CEJOR (2012) 20:337–354
DOI 10.1007/s10100-010-0184-1

ORIGINAL PAPER

A heuristic for solving large bin packing problems in two and three dimensions

Daniel Mack · Andreas Bortfeldt

Published online: 31 December 2010

Abstract The more-dimensional bin packing problem (BPP) considered here requires packing a set of rectangular-shaped items into a minimum number of identical rectangular-shaped bins. All items may be rotated and the guillotine cut constraint has to be respected. A straightforward heuristic is presented that is based on a method for the container loading problem following a wall-building approach and on a method for the one-dimensional BPP. 1,800 new benchmark instances are introduced for the two-dimensional and three-dimensional BPP. The instances include more than 1,500 items on average. Applied to these very large instances, the heuristic generates solutions of acceptable quality in short computation times. Moreover, the influence of different instance parameters on the solution quality is investigated by an extended computational study.

Keywords Bin packing · Heuristics · Single bin-size bin packing problem · SBSBPP · Single stock-size cutting stock problem · SSSCSP

1 Introduction

This article deals with the bin packing problem in two and three spatial dimensions, referred to hereafter as the “multi-dimensional BPP”. In the three-dimensional case (3D-BPP) it can be defined as follows: Let J be a set of rectangular-shaped items and let an infinite number of identical rectangular containers (or bins) be given. Each item j of J is specified by its dimensions length l_j, width w_j and height h_j ($j = 1, \dots, n, n = |J|$). The containers have the common length L, width W and height H.

D. Mack (✉) · A. Bortfeldt
Department of Information Systems, FernUniversität in Hagen, 58084 Hagen, Germany
e-mail: daniel.mack@fernuni-hagen.de

A. Bortfeldt
e-mail: andreas.bortfeldt@fernuni-hagen.de

The objective is to find a feasible arrangement of all items within a minimal number of bins. Obviously, the two-dimensional BPP (2D-BPP) can be viewed as a special case of the 3D-BPP. An instance of the multi-dimensional BPP can be denoted as a tuple (L, W, H, J).

An arrangement, also called packing plan, is considered as feasible if each item lies completely within a container and is arranged parallel to its side walls. Furthermore, no item must overlap with another item placed within the same container.

In the cutting and packing (C&P) area, a distinction is usually made between weakly and strongly heterogeneous item sets. An item set is considered as weakly heterogeneous if it consists of only a few distinct item types, whereas strongly heterogeneous item stocks have more distinct item types, but only a few individual items per type. Two items are of the same type if they are congruent. In this article, both strongly and weakly heterogeneous sets are dealt with. According to the typology for C&P problems proposed by Wäscher et al. (2007), the BPP is referred to as Single Stock-Size Cutting Stock Problem (SSSCSP) or Single Bin-Size Bin Packing Problem (SBSBPP).

In the relevant literature the following constraints are commonly imposed:

(C1) Orientation constraint

While a rotation of the rectangular items by 90° is generally feasible, the orientation constraint fixes a specific orientation variant for each item, thus forbidding any rotation.

(C2) Guillotine cut constraint

This constraint demands that all packed items are reproducible by a series of guillotine cuts. A guillotine cut splits a block into two smaller blocks, where the slice plane is parallel to one side of the initial block.

Regarding the constraints (C1) and (C2) Lodi et al. (1999a) introduce four possible BPP subtypes:

RF: Items may be rotated by 90° (R), guillotine cut constraint not imposed (F);
RG: Items may be rotated by 90° (R), guillotine cut constraint is imposed (G);
OF: Orientation of items is fixed (O), guillotine cut constraint not imposed (F);
OG: Orientation of items is fixed (O), guillotine cut constraint is imposed (G).

Of course, a feasible solution for the subtype OG is also valid for the other three subtypes and a feasible solution for subtype OF or RG is also a valid solution for subtype RF.

