A tree-search heuristic for the container loading problem

David Pisinger
Dept. of Computer Science, University of Copenhagen,
Universitetsparken 1, DK-2100 Copenhagen, Denmark

Abstract

We consider the container loading problem where a number of rectangular boxes should be loaded into a container of fixed dimensions. In order to develop an effective heuristic, we restrict our problem to consider loadings which are guillotine cuttable, i.e. where the container may be filled by dividing it into layers and subdividing each layer into strips, a strip usually being a sequence of boxes. This restriction implies that every strip may be optimally filled by solving a knapsack problem with capacity equal to the length of the strip. The depth of a layer as well as the thickness of each strip is decided through a branch-and-bound approach where at each node only a subset of branches is explored.

Computational experiments involving up to 120 boxes are reported, showing that several loading problems from the literature are solved to optimality in reasonable time, achieving an average filling of 91% of the container volume.

1 Introduction

The problem addressed in this paper is that of orthogonally packing a subset of some given rectangular-shaped boxes into a rectangular container of fixed dimensions. The problem has numerous applications, since it directly models several problems in production and transportation planning. An optimal filling of a container reduces the shipping costs, and since the packing will be dense, it also decreases the chance of breaking articles. The problem has been studied since the seminal work of Gilmore and Gomory [7] in the early sixties, and numerous papers and algorithms have been presented for its solution. Although authors talk about the "container loading problem" or "loading problem" there are however several versions of the problem with respect to objective function and constraints.

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 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 [5] and Scheithauer [14].

Minimizing Depth. All boxes have to be packed into one container which has fixed width and height but infinite depth. The objective is to find a feasible solution which minimizes the depth. Several algorithms for this problem are compared in Bischoff and Marriott [1].

Bin-Packing. All containers have fixed dimensions, and all the boxes are to be orthogonally packed into a minimum number of containers. This problem has recently been considered in Scheithauer [13], Chen, Lee and Shen [2], and Martello, Pisinger, and Vigo [11]. The last two consider exact methods for the solution.

For a general classification of packing and loading problems we refer to Dyckhoff [3] and Dyckhoff, Scheithauer and Terno [4]. Several other constraints may be imposed to the above problems: Feasible solutions might be restricted to those which are guillotine cuttable, only specific rotations (if any) may be allowed, it may be demanded that the weight of the container is balanced, there may be restrictions on which or how many boxes may be put on top of each other, or it may be necessary to place boxes from one lot next to each other.

The problem considered in this paper deals with the knapsack loading version, where we assume that the profit of a box equals its volume (although the general version with distinct profits may be handled in a similar way). The boxes may be rotated in any orthogonal directions, but no other restrictions are put on the solution. We will denote this variant of the problem by *KCLP* (*Knapsack Container Loading Problem*).

In the following we will assume that the container has width W, height H and depth D. A set $N = \{1, \ldots, n\}$ of boxes is given, each box j having width w_j, height h_j and depth d_j. The volume of each box is $v_j = w_j h_j d_j$.

The organization of this paper is the following: The next section presents several heuristics for KCLP based on the George-Robinson framework [6]. These heuristics fill the container layer by layer in a single-pass way (see Figure 1 and 2), where the loading order of the boxes is ruled by some ranking functions. In Section 3 we present an improved algorithm based on the layer building approach which however incorporates a backtracking step to improve the solution quality. Several different layer depths are thus investigated in an enumerative way, and the layer itself is packed by considering several strip widths. Computational experiments with loading problems of about 100 boxes of 20 types are brought in Section 4 where the presented strip packing approach is compared to an unrestricted algorithm for filling a layer. The paper is concluded by some final remarks in Section 5.

