A simple and effective recursive procedure for the manufacturer's pallet loading problem

Article *in* Journal of the Operational Research Society · August 1998

DOI: 10.1038/sj.jors.2600588

2 authors, including:

Reinaldo Morabito
Universidade Federal de São Carlos
286 PUBLICATIONS **5,508** CITATIONS

SEE PROFILE

Journal of the Operational Research Society (1998) 49, 819–828
http://www.stockton-press.co.uk/jor

A simple and effective recursive procedure for the manufacturer's pallet loading problem

R Morabito and S Morales

Universidade Federal de São Carlos, Brazil

In this paper we present a simple and effective heuristic to solve the problem of packing the maximum number of rectangles of sizes (l, w) and (w, l) into a larger rectangle (L, W) without overlapping. This problem appears in the loading of identical boxes on pallets, namely the manufacturer's pallet loading (MPL), as well as in package design and truck or rail car loading. Although apparently easy to be optimally solved, the MPL is claimed to be NP-complete and several authors have proposed approximate methods to deal with it. The procedure described in the present paper can be seen as a refinement of Bischoff and Dowsland's heuristic and can easily be implemented on a microcomputer. Using moderate computational resources, the procedure was able to find the optimal solution of 99.9% of more than 20 000 examples analysed.

Keywords: combinatorial analysis; cutting stock problem; heuristics; manufacturer's pallet loading; packing

Introduction

Packing items (for example, boxes) into larger objects (pallets) can be seen as similar to cutting objects (for example, plates) to produce smaller items (ordered pieces). Hundreds of papers have been published in the literature dealing with cutting and packing problems; see for example the surveys in Dowsland and Dowsland,[1] Dyckhoff and Finke,[2] Sweeney and Paternoster,[3] Bischoff and Waescher.[4] In this present paper we are paticularly concerned with the problem of loading products (packaged in boxes) on a rectangular pallet in such a way as to optimise the pallet utilisation. The problem appears in the logistical move-and-store activities such as transportation and carrying inventories and, depending on the scale of the supply/distribution channel, a small increase in the number of products loaded on each pallet may result in substantial savings.

The problem is known as the manufacturer's pallet loading (MPL) if all boxes are identical. This is the case of a manufacturer that produces goods packaged in identical boxes of size (l, w, h), which are then arranged in horizontal layers on pallets of size (L, W, H) (where H is the maximum height of the loading). We assume that the boxes are available in large quantities and are orthogonally loaded on each pallet (that is, with their sides parallel to the pallet sides). If no orientation is fixed (that is, the boxes can be placed on any of their faces (l, w), (l, h) or (w, h)) and because of loading stability requirements, it is usual to divide the MPL into two subproblems: (i) for each face, the two-dimensional problem of arranging the maximum number of rectangles (faces) into the large rectangle (L, W) (pallet surface) without overlapping, and (ii) the one-dimensional problem of stacking the most valuable layers along the width H of the pallet, where the value of each layer corresponds to its number of boxes (note that this is a knapsack problem).

When boxes are not of the same size, the problem is called the distributor's pallet loading. Different approaches are found in the literature, for example, in Hodgson,[5] Dowsland,[6] Abdou and Yang[7] and Arenales and Morabito[8]; see also the references in[1,2]. Under these conditions, the distributor's problem can be seen as a container loading problem; see for example Bischoff and Ratcliff[9] and Morabito and Arenales.[10]

In this paper we assume that the boxes are identical and an orientation is fixed. The problem consists of packing the maximum number of rectangles (l, w) and (w, l) orthogonally into a larger rectangle (L, W) without overlapping (as in (i) above), yielding a layer of height h. We also assume that there are no constraints related to the weight, density, fragility, etc., of the cargo loading. Besides the pallet loading, the MPL is also of concern in the package design and the truck or rail car loading. It can be classified as 2/B/O/C according to Dychoff's typology of cutting and packing problems.[11]

Apparently easy to be optimally solved, the MPL is claimed to be NP-complete although this has not been proven yet.[12] Several authors have proposed approximate methods to deal with the MPL. The approach presented in this paper generalises the heuristic of Bischoff and Dowsland[13] (B&D heuristic) in order to produce more generic

Correspondence: Dr R Morabito, Departamento de Engenharia de Produção, Universidade Federal de São Carlos, 13565-905, São Carlos, SP, Brazil
E-mail: morabito@power.ufscar.br

loadings. Requiring moderate computational efforts, the algorithms were able to find the optimal solutions of almost all examples analysed (only 18 examples out of more than 20 000 were not optimally solved).

In the next section, we define packing patterns and normal and raster point sets, which are useful to the recursive procedure of section 3. In section 2.1 we briefly discuss exact approaches and present a 0–1 linear model to the MPL. The LP-relaxation of this model is utilised to certify that the solutions of the examples examined in section 4 are indeed optimal. In section 2.2 we concisely review approximate methods and upper/lower bounds to the MPL, and present the B&D heuristic (algorithm 1) which is part of the recursive procedure. In section 3 this procedure is firstly described as a simple extension of the B&D heuristic (algorithm 2), and then is refined to produce more effective solutions (algorithm 3). Section 4 evaluates the computational performance of the algorithms to solve a number of examples known in the literature as well as random examples. Finally, the concluding remarks are discussed in section 5.

MPL modelling

A two-dimensional cutting (or packing) pattern lays out a set of rectangles on a larger rectangle (see Figure 1(b)) and is well defined by a sequence of cuts performed on the large rectangle. For the purpose of this work, the packing patterns are classified[8] as:

(i) Guillotine and non-guillotine patterns: A cut is guillotine-type if, when applied on a rectangle, it produces two new rectangles; otherwise it is called non-guillotine. A pattern is guillotine-type if it is obtained by successive guillotine cuts (Figure 4(a)). A pattern in non-guillotine if it is obtained by successive guillotine and/or non-guillotine cuts (Figures 1(b), 4(b), 5(a, b)). Therefore, the guillotine patterns are particular non-guillotine patterns.

(ii) 1st-order and superior-order non-guillotine patterns: A cut is 1st-order non-guillotine-type if, when applied on a rectangle, it produces five new rectangles arranged in such a way as not to form a guillotine pattern (Figure 1(a)). A pattern is 1st-order non-guillotine-type if it is obtained by successive guillotine and/or 1st-order non-guillotine cuts (Figures 1(b), 5(b)). Obviously there are non-guillotine patterns that are not 1st-order, namely superior-order, as the ones depicted in Figures 4(b), 5(a).

The optimal solution of the MPL problem corresponds to a non-guillotine pattern (that is either a guillotine, 1st-order or superior-order non-guillotine pattern).

