we adapted the approach introduced by Martello et al. [10] for the three-dimensional oriented bin packing problem, thus reducing the possible points on which to allocate an item vertex to the points for which no already assigned item has some part right and above the vertex. If no feasible single bin packing is determined by the inner tree, the outer tree node is fathomed.

Whenever an outer tree node is not fathomed, lower bound LB is used for trying to establish that no further item can be assigned to the current bin. We compute LB for all sub-instances induced, for each unassigned item j, by item set $J \cup \{k\} \cup \{j\}$: the result is established if $LB = 2$ for all j, hence the bin is considered *closed*. When a new bin is closed, LB is used again in an additional attempt to fathom the current node. Let B be the set of closed bins, and $J(B)$ the set of items assigned to these bins: we compute LB for the sub-instance induced by item set $\{1,\dots,n\} \setminus J(B)$, and fathom the node if $|B| + LB \geqslant z$, where z is the incumbent solution value.

4. Computational experiments

Lower bound LB and the branch-and-bound algorithm of the previous section were coded in FORTRAN 77 and run on a Digital Alpha 533 MHz. Computational experiments on two-dimensional bin packing problems (see, e.g., Martello and Vigo [11]) show that randomly generated instances are generally harder to solve exactly than real-world ones. Therefore, we tested our bound and the branch-and-bound algorithm on random instances, generated so as to consider different bin types (square and rectangular) and item characteristics.

We considered nine classes of instances, subdivided into three groups. Group 1 involves square bins with $W = H = 100$. The items are generated in three classes by increasing average area:

Class 1.1: w_j, h_j uniformly random in $[10, 100]$;

Class 1.2: w_j, h_j uniformly random in $[20, 100]$;

Class 1.3: w_j, h_j uniformly random in $[30, 100]$.

The second group of classes involves rectangular bins with $W = 100$ and $H = 50$. In this case too, the items are generated in three classes by increasing average area:

Class 2.1: w_j uniformly random in $[10, 100]$, h_j uniformly random in $[5, 50]$;

Class 2.2: w_j uniformly random in $[20, 100]$, h_j uniformly random in $[10, 50]$;

Class 2.3: w_j uniformly random in $[30, 100]$, h_j uniformly random in $[15, 50]$.

Finally, in Group 3, we have square bins with $W = H = 100$, and a mix of thin and square items. Namely, each item belongs, with equal probability, to one of two types, i.e.,

thin: w_j uniformly random in $[10, 50]$, h_j uniformly random in $[h_{\min}, 100]$;

square: $w_j = h_j$ uniformly random in $[20, 100]$,

Table 1
Results on test instances of Group 1[a]

				B&B (continuous bound)			B&B (bound LB)			
Class	n	$\%\frac{LB_C}{z}$	$\%\frac{LB}{z}$	#Opt	Nodes	Time	#Opt	Nodes	Time	#Squares
1.1	10	88.0	95.0	10	6	0.00	10	3	0.00	142.0
	20	86.3	94.7	10	227	0.03	10	56	0.02	180.4
	30	86.2	96.6	10	4610	3.00	10	419	0.12	289.3
	40	88.2	93.0	7	355 328	46.95	8	101 055	38.41	385.7
	50	88.0	94.2	4	57 329	6.77	5	20 008	4.46	461.7
	75	90.1	93.0	2	44 556	2.66	2	2638	2.03	699.9
	100	88.0	95.0	1	1 485 071	96.78	2	0	0.06	938.4
1.2	10	88.3	92.0	10	9	0.00	10	2	0.00	185.1
	20	79.9	90.3	10	482	0.05	10	149	0.05	197.7
	30	82.5	94.7	10	5842	0.85	10	1789	0.84	315.0
	40	84.4	92.7	7	281 926	24.60	7	84 681	24.66	403.4
	50	83.5	95.1	3	2997	0.27	5	0	0.03	498.5
	75	86.7	92.2	0	0	0.00	2	4098	2.60	753.6
	100	83.9	93.3	0	0	0.00	1	0	0.08	994.5
1.3	10	72.8	88.2	10	39	0.01	10	11	0.00	189.7
	20	78.3	93.3	10	1705	0.15	10	246	0.13	213.8
	30	74.3	95.5	10	40 170	4.81	10	4987	3.92	319.4
	40	77.9	93.6	5	339 662	31.37	5	36 821	25.19	440.5
	50	78.4	93.5	1	699	0.07	3	0	0.02	536.4
	75	80.8	92.6	0	0	0.00	0	0	0.00	785.6
	100	78.2	94.6	0	0	0.00	3	0	0.08	1052.8

[a] Time limit of 120 CPU seconds on a Digital Alpha 533 MHz. Average nodes and times computed over the solved instances.

and three classes are obtained as

Class 3.1: $h_{\min}=40$;

Class 3.2: $h_{\min}=60$;

Class 3.3: $h_{\min}=80$.

For each class, we considered seven values of n : 10, 20, 30, 40, 50, 75 and 100.

Ten instances were generated for each class and value of n, yielding a total of 630 test problems. A time limit of 120 seconds was given to each instance. Tables 1–3 give the outcome of the experiments for the three groups. The entries give, for each class and value of n,

(i) average values of the ratios $100 \cdot LB_C/z$ and $100 \cdot LB/z$, where z is the value of the best solution found by the branch-and-bound algorithm (optimal or incumbent);

(ii) number of instances solved to proved optimality, average number of decision nodes and average CPU time (both computed over the optimally solved instances) for the branch-and-bound algorithm, if run with the continuous lower bound;

(iii) same information as (ii) for the branch-and-bound algorithm, if run with lower bound LB;

(iv) average number of squares generated by procedure *CUTSQ*.

Table 2
Results on test instances of Group 2[b]

				B&B (continuous bound)			B&B (bound LB)			
Class	n	$\%\frac{LB_C}{z}$	$\%\frac{LB}{z}$	#Opt	Nodes	Time	#Opt	Nodes	Time	#Squares
2.1	10	84.7	95.8	10	6	0.00	10	1	0.00	178.3
	20	85.8	97.5	9	134	0.01	10	10	0.02	161.6
	30	86.4	97.7	10	2188	0.45	10	29	0.04	245.5
	40	85.2	95.2	6	74 889	7.74	8	10 559	11.68	339.5
	50	86.0	94.4	1	282 826	12.47	3	20	0.07	422.4
	75	90.1	94.6	0	0	0.00	1	0	0.02	614.6
	100	88.4	95.9	0	0	0.00	1	108	0.28	823.6
2.2										
	10	85.0	94.3	10	9	0.00	10	1	0.00	162.3
	20	82.2	94.6	10	659	0.07	10	63	0.07	168.1
	30	82.1	99.2	9	214 474	13.34	10	58	0.10	262.6
	40	83.9	95.5	6	347 152	38.80	6	1575	3.12	343.7
	50	84.4	97.5	2	61 151	8.19	6	3174	6.31	420.5
	75	87.0	96.3	1	501 704	38.80	2	23	0.13	634.4
	100	84.3	96.8	0	0	0.00	4	30	0.35	847.7
2.3	10	75.4	96.1	10	19	0.00	10	2	0.00	159.0
	20	77.9	98.3	10	1369	0.13	10	13	0.03	167.1
	30	74.4	100.0	10	10 494	0.92	10	14	0.11	259.9
	40	78.5	96.7	6	475 834	49.23	7	3323	13.39	355.3
	50	79.7	96.9	0	0	0.00	3	1	0.06	434.0
	75	81.4	96.7	0	0	0.00	2	13	0.20	637.3
	100	78.9	97.3	0	0	0.00	5	6	0.34	877.8

[b] Time limit of 120 CPU seconds on a Digital Alpha 533 MHz. Average nodes and times computed over the solved instances.

The tables show that the proposed bound is considerably better than the continuous lower bound. By embedding it in a standard branch-and-bound algorithm we could easily solve instances of moderate size. Indeed, practically all instances with $n \leqslant 30$, 90% of those with $n \leqslant 40$, and 80% of those with $n \leqslant 50$ were solved to optimality within very short computing time. In total, we solved almost two-thirds of the instances within a time limit of 2 minutes (i.e., 398 out of 630). Increasing the time limit to 600 seconds produced just a slight improvement: 25 more instances in total were solved to optimality.

The same branch-and-bound algorithm with the continuous lower bound solved to optimality 15% less instances and the number of branch-decision nodes was one order of magnitude higher. As to the CPU times, the algorithm with bound LB is generally faster and/or solves more instances (recall that the times in the tables are averages over the solved instances only).