The 3D-BPP occurs in practice when goods that are wrapped in rectangular boxes have to be stowed into standard containers, trucks or railway wagons. Several practical applications of the 2D-BPP are mentioned in Lodi et al. (1999a). Both the orientation and the guillotine constraint often have to be observed in cutting problems, due to the nature of the material surfaces (e.g. as a result of rolling) or the cutting technology.

Considering 2D and 3D BPP-instances with a very large number of items can be justified as follows. On the one hand, large scale benchmark instances were introduced for other C&P problems. As an example Mumford-Valenzuela et al. (2003), proposed instances with up to 5,000 items for the two dimensional strip packing problem. On the other hand, larger bin packing problems can be found in real life applications.

One of the authors was involved in developing a software system for an Australian steel producer. In this particular case, the bin packing problem occurred as a cutting problem where ordered steel blocks of given dimensions were to be sawn out of larger steel blocks of identical size. The software was designed to accept problems with up to 5,000 order blocks. In the area of transportation, often more than 200 items fit into a single standard container (e.g. household consignements, baggage, export shipping etc.).

In this article a heuristic for the multi-dimensional BPP and the subtype RG is proposed which may be applied to problems with 1,000 items and more. The following section gives an overview of literature dealing with the multi-dimensional bin packing problem. Section 3 describes the heuristic. In Sect. 4 new benchmark instances with large items sets are introduced and the results of extensive tests are evaluated. Section 5 resumes the article.

2 Literature overview

The bin packing problem is NP-hard, regardless of its dimensionality. Therefore heuristic approaches for the BPP dominate in the relevant literature, at least in the multi-dimensional case. These approaches can be classified into constructive heuristics (CH), improvement heuristics (IH) and, most popular in recent years, metaheuristics (cf. Glover and Kochenberger 2003). Metaheuristics include advanced local search strategies such as Simulated Annealing (SA), Tabu Search (TS) or Guided Local Search (GLS) and population-based approaches as genetic algorithms (GA). However, other paradigms such as Constrained Programming (CP) for example are also viewed as metaheuristic approaches. Branch and Bound (B&B) and related concepts (Tree Search, TRS; Branch and Cut, B&C) are applied in exact, but also in heuristic algorithms. Table 1 presents a representative sample of methods for the multi-dimensional bin packing problem and its different subtypes.

For each algorithm in Table 1, the problem variant is given by the dimensionality, the number of distinct container types (1 or "M", "M" stands for "multiple") and the subtype addressed. In the case of multiple container types, the objective is not to minimize the number of containers, but rather to minimize the overall shipping costs. The method type is specified and exact algorithms are labelled as such. In the last column, the maximal problem size in terms of number of items is given for which results were reported. For an extensive bibliography for the Bin Packing Problem the reader is referred to Coffman et al. (2004).

In the relevant literature relatively few publications deal with the multi-dimensional bin packing problem. Algorithms are rare, especially for the three-dimensional BPP. The problem size addressed by the exact, but also by most heuristic algorithms is rather small, increasing to only 200 items. It can also be observed that the benchmark instances often include large items compared to the container dimensions. As an example, for the instances introduced by Ivancic et al. (1989), a single container may only accommodate around 10 boxes. Several important practical situations are not covered by those instances; one need only consider packing retail household appliances into a

Table 1 Algorithms for multi-dimensional bin packing problems with rectangular items

Authors, source	Dimension	No. of cont. types	Sub-type	Method type	Max. calculated problem size
Bengtsson (1982)	2	1	RF	IH	200
Berkey and Wang (1987)	2	1	OF	CH	100
Ivancic et al. (1989)	3	M	RF	CH	180
El Bouri et al. (1994)	2	1	RF	TRS	200
Bischoff and Ratcliff (1995)	3	1	RF	CH	180
Martello and Vigo (1998)	2	1	OF	B&B, exact	200
Lodi et al. (1998)	2	1	RG	TS	164
Lodi et al. (1999a)	2	1	all	TS	100
Lodi et al. (1999b)	2	1	OG	TS	100
Martello et al. (2000)	3	1	OF	B&B, exact	90
Terno et al. (2000)	3	1	RF	B&B	–
Bortfeldt (2000)	3	M	RF	CH, IH	180
Dell'Amico et al. (2002)	2	1	RF	B&B, exact	100
Lodi et al. (2002)	3	1	OF	TS	200
Faroe et al. (2003)	3	1	OF	GLS	100
Eley (2003)	3	M	RF	TRS	180
Lodi et al. (2004)	3	1	RF	TS	200
Pisinger and Sigurd (2007)	2	1	OF, OG	CP, B&C, exact	100
Crainic et al. (2008)	2, 3	1	OF	CH, IH	200

standard container. Furthermore, most algorithms are tested using either only weakly or only strongly heterogeneous problem instances.

Given these arguments, it seems useful to conceive a heuristic capable of calculating problem instances of much greater size (1,000 items and more). Accordingly, new large problem instances are needed. These instances should vary along with different parameters, such as the average item size compared to the container size, in order to study the influence of basically different instance types on the solution quality empirically.

3 The heuristic for the multi-dimensional BPP

In the sequel, the heuristic is described for the 3D case. It consists of five modules (see Fig. 1):

– *3CLH*: an algorithm for the three dimensional container loading problem (CLP). The CLP is defined as follows: Let a set J of rectangular items (boxes) and a single container of length L, width W and height H be given. Consider the same feasibility conditions and constraints concerning the placement of items as for the multi-dimensional BPP. The objective is to find a feasible arrangement of a subset of items in the container so that the packed volume is maximised. A CLP instance

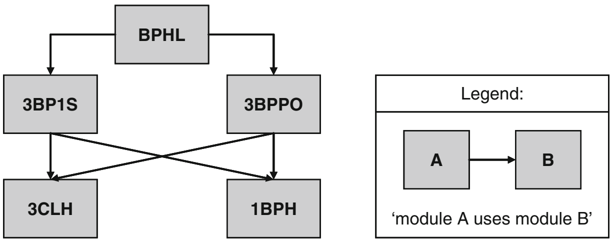

Fig. 1 Modular layout of the BPP heuristic

I_{CL} can be denoted as a tuple (L, W, H, J). Here, the algorithm proposed by Pisinger (2002) is applied to solve CLP instances.

- *1BPH*: an algorithm for solving the 1D-BPP. The heuristic of Bortfeldt (2005) is used here (see also Bortfeldt (2007)).
- *3BP1S*: a heuristic to generate a single solution for the 3D-BPP using a specific parameter set.
- *3BPPO*: a heuristic for further optimising a solution found with *3BP1S*.
- *BPHL*: the main module of the 3D-BPP-heuristic ("*H*" stands for heuristic, "*L*" for "Layer").

Obviously, the 3D-heuristic is also applicable to 2D-problems since 2D-items (rectangular containers, packing pieces) can be considered as 3D-items with height (or minimum dimension) 1. Each module will be specified in the following in greater detail. No distinction will be made between 3D and 2D problems.

3.1 The CLP-algorithm (module 3CLH)

The CLP-heuristic by Pisinger (2002) follows a layer-building approach. A generated packing plan consists of several vertical layers (see Fig. 2). These layers follow one after another in the longitudinal direction of the container. Each layer is composed by a rectangular block $l_l \times W \times H$ wherein several boxes are arranged. l_l is the layer depth. The volume utilisation of the layer is defined as the ratio of the total volume of packed items and the layer volume while the volume utilisation of the container is defined analogously. A layer consists in general of several horizontal and/or vertical strips. A horizontal strip is formed by several items that are placed one after another in the width direction without leaving gaps, while vertical strips are formed by items being placed one upon the other (see Fig. 2).

The CLP-heuristic is devised as a two-level tree search algorithm, where branