The Three-Dimensional Bin Packing Problem

Article *in* Operations Research · February 1998

DOI: 10.1287/opre.48.2.256.12386 · Source: CiteSeer

3 authors, including:

David Pisinger
Technical University of Denmark
168 PUBLICATIONS **10,771** CITATIONS

SEE PROFILE

Daniele Vigo
University of Bologna
176 PUBLICATIONS **11,301** CITATIONS

SEE PROFILE

Università degli Studi di Bologna

Dipartimento di Elettronica, Informatica e Sistemistica
Viale Risorgimento, 2 40136 - Bologna (Italy)
phone : + 39-51-6443001 fax : + 39-51-6443073

The Three-Dimensional Bin Packing Problem

Silvano Martello, David Pisinger, Daniele Vigo

Technical Report DEIS - OR - 97 - 6

The Three-Dimensional Bin Packing Problem

Silvano Martello*, David Pisinger†, Daniele Vigo*
*DEIS, Univ. of Bologna, Viale Risorgimento 2, Bologna
†DIKU, Univ. of Copenhagen, Univ.parken 1, Copenhagen

May 1997

Under revision for *Operations Research*

Abstract

The problem addressed in this paper is that of orthogonally packing a given set of rectangular-shaped boxes into the minimum number of rectangular bins. The problem is strongly NP-hard and extremely difficult to solve in practice. Lower bounds are discussed, and it is proved that the asymptotical worst-case performance of the continuous lower bound is $\frac{1}{8}$. An exact algorithm for filling a single bin is developed, leading to the definition of an exact branch-and-bound algorithm for the three-dimensional bin packing problem, which also incorporates original approximation algorithms. Extensive computational results, involving instances with up to 60 boxes, are presented: it is shown that many instances can be solved to optimality within a reasonable time limit.

1 Introduction

We are given a set of n rectangular-shaped *boxes*, each characterized by width w_j, height h_j and depth d_j ($j \in J = \{1, \ldots, n\}$), and an unlimited number of identical three-dimensional containers (*bins*) having width W, height H and depth D. The *Three-Dimensional Bin Packing Problem* (3D-BPP) consists of orthogonally packing all the boxes into the minimum number of bins. We assume that the boxes may not be rotated, i.e. that they are packed with each edge parallel to the corresponding bin edge. We also assume, without loss of generality, that all the input data are positive integers satisfying $w_j \leq W$, $h_j \leq H$ and $d_j \leq D$ ($j \in J$). No further restriction is imposed: the boxes need not be packed in layers,

and *guillotine cutting* (imposing that the packing be such that the boxes can be obtained by sequential face-to-face cuts parallel to the faces of the bin) is not required.

Problem 3D-BPP is strongly NP-hard, since it is a generalization of the well-known (one-dimensional) *Bin Packing Problem* (1D-BPP), in which a set of n positive values w_j has to be partitioned into the minimum number of subsets so that the total value in each subset does not exceed a given bin capacity W. It is clear that 1D-BPP is the special case of 3D-BPP arising when $h_j = H$ and $d_j = D$ for all $j \in J$. Another important related problem arises when $d_j = D$ for all $j \in J$: we have in this case the *Two-Dimensional Bin Packing Problem* (2D-BPP), calling for the determination of the minimum number of identical rectangular bins of size $W \times H$ needed to pack a given set of rectangles of sizes $w_j \times h_j$ $(j \in J)$.

For a general classification of packing and loading problems we refer to Dyckhoff [7], Dyckhoff and Finke [8], and Dyckhoff, Scheithauer and Terno [9]. The 3D-BPP is closely related to other container loading problems:

- *Knapsack Loading.* In the knapsack loading of a container each box has an associated profit, and the problem is to choose a subset of the boxes that fits into a single container (bin) so that maximum profit is loaded. If the profit of a box is set to its volume, this problem corresponds to the minimization of wasted space. Heuristics for the knapsack loading problem have been presented in Gehring, Menscher and Meyer [10] and Pisinger [17].

- *Container Loading.* In this version, all the boxes have to be packed into a single container (bin), which does however have an infinite length. The problem is thus to find a feasible solution such that the length to which the bin is filled is minimized. The first heuristic for the container loading problem was presented by George and Robinson [11], but several variants have been presented since then. Bischoff and Marriott [2] compare 14 different heuristics based on the George-Robinson framework.

- *Bin Packing.* The 3D-BPP calls for a solution where all the boxes are packed into the bins, but in contrast to the container loading problem, all the bins have finite dimensions, and the objective is to find a solution using the smallest possible number of bins. An approximation algorithm for the three-dimensional bin-packing problem was presented by Scheithauer [18]. Chen, Lee and Shen [3] consider a generalization of the problem, where the bins may have different dimensions. An integer programming formulation is developed and a small instance with $n = 6$ boxes is solved to optimality using an MIP solver.

To our knowledge no algorithms for the exact solution of 3D-BPP have been published, thus we start this presentation by considering a number of lower bounds. In Section 2 we

determine the worst-case behaviour of the so-called continuous lower bound for 3D-BPP. It is proved that the asymptotical worst-case behavior of the continuous bound is 1/8, and a constructive algorithm is presented which meets this bound. New bounds are introduced and analyzed in Section 3. In Section 4 we present an exact algorithm for selecting a subset of boxes which can be packed into a single bin by maximizing the total volume packed. These results are used in Section 5 to obtain two approximation algorithms and an exact branch-and-bound algorithm. An extensive computational testing of these algorithms is presented in Section 6, showing that the exact algorithm is able to solve instances with up to 60 boxes to optimality within reasonable time.

In the following we will denote by Z the optimal solution value of 3D-BPP. The volume of box j will be denoted by $v_j = w_j h_j d_j$, the total volume of the boxes in J by $V = \sum_{j=1}^{n} v_j$, and the bin volume by $B = WHD$.

2 The continuous lower bound

An obvious lower bound for 3D-BPP comes from a relaxation in which each box j is cut into $w_j h_j d_j$ unit-length cubes, thus producing a *continuous lower bound*

$$L_0 = \left\lceil \frac{\sum_{j=1}^{n} v_j}{B} \right\rceil \tag{1}$$

L_0 can clearly be computed in $O(n)$ time.

We next examine the worst-case behaviour of L_0. Let $Z(I)$ be the value of the optimal solution to an instance I of a minimization problem, $L(I)$ the value provided by a lower bound L and let $\rho(I) = L(I)/z(I)$. The *absolute worst-case performance ratio* of L is then defined as the smallest real number ρ such that $\rho(I) \geq \rho$ for any instance I of the problem. The *asymptotic worst-case performance ratio* of L is instead the smallest real number ρ^{∞} such that there is a (large) $N \in Z^{+}$ for which $\rho(I) \geq \rho^{\infty}$ for all instances I satisfying $Z(I) \geq N$.

It has been proved by Martello and Vigo [16] that, for the two-dimensional bin packing problem, the continuous lower bound ($\lceil \sum_{j=1}^{n} w_j h_j/(WH) \rceil$) has absolute worst-case performance ratio equal to $\frac{1}{4}$. The next theorem defines, for the three-dimensional case, the asymptotic worst-case performance of L_0.

Theorem 1 *The asymptotic worst-case performance ratio of lower bound L_0 is $\frac{1}{8}$.*

Proof. We prove the thesis by introducing a heuristic algorithm which, given an instance I of 3D-BPP with a sufficiently large solution value $Z(I)$, produces a feasible solution requiring a number of bins arbitrarily close to $8L_0(I)$.

The heuristic applies, at each iteration, an algorithm (called H2D hereafter) proposed by Martello and Vigo [16] for the two-dimensional bin packing problem. Given an instance $\widetilde{I}$ of 2D-BPP, algorithm H2D determines a feasible solution requiring, say, $U(\widetilde{I})$ bins such that the first $U(\widetilde{I})-2$ of them are filled, on average, to at least one quarter of their area, i.e.,

$$\sum_{i=1}^{U(\widetilde{I})-2} A_i \geq (U(\widetilde{I})-2)\frac{HW}{4},$$

where A_i denotes the total area occupied by the rectangles packed by H2D into bin i. The heuristic algorithm for 3D-BPP works as follows.

1. Partition the boxes, according to their depth, into a number $q = \lceil \log D \rceil$ of subsets $J_0, \ldots, J_{q-1}$, using the following rule:

$$\text{box } j \text{ is assigned to set } J_i \iff \frac{D}{2^{i+1}} \leq d_j < \frac{D}{2^i}$$

   and boxes with $d_j = D$ are assigned to set J_0. In other words, J_0 contains all the boxes with a depth of at least $D/2$, J_1 contains all the boxes with a depth less than $D/2$ and at least $D/4$, and so on.

2. For each nonempty subset J_i do the following:

   - let $\widetilde{I}_i$ be the instance of 2D-BPP defined by $|J_i|$ rectangles having sizes w_j and h_j for each $j \in J_i$, and bin sizes W and H;
   - apply algorithm H2D to $\widetilde{I}_i$, thus obtaining a solution of value $U(\widetilde{I}_i) = u_i + 2$ with the first u_i bins filled, on average, to one quarter of their area;
   - derive the corresponding three-dimensional solution using $u_i + 2$ *bin slices* with width W, height H and depth $D/2^i$. Observe that the first u_i slices are filled, on average, to one eighth of their volume, i.e.,

$$\sum_{k=1}^{u_i} V_k \geq \frac{1}{8} u_i \frac{B}{2^i}$$

     where V_k denotes the total volume occupied by the boxes packed into bin slice k. Call *fractional* bin slices the last two (possibly less filled) slices.

3. Observe that all the bin slices have width W, height H and a depth which is a "power-of-2" fraction of D. Hence consider all the bin slices, part from the fractional ones, according to decreasing depth and combine them into full bins of depth D by simply filling up the bins with slices from one end. This will give a number u of bins which all have a filling of no less than $\frac{B}{8}$, plus at most one possibly less filled bin, $u+1$.

4. Consider now the fractional bin slices and combine them, as done in Step 3, into full bins of depth D. Observe that at most four new bins are needed, since there are two such slices per subset and their total depth is

$$\sum_{i=0}^{q-1} 2\frac{D}{2^i} \leq 4D$$

We have thus obtained a solution which uses $u+5$ bins. The total volume of all the boxes is $V = \sum_{j=1}^{n} v_j \geq u\frac{B}{8}$, and we have

$$L_0(I) = \left\lceil \frac{V}{B} \right\rceil \geq \frac{u}{8} \geq \frac{Z(I)}{8} - 5 \tag{2}$$

since obviously $Z(I) \leq u+5$.

To show that the bound is tight it is sufficient to consider an instance in which n is a multiple of eight, $W = H = D = a$ (where a is a sufficiently large even value) and, for each $j \in J$, $w_j = h_j = d_j = \frac{a}{2}+1$. The optimal solution value is clearly $Z = n$. If we denote with $\frac{a^3}{8} + \Delta$ the volume of a box, we have $L_0 = \lceil \frac{n}{8} + \frac{n\Delta}{a^3} \rceil = \frac{n}{8}+1$, so the ratio L_0/Z is arbitrarily close to $\frac{1}{8}$ for sufficiently large n. □

As previously mentioned, a variant of the problem may admit that boxes are rotated in order to obtain a better filling. Given any instance I, let $Z^r(I)$ denote the solution value in such hypothesis. Since $Z^r(I) \leq Z(I)$, we immediately have from (2) that $L_0(I) \geq \frac{Z^r(I)}{8} - 5$. On the other hand, the tightness example above holds for this variant too. Hence we have

Corollary 1 *The asymptotic worst-case performance ratio of lower bound L_0 is $\frac{1}{8}$ even if rotation of the boxes (by any angle) is allowed.*

3 New lower bounds

The continuous lower bound of the previous section is likely to produce tight values when the box sizes are small with respect to the bin size. The lower bound presented in this and the next section is better suited to the cases in which there are relatively large boxes.

Our first bound is obtained by reduction to the one-dimensional case. Let

$$J^{WH} = \left\{ j \in J : w_j > \frac{W}{2} \text{ and } h_j > \frac{H}{2} \right\} \tag{3}$$

By this definition, boxes of J^{WH} can possibly be packed into the same bin only by placing them one above the other. Hence, the relaxed instance of 3D-BPP consisting of only such boxes is equivalent to an instance of 1D-BPP defined by the values d_j ($j \in J^{WH}$) with bin

size D. A valid lower bound for 3D-BPP is thus given by any valid lower bound for 1D-BPP. In particular, we consider the following bound, introduced in Martello and Toth [15] and Dell'Amico and Martello [6]:

$$L_1^{WH} = \left|\left\{j \in J^{WH} : d_j > \frac{D}{2}\right\}\right| + \max_{1 \le p \le \frac{D}{2}} \left\{ \left\lceil \frac{\sum_{j \in J_s(p)} d_j - (|J_\ell(p)|D - \sum_{j \in J_\ell(p)} d_j)}{D} \right\rceil, \left\lceil \frac{|J_s(p)| - \sum_{j \in J_\ell(p)} \lfloor \frac{D-d_j}{p} \rfloor}{\lfloor \frac{D}{p} \rfloor} \right\rceil \right\} \quad (4)$$

where

$$J_\ell(p) = \{j \in J^{WH} : D - p \ge d_j > \tfrac{D}{2}\} \quad (5)$$

$$J_s(p) = \{j \in J^{WH} : \tfrac{D}{2} \ge d_j \ge p\} \quad (6)$$

Analogous lower bounds, L_1^{WD} and L_1^{HD}, may be obtained by defining $J^{WD} = \{j \in J : w_j > \frac{W}{2} \text{ and } d_j > \frac{D}{2}\}$ and $J^{HD} = \{j \in J : h_j > \frac{H}{2} \text{ and } d_j > \frac{D}{2}\}$ and determining L_1^{WD} and L_1^{HD} through congruent modification of (3)-(6). We have thus proved the following

Theorem 2 *A valid lower bound for 3D-BPP is*

$$L_1 = \max\{L_1^{WH}, L_1^{WD}, L_1^{HD}\} \quad (7)$$

where L_1^{WH} is defined by (3)-(6), while L_1^{WD} (resp. L_1^{HD}) are obtained from (3)-(6) by interchanging d_j with h_j (resp. w_j) and D with H (resp. W).

The above bound can be computed in $O(n^2)$ time, since it has been proved in Martello and Toth [15] and Dell'Amico and Martello [6] that only the p values corresponding to actual box sizes need be considered, so the computation of (4) may be performed in $O(n^2)$ time.

Lower bounds L_0 and L_1 do not dominate each other. For the instance introduced in the tightness proof of Theorem 1 we have $\{j \in J^{WH} : d_j > \frac{D}{2}\} = J$ so from (4) we obtain $L_1 = n\ (= Z)$, while $L_0 = \frac{n}{8} + 1$. Now consider a similar instance with n multiple of eight, W, H and D even and, for each $j \in J$, $w_j = \frac{W}{2}$, $h_j = \frac{H}{2}$ and $d_j = \frac{D}{2}$. In this case $Z = \frac{n}{8} = L_0$ whereas $L_1 = 0$, since $J^{WH} = J^{WD} = J^{HD} = \emptyset$. The latter instance also shows that the worst-case performance of L_1 can be arbitrarily bad.

A better bound, which explicitly takes into account the three box dimensions, is provided by the following theorem.

Theorem 3 *Given any pair of integers (p, q), with $1 \le p \le \frac{W}{2}$ and $1 \le q \le \frac{H}{2}$, define*

$$K_v(p,q) = \{j \in J : w_j > W - p \text{ and } h_j > H - q\} \quad (8)$$

$$K_\ell(p,q) = \{j \in J \setminus K_v(p,q) : w_j > \tfrac{W}{2} \text{ and } h_j > \tfrac{H}{2}\} \quad (9)$$

$$K_s(p,q) = \{j \in J \setminus (K_v(p,q) \cup K_\ell(p,q)) : w_j \ge p \text{ and } h_j \ge q\} \quad (10)$$

A valid lower bound for 3D-BPP is

$$L_2^{WH}(p,q) = L_1^{WH} + \max\left\{0, \left\lceil \frac{\sum_{j\in K_\ell(p,q)\cup K_s(p,q)} v_j - (D\cdot L_1^{WH} - \sum_{j\in K_v(p,q)} d_j)WH}{B} \right\rceil\right\} \quad (11)$$

Proof. We will show that $L_2^{WH}(p,q)$ is a valid lower bound for the relaxed instance of 3D-BPP consisting of only the boxes in $K_v(p,q) \cup K_\ell(p,q) \cup K_s(p,q)$. For any pair (p,q), $K_v(p,q) \cup K_\ell(p,q)$ coincides with set J^{WH} (see (3)), so L_1^{WH} is a valid lower bound on the number of bins needed for such boxes. The second term in (11) increases this value through a lower bound on the number of additional bins needed for the boxes in $K_s(p,q)$. To this end observe that a box of $K_\ell(p,q) \cup K_s(p,q)$ and one of $K_v(p,q)$ could be packed into the same bin only by placing them one behind the other. In other words, with respect to this relaxed instance, any box of $K_v(p,q)$ of sizes $w_j \times h_j \times d_j$ can be seen as a larger *equivalent box* of sizes $W \times H \times d_j$. Hence, the total volume of the L_1^{WH} bins which can be used for the boxes in $K_s(p,q)$ is given by BL_1^{WH} minus the volume of the equivalent boxes of $K_v(p,q)$ minus the volume of the boxes of $K_\ell(p,q)$. It follows that at least

$$\max\left\{0, \left\lceil \frac{\sum_{j\in K_s(p,q)} v_j - (B\cdot L_1^{WH} - WH\sum_{j\in K_v(p,q)} d_j - \sum_{j\in K_\ell(p,q)} v_j)}{B} \right\rceil\right\} \quad (12)$$

additional bins are needed for the boxes of $K_s(p,q)$, hence the thesis. □

Corollary 2 *A valid lower bound for 3D-BPP is*

$$L_2 = \max\{L_2^{WH}, L_2^{WD}, L_2^{HD}\} \quad (13)$$

where

$$L_2^{WH} = \max_{1\le p\le \frac{W}{2}; 1\le q\le \frac{H}{2}} \left\{L_2^{WH}(p,q)\right\} \quad (14)$$

and L_2^{WD} (resp. L_2^{HD}) are obtained from (8)-(11) and (14) by interchanging h_j (resp. w_j) with d_j and H (resp. W) with D. Bound L_2 can be computed in $O(n^2)$ time.

Proof. The validity of L_2 is immediate from Theorem 3. We prove the second part of the thesis by showing how to compute L_2^{WH} in $O(n^2)$ time. First observe that L_1^{WH} is independent of p and q, hence it can be computed once (in $O(n^2)$ time).

We now show that in the computation of (14) it is sufficient to consider the values of p and q which correspond to distinct values of w_j and h_j, respectively. Indeed, given any p value, let q_1 and q_2 (with $q_1 < q_2$) be two distinct q values such that $K_s(p,q_1) = K_s(p,q_2)$, and note that the increase of q from q_1 to q_2 may cause one or more boxes to move from $K_\ell(p,q)$ to $K_v(p,q)$, i.e., $K_\ell(p,q_2) = K_\ell(p,q_1) \setminus K$ and $K_v(p,q_2) = K_v(p,q_1) \cup K$. Hence,

$L_2^{WH}(p,q_1) \leq L_2^{WH}(p,q_2)$ since for each box $i \in K$ the value of the numerator in (11) is increased by $d_i(WH - w_i h_i) \geq 0$. Therefore, only distinct values $q = h_j \leq \frac{H}{2}$ need be considered since they induce different sets $K_s(p,q)$, while all the intermediate q values produce dominated lower bounds. Analogously, we obtain that for any q value only the values $p = w_j \leq \frac{W}{2}$ need be considered.

We conclude the proof by showing that given a p value, the computation of the bounds $L_2(p,q)$ for all q may be parametrically performed in $O(n)$ time. Indeed, let us assume that boxes are renumbered according to nondecreasing h_j values. The determination of the initial sets $K_v(p,h_1), K_\ell(p,h_1)$ and $K_s(p,h_1)$, as well as the computation of $L_2^{WH}(p,h_1)$ may be clearly performed in $O(n)$ time. As to the remaining q values, the computation of the corresponding bounds simply requires the updating of $\sum_{j \in K_\ell(p,q) \cup K_s(p,q)} v_j$ and $\sum_{j \in K_v(p,q)} d_j$. Indeed $K_v(p,q) \cup K_\ell(p,q) = J^{WH}$ is invariant while, for each new q value, some boxes may move from $K_\ell(p,q)$ to $K_v(p,q)$, and some new boxes may enter $K_s(p,q)$. Hence, for each p value the overall computation requires $O(n)$ time. □

Although the worst-case time complexity of L_2 is equal to that of L_1, it should be noted that the computational effort required to compute L_2 is considerably greater than that required by L_1. However,

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

Proof. For $p = q = 1$ we have $K_v(p,q) = \{j \in J : w_j = W \text{ and } h_j = H\}$ and $K_v(p,q) \cup K_\ell(p,q) \cup K_s(p,q) = J$. Hence, from (11)

$$L_2^{WH}(1,1) = L_1^{WH} + \max\left\{0, \left\lceil \frac{\sum_{j \in J} v_j - B L_1^{WH}}{B} \right\rceil\right\} = \max\left\{L_1^{WH}, \left\lceil \frac{\sum_{j \in J} v_j}{B} \right\rceil\right\} \qquad (15)$$

from which $L_2^{WH} \geq \max\left(L_1^{WH}, L_0\right)$. Analogously we have $L_2^{WD} \geq \max\left(L_1^{WD}, L_0\right)$ and $L_2^{HD} \geq \max\left(L_1^{HD}, L_0\right)$. □

4 Filling a single bin

In Section 5 we describe an enumerative algorithm for the exact solution of 3D-BPP. This algorithm repeatedly solves associated subproblems in which a given subset $\overline{J}$ of boxes has to be packed, if possible, into a single bin. Also this problem is NP-hard in the strong sense, since solving a special case in which all the boxes have the same depth $d_j = D$ and the same height $h_j = 1$ answers the question whether an instance of the one-dimensional bin packing problem admits a solution requiring no more than H bins. In this section we

describe a branch-and-bound algorithm for the exact solution of a maximization version of the problem, which is also used within one of the heuristic algorithms presented in Section 5.3.

The problem we consider is that of selecting a subset $\overline{J}' \subseteq \overline{J}$ of boxes and assigning coordinates (x_j, y_j, z_j) to each box $j \in \overline{J}'$ such that no box goes outside the bin, no two boxes overlap, and the total volume of the boxes in $\overline{J}'$ is a maximum. For the two-dimensional case, a similar problem has been considered in Hadjiconstatinou and Christofides [12]. In the following we present a non trivial generalization of the two-dimensional approach to the three-dimensional case, and give an effective algorithm for this problem. We assume in the following that the origin of the coordinate system is in the left-bottom-back corner of the bin.

At each node of the branch-decision tree, described in more detail in Section 4.2, a current partial solution, which packs the boxes of a certain subset $I \subset \overline{J}$, is increased by selecting in turn each box $j \in \overline{J} \setminus I$, and generating descendant nodes by placing j into all the admissible points. By placing a box into a point p we mean that its left-bottom-back corner is positioned at p.

Let (x_p, y_p, z_p) be the coordinates of point p: obviously box j cannot be placed at p if $x_p + w_j > W$ or $y_p + h_j > H$ or $z_p + d_j > D$. The set of admissible points to be considered may be further drastically reduced through the following properties.

Property 1 *Any packing of a bin can be replaced by an equivalent packing where no box may be moved leftwards, downwards or backwards.*

Proof. Obvious. A similar property was observed by Christofides and Witlock [4] for the two-dimesional case. □

Property 2 *An ordering of the boxes in an optimal solution exists such that*

$$x_i + w_i \leq x_j \quad \text{or} \quad y_i + h_i \leq y_j \quad \text{or} \quad z_i + d_i \leq z_j \tag{16}$$

if $i < j$.

Proof. Given any optimal solution, define an associated digraph with a vertex for each box and an arc from vertex i to vertex j if and only if (16) holds. It is clear that the resulting digraph is acyclic, since otherwise we would have a "cycle" of boxes in which two boxes are both one to the right of the other, or one above the other, or one behind the other. It is

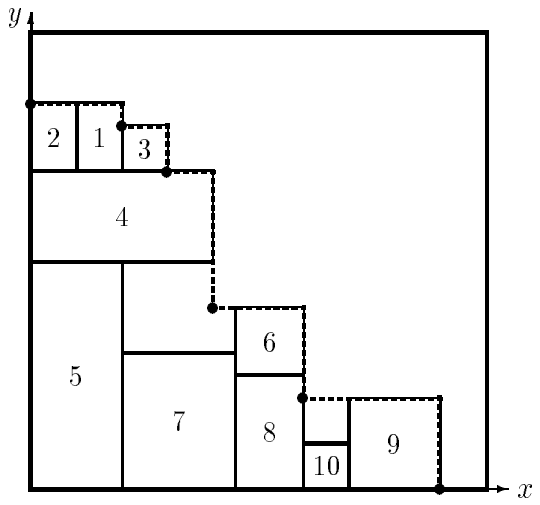

Figure 1: Two-dimensional single bin filling. The envelope associated with the placed boxes is marked by a dashed line, while black points indicate corner points in $\widehat{C}(I)$.

well-known that the vertices of an acyclic digraph can be re-numbered so that $i < j$ if arc (i, j) exists. □

It follows that the enumeration scheme for filling a single bin can be limited to only generate solutions which: (i) satisfy Property 1, and (ii) are such that the sequence in which the boxes are assigned (starting from the root node) constitute a box numbering satisfying Property 2.

An important consequence of (ii) is that at any decision node, where the boxes of I are already packed and box $j \in \overline{J} \setminus I$ is selected for branching, j may only be placed at points p such that no box of I has some part right of p or above p or in front of p. In other words, the boxes of I define an "envelope" separating the two regions where the boxes $\overline{J} \setminus I$ may (resp. may not) be placed. More formally, a new box may only be placed in the set

$$S(I) = \{(x, y, z) : \forall i \in I, x \geq x_i + w_i \text{ or } y \geq y_i + h_i \text{ or } z \geq z_i + d_i\} \tag{17}$$

Figure 1 shows, for the two-dimensional case (in which the feasible region is $\widehat{S}(I) = \{(x, y) : \forall i \in I, x \geq x_i + w_i \text{ or } y \geq y_i + h_i\}$), a set of already placed boxes and the corresponding envelope. Observe that, due to Property 1, the next box j may only be placed at points where the slope of the envelope changes from vertical to horizontal (black points in Figure 1): such points are called in the following *corner points* of the envelope. In the next subsection we show how the set of feasible corner points can be efficiently determined.

4.1 Finding possible positions for placing a box

We solve this problem in two stages, starting from the two-dimensional case.

Assume, for the moment, that $d_j = D = 1$ for each box j so that two-dimensional packing of the x-y faces is considered. Given a box set $\widehat{I}$, it is quite easy to find, in two dimensions, the set $\widehat{C}(\widehat{I})$ of corner points of the envelope associated with $\widehat{S}(\widehat{I})$. Assume that the boxes are ordered according to their *end-points* $(x_j + w_j, y_j + h_j)$, so that the values of $y_j + h_j$ are nonincreasing, breaking ties by the largest value of $x_j + w_j$ (see Figure 1). The following algorithm for determining the corner points set, consists of three phases: First, *extreme* boxes are found, i.e., boxes whose end-point coincides with a point where the slope of the envelope changes from horizontal to vertical. In the second phase, corner points are defined at intersections between the lines leading out from end-points of extreme boxes. Finally, infeasible corners, where no further box of $\overline{J} \setminus I$ can fit, are removed. Thus we get the following algorithm:

algorithm 2D-CORNERS:
begin
  if $\widehat{I} = \emptyset$ **then** $\widehat{C}(\widehat{I}) := \{(0,0)\}$ and **return**;
  comment: Phase 1: identify the extreme boxes $e_1, \ldots, e_m$;
  $\overline{x} := 0$;
  $m := 0$;
  for $j := 1$ **to** $|\widehat{I}|$ **do**
    if $(x_j + w_j > \overline{x})$ **then**
      begin
        $m := m + 1$;
        $e_m := j$;
        $\overline{x} := x_j + w_j$
      end;
  comment: Phase 2: determine the corner points;
  $\widehat{C}(\widehat{I}) := \{(0, y_{e_1} + h_{e_1})\}$;
  for $j := 2$ **to** m **do** $\widehat{C}(\widehat{I}) := \widehat{C}(\widehat{I}) \cup \{(x_{e_{j-1}} + w_{e_{j-1}}, y_{e_j} + h_{e_j})\}$;
  $\widehat{C}(\widehat{I}) := \widehat{C}(\widehat{I}) \cup \{(x_{e_m} + w_{e_m}, 0)\}$;
  comment: Phase 3: remove infeasible corner points;
  for each $(x'_j, y'_j) \in \widehat{C}(\widehat{I})$ **do**
    if $x'_j + \min_{i \in \overline{J} \setminus I}\{w_i\} > W$ or $y'_j + \min_{i \in \overline{J} \setminus I}\{h_i\} > H$ **then** $\widehat{C}(\widehat{I}) := \widehat{C}(\widehat{I}) \setminus \{(x'_j, y'_j)\}$
end.

Consider the example in Figure 1: the extreme boxes are 1, 3, 4, 6 and 9, and the resulting corner points are indicated by filled circles; Phase 3 could remove some of the first and/or last corner points.

The time complexity of 2D-CORNERS is $O(|\widehat{I}|)$, plus $O(|\widehat{I}| \log |\widehat{I}|)$ for the initial box

sorting.

Assume that the resulting corner points are $\widehat{C}(\widehat{I}) = \{(x'_1, y'_1), \ldots, (x'_\ell, y'_\ell)\} \neq \emptyset$. Then the area occupied by the the envelope is

$$A(\widehat{I}) = x'_1 H + \sum_{i=2}^{\ell} (x'_i - x'_{i-1}) y'_{i-1} + (W - x'_\ell) y'_\ell \tag{18}$$

where the first (resp. last) term is nonzero whenever the first (resp. last) corner point found in Phase 2 has been removed in Phase 3. In the special case where $\widehat{C}(\widehat{I}) = \emptyset$, we obviously set $A(\widehat{I}) = WH$.

The algorithm above can be used to determine the set $C(I)$ of corner points in three dimensions, where I is the set of three-dimensional boxes currently packed into the bin. One may apply algorithm 2D-CORNERS for $z = 0$ and for each distinct z coordinate where a box of I ends, by increasing values. For each such coordinate z', 2D-CORNERS can be applied to the subset of those boxes $i \in I$ which end after z', i.e., such that

$$z_i + d_i > z' \tag{19}$$

adding the resulting corner points to $C(I)$. In this way, one may however obtain some false corner points, since they are corner points in the two-dimensional case, but not in the three-dimensional case (see, e.g., Figure 2). Such points are easily removed by dominance, where we say that a corner point (x'_a, y'_a, z'_a) dominates another corner point (x'_b, y'_b, z'_b) that is "hidden" behind it. Formally this may be written as

$$x'_a = x'_b \;\text{ and }\; y'_a = y'_b \;\text{ and }\; z'_a < z'_b \tag{20}$$

where we have equalities in the first two expressions since (19) ensures that all the boxes in front of z'_k are chosen, and thus no corner point will be generated inside the three-dimensional envelope. This is done in the following algorithm, where the generation of corner points ends as soon as a z coordinate is found such that no further box could be placed after it.

algorithm 3D-CORNERS:
begin
  if $I = \emptyset$ **then** $C(I) := \{(0,0,0)\}$ and **return**;
  $T := \{0\} \cup \{z_i + d_i : i \in I\}$;
  sort T by increasing values, and let $T = \{z'_1, \ldots, z'_r\}$;
  $C(I) := \emptyset$;
  $\widehat{C}(\widehat{I}_0) := \emptyset$;
  $k := 1$;
  while $k \leq r$ and $z'_k + \min_{i \in \overline{J} \setminus I}\{d_i\} \leq D$ **do**

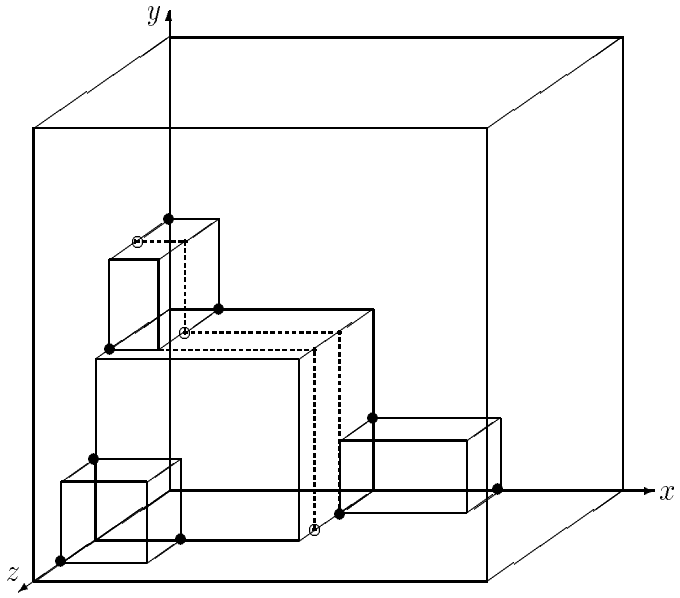

Figure 2: Three-dimensional single bin filling. Corner points $C(I)$ are found by applying algorithm 2D-CORNERS five times, one for each value of z'_k. True corner points are filled circles, false corner points are empty circles.

begin
  $\widehat{I}_k := \{i \in I : z_i + d_i > z'_k\}$;
  apply 2D-CORNERS to $\widehat{I}_k$ yielding $\widehat{C}(\widehat{I}_k)$;
  (**comment**: add true corner points to $C(I)$)
  for each $(x'_j, y'_j) \in \widehat{C}(\widehat{I}_k)$ **do**
    if $(x'_j, y'_j) \notin \widehat{C}(\widehat{I}_{k-1})$ **then** $C(I) := C(I) \cup \{(x'_j, y'_j, z'_k)\}$;
  $k := k + 1$
end
end.

The time complexity of 3D-CORNERS is $O(n^2)$. Indeed, there are at most $|I| + 1$ distinct z coordinates in T, and each set $C(\widehat{I}_k)$ is derived in $O(|\widehat{I}_k|)$ time, since the sorting of the boxes can be done once and for all at the beginning of the algorithm. For each z'_k value, the check of corner points prior to their addition to $C(I)$ requires $O(|\widehat{C}(\widehat{I}_{k-1})|)$ time in total, since both $\widehat{C}(\widehat{I}_{k-1})$ and $\widehat{C}(\widehat{I}_k)$ are ordered by increasing x'_j and decreasing y'_j. The overall complexity follows from the observation that both $|I|$ and $|\widehat{C}(\widehat{I}_k)|$ are $O(n)$.

Assuming that k^* is the index of the last $\widehat{I}_k$ generated by 3D-CORNERS, the volume $V(I)$ occupied by the envelope associated with I is then

$$V(I) = \sum_{k=2}^{k^*} (z'_k - z'_{k-1}) A(\widehat{I}_{k-1}) + (D - z'_{k^*}) A(\widehat{I}_{k^*}) \tag{21}$$

where the last term is nonzero whenever $\overline{k} < r$.

4.2 A branch-and-bound algorithm for filling a single bin

We can now easily derive, in a recursive way, a branch-and-bound algorithm for finding the best filling of a single bin using boxes from a given set $\overline{J}$. Initially no box is placed, so $C(\emptyset) = \{(0,0,0)\}$. At each iteration, given the set $I \subset \overline{J}$ of currently packed boxes, set $C(I)$ is determined through 3D-CORNERS, together with the corresponding volume $V(I)$. If F is the total volume achieved by the current best filling, we may backtrack whenever

$$\sum_{i \in I} v_i + \Big(B - V(I)\Big) \leq F \tag{22}$$

since even if the remaining volume was completely filled, we would not improve on F.

If no more boxes fit into the bin (i.e., if $C(I) = \emptyset$) we possibly update F, and backtrack. Otherwise for each position $(x', y', z') \in C(I)$ and for each box $j \in \overline{J} \setminus I$, we assign the box to this position (provided that its end points do not exceed the bin limits), and call the procedure recursively.

The best performance of the branch-and-bound algorithm was obtained when the boxes were a-priori ordered according to nonincreasing volumes.

4.3 Small instances

Although the branch-and-bound approach based on 3D-CORNERS considerably limits the enumeration compared to a naive technique trying all placings of boxes, practical experiments have shown that the algorithm is very time consuming. However, in the branch-and-bound algorithm decribed in the next section the single bin filling has often to be solved for a small subset $\overline{J}$ of boxes. Thus a specific procedure was derived for solving the subproblem when $|\overline{J}| \leq 4$, through direct evaluation of all possible placings.

For the case of two boxes, there are only three arrangements in which the boxes may be placed with respect to each other: one beside the other, one above the other, one behind the other. Thus these three arrangements are simply tested.

With three boxes it is obvious that a guillotine packing always exists. Thus the problem may be reduced to that of placing two boxes at one side of the cut, and the remaining box at the other side. Hence, there are three ways in which the boxes can be partitioned, and for each partition the cut can be made in three orthogonal orientations: the two boxes at one side of the cut can then be handled as the packing problem with two boxes.

With four boxes the case is more involved, since non-guillotine packing is also possible. If the boxes are guillotine packed, the first cut will either separate one box from the remaining

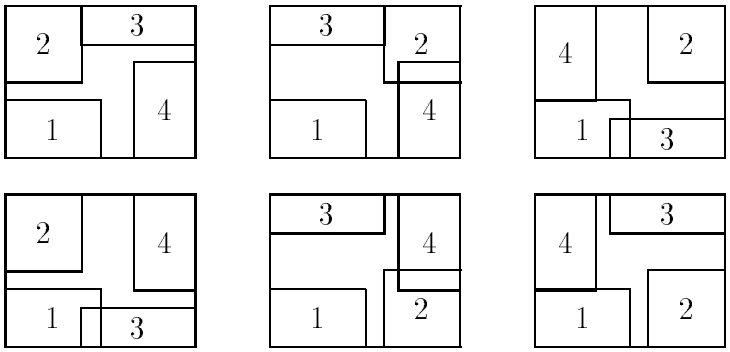

Figure 3: Non-guillottine patterns with four boxes on the $W - H$ plane (two feasible).

three, or it will separate two boxes from the other two. In the first case there are four different partitions: for each partition and for each orientation of the cut, the placing of the three boxes can be handled as the packing problem In the second case there are three different partitions: for each partition and for each orientation of the cut, three placings of each pair of boxes can be handled as the packing problem with two boxes. Finally, non-guillotine packings are possible when the four boxes lay on the same plane as illustrated, for the $W - H$ plane, in Figure 3: for reasons of symmetry, we may fix one of the boxes in the lower left corner of the bin and consider the 3! placings of the remaining boxes in the three remaining corners of the same plane, checking that no two boxes overlap.

5 Exact and approximation algorithms

The exact algorithm for the three-dimensional bin packing problem is based on a two-level decomposition principle presented by Martello and Vigo [16] for the two-dimensional bin packing problem. A *main branching tree* assigns boxes to bins without specifying the actual position of the boxes, while a specialized version of the branch-and-bound algorithm of Section 4 is used, at certain decision nodes, to test whether a subset of boxes can be placed inside a single bin and to determine the placing when the answer is affirmative. In order to construct a good starting solution, two heuristic algorithms have been defined, one based on a first-fit-decreasing approach and one based on an m-cut version of the single bin filling algorithm. In the following sections we describe the main branching tree, the specialized single bin filling algorithm and the two heuristics.

5.1 Main Branching Tree

The main branching tree assigns boxes to the different bins without specifying their actual position. The boxes are previously sorted according to nonincreasing volumes, and the exploration follows a depth-first strategy. Let Z be the incumbent solution value and $M = \{1, \ldots, m\}$ the current set of bins used to allocate boxes in the ascendant decision nodes. A bin of M is called *open* if at least one box is currently placed into it and there is no evidence that no further box can be placed into it; otherwise, it is called *closed* (i.e., $M = M_o \cup M_c$, where M_o (resp. M_c) denotes the set of currently open (resp. closed) bins).

At each decision node the next free box is assigned, in turn, to all the open bins; in addition, if $|M| < Z - 1$ the box is also assigned to a new bin (hence opening it).

When a box j is assigned to a bin i which already contains, say, a subset $J_i \neq \emptyset$ of boxes, the actual feasibility of the assignment is checked as follows. First, lower bound L_2 is computed for the sub-instance defined by the boxes of $\overline{J} = J_i \cup \{j\}$: if $L_2 \geq 2$ the node is immediately killed. Otherwise, the two heuristics of Section 5.3 are executed in sequence for the sub-instance: if a solution requiring a single bin is obtained, the assignment is accepted. If, instead, no such solution is found, the optimal solution for the sub-instance is determined through the algorithm of Section 5.2, and the node is accepted if a single bin solution is found, or killed otherwise.

When the current node is accepted, an attempt is made to close the current bin. For each free box j' we compute lower bound L_2 for the sub-instance defined by the boxes of $\overline{J} \cup \{j'\}$. If the lower bound is greater than one for each j' then the bin is closed, since we know that no further box could be placed into it. Otherwise, the following dominance rule is tested. Let J' be the subset of those free boxes for which the above lower bound computation has given value one: if a single bin solution can be found by one of the two heuristics for the sub-instance defined by the boxes in $\overline{J} \cup J'$, then we know that no better placing is possible for these boxes, so we assign all of them to bin i and close it.

Whenever a bin is closed, lower bound L_2 is computed for the instance defined by all the boxes not currently assigned to closed bins: if $L_2 + |M_c| \geq Z$ we backtrack. Since closed bins are seldom completely filled, this improved bound generally increases as the branching propagates.

5.2 Single bin filling

As previously mentioned, certain nodes of the main decision tree require implicit enumeration. Indeed, when a decision node is neither killed by the lower bound computation, nor accepted by the heuristics, the feasibility of the current assignment of the boxes in $\overline{J}$ is tested through the single bin filling algorithm of Section 4. Since we are only interested in finding

solutions where all the boxes currently assigned to a bin are placed inside it, in the filling procedure of Section 4.2 we initialize the current best filling to

$$F = \sum_{j' \in \overline{J}} v_i - 1 \tag{23}$$

Hence, if the algorithm returns an unchanged value of F we may conclude that no filling exists with all the boxes inside the same bin.

5.3 Approximation algorithms

In order to obtain a good upper bound at the root node of the branching tree and to limit the number of executions of the single bin filling algorithm, two different heuristics were defined. The two approaches computationally proved to have a "complementary" behaviour: for many instances a poor performance of one of them corresponded to a good performance of the other one.

The first heuristic is based on a layer building principle derived from shelf approaches used by several authors for the two-dimensional bin packing problem (see, e.g., Chung, Garey and Johnson [5] and Berkey and Wang [1]). In such approaches the rectangles are sorted by nonincreasing height and packed, from left to right, in rows forming shelves: the first shelf is the bottom of the two-dimensional bin; when a new shelf is needed, it is created along the horizontal line which coincides with the top of the tallest rectangle packed into the highest shelf.

For the three-dimensional case we first construct *bin slices* having width W, height H and different depths. Each slice is obtained through a two-dimensional shelf algorithm applied to a subset containing the deepest boxes not yet packed. The slices are then combined into three-dimensional bins. Let $\overline{J}$ be the set of boxes to be packed. The algorithm works as follows.

algorithm H1:
begin
  $T := \overline{J}$;
  sort the boxes in T by nonincreasing values of d_j;
  while $T \neq \emptyset$ **do**
    begin
      let $T = \{j_1, \ldots, j_{|T|}\}$;
      $k := \max\{r : \sum_{s=1}^{r} w_{j_s} h_{j_s} < 2WH\}$;
      construct a single slice of depth d_{j_1} by applying the following
      two-dimensional shelf algorithm to the boxes of $T' = \{j_1, \ldots, j_k\}$:

```
        sort the boxes of T' by nondecreasing height;
        for each box j ∈ T' do
          if j fits into an existing shelf then pack j into the lowest shelf where it fits
          else if there is enough vertical space for j then
                create a new shelf and pack j into it;
        remove the packed boxes from T
      end;
    let d'_1, ..., d'_ℓ be the depths of the resulting slices;
    use a one-dimensional bin packing algorithm to combine the slices
    into the minimum number of three-dimensional bins of depth D
end.
```

For the final step we used the FORTRAN code MTP, provided in the book by Martello and Toth [14], with a limit of 10^6 backtrackings. Algorithm H1 could also construct bin slices with width W, depth D and different heights, or bin slices slices with heigth H, depth D and different widths. In practice, H1 is run three times by interchanging the dimension of the boxes and of the bins, and the best solution is taken.

The second heuristic repeatedly fills a single bin through the exact algorithm in Section 4. Let $\overline{J}$ be the set of boxes to be packed.

```
algorithm H2:
begin
  T := J̄;
  sort the boxes of T according to nondecreasing volume;
  while T ≠ ∅ do
    begin
      apply the single bin filling algorithm to T;
      remove the packed boxes from T
    end
end.
```

Since the solution times of the single bin filling algorithm may be unacceptable when $|T|$ is large, the branching scheme of Section 4.2 has been changed to an m-cut enumeration as described in Ibaraki [13], where at each decision node only the first m branches are considered. Good values for m were experimentally found to be $m = 4$ when $|T| \leq 10$, $m = 3$ when $|T| \leq 15$ and $m = 2$ for larger problems. Moreover, a limit of 5000 decision nodes was imposed on each filling of a bin, and the best solution found within this limit was returned. Because of the limit on the number of branches, also algorithm H2 produces

different solutions if the dimensions of the boxes and of the bins are interchange, hence this algorithm too is run three times.

6 Computational Experiments

To our knowledge no test instances have been published for the three-dimensional bin packing problem. Thus in our computational experiments we have chosen to generate three-dimensional instances by generalizing some classes of randomly generated two-dimensional instances. Also a new class of "all-fill" instances was introduced. For short in the following "u.r." means "uniformly random" (or "uniformly randomly").

The first classes of instances are generalizations of the instances considered by Martello and Vigo in [16]. The bin size is $W = H = D = 100$ and five types of boxes are considered:

- *Type 1*: w_j u.r. in $[1, \frac{1}{2}W]$, h_j u.r. in $[\frac{2}{3}H, H]$, d_j u.r. in $[\frac{2}{3}D, D]$.
- *Type 2*: w_j u.r. in $[\frac{2}{3}W, W]$, h_j u.r. in $[1, \frac{1}{2}H]$, d_j u.r. in $[\frac{2}{3}D, D]$.
- *Type 3*: w_j u.r. in $[\frac{2}{3}W, W]$, h_j u.r. in $[\frac{2}{3}H, H]$, d_j u.r. in $[1, \frac{1}{2}D]$.
- *Type 4*: w_j u.r. in $[\frac{1}{2}W, W]$, h_j u.r. in $[\frac{1}{2}H, H]$, d_j u.r. in $[\frac{1}{2}D, D]$.
- *Type 5*: w_j u.r. in $[1, \frac{1}{2}W]$, h_j u.r. in $[1, \frac{1}{2}H]$, d_j u.r. in $[1, \frac{1}{2}D]$.

We have obtained five classes of instances as follows. For *Class k* ($k = 1, \ldots, 5$), each box is of type k with probability 60%, while it is of the other four types with probability 10% each.

The second group of classes is a generalization of the instances presented by Berkey and Wang [1]. The three classes may be described as:

Class 6: bin size $W = H = D = 10$; boxes with w_j, h_j, d_j u.r. in $[1, 10]$.

Class 7: bin size $W = H = D = 40$; boxes with w_j, h_j, d_j u.r. in $[1, 35]$.

Class 8: bin size $W = H = D = 100$; boxes with w_j, h_j, d_j u.r. in $[1, 100]$.

Finally, a new difficult class of *all-fill* problems has been introduced (*Class 9*). These instances have a known solution with three bins, since the boxes are generated by cutting the bins into smaller parts. For a problem with n boxes, bin 1 and 2 are cut into $m_1 = m_2 = \lfloor n/3 \rfloor$ boxes, while bin 3 is cut into $n - m_1 - m_2$ boxes. The cutting is made using the following recursion, where J is the set of boxes which should be generated, and the initial values of w, h and d are W, H and D, respectively.

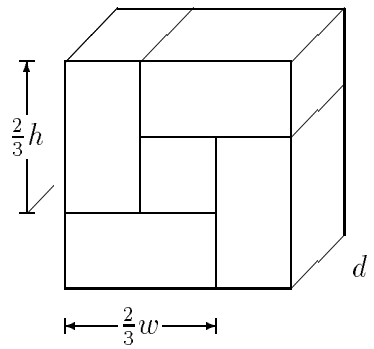

Figure 4: A non-guillottine cuttable pattern with five boxes

procedure CUT(J, w, h, d)
begin
  if $|J| = 1$ **then** generate a box with dimension (w, h, d);
  else
    if $|J| = 5$ **then** generate a non-guillotine cutting pattern as shown in Figure 4;
    else
      begin
        u.r. partition J into J_1 and J_2;
        u.r. execute one of the following three steps:
        1. generate w' u.r. in $[1, w-1]$; **call** CUT(J_1, w', h, d); **call** CUT(J_2, $w-w'$, h, d);
        2. generate h' u.r. in $[1, h-1]$; **call** CUT(J_1, w, h', d); **call** CUT(J_2, w, $h-h'$, d);
        3. generate d' u.r. in $[1, d-1]$; **call** CUT(J_1, w, h, d'); **call** CUT(J_2, w, h, $d-d'$)
      end
end.

Pathological situations may occur, in which it is not possible to arrange the boxes of J within (w, h, d), but in this case a new problem is simply generated from scratch.

The exact code was implemented in C, and the experiments were run on a SUN UltraSparc 175 Mhz. A time limit of 1000 seconds was given to each instance, and 10 instances were run for each class and size of a problem. Fine-tuning of the algorithm showed that the best performance was obtained if the principle of closing a bin described in Section 5.1 was only tested at branching nodes not farther than 15 nodes from the root. The outcome of our experiments is given in Tables I to VI.

Table I shows that we solved a large majority of the problems within the given time limit. Nearly all the instances with up to $n = 25$ were solved to optimality, and even for $n = 50$ we were able to solve one half of the instances to optimality (but the very difficult all-fill instances). Notice that the almost all the instances of classes 4 and 6 were solved to

optimality also for large values of n. The average solution times are given in Table II.

Table I: Number of instances solved to optimality

	Class								
n	1	2	3	4	5	6	7	8	9
10	10	10	10	10	10	10	10	10	10
15	10	10	10	10	10	10	10	10	10
20	10	10	10	10	10	10	10	10	10
25	10	10	10	10	10	10	8	10	10
30	10	10	10	10	9	10	5	10	7
35	9	8	10	10	6	10	5	7	9
40	9	8	7	10	4	10	3	7	3
45	5	6	4	10	8	9	1	7	–
50	4	2	4	10	4	9	1	5	–
55	2	–	–	7	4	9	–	3	–
60	1	1	–	9	2	6	–	3	–

Table II: Average solution times

	Class								
n	1	2	3	4	5	6	7	8	9
10	0.12	0.07	0.10	0.02	0.54	0.23	0.45	0.35	0.00
15	0.16	0.12	0.15	0.02	1.01	0.43	14.69	0.78	0.02
20	0.28	0.21	0.26	0.06	1.58	1.00	1.17	1.06	0.21
25	0.76	0.33	0.86	0.09	48.86	1.57	226.64	1.65	1.86
30	47.72	2.63	5.45	0.40	106.35	4.84	533.05	5.65	344.97
35	101.78	221.12	51.03	0.47	426.74	12.79	580.78	319.66	116.42
40	171.38	212.45	472.85	0.50	616.97	100.24	720.29	316.54	794.77
45	546.90	538.35	615.99	1.10	215.52	118.37	922.12	344.35	1040.43
50	645.48	831.64	617.00	36.61	619.16	144.33	924.19	582.88	1026.92
55	837.61	1000.43	1000.30	320.79	643.25	256.58	1025.96	765.32	1028.15
60	917.52	938.17	1000.19	129.63	823.29	606.77	1024.93	733.68	1019.31

Table III shows the average value of the found solutions. The entries should be compared to the three lower bound values given in Tables IV to VI. In Theorem 4 we proved that L_2 dominates L_0 and L_1 but it is interesting to see, that L_2 also in practice is considerably better than both of the other. Comparing L_2 to the found solution values, it is seen that L_2 is extremely tight for the considered instances, generally differing by about one bin from the optimal solution value.

7 Conclusions

Our paper is the first work on exact algorithms for the three-dimensional bin-packing problem; thus several important aspects have been addressed. We have presented a number of lower bounds, and compared the theoretical as well as practical performance of them. The

Table III: Average solution found

	Class								
n	1	2	3	4	5	6	7	8	9
10	3.3	3.5	3.6	6.6	2.5	2.9	2.5	2.8	3.0
15	4.7	4.5	4.8	8.7	2.9	4.4	3.0	3.7	3.0
20	6.0	6.6	6.0	12.3	3.9	5.4	3.8	4.9	3.0
25	7.4	7.0	7.1	15.4	4.6	5.9	4.5	5.5	3.0
30	8.6	8.0	8.6	17.2	5.5	6.8	4.5	6.0	3.3
35	9.4	9.5	10.3	21.1	6.7	7.7	5.5	7.3	3.2
40	11.0	10.8	11.6	24.3	7.2	8.8	6.5	8.1	4.0
45	12.4	12.6	12.2	27.6	7.6	9.8	7.1	8.6	4.8
50	13.6	13.9	13.6	29.4	9.0	9.9	8.4	9.9	5.0
55	14.8	14.7	14.7	32.6	8.8	11.8	9.4	11.6	5.0
60	15.5	15.4	15.2	36.4	9.3	12.8	9.6	11.9	5.0

Table IV: Average value of L_0

	Class								
n	1	2	3	4	5	6	7	8	9
10	2.5	2.2	2.2	3.6	1.5	2.0	1.5	2.0	3.0
15	3.2	3.1	3.2	5.0	1.8	3.0	2.1	2.7	3.0
20	4.2	4.3	4.2	6.6	2.7	3.9	2.5	3.3	3.0
25	5.4	4.9	4.7	8.3	3.2	4.5	3.1	3.8	3.0
30	6.0	5.6	6.1	9.5	3.8	5.4	2.9	4.3	3.0
35	6.9	6.9	7.5	11.3	4.4	6.3	3.7	5.0	3.0
40	8.3	7.7	8.1	12.8	4.7	7.1	4.2	5.7	3.0
45	8.8	8.9	8.7	14.4	5.3	7.9	4.4	6.3	3.0
50	9.6	9.9	9.6	15.8	6.1	8.4	5.5	7.0	3.0
55	10.7	10.5	10.5	17.4	6.1	9.9	5.9	7.9	3.0
60	11.3	11.0	11.0	18.8	6.2	10.7	6.3	8.5	3.0

Table V: Average value of L_1

	Class								
n	1	2	3	4	5	6	7	8	9
10	3.1	3.1	2.8	6.5	2.1	2.1	1.7	2.3	3.0
15	3.9	3.7	3.7	8.6	2.2	3.1	2.4	3.2	2.7
20	5.2	5.7	4.9	12.2	3.3	4.1	3.2	4.2	2.8
25	6.5	5.9	6.2	15.1	3.7	4.6	3.4	4.6	3.0
30	7.4	6.7	7.2	16.6	4.5	4.8	3.1	4.9	2.6
35	8.4	8.4	8.7	20.6	5.4	5.7	4.0	6.0	2.7
40	9.6	9.4	9.7	23.8	5.2	7.2	4.5	6.5	2.7
45	10.6	10.9	10.6	27.0	5.7	7.8	4.8	7.1	2.6
50	11.7	12.0	11.4	28.7	6.7	7.6	6.0	7.5	2.8
55	12.4	12.5	12.5	31.9	6.3	9.6	6.5	9.4	2.1
60	13.1	13.7	13.5	35.7	6.5	10.0	6.4	10.0	2.0

Table VI: Average value of L_2

	Class								
n	1	2	3	4	5	6	7	8	9
10	3.1	3.1	2.9	6.5	2.1	2.4	1.8	2.4	3.0
15	4.1	3.9	4.0	8.6	2.2	3.5	2.7	3.3	3.0
20	5.2	5.8	5.0	12.2	3.3	4.6	3.3	4.3	3.0
25	6.7	6.4	6.3	15.1	4.0	5.1	3.5	4.9	3.0
30	7.6	7.2	7.5	16.6	4.7	6.0	3.3	5.2	3.0
35	8.8	8.8	9.2	20.6	5.7	6.8	4.4	6.2	3.0
40	10.3	9.8	10.4	23.8	5.6	8.2	4.8	7.0	3.0
45	11.2	11.5	11.3	27.0	6.3	8.7	5.1	7.4	3.0
50	12.5	12.7	12.3	28.7	7.3	8.7	6.3	8.0	3.0
55	13.2	13.1	13.1	31.9	6.7	10.7	6.8	9.7	3.0
60	14.1	14.1	13.9	35.7	7.0	11.3	6.9	10.2	3.0

branch-and-bound algorithm for the filling of a single bin plays a central role in our overall algorithm for 3D-BPP, and it may be useful for other research projects in the field of cutting and packing. Finally we have demonstrated that the framework proposed by Martello and Vigo [16] for the exact solution of 2D-BPP may be adapted to the three-dimensional case with small modifications. The computational results illustrate the applicability of our results.

Acknowledgments

The first and third authors acknowledge Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST) and Consiglio Nazionale delle Ricerche (CNR) for the support of this project. The second author would like to thank the EC Network DIMANET for supporting the research by European Research Fellowship No. ERBCHRXCT-94 0429.

References

[1] J.O. Berkey and P.Y. Wang. Two dimensional finite bin packing algorithms. *Journal of the Operational Research Society*, 38:423–429, 1987.

[2] E.E. Bischoff and M.D. Marriott. A comparative evaluation of heuristics for container loading. *European Journal of Operational Research*, 44:267–276, 1990.

[3] C.S. Chen, S.M. Lee, and Q.S. Shen. An analytical model for the container loading problem. *European Journal of Operational Research*, 80:68–76, 1995.

[4] N. Christofides and C. Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, 25:30–44, 1977.

[5] F.K.R. Chung, M.R. Garey, and D.S. Johnson. On packing two-dimensional bins. *SIAM Journal of Algebraic and Discrete Methods*, 3(1):66–76, 1982.

[6] M. Dell'Amico and S.Martello. Optimal scheduling of tasks on identical parallel processors. *ORSA Journal on Computing*, 7(2):191–200, 1995.

[7] H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159, 1990.

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

[9] H. Dyckhoff, G. Scheithauer, and J. Terno. Cutting and Packing (C&P). In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*. John Wiley & Sons, Chichester, 1997.

[10] M. Gehring, K. Menscher, and M. Meyer. A computer-based heuristic for packing pooled shipment containers. *European Journal of Operational Research*, 44:277–288, 1990.

[11] J.A. George and D.F. Robinson. A heuristic for packing boxes into a container. *Computers and Operations Research*, 7:147–156, 1980.

[12] H. Hadjiconstantinou and N. Christofides. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *European Journal of Operational Research*, 83:39–56, 1995.

[13] T. Ibaraki. *Enumerative Approaches to Combinatorial Optimization – Part 2*, volume 11 of *Annals of Operations Research*. Baltzer, Basel, 1987.

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

[15] S. Martello and P. Toth. Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics*, 28:59–70, 1990.

[16] S. Martello and D. Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management Science*, 1997. (to appear).

[17] D. Pisinger. The container loading problem. In *Proceedings NOAS'97*, 1997.

[18] G. Scheithauer. A three-dimensional bin packing algorithm. *J. Inform. Process. Cybernet.*, 27:263–271, 1991.