2 Heuristic Algorithms

The KCLP is $\mathcal{NP}$-hard in the strong sense, since solving a special case where all 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 (BPP) admits a solution requiring

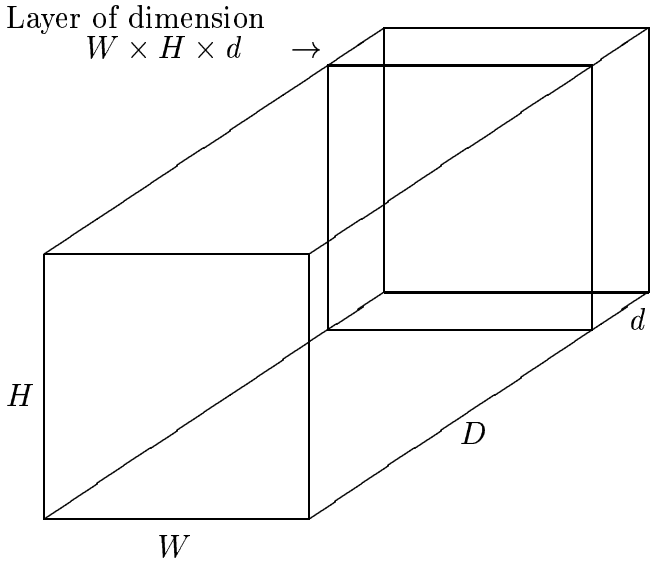

Figure 1: The George-Robinson algorithm fills the container layer by layer

no more than H bins. In practice it is however considerably more difficult to solve KCLP than BPP, thus no exact algorithm, being capable of solving real-life instances, has been presented for the first problem. This paper thus only deals with heuristics for KCLP.

George and Robinson [6] presented the first heuristic algorithm for KCLP. The algorithm is based on the concept of filling the container in a number of layers across the depth of the container as illustrated in Figure 1. Obviously only normalized layer depths need to be considered (i.e. layer depths d which equal some box dimensions), and the depth must be carefully selected to obtain a good performance. Hence when opening a new layer, George and Robinson apply a ranking rule for selecting d: Among the remaining boxes they choose the box ℓ which has largest size of the smallest dimension, the rationale being that such a box may be difficult to accommodate later in the packing procedure. The box is rotated such that its depth is maximized, with the restriction that it may not exceed a given parameter k. The layer depth d is then set equal to the depth d_ℓ of the chosen box.

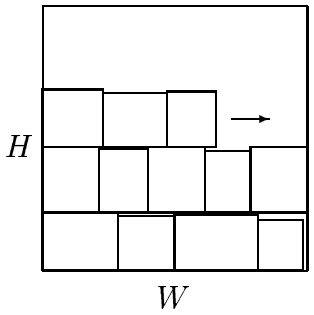

Figure 2: Each layer is filled by a number of horizontal strips

Having determined the depth of a layer, the face is then packed as a number of horizontal strips (see Figure 2). The boxes are considered in an order of priority corresponding to their ranking. The first ranking criterion is the smallest dimension of a box. The larger it is, the larger ranking. In case of ties, ranking is chosen according to the number of remaining boxes of a given type and finally according to the length of the largest dimension. Every strip is packed by consecutively inserting boxes with the largest ranking.

Bischoff and Marriott [1] compare 14 heuristics based on the George-Robinson approach, where different ranking functions are applied. Their results indicate that there are no clear winning strategy thus they propose a "hybrid" version where all 14 heuristics are run in parallel.

A major disadvantage about the George-Robinson heuristics is that they are of the single-pass type, and thus easily may be trapped with a bad combination of boxes for the last layers. To obtain a better performance it is necessary to have a kind of backtracking possibility. Hemminki [9] presented a genetic algorithm for choosing the depths of the layers. Each chromosome consists of a string of layer depths, and it is the hope that "good" combinations of layers will evolve during the generations. Hemminki also used a specialized algorithm for the filling of a single layer [8] obtaining an algorithm which on average fills 87.0% of the volume of the container.

3 A tree-search heuristic

Our algorithm is an extension of the George and Robinson approach where a tree-search algorithm is used to find the set of layer depths and strip widths which results in the best overall filling. Ideally we would like to consider all normalized layer depths and strip widths, but this would be computationally too expensive. To decrease the complexity an m-cut approach [10] is used for the enumeration, where only a fixed number of sub-nodes are considered for every branching node.