Let l, w, L and W be positive integers satisfying: $l \geqslant w, l \leqslant L$ and $l \leqslant W$, where (l, w) and (L, W) are the sizes of the box face and the pallet surface, respectively. Obviously we can place a face in many positions along the length L and the width W of the pallet. Let (x, y) be the coordinates of the left-lower-corner of a face placed on the pallet (we assume that the left-lower-corner of the pallet is $(0, 0)$). Note that x and y belong to the sets $\{x|0 \leqslant x \leqslant L - w, \text{integer}\}$ and $\{y|0 \leqslant y \leqslant W - w, \text{integer}\}$, respectively. It can be shown that these sets can be reduced without loss of generality to the normal sets (Beasley)[14]:

$$X = \{x|x = rl + sw, x \leqslant L - w, r, s \geqslant 0 \text{ and integer}\} \quad (1)$$

$$Y = \{y|y = tw + ul, y \leqslant W - w, t, u \geqslant 0 \text{ and integer}\} \quad (2)$$

or to the sets of raster points (Scheithauer and Terno[15]):

$$\begin{aligned} X' &= \{\langle L - x\rangle_X | x \in X\} \cup \{0\}, \\ \langle x'\rangle_X &= \max\{x|x \in X, x \leqslant x'\} \end{aligned} \quad (3)$$

$$\begin{aligned} Y' &= \{\langle W - y\rangle_Y | y \in Y\} \cup \{0\}, \\ \langle y'\rangle_Y &= \max\{y|y \in Y, y \leqslant y'\} \end{aligned} \quad (4)$$

Exact approaches and a 0–1 linear formulation

The MPL can be modeled as a special case of the two-dimensional non-guillotine cutting formulation proposed in Beasley.[14] For simplicity, we define l_k and w_k as the face length and width under orientation $k = 1, 2$ and so, $l_1 = l, w_1 = w, l_2 = w$ and $w_2 = l$. In this way, the decision of placing a face on the pallet is limited to the orientations $k = 1, 2$ and the positions $(x, y), x \in X, y \in Y$ or, similarly, the positions $(x, y), x \in X', y \in Y'$. Let $I = \{1, 2, \ldots, |X|\}$, $J = \{1, 2, \ldots, |Y|\}$ and $K = \{1, 2\}$ ($|A|$ denotes the number of elements of set A). Therefore, $x_i \in X, i \in I$, corresponds to the ith element of set X (similarly to $y_j \in Y, j \in J$). As we decide to place a face on a position (x_i, y_j) with orientation $k \in K$, we can not place another face in a position (x_p, y_q) such that $x_i \leqslant x_p \leqslant x_i + l_k - 1$ and $y_j \leqslant y_q \leqslant y_j + w_k - 1$, $p \in I, q \in J$. In order to avoid face overlapping, we define a function:

$$g_{ijpqk} = \begin{cases} 1, & \text{if } x_i \leqslant x_p \leqslant x_i + l_k - 1 \leqslant l - 1 \text{ and} \\ & y_j \leqslant y_q \leqslant y_j + w_k - 1 \leqslant w - 1 \\ 0, & \text{otherwise.} \end{cases}$$

which must be computed a priori for each position (x_i, y_j), $i \in I, j \in J$, for each position (x_p, y_q), $p \in I, q \in J$, and for each orientation $k \in K$. Moreover, for each $i \in I, j \in J$ and $k \in K$, we define the decision variables:

$$a_{ijk} = \begin{cases} 1, & \text{if a face is placed in a position } (x_i, y_j) \\ & \text{with orientation } k \\ 0, & \text{otherwise} \end{cases}$$

The MPL can be formulated as the following 0–1 integer program:

$$\max \sum_{k\in K} \sum_{i\in I|x_i \leqslant L-l_k} \sum_{j\in J|y_j \leqslant W-w_k} a_{ijk} \tag{5}$$

subject to

$$\sum_{k\in K} \sum_{i\in I} \sum_{j\in J} g_{ijpqk} a_{ijk} \leqslant 1, \qquad p \in I, q \in J \tag{6}$$

with

$$a_{ijk} \in \{0, 1\}, \qquad i \in I, j \in J, k \in K \tag{7}$$

Note that model 5–7 has $O(|X||Y|)$ 0–1 variables and $O(|X||Y|)$ constraints and, since X and Y can be large in practical cases, it may be hard to solve this model. An optimal solution to 5–7 corresponds to a non-guillotine pattern (that is, either a guillotine, 1st-order or superior-order non-guillotine pattern). In order to solve cutting problems with relatively many different rectangles of low demand, Beasley[14] suggested applying a branch-and-bound procedure with upper bounds provided by a Lagrangean relaxation of 5–7 (under surrogate constraints) and using the subgradient optimisation method to obtain the Lagrangean multipliers. However, this approach does not perform well on typical MPL as pointed out in Dowsland.[16]

A few authors have proposed other exact methods to solve the MPL. Dowsland[16] presented an interesting approach that basically consists of finding the maximum stable set of a particular finite graph where the nodes represent the possible box positions on the pallet, such that two nodes are adjacent if the box positions they represent overlap (a stable set of a graph $G(N, A)$ is a node subset of N such that there are no two nodes connected by an arc in A). The approach was based on the observations that there is a one-to-one correspondence between the maximum stable sets and the optimal layouts. The algorithm performs well over problems where the number of boxes by layer is not large (less than 50). Tsai *et al*[17] formulated the MPL as a 0–1 program (different from 5–7) with disjunctive constraints (multiple choice) and suggested to solve it by a branch-and-bound algorithm exploring the particular structure of these constraints. However, the number of such constraints grows exponentially with the number of available boxes.

Approximate methods and the B&D heuristic

Due to the complexity of the exact methods, a number of block heuristics has been proposed in the literature to solve the MPL. Each block corresponds to a rectangle containing a set of boxes arranged under the same orientation. Examples of block heuristics can be found in Steudel[18] and Smith and De Cani[19] where the pallet is divided into, at most, four blocks and the boxes are simply arranged inside each one according to a previously defined orientation. Other approaches can be found in Nelissen,[12,20] where upper bound methods and a genetic algorithm were proposed to the MPL, and more recently in Scheithauer and Terno,[15] where an effective heuristic namely G4 was defined. The G4 heuristic, in some sense similar to the basic idea of the approach presented in section 3, explores a recursive division of the pallet into, at most, four rectangles and the use of dynamic programming to obtain the best pattern with a G4 structure.

An interesting block heuristic was presented in Bischoff and Dowsland[13] which can be viewed as a refinement of the heuristic in[19]. The pallet (L, W) is divided into, at most, five blocks, namely 1, 2, 3, 4 and 5, with sizes (L_i, W_i) according to Figure 1(a). Each block has a pre-fixed orientation for the arrangement of the faces as indicated in Figure 1(a), except for block 3 whose orientation is post-fixed. Basically, the method examines a number of guillotine and 1st-order non-guillotine cuts or, equivalently, a number of different sizes (L_i, W_i) of the external blocks 1, 2, 4 and 5 (including $L_i = 0$ and $W_i = 0$), and returns the pattern that results in the maximum number of faces.

The search for the best pattern can be interrupted as soon as a pattern satisfying the condition $LW - z(lw) < lw$ is found (that is, the remaining area in the pallet is less than the area of the face). It is worth noting that this is a sufficient but not a necessary optimality condition. In spite of not being (explicitly) explored,[13] the use of lower and upper bounds to reduce the search can substantially improve the computational performance of the method. Let z_{lb} and z_{ub} be two simple bounds to (L, W) defined as ($\lfloor a \rfloor$ denotes the largest integer less than or equal to a):

$$z_{lb} = \max\{\lfloor L/l \rfloor \lfloor W/w \rfloor, \lfloor L/w \rfloor \lfloor W/l \rfloor\} \tag{8}$$

$$z_{ub} = \lfloor L^* W^*/lw \rfloor \tag{9}$$

where $L^* = w + \max\{x|x \in X\}$ and $W^* = w + \max\{y|y \in Y\}$. The lower bound z_{lb} can be updated as the search proceeds and better patterns are found (that is, if $z > z_{lb}$ then $z_{lb} = z$). Obviously, if $z_{lb} = z_{ub}$ then the search can be interrupted. Other lower and upper bounds better than (8) or (9) might be defined; see the discussion in[20]. In particular, an interesting upper bound was proposed in

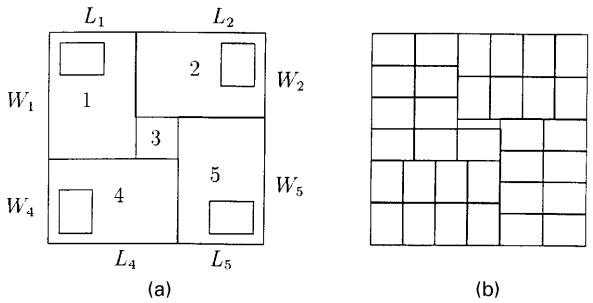

Figure 1 (a) Division of the pallet into five blocks with orientation of the boxes, (b) pattern with 33 boxes obtained by algorithm 1 for example $(L, W) = (20, 20)$ and $(l, w) = (4, 3)$.

Barnes[21] which is based on the consideration that each packing of an (l, w) rectangle on (L, W) is also a packing of either $(l, 1)$ or $(1, w)$ tiles on (L, W).

A formal algorithm for the B&D heuristic is presented below—the solution produced by this algorithm with parameters (L, W) and (l, w), or (W, L) and (l, w), corresponds to the best 1st-order non-guillotine pattern of, at most, five blocks.

Algorithm 1 B&D heuristic

Step 1: Determine the sets: $P = \{(r, s) | L - w < rl + sw \leqslant L, r, s \geqslant 0$ and integer$\}$ and $Q = \{(t, u) | W - w < tl + uw \leqslant W, t, u \geqslant 0$ and integer$\}$, and compute z_{lb} (8) and z_{ub} (minimum of (9) and Barnes's bound). If $z_{lb} = z_{ub}$ then go to step 3, otherwise go to step 2.

Step 2: For each $(r_1, s_1) \in P$ and $(r_2, s_2) \in P$ such that $r_2 l + s_2 w \geqslant r_1 l + s_1 w$, and for each $(t_1, u_1) \in Q$ and $(t_2, u_2) \in Q$ such that $t_2 l + u_2 w \geqslant t_1 l + u_1 w$, do:

2.1: Determine the sizes $(L_1, W_1) = (r_1 l, u_1 w)$, $(L_2, W_2) = (s_1 w, t_2 l)$, $(L_3, W_3) = (L - L_1 - L_5, W - W_2 - W_4)$, $(L_4, W_4) = (s_2 w, t_1 l)$ and $(L_5, W_5) = (r_2 l, u_2 w)$.

2.2: If there is no overlap, that is, if $(L_1 + L_5 > L, W_1 + W_5 > W)$ and $(W_2 + W_4 > W, L_2 + L_4 > L)$ are false, then:

2.2.1: Determine $z = \sum_{i=1}^{5} z_i$, where $z_i = L_i/l \rfloor \lfloor W_i/w \rfloor$, $i = 1, 5$, $z_i = \lfloor L_i/w \rfloor \lfloor W_i/l \rfloor$, $i = 2, 4$, and $z_3 = \max\{\lfloor L_3/l \rfloor \lfloor W_3/w \rfloor, \lfloor L_3/w \rfloor \lfloor W_3/l \rfloor\}$.

2.2.2: If $z > z_{lb}$ then update z_{lb}, the maximum number of boxes obtained so far. In this case, if $z_{lb} = z_{ub}$ then exit to step 3.

Step 3: Return the best found solution z_{lb}. If $z_{lb} = z_{ub}$ then the pattern to (L, W) is optimal.

The optimal pattern of Figure 1(b) was generated by algorithm 1. Note that $P = Q = \{(0, 6), (1, 5), (2, 4), (3, 2), (4, 1), (5, 0)\}$. Other block heuristics of 7 and 9 blocks and diagonal solution are described.[20] Figures 2(b), 5(b) and 6(c) illustrate patterns with much more than 5 blocks. Note that none of them can be generated by algorithm 1 which is unable to find patterns with more than 5 blocks. In the next section we present a simple and effective refinement of algorithm 1 which is able to produce 1st-order patterns of any number of blocks.

Recursive procedure

The B&D heuristic can be seen as an implicit enumeration method of a number of guillotine and 1st-order non-guillotine cuts to the rectangle (L, W). This heuristic can be improved as follows: During the enumeration of algorithm 1 to (L, W), for each cut examined, apply recursively algorithm 1 to each block (L_i, W_i) generated by this cut, and so on, until all blocks are generated by all cuts are so small that the algorithm cannot be further applied. Note that this is a simple recursive procedure based on the non-guillotine B&D heuristic (recursive procedures for guillotine problems were explored by Herz[22]).

This procedure can be represented as a tree search, where each node of the tree corresponds to a block and each set of, at most, 5 arcs emerging from a node corresponds to a set of guillotine cuts of a 1st-order cut. Each time that algorithm 1 is applied to a block, the search goes down one level in the tree. If we always apply the algorithm to the most recent generated node, we traverse the tree according to a depth-first search strategy (in fact, the recursive procedure utilises a backtracking strategy which becomes more evident with the description of algorithm 2).

The depth of the tree depends on the sizes of the boxes and the pallet. Let n be the level of a node in the tree. In problems where the tree is deep, it may be necessary to impose an upper bound N on the level of the traversed nodes in order to limit the search. In this way, the B&D heuristic can be seen as the particular case of the recursive procedure as we fix $N = 1$.

A procedure which can be easily coded in a computer language supporting recursion, such as Pascal and C, is presented below as algorithm 2. Note that the routine B&D (L_0, W_0, n) is recursive (see step 2.2.2.1 of the algorithm) and has the parameters: length $L_0 \leqslant L$, width $W_0 \leqslant W$ and level n of the block or node in the tree. Unlike algorithm 1, the best solution produced by algorithm 2 corresponds to a 1st-order pattern not limited to 5 blocks.

Algorithm 2 Recursive procedure

Initialisation: Determine the sets P and Q to (L, W) (as defined in step 1 of algorithm 1). Make $z =$ B&D $(L, W, 1)$, that is, call the recursive routine B&D with the pallet size at level 1.

Routine B&D (L_0, W_0, n): Returns the maximum number of faces in the block (L_0, W_0) at level n:

Step 1: Determine the subsets of P and Q to (L_0, W_0), say $P(L_0)$ and $Q(W_0)$. Compute z_{lb} (8) and z_{ub} (minimum of (9) and Barnes's bound) to (L_0, W_0). If $z_{lb} = 0$ or $z_{lb} = z_{ub}$ then go to step 3, otherwise go to step 2.

Step 2: For each $(r_1, s_1) \in P(L_0)$ and $(r_2, s_2) \in P(L_0)$ such that $r_2 l + s_2 w \geqslant r_1 l + s_1 w$, and for each $(t_1, u_1) \in Q(W_0)$ and $(t_2, u_2) \in Q(W_0)$ such that $t_2 l + u_2 w \geqslant t_1 l + u_1 w$, do:

2.1: Determine the sizes (L_i, W_i), $i = 1, \ldots, 5$ (the same as step 2.1 of algorithm 1).

2.2: If L_1, L_4, W_4 and W_5 are not null and there is no overlap (as in step 2.2 of algorithm 1), then:

2.2.1: If the depth limit of the tree is not reached (that is, if $n < N$):

2.2.2.1: Then, determine $z_0 = \sum_{i=1}^{5}$ B&D $(L_i, W_i, n+1)$.

2.2.2.2: Else, determine $z_0 = \sum_{i=1}^{5} z_i$, where:

$$z_i = \max\{\lfloor L_i/l \rfloor \lfloor W_i/w \rfloor, \lfloor L_i/w \rfloor \lfloor W_i/l \rfloor\}, \quad \text{if } i = 1, \ldots, 5 \quad (10)$$

2.2.2: If $z_0 > z_{lb}$ then update z_{lb}, the maximum number of boxes obtained so far to (L_0, W_0). In this case, if $z_{lb} = z_{ub}$ then exit to step 3.

Step 3: Return the best found solution z_{lb}. If $z_{lb} = z_{ub}$ then the pattern to (L_0, W_0) is optimal.

It should be remarked that even for $N = \infty$:

(i) Algorithm 2 is heuristic, that is, it does not provide a guarantee of finding the optimal non-guillotine pattern of the MPL.

(ii) Moreover, algorithm 2 does not provide a guarantee of finding the best 1st-order non-guillotine pattern of the MPL.

Remark (ii) can be shown by the following example: Consider a pallet of size $(42, 39)$ and boxes of size $(9, 4)$. Then, $P = \{(4, 1), (3, 3), (2, 6), (1, 8), (0, 10)\}$ and $Q = \{(4, 0), (3, 3), (2, 5), (1, 7), (0, 9)\}$. Figure 2(a) illustrates the best pattern obtained by algorithm 2 without limiting the tree depth (that is, $N = \infty$) and Figure 2(b) illustrates the optimal pattern. Note that both patterns are 1st-order type, however, the optimal pattern lays out one more box. To be obtained, this pattern involves the examination of a partition $(r, 4)$ in the upper border of the pallet (see Figure 2(b)), but there is no such a partition in P. Therefore, this path cannot be followed by algorithm 2.

We can modify algorithm 2 in order to guarantee that it does find the best 1st-order non-guillotine pattern of the MPL, however, requiring higher computational efforts. This can be done as follows: In step 2, instead of examining the partitions in sets P and Q, the algorithm should examine all points of sets X and Y defined in 1 and 2 (or, similarly, the raster points of sets X' and Y' in 3 and 4). Thus, steps 2 and 2.1 should be substituted by: For each $x_1 \in X, x_2 \in X$, $y_1 \in Y$ and $y_2 \in Y$, examine the following five blocks:

$$(L_1, W_1) = (x_1, W - W_4) \quad (11)$$

$$(L_2, W_2) = (L - L_1, W - W_5) \quad (12)$$

$$(L_3, W_3) = (L - L_1 - L_5, W - W_2 - W_4) \quad (13)$$

$$(L_4, W_4) = (x_2, y_1) \quad (14)$$

$$(L_5, W_5) = (L - L_4, y_2) \quad (15)$$

It is easy to show that algorithm 2 with this modification finds the optimal pattern of Figure 2(b) for $N \geqslant 3$. Before presenting an improved version of algorithm 2, let us discuss how step 2.2 can be modified to reduce the search by discarding symmetrical patterns (note that symmetrical patterns can be discardable since they lay out the same number of boxes). Four types of symmetry can be found (Figures 3(a)–(c) depict three of them)—they are avoided if x_1, x_2, y_1 and y_2 in 11–15 are such that one of the four (mutually exclusive) rules is satisfied:

5-blocks: $x_1 > 0$, $y_1 > 0$, $x_2 > x_1$, $y_2 > y_1$ and either $x_1 + x_2 < L$ or $(x_1 + x_2 = L$ and $y_1 + y_2 \leqslant W)$.[8]

4-blocks: $x_1 > 0, y_1 > 0$ and either $(x_2 = x_1, y_2 > y_1$ and $x_1 \leqslant \lfloor L/2 \rfloor)$ or $(x_2 > x_1, y_2 = y_1$ and $y_1 \leqslant \lfloor W/2 \rfloor)$ or $(x_2 = x_1,\ y_2 = y_1,\ x_1 \leqslant \lfloor L/2 \rfloor$ and $y_1 \leqslant \lfloor W/2 \rfloor)$.

3-blocks: either $(x_1 > 0,\ y_1 = 0,\ x_2 = x_1$ and $0 < y_2 \leqslant \lfloor W/2 \rfloor)$ or $(x_1 = 0,\ y_1 > 0,\ 0 < x_2 \leqslant \lfloor L/2 \rfloor$ and $y_2 = y_1)$.

2-blocks: either $(0 < x_1 \leqslant \lfloor L/2 \rfloor,\ y_1 = 0,\ x_2 = x_1$ and $y_2 = 0)$ or $(x_1 = 0, 0 < y_1 \leqslant \lfloor W/2 \rfloor,\ x_2 = 0$ and $y_2 = y_1)$.

It is worth noting that these rules also prevent overlapping. Step 2.2.2.1 in algorithm 2 can also be modified to avoid unnecessary recursions. Before computing all B&D $(L_i, W_i,\ n+1)$, $i = 1, \ldots, 5$, we can check if the best

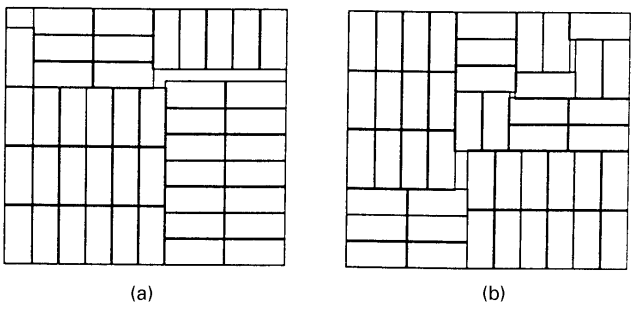

Figure 2 Example N4 with $(L, W) = (42, 39)$ and $(l, w) = (9, 4)$: (a) pattern with 44 boxes obtained by algorithm 2, (b) optimal pattern with 45 boxes obtained by algorithm 3.

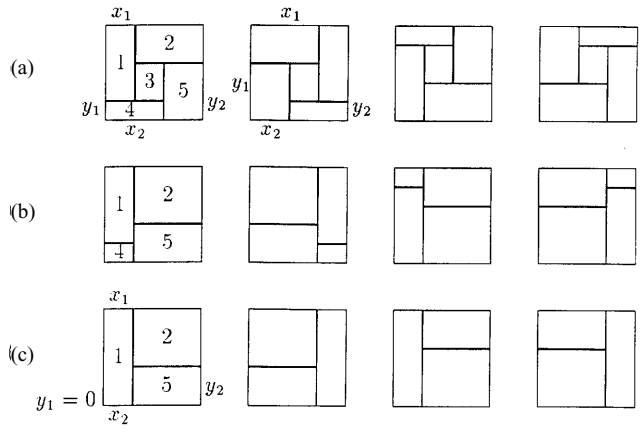

Figure 3 Examples of: (a) 5-block, (b) 4-block and (c) 3-block symmetries.

current solution to (L_0, W_0), z_{lb}, is attainable by z_0. While there exists a chance to find a solution better than z_{lb} we continue evaluating each B&D $(L_i, W_i, n+1)$; otherwise, we abandon the current examination.

The modified recursive procedure is presented below as algorithm 3. Similarly to algorithm 2, the routine B&D (L_0, W_0, n) is recursive (see step 2.1.2.1 of the algorithm) and has the parameters: length $L_0 \leqslant L$, width $W_0 \leqslant W$, and level n of the block or node in the tree. The solution produced by algorithm 3 with $N = \infty$ corresponds to the best 1st-order non-guillotine pattern to (L, W).

Algorithm 3 Modified recursive procedure

Initialisation: Determine the sets X (1) and Y (2) to (L, W). Make $z = \text{B\&D}\ (L, W, 1)$, that is, call the recursive routine B&D with the pallet size at level 1.

Routine B&D (L_0, W_0, n): Returns the maximum number of faces in the block (L_0, W_0) at level n:

Step 1: Determine the subsets of X and Y to (L_0, W_0), say $X(L_0)$ and $Y(W_0)$. Compute z_{lb} (8) and z_{lb} (minimum of (9) and Barnes's bound) to (L_0, W_0), where L^* and W^* in (9) is now defined as $L^* = w + \max\{x | x \in X(L_0)\}$ and $W^* = w + \max\{y | y \in Y(W_0)\}$. If $z_{lb} = 0$ or $z_{lb} = z_{ub}$ then go to step 3, otherwise go to step 2.

Step 2: For each $x_1 \in X(L_0)$ and $x_2 \in X(L_0)$ such that $x_2 \geqslant x_1$, and for each $y_1 \in Y(W_0)$ and $y_2 \in Y(W_0)$ such that $y_2 \geqslant y_1$, do:

2.1: If one of the four rules (5-blocks, 4-blocks, 3-blocks or 2-blocks) is hold then:

2.1.1: Determine the sizes (L_i, W_i), $i = 1, \ldots, 5$ according to 11–15.

2.1.2: If the depth limit of the tree is not reached (that is, if $n < N$):

2.1.2.1: Then, make $z_0 = 0$ and compute the upper bound z_{ub} for each (L_i, W_i), say z^i_{ub}. If $z_{lb} \geqslant \sum_{i=1,\ldots,5} z^i_{ub}$ then go to step 2.1.3 else determine $z_1 = \text{B\&D}\ (L_1, W_1, n+1)$. If $z_{lb} \geqslant z_1 + \sum_{i=2,\ldots,5} z^i_{ub}$ then go to step 2.1.3, else determine $z_2 = \text{B\&D}\ (L_2, W_2, n+1)$. If $z_{lb} \geqslant \sum_{i=1,\ldots,2} z_i + \sum_{i=3,\ldots,5} z^i_{ub}$ then go to step 2.1.3, else determine $z_3 = \text{B\&D}\ (L_3, W_3, n+1)$. If $z_{lb} \geqslant \sum_{i=1,\ldots,3} z_i + \sum_{i=4,\ldots,5} z^i_{ub}$ then go to step 2.1.3, else determine $z_4 = \text{B\&D}\ (L_4, W_4, n+1)$. If $z_{lb} \geqslant \sum_{i=1,\ldots,4} z_i + z^5_{ub}$ then go to step 2.1.3, else determine $z_0 = \sum_{i=1,\ldots,4} z_i + \text{B\&D}\ (L_5, W_5, n+1)$.

2.1.2.2: Else, determine $z_0 = \sum_{i=1}^{5} z_i$ using 10.

2.1.3: If $z_0 > z_{lb}$ then update z_{lb}, the maximum number of boxes obtained so far to (L_0, W_0). In this case, if $z_{lb} = z_{ub}$ then exit to step 3.

Step 3: Return the best found solution z_{lb}. If $z_{lb} = z_{ub}$ then the pattern to (L_0, W_0) is optimal.

Motivated by the discussion in[22], two additional modifications in algorithm 3 can be considered in order to improve its computational performance. Firstly, in step 1, there is a tradeoff between the undesirable cost of obtaining high quality lower bounds z_{lb} to (L_0, W_0) and the benefits of their savings on the search process. Our computational experiments showed that it is very effective to start with a good lower bound. Thus, for $(L_0, W_0) = (L, W)$, we obtain z_{lb} calling algorithm 1 instead of computing it from (8).

In spite of being drastically reduced by step 2.1, the tree search may still involve multiple B&D callings for the same node (L_0, W_0). After the first time B&D (L_0, W_0) is computed, we can update the lower bound z_{lb} of (L_0, W_0), store its value in the memory, and simply retrieve it each time B&D (L_0, W_0) is re-called. Instead of $O(LW)$ numbers, the memory requirements are $O(|X||Y|)$ since only the values with respect to the elements in $X \cup \{L^*\}$ and $Y \cup \{W^*\}$ must be stored (these requirements can even be diminished if we use the raster points). The lower bound z_{lb} of (L_0, W_0) is placed at the position (i_0, j_0) of an integer $(|X|+1)$ by $(|Y|+1)$-matrix, where $i_0 = \arg\max_{i \in I}\{x_i | x_i \leqslant L_0,\ x_i \in X \cup \{L^*\}\}$ and $j_0 = \arg\max_{j \in J}\{y_j | y_j \leqslant W_0, y_j \in Y \cup \{W^*\}\}$. Moreover, for $L_0, W_0 \leqslant \min\{L, W\}$, the position (j_0, i_0) is also updated since $x_{i_0} y_{j_0} = x_{j_0} y_{i_0}$ (this is true only for the normal sets). Similarly for the upper bound z_{ub} of (L_0, W_0).

It should be noted that algorithms 2 and 3 perform a backtracking search that traverse the tree without using additional information which would indicate the most promising paths to be firstly followed and would enable the search to be (eventually) more efficient. This can be accomplished, for example, utilising an evaluation function to estimate the merits of each node of the tree. This function can be defined as the sum of two components, g and h,[23] where g is the solution value of the best path to the current node, and h is an overestimate of the additional value of getting from the current node to final nodes (that is, either nodes representing sufficiently small rectangles or nodes at level N). Two difficulties arise: (i) how to define an effective overestimating function h; and (ii) how to modify the algorithms in order to perform such a best-first search without losing their simplicity? The use of best-first searches will not be considered in the present work.

Computational results

Algorithms 1 and 3 were coded in Pascal language (Delphi 2) and ran in a microcomputer (Pentium 100 Mhz). It is worth recalling that algorithm 1 corresponds to the original B&D heuristic and its solution is equal to the solution of algorithm 3 with $N = 1$. We implemented algorithm 3 in two versions, one with the normal sets X and Y defined in 1 and 2 (namely algorithm 3a), and the other with the raster point sets X' and Y' in 3 and 4 (algorithm 3b). Both versions incorporated the two modifications discussed at the end of section 3.

Initially the algorithms were applied to solve random examples with L, W, l and w uniformly sampled in the ratios L/W, l/w and LW/lw, subject to the constraints: $1 \leqslant L/W \leqslant 2$, $1 \leqslant 1/w \leqslant 4$, and either $1 \leqslant LW/lw \leqslant 51$ or $51 \leqslant LW/lw \leqslant 101$. The use of these data sets is usual in the literature.[15,16,20] The first two sets (S1 and S2) of Table 1 contain 100 randomly generated examples with respectively $1 \leqslant LW/lw \leqslant 51$ and $51 \leqslant LW/lw \leqslant 101$ (as indicated in the second and third columns of Table 1). For simplicity, each lengh L was uniformly sampled from the interval $[50, 100]$. Columns $\overline{|X|}$, $\overline{|Y|}$, $\overline{|X'|}$, $\overline{|Y'|}$, Alg., $N, \bar{z}$, Time and Opt. present respectively the mean number of elements of the normal and raster point sets, the algorithm (1, 3a or 3b), the tree depth limit, the mean solution obtained by the algorithm, the mean runtime in seconds, and the percentage of optimal solutions.

All solutions of sets S1 and S2 obtained by algorithms 3a and 3b with $N = 3$ are optimal (Table 1). To prove their optimality in examples not satisfying the condition $z_{lb} = z_{ub}$ of step 3 of the algorithms (a total of 36 examples in S1 and S2), we coded model 5–7 of section 2.1 in the modelling language GAMS[25] and ran it in the same microcomputer using the integer programming solver GAMS/OSL. For all examples, the LP-relaxation of 5–7 provided a solution less than $z + 1$, where z is the solution produced by algorithms 3a and 3b, and thus z is optimal. Note in Table 1 that, despite being much faster than algorithms 3a and 3b, algorithm 1 was unable to find the optimal solution in 15 examples of sets S1 and S2.

In order to evaluate the performance of the algorithms with examples known in the literature, sets S3 and S4 (Table 1) were taken from the data sets Cover I and Cover II[15] with respectively $1 \leqslant LW/lw \leqslant 51$ and $51 \leqslant LW/lw \leqslant 101$ (J Nelissen and G Scheithauer kindly placed these sets at our disposal). For simplicity, we consider only examples with $L, W \leqslant 1000$. In this way, set S3 has 3183 examples (out of the 8274 examples of Cover I), including the 4 hardest instances analysed in[15]. Set S4 contains 16938 examples out of the 41831 examples of Cover II.

Algorithms 3a and 3b with $N = 3$ found the optimal solutions of all examples of S3, including the 4 hardest ones, namely ST1–ST4, which are detailed in Table 2. On the other hand, the algorithms were unable to find the optimal solutions of 18 examples of set S4, even for $N = \infty$ (the best 1st-order patterns laid out one box less than the optimal patterns). One of these examples, namely N1, is detailed in Table 2. Figure 4 depicts its best 1st-order pattern (52 boxes) obtained by algorithm 3a, together with its optimal superior-order pattern (53 boxes) obtained by model 5–7. To find this pattern, the solver GAMS/OSL required more than 30 minutes of runtime on the microcomputer. Although algorithm 1 was very fast (its mean runtime in sets S1–S4 was less than 0.1 seconds), it failed to obtain the optimal solution of 11.4% of the examples of S4. Note also that, in spite of handling larger sets (the

Table 1 Computational results of sets S1–S4

Set	*Num. Exp.*	$\left[\underline{\left(\frac{LW}{lw}\right)}, \overline{\left(\frac{LW}{lw}\right)}\right]$	$\overline{\lvert X\rvert}$, $\overline{\lvert Y\rvert}$	$\overline{\lvert X'\rvert}$, $\overline{\lvert Y'\rvert}$	*Alg*	*N*	$\bar{z}$	*Time (sec)*	*Opt. (%)*
S1	100	[1, 51]	19.6, 11.5	13.8, 8.3	1	1	25.2	0.1	91
					3a	2	25.3	0.2	100
						3	25.3	0.2	100
					3b	2	25.3	0.4	100
						3	25.3	0.6	100
S2	100	[51, 101]	38.5, 23.6	29.7, 17.2	1	1	69.8	0.1	94
					3a	2	69.8	5.8	99
						3	69.9	8.5	100
					3b	2	69.8	7.4	99
						3	69.9	10.6	100
S3	3183	[1, 51]	25.9, 14.2	19.8, 11.0	1	1	36.1	0.1	95.9
					3a	3	36.2	0.2	100
					3b	3	36.2	0.2	100
S4	16938	[51, 101]	54.6, 29.7	38.4, 20.8	1	1	80.1	0.1	88.6
					3a	3	80.2	2.2	99.9
					3b	3	80.2	3.7	99.9

Table 2 Computational results of examples D1–D2, N1–N5 and ST1–ST5

Exp.	(L, W)	(l, w)	$\|X\|, \|Y\|$	*Alg*	*N*	*z*	*Time*	C_1	C_2
D1	(22, 16)	(5, 3)	16, 10	1	1	22	0.1		
				3a	3	23*	0.1	23	19
D2	(86, 82)	(15, 11)	24, 22	1	1	41	0.1		
				3a	3	42*	0.5	57	38
N1	(43, 26)	(7, 3)	35, 18	1	1	52	0.1		
				3a	3	52	1.6	276	233
N2	(87, 47)	(7, 6)	67, 27	1	1	95	0.1		
				3a	3	97*	46.3	1853	1683
N3	(153, 100)	(24, 7)	78, 35	1	1	90*	0.1		
				3a	any	90*	0.1	1	0
N4	(42, 39)	(9, 4)	27, 24	1	1	44	0.1		
				3a	3	45*	2.0	435	369
N5	(124, 81)	(21, 10)	41, 18	1	1	46	0.1		
				3a	3	47*	5.5	8340	8121
ST1	(40, 25)	(7, 3)	32, 17	1	1	46	0.1		
				3a	3	47*	2.1	981	899
ST2	(52, 33)	(9, 4)	37, 18	1	1	46	0.1		
				3a	3	47*	3.1	4090	3944
ST3	(57, 44)	(12, 5)	31, 19	1	1	40	0.1		
				3a	3	41*	0.9	453	382
ST4	(56, 52)	(12, 5)	30, 26	1	1	47	0.1		
				3a	3	48*	2.2	562	464
ST5	(300, 200)	(21, 19)	113, 51	1	1	149*	0.1		
				3a	any	149*	0.1	1	0

*Optimal solution.

normal sets are often larger than the raster point sets), algorithm 3a was (in average) faster than algorithm 3b, particularly because of the second modification discussed at the end of section 3.