Finally, from the comparison of the three tables it is seen that both the bound and the branch-and-bound algorithm have a rather stable behavior over the three groups: the bound quality and the number of solved instances are indeed quite insensitive to bin and item characteristics.

Table 3
Results on test instances of Group 3[c]

				B&B (continuous bound)			B&B (bound LB)			
Class	n	$\%\frac{LB_C}{z}$	$\%\frac{LB}{z}$	#Opt	Nodes	Time	#Opt	Nodes	Time	#Squares
3.1	10	79.5	97.5	10	5	0.00	10	0	0.00	44.7
	20	82.5	92.1	10	345	0.03	10	95	0.02	97.1
	30	80.5	91.0	10	90 310	7.42	10	37 561	6.66	163.8
	40	81.7	97.7	8	156 909	14.54	10	44 881	26.56	196.1
	50	81.5	95.5	6	1 057 470	61.44	6	97 632	22.15	254.9
	75	83.7	97.8	0	0	0.00	4	0	0.02	357.3
	100	83.7	95.5	0	0	0.00	1	0	0.02	518.6
3.2	10	82.7	95.8	10	7	0.00	10	1	0.00	107.0
	20	81.5	91.0	10	1187	0.17	10	746	0.16	115.3
	30	81.6	89.0	9	111 017	9.35	9	48 003	8.76	166.1
	40	82.0	95.5	5	194 792	20.91	7	99 568	27.93	216.9
	50	83.6	94.1	1	1 078 344	68.43	3	42 962	17.98	289.1
	75	85.5	96.4	0	0	0.00	2	0	0.02	383.5
	100	86.1	94.7	0	0	0.00	0	0	0.00	553.4
3.3	10	86.8	98.3	10	7	0.00	10	1	0.00	109.3
	20	83.4	94.2	10	2424	0.35	10	826	0.18	116.3
	30	82.6	89.1	8	96 114	6.93	8	25 688	5.53	164.5
	40	83.3	94.2	3	162 047	14.19	4	8363	5.13	215.2
	50	84.4	93.4	1	1 029 222	61.52	3	17 445	15.95	284.5
	75	86.9	94.8	0	0	0.00	1	0	0.02	391.9
	100	87.6	93.0	0	0	0.00	0	0	0.00	575.8

[c] Time limit of 120 CPU seconds on a Digital Alpha 533 MHz. Average nodes and times computed over the solved instances.

Acknowledgements

This work was supported by Ministero dell'Università e della Ricerca Scientifica e Tecnologica, and by Consiglio Nazionale delle Ricerche, Italy. We thank Lab.O.R. (Laboratory of Operations Research of the University of Bologna) for the support given in the computational testings.

References

[1] B.S. Baker, A.R. Calderbank, E.G. Coffman Jr., J.C. Lagarias, Approximation algorithms for maximizing the number of squares packed into a rectangle, SIAM J. Algebraic Discrete Methods 4 (3) (1983) 383–397.

[2] B.E. Bengtsson, Packing rectangular pieces — a heuristic approach, Comput. J. 25 (1982) 353–357.

[3] K.A. Dowsland, W.B. Dowsland, Packing Problems, European J. Oper. Res. 56 (1992) 2–14.

[4] H. Dyckhoff, U. Finke, Cutting and Packing in Production and Distribution, Physica-Verlag, Heidelberg, 1992.

[5] H. Dyckhoff, G. Scheithauer, J. Terno, Cutting and Packing (C&P), in: M. Dell'Amico, F. Maffioli, S. Martello (Eds.), Annotated Bibliographies in Combinatorial Optimization, Wiley, Chichester, 1997.

[6] 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.

[7] S.P. Fekete, J. Schepers, On more-dimensional packing II: Bounds, Technical Report ZPR97-289, Mathematisches Institut, Universität zu Köln, 2000.

[8] C.E. Ferreira, F.K. Miyazawa, Y. Wakabayashi, Packing squares into squares, Technical Report, Instituto de Computação, Universidade Estadual de Campinas, Brazil, 1999.

[9] 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.

[10] S. Martello, D. Pisinger, D. Vigo, The three-dimensional bin packing problem, Oper. Res. 48 (2000) 256–267.

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