Hence, at each branching node we consider M_1 different layer depths selected according to a specific ranking rule. Each layer is then filled as a number of strips, where the strips may be oriented horizontally or vertically. Again, only a limited number M_2 of different strip widths are considered. Once the layer depth and strip width has been settled, a single strip may filled optimally by solving a Knapsack Problem (KP). For every layer depth $d'_1, \ldots, d'_{M_1}$ we choose the layer filling which has the overall best use of the volume before proceeding to the next layer. This leads to the following algorithm:

For a given subset of boxes N', residual container depth d and previous packing of volume V, the recursive procedure `choose_depth` selects M_1 different layer depths according to the ranking rule to be described in Section 3.1. For each depth d', it is attempted to pair boxes two by two in order to obtain uniform dimensions. This approach will be described in Section 3.3. The layer of dimensions $W \times H \times d'$ is then filled by calling a procedure `fill_layer`. The chosen boxes L for the current layer are removed from the problem, and `choose_depth` is called recursively. The algorithm backtracks when there is no more space for a layer. The procedure makes use of two global variables:

X^* which is the currently best solution (i.e. a set of chosen boxes and their positions), and V^* which is the corresponding volume. Initially, one should set $V^* = 0$ before calling `choose_depth`$(D, 0, N)$. Upon completion, X^* is the heuristic solution and V^* its volume.

algorithm choose_depth(d, V, N')
if $V > V^*$ **then** save solution in X^*, set $V^* = V$.
if $d < \min_{j \in N'} \min\{w_j, h_j, d_j\}$ **then return**;
select M_1 best ranked layer depths $\{d'_1, d'_2, \ldots, d'_{M_1}\}$.
for each depth $d' \in \{d'_1, d'_2, \ldots, d'_{M_1}\}$ **begin**
  Pair boxes to obtain a depth close to d', and set $U^* \leftarrow 0$;
  Call fill_layer$(W, H, d', 0, N')$; Let L be the chosen boxes, and U^* their volume.
  Call choose_depth$(d - d', V + U^*, N' \backslash L)$;
end;

The recursive procedure `fill_layer` packs a layer of dimensions $\underline{w} \times \underline{h} \times d'$ with a subset of the items N''. The volume of already placed boxes in this layer is given by the parameter U. The layer is filled by a number of strips which may be either vertical or horizontal. Before filling a strip, some preprocessing of the items in N'' is made, such that they are rotated to occupy least possible height (or width). This is further described in Section 3.2. The procedure makes use of the global variables L and U^*, holding the current best filling of the layer and the corresponding objective value.

algorithm fill_layer$(\underline{w}, \underline{h}, d', U, N'')$
if $U > U^*$ **then** save solution in L, set $U^* \leftarrow U$;
$\ell \leftarrow \min_{j \in N''} \min\{w_j, h_j, d_j\}$;
if $(\underline{w} < \ell)$ **or** $(\underline{h} < \ell)$ **then return**;
comment : pack vertical strip:
select M_2 best ranked widths $\{w'_1, w'_2, \ldots, w'_{M_2}\}$.
for each width $w' \in \{w'_1, w'_2, \ldots, w'_{M_2}\}$ **begin**
  Rotate the boxes and solve a KP with capacity $\underline{h}$. Let K be the set of chosen boxes.
  Call fill_layer$(\underline{w} - w', \underline{h}, U + \sum_{j \in K} v_j, N'' \backslash K)$;
end;
comment : pack horizontal strip:
select M_2 best ranked heights $\{h'_1, h'_2, \ldots, h'_{M_2}\}$.
for each height $h' \in \{h'_1, h'_2, \ldots, h'_{M_2}\}$ **begin**
  Rotate the boxes and solve a KP with capacity $\underline{w}$. Let K be the set of chosen boxes.
  Call fill_layer$(\underline{w}, \underline{h} - h', U + \sum_{j \in K} v_j, N'' \backslash K)$;
end;

3.1 Ranking of depths and widths

At each branching node we consider the M_1 best ranked depths of a layer, or the M_2 best ranked widths/heights of a strip. Computational experiments showed that the performance of the algorithm strictly depends on the ranking of these dimensions, thus the following ranking functions were investigated:

(i) The largest value of the smallest dimension of a box. This corresponds to the ranking function of George and Robinson.