Two other examples of set S4, namely N2 and N3, are detailed in Table 2. The asterisk in column z of Table 2 indicates if the solution is optimal, and the last two columns present two counters: C_1 corresponds to the number of times that the recursive routine B&D was called, and C_2 counts how many of the C_1 calls were retrieved from the memory (according to section 3). As the size of X and Y increases, the problem is expected to be more difficult to solve with algorithm 3a if the initial lower bound z_{lb} of step 1 (provided by algorithm 1) does not satisfy the condition $z_{lb} = z_{ub}$ of step 3 (as in example N3). Example N2 ($|X| = 67$ and $|Y| = 27$) was one of the most difficult to solve—in fact, as pointed out,[12] it required more than 8 hours of runtime on a HP9000/720 workstation to be optimally solved by an exact algorithm. Note that its optimal solution (97 boxes) loaded two more boxes per layer than the solution given by algorithm 1.

Table 2 also presents the computational performance of algorithms 1 and 3a for other 5 examples published in the literature—examples D1–D2 were taken from Dowsland,[24] examples N4–N5 from Nelissen,[20] and example ST5 from Scheithauer and Terno.[15] All solutions of algorithm 3a with $N = 3$ satisfy the condition $z_{lb} = z_{ub}$ and, therefore, are optimal. The pattern of example N4 was illustrated in Figure 2(b). Figures 5–7 compare the patterns for examples N5, ST3 and ST5 published[12,15,20] with the ones obtained

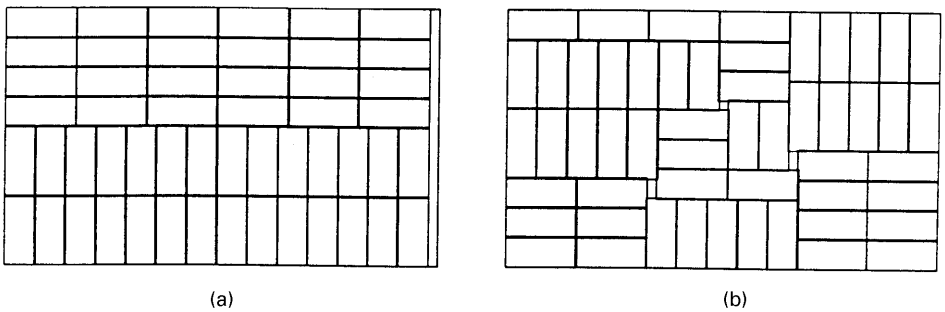

Figure 4 Example N1 with $(L, W) = (43, 26)$ and $(l, w) = (7, 3)$: (a) pattern with 52 boxes obtained by algorithm 3a with $N = 3$, (b) optimal pattern with 53 boxes obtained by model (5)–(7).

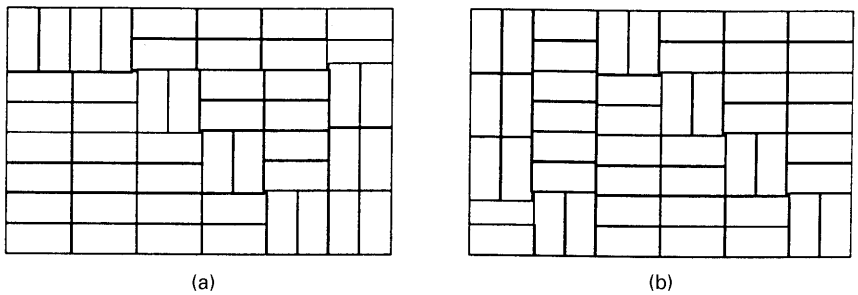

Figure 5 Example N5 with $(L, W) = (124, 81)$ and $(l, w) = (21, 10)$: (a) pattern with 47 boxes in Nelissen, (b) pattern with 47 boxes obtained by algorithm 3a with $N = 3$.

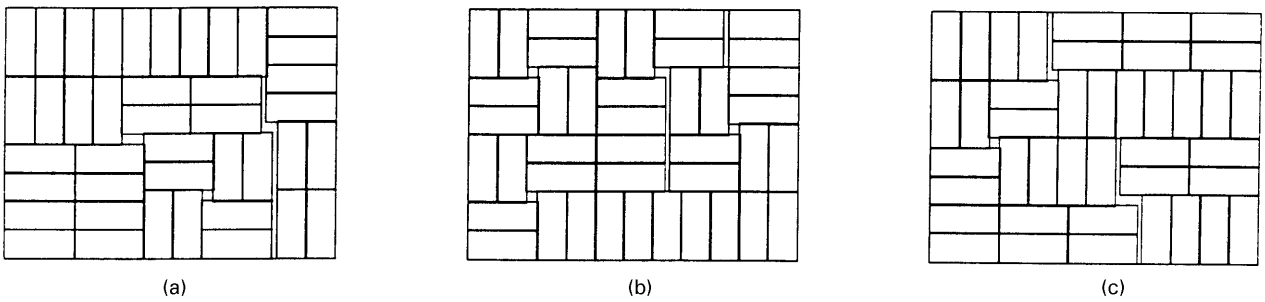

Figure 6 Example ST3 with $(L, W) = (57, 44)$ and $(l, w) = (12, 5)$: (a) pattern with 41 boxes in Nelissen, (b) pattern with 41 boxes in Scheithauer and Terno. (c) pattern with 41 boxes obtained by algorithm 3a with $N \geqslant 2$.

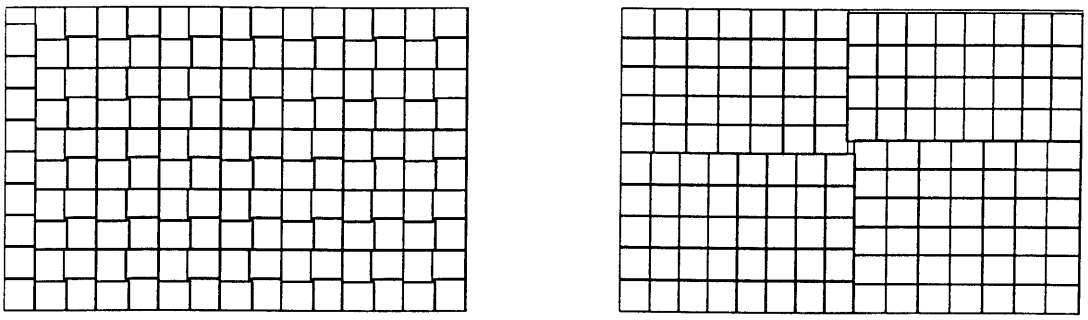

Figure 7 Example ST5 with $(L, W) = (300, 200)$ and $(l, w) = (21, 19)$: (a) pattern with 149 boxes in Scheithauer and Terno, (b) pattern with 149 boxes obtained by algorithms 1 and 3a with $N \geqslant 1$.

by algorithm 3a. Observe that the patterns in Figures 5(a) and 6(a) are superior-order non-guillotine whereas the patterns in Figures 5(b), 6(b, c) and 7(a, b) are 1st-order.

In order to evaluate the performance of algorithms 1 and 3a with instances closer to practical MPL, we generated 100 random examples sorting l and w from uniform distributions in the intervals $[200, 600]$ and $[150, 450]$ millimeters (mm) respectively. These intervals seem to be represented in the USA[26] and in Brazil.[27] Table 3 presents the results obtained by the algorithms with three standard pallets: the first two (sets S5 and S6) are recommended by ISO (International Standards Organisation) and have sizes of $(1000, 1000)$ and $(1200, 1000)$ mm, the third (set S7) is the Euro-pallet adopted by UIC (Union Internationale des Chemins de Fer) and have a size of $(1200, 800)$ mm. Contrary to the results of Tables 1 and 2, the average performance of algorithm 1 for the examples of Table 3 was surprisingly good compared to algorithm 3a. Algorithm 3a produced better solutions than algorithm 1 in only two examples of set S6—these solutions were proved to be optimal solving the LP-relaxation of 5–7.

Concluding remarks

In this paper we presented a simple and effective heuristic (algorithm 3) to solve the MPL which can easily be implemented in a computer language supporting recursion. The procedure may be seen as an extension of Bischoff and

Table 3 Computational results of sets S5–S7

Set	Num. Exp.	(L, W)	$[\underline{l}, \bar{l}]$, $[\underline{w}, \bar{w}]$	$\overline{\lvert X\rvert}$, $\overline{\lvert Y\rvert}$	Alg	N	$\bar{z}$	Time (sec)	Opt. (%)
S5	100	(1000, 1000)	[200, 600]	6.1, 6.1	1	1	7.2	0.1	100
			[150, 450]		3a	3	7.2	0.1	100
S6	100	(1200, 1000)	[200, 600]	8.2, 6.1	1	1	9.1	0.1	98
			[150, 450]		3a	3	9.2	0.1	100
S7	100	(1200, 800)	[200, 600]	8.2, 4.4	1	1	7.0	0.1	100
			[150, 450]		3a	3	7.0	0.1	100

Dowsland's heuristic. Requiring moderate computational efforts, it was able to find the optimal solutions of almost all examples analysed (only 18 examples out of more than 20 000 were not optimally solved). This suggests that the best 1st-order non-guillotine pattern is a very effective solution to the MPL.

The computational efforts of algorithm 3 may be reduced by using additional upper bounds in step 1, such as those discussed.[12,15] An interesting perspective for future research is to explore in the algorithm the use of more informed search strategies,[23] such as best-first searches, instead of depth-first or backtracking.

Acknowledgements—The authors thank Prof M Arenales and the referees for their helpful comments, and Drs J Nelissen and G Scheithauer for kindly providing their data sets. This research was partially supported by CNPq (grants #522973/95-7, #680082/95-6) and FAPESP (grant #9522-0).

References

1 Dowsland K and Dowsland W (1992). Packing problems. *Eur J Opl Res* **56**: 2–14.
2 Dychoff H and Finke U (1992). *Cutting and Packing in Production and Distribution: Typology and Bibliography.* Springer-Verlag Co.: Heidelberg.
3 Sweeney P and Paternoster E (1992). Cutting and packing problems: A categorized, application-oriented research bibliography. *J Opl Res Soc* **43**: 691–706.
4 Bischoff E and Waescher G (eds) (1995). Special issue on cutting and packing. *Eur J Opl Res* **84**: 503–712.
5 Hodgson T (1982). A combined approach to the pallet loading problem. *IIE Trans* **14**: 176–182
6 Dowsland K (1993). Packing problems. *Eur J Opl Res* **68**: 389–399.
7 Abdou G and Yang M (1994). A systematic approach for the three-dimensional palletization problem. *Int J Prod Res* **32**: 2381–2394.
8 Arenales M and Morabito R (1995). An AND/OR-graph approach to the solution of two-dimensional non-guillotine cutting problems. *Eur J Opl Res* **84**: 599–617.
9 Bischoff E and Ratcliff M (1995). Loading multiple pallets. *J Opl Res Soc* **46**: 1322–1336.
10 Morabito R and Arenales M (1994). An AND/OR-graph approach to the container loading problem. *Int Trans Opl Res* **1**: 59–73.
11 Dychoff H (1990). A typology of cutting and packing problems. *Eur J Opl Res* **44**: 145–159.
12 Nelissen J (1995). How to use structural constraints to compute an upper bound for the pallet loading problem. *Eur J Opl Res* **84**: 662–680.
13 Bischoff E and Dowsland W (1982). An application of the micro to product design and distribution. *J Opl Res Soc* **33**: 271–280.
14 Beasley J (1985). An exact two-dimensional non-guillotine tree search procedure. *Opns Res* **33**: 49–64.
15 Scheithauer G and Terno J (1996). The G4-heuristic for the pallet loading problem. *J Opl Res Soc* **47**: 511–522.
16 Dowsland K (1987). An exact algorithm for the pallet loading problem. *Eur J Opl Res* **31**: 78–84.
17 Tsai R, Malstrom E and Kuo W (1993). Three dimensional palletization of mixed box sizes. *IEE Trans* **25**: 64–75.
18 Steudel H (1979). Generating pallet loading patterns: A special case of the two-dimensional cutting stock problem. *Mgmt Sci* **10**: 997–1004.
19 Smith A and De Cani P (1980). An algorithm to optimize the layout of boxes in pallets. *J Opl Res Soc* **31**: 573–578.
20 Nelissen J (1994). Solving the pallet loading problem more efficiently. *Working paper*, Graduiertenkolleg Informatik und Technik, Aachen, August.
21 Barnes F (1979). Packing the maximum number of $m \times n$ tiles in a large $p \times q$ rectangle. *Discrete Mathe* **26**: 93–100.
22 Herz J (1972). Recursive computational procedure for two dimensional stock cutting. *IBM J Res Develop* **16**: 462–469.
23 Nilsson N (1971). *Problem Solving Methods in Artificial Intelligence*. McGraw-Hill: New York.
24 Dowsland K (1984). The three-dimensional pallet chart: An analysis of the factors affecting the set of feasible layouts for a class of two-dimensional packing problems. *J Opl Res Soc* **35**: 895–905.
25 Brooke A, Kendrick D and Meeraus A (1992). *GAMS: A user's guide, Release 2.25*. The Scientific Press, USA.
26 Wright P (1984). Pallet loading configurations for optimal storage and shipping. *Paperboard and Packing*, December, 46–49.
27 Morales S (1995). Optimização do carregamento de paletes: Uma abordagem heurística para resolver o problema do produtor. *Dissertation*, Departamento de Engenharia de Produção, Universidade Federal de São Carlos, Brazil.

Received March 1997;
accepted February 1998 after one revision