(ii) Most frequent dimension. By considering the most frequent dimension, there is a large chance of obtaining a regular layer or strip.

(iii) Largest dimension. If we get rid of boxes with large dimensions early in the algorithm, small boxes are easier to handle for the residual problem.

(iv) Largest dimension with increasing frequency. This is somehow a trade-off between the previous two ranking function: The largest dimensions are selected, such that every chosen dimension has a higher frequency than any larger dimension.

Computational experiments showed that best results were obtained if ranking function (iv) was used for the layer depths and ranking function (ii) for the widths/heights of the strips.

Thus let f_k be the frequency of dimension k, i.e. the number of boxes $j \in N'$ with w_j, h_j or d_j equal to k. The M_1 depths are chosen as follows: Initially set $d'_1 = \max\{k : f_k > 0\}$. Subsequent values are found as $d'_i = \max\{k : f_k > f_{d'_{i-1}}\}$ for $i = 2, \ldots, M_1$. The M_2 widths/heights are chosen by setting $w'_i = \arg\max\{f_k : k \neq w'_1, \ldots, w'_{i-1}\}$ for $i = 1, \ldots, M_2$.

In certain situations, it may not be possible to generate all the M_1 different depths or M_2 different widths/heights, but in such cases a smaller set is simply returned. Only dimensions which results in a layer or strip that fits within the container are considered.

3.2 Filling a single strip

Strips may be filled horizontally or vertically. Since the cases are symmetrical, we will consider a vertical filling in the following. The filling of a single strip of width w' and depth d' may be formulated as a Knapsack Problem. First, each box j is rotated in one of six directions such that $w_j \leq w'$, $d_j \leq d'$ and h_j is minimized. If it is not possible to fit the box j within $w' \times d'$ then add it to the set of discarded boxes D. Otherwise set

$$a_j = h_j \qquad\qquad c_j = v_j = w_j h_j d_j \tag{1}$$

for each feasible box $j \in N \backslash D$. Then the strip packing problem becomes

$$\begin{aligned} \text{maximize} \quad & \sum_{j \in N \backslash D} c_j x_j \\ \text{subject to} \quad & \sum_{j \in N \backslash D} a_j x_j \leq b \\ & x_j \in \{0, 1\}, \quad j \in N \backslash D \end{aligned} \tag{2}$$

where $N \backslash D$ is the set of feasible boxes, and b is the strip height $\underline{h}$ from algorithm `fill_layer`. The resulting problem is a Knapsack Problem (KP) which may be solved in $O(nb)$ time through dynamic programming, where $n = |N \backslash D|$.

3.3 Pairing of boxes

The solution found by the KP may be improved by pairing boxes two by two whenever possible. Since the complexity of this algorithm is $O(n^2)$ it is only executed once for each layer having depth d'. Thus assume that a box j has been rotated such that its depth d_j is largest possible satisfying $d_j \leq d'$. A measure of how well it fills the depth is $\alpha(j) = v_j/(w_j h_j d') = d_j/d'$. We aim at improving this filling by pairing box j with another box i. Thus for each box $i \neq j$ all different rotations of i and j are considered where $d_i + d_j \leq d'$, each time deriving the filling ratio $\beta(i,j) = (v_i + v_j)/(d' \max\{w_i, w_j\} \max\{h_i, h_j\})$. If $\beta(i,j) \leq \alpha(j)$ for all boxes i and all rotations, then the box j remains alone. Otherwise, the box i (and corresponding rotation) leading to the largest value of $\beta(i,j)$ is chosen and a new box k is constructed with dimensions

$$w_k = \max\{w_i, w_j\}; \qquad h_k = \max\{h_i, h_j\}; \qquad d_k = d' \tag{3}$$

The original boxes i and j are temporarily removed from the problem, to avoid that they are selected twice in the layer filling.

4 Computational Experiments

The above algorithm was implemented in C and tests were run on a HP Visualize C200. The Knapsack Algorithm for solving the strip problem (2) was the `minknap` algorithm from Pisinger [12]. Appropriate values of M_1 and M_2 were experimentally found to 4 and 8 respectively, since this gave a good trade-off between solution quality and computational time. The code is available from `http://www.diku.dk/~pisinger`.

Data instances were generated as described in Hemminki [9]. Although the instances are randomly generated, they should reflect typical properties of loading problems. The container has dimensions $W = H = 230$ and $D = 590$ which reflects the size of a 20 feet container measured in centimetres. 20 different box types are generated with w_j, h_j and d_j randomly distributed in $[25, 115]$. Then the cargo is generated by randomly setting box i to one of the 20 box types. New boxes are generated until the overall volume of the boxes exceeds 90% of the containers volume WHD. In this way each instance consists of about 60–130 boxes. Table 1 shows a typical instance, where the box types are given by their width w_j, height h_j and depth d_j and the last row shows how many boxes were generated of the given type.

The algorithm was run on 20 randomly generated instances, and the obtained filling as well as solution time is given in Table 2. All the 20 instances were solved to optimality,

Table 1: Dimensions of boxes for instance number 13

box type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
width w_j	100	78	57	54	42	48	82	55	67	87	45	66	71	93	47	56	82	28	96	70
height h_j	42	36	38	42	45	71	77	91	76	31	26	39	27	30	69	30	100	51	49	67
depth d_j	75	109	41	27	66	38	68	71	79	107	27	86	95	31	46	67	74	39	25	67
number of boxes	5	6	3	7	8	10	11	10	6	3	11	7	7	7	5	3	4	2	5	7

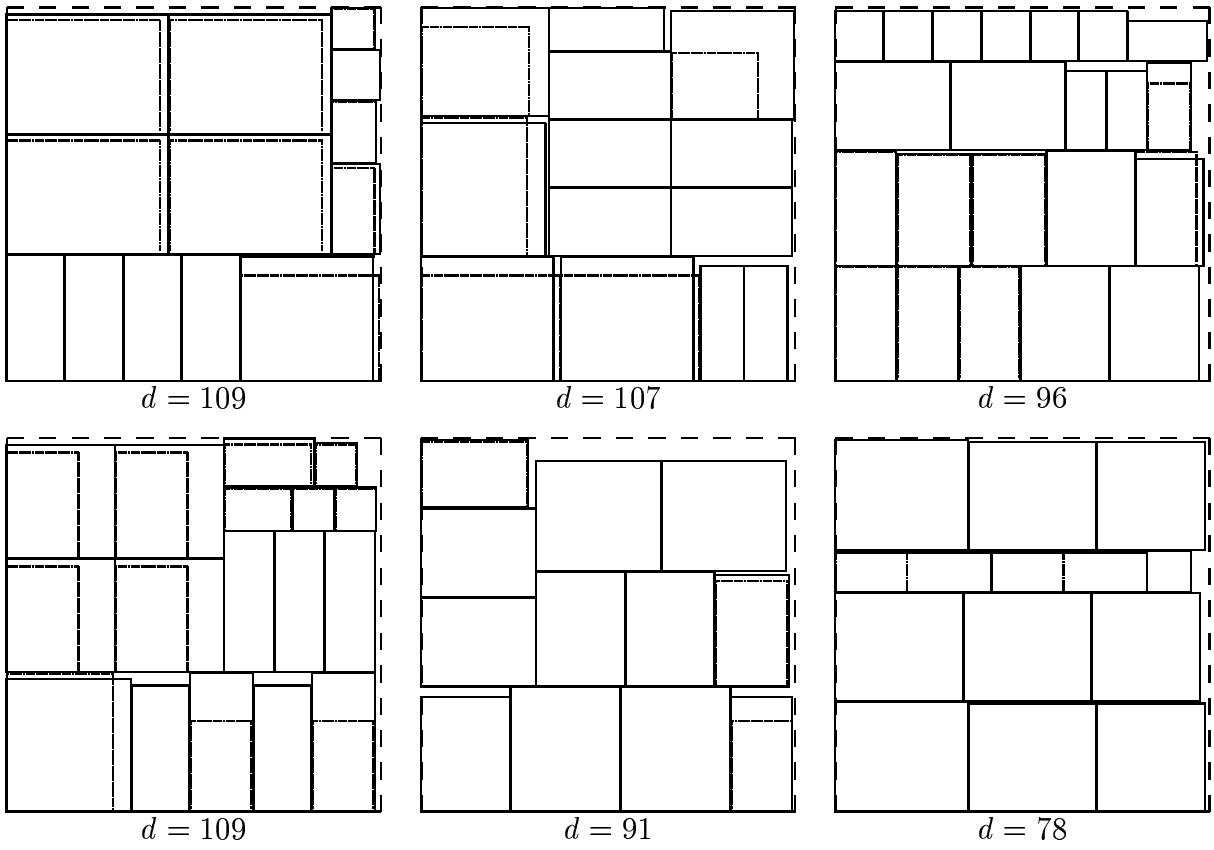

Figure 3: Optimal solution of instance 13. All boxes are packed into the container. The values d state the depth of each layer. Dotted boxes are placed behind the full-drawn boxes forming a pair.

meaning that a solution was found where all the boxes could be placed in the container. On average 90.8% of each container was filled using about half a minute of CPU-time. This involved solving on average 466388 Knapsack Problems, and packing 221 layers for each instance. Figure 3 shows a packing chart of the instance given in Table 1.

For the same data sets, Hemminki [9] considers 12 possible ranking strategies used in the George-Robinson algorithm. If all variants of the algorithm are run simultaneously and the best solution is chosen for each instance, the average filling ratio becomes about 84.5%. A second strategy is to choose each layer depth as the one which yields the best overall filling of the current layer, before proceeding to the next layer. This greedy method yields filling ratios of about 85.4%. If genetic algorithms are used to choose the depths of the layers, Hemminki obtains an average filling ratio of 87.0%.

Restricting the solution space to strip packings is a serious limitation, thus finally a test was run, where the same scheme was used for choosing the layer depths d but where an exact knapsack filling algorithm was used to fill each layer. For this purpose, the knapsack filling algorithm from Martello, Pisinger and Vigo [11] was adapted to the two-dimensional case. Thus, having chosen a layer depth d, the objective is to arrange

Table 2: Solution times and solution quality for 20 randomly generated instances using the strip packing algorithm

instance number	n (boxes)	not packed (boxes)	filling (% volume)	layers	CPU time (sec)
1	76	0	90.5	6	21.19
2	92	0	90.1	6	30.11
3	66	0	90.4	6	11.78
4	68	0	91.4	6	31.58
5	65	0	90.3	6	7.04
6	74	0	91.0	6	1.58
7	105	0	90.3	6	6.30
8	73	0	90.6	6	4.27
9	88	0	90.7	6	2.12
10	78	0	90.5	6	2.26
11	73	0	91.6	6	35.31
12	102	0	92.4	6	43.82
13	127	0	90.2	6	115.33
14	86	0	91.6	6	12.71
15	86	0	90.6	6	9.82
16	114	0	90.2	6	15.38
17	98	0	90.7	6	16.93
18	83	0	92.1	6	63.44
19	72	0	90.6	6	7.81
20	92	0	90.0	6	2.57
avrg.	86	0.0	90.8	6.0	22.07

Table 3: Solution times and solution quality for 20 randomly generated instances using unrestricted two-dimensional knapsack packing of each layer

instance number	n (boxes)	not packed (boxes)	filling (% volume)	layers	CPU time (sec)
1	76	0	90.5	6	3476
2	92	0	90.1	6	7156
3	66	0	90.4	6	8983
4	68	0	91.4	6	5700
5	65	0	90.3	6	18124
6	74	0	91.0	6	1167
7	105	0	90.3	6	2659
8	73	1	90.0	6	22392
9	88	3	89.1	6	39020
10	78	0	90.5	6	23286
11	73	3	90.6	6	14936
12	102	4	89.7	6	10814
13	127	2	88.8	7	40193
14	86	2	91.1	6	12615
15	86	0	90.6	6	1449
16	114	1	89.9	6	11715
17	98	1	89.8	6	18749
18	83	0	92.1	6	9114
19	72	0	90.6	6	5806
20	92	0	90.0	6	167
avrg.	86	0.8	90.3	6.1	12876

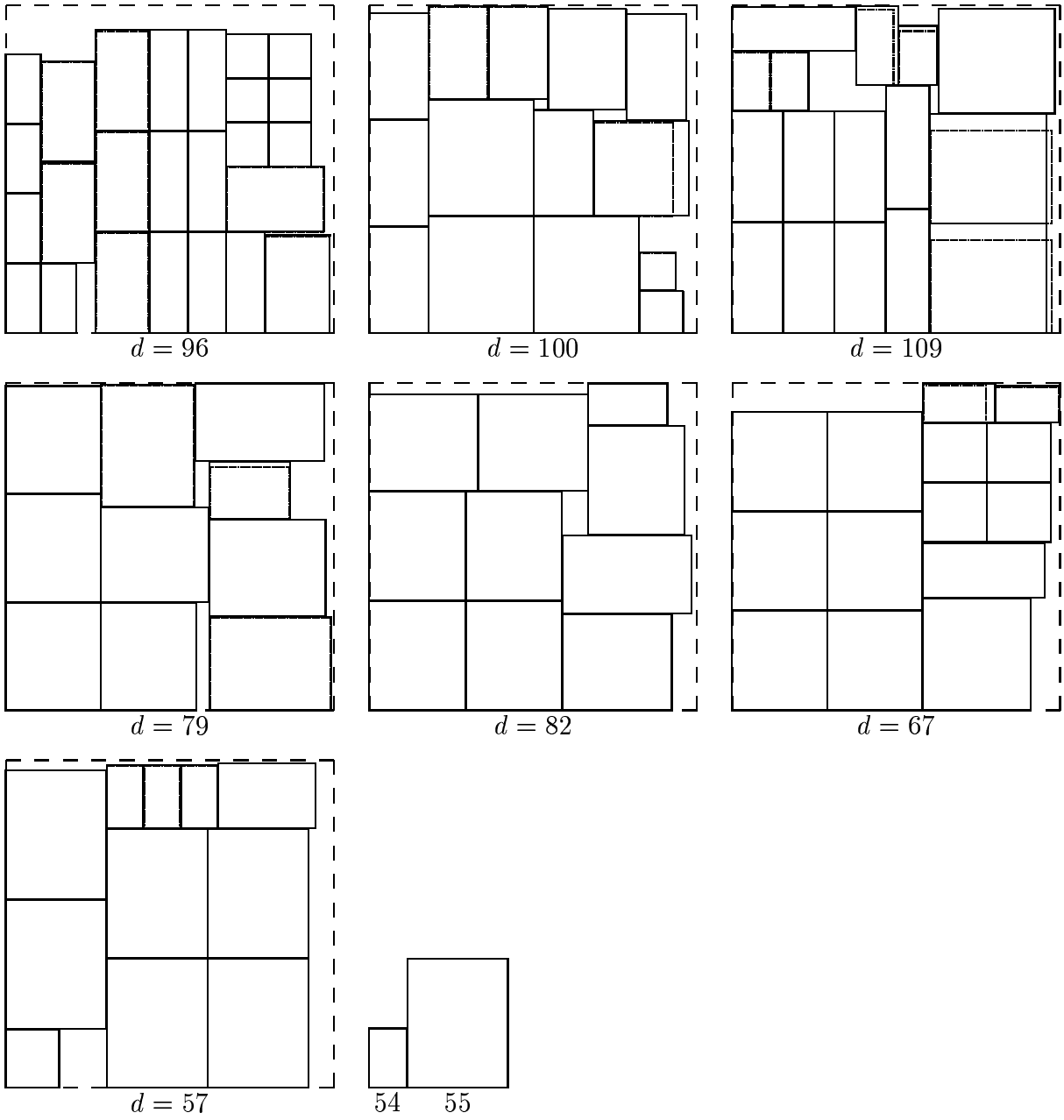

Figure 4: Found solution of instance 13, using unrestricted two-dimensional knapsack filling for each layer. The two last boxes with depth 54 and 55 respectively could not be packed into the container.

boxes into the layer such that the largest possible box volume is obtained. As in the previous algorithm, boxes are paired for each layer to obtain a uniform depth.

The two-dimensional knapsack problem is however $\mathcal{NP}$-hard in the strong sense and it quickly appeared that the problem was too difficult to solve to optimality with that many boxes present. Thus an upper limit on the investigated branching nodes was imposed for the two-dimensional knapsack problem, after which the algorithm returned the so far best solution. This limit was set to a large value of 5 million nodes.

Computational experiments with this algorithm are reported in Table 3, and an example of the filling is illustrated in Figure 4. Although the filling of 90.3% is better than that of George-Robinson and Hemminki, it is not impressive compared to the very huge time consumption.

One can draw several conclusions from this experiment: By dropping the guillotine cutting constraint, it is possible to pack each layer better than by the strip packing approach. But current techniques are simply not good enough for solving two-dimensional knapsack problems with about 100 boxes to optimality. Thus the layer filling algorithm must be aborted after a given number of iterations. Even with a very huge amount of iterations in every layer, the found solutions cannot compete with those found by the strip packing algorithm.

5 Conclusion

The presented algorithm generates packing schemes where almost 91% of the available volume is used. On average the instances are solved in less than half a minute, indicating the industrial applicability of this approach. The previously best algorithm [9] generates packing schemes with an average volume of 87.0%. This means that we reduced the vasted volume from 13.0% to 9.2% — a relative improvement of nearly 30%.

The algorithm differs from previous KCLP algorithms in several respects: by using the m-cut enumeration scheme for choosing depths and widths/heights, the algorithm has a possibility of backtracking in order to reach depth and width/height combinations that fit well together. Each strip filling problem is solved to optimality by use of an efficient KP algorithm, and pairings of boxes are allowed to form layers of uniform depth. Finally, the alternating directions of strips (and thus varying lengths of strips) allow us to place small series of similar boxes next to each other.

Relaxing the guillotine cutting constraint leads to an almost intractable problem, even when the layer approach is maintained. Thus the presented heuristic finds structured solutions of better quality in shorter time than by unrestricted approaches.

References

[1] E.E. Bischoff, M.D. Marriott (1990), "A comparative evaluation of heuristics for container loading", *European Journal of Operational Research*, **44**, 267–276.

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

[3] H. Dyckhoff (1990), "A typology of cutting and packing problems", *European Journal of Operational Research*, **44**, 145–159.

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

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

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

[7] P.C. Gilmore and R.E. Gomory (1965), "Multistage cutting stock problems of two and more dimensions", *Operations Research*, **13**, 94–120.

[8] J. Hemminki (1993), "A heuristic for container loading", Report 141, Department of Applied Mathematics, University of Turku, Finland.

[9] J. Hemminki (1994), "Container loading with variable strategies in each layer", presented at *ESI-X, EURO Summer Institute*, Jouy-En-Josas, France, July 2–15, 1994.

[10] T. Ibaraki (1987), "Enumerative Approaches to Combinatorial Optimization – Part 2", *Annals of Operations Research*, **11**, Baltzer, Basel.

[11] S. Martello, D. Pisinger, D. Vigo (1998), "The Three-Dimensional Bin Packing Problem", accepted for publication *Operations Research*.

[12] D. Pisinger (1997), "A minimal algorithm for the 0-1 Knapsack Problem", *Operations Research*, **45**, 758–767.

[13] G. Scheithauer (1991), "A three-dimensional bin packing algorithm", *Journal of Information Processing and Cybernetics*, **27**, 263–271.

[14] G. Scheithauer (1992), "Algorithms for the container loading problem", *Operations Research Proceedings 1991*, Springer-Verlag Berlin, Heidelberg, 445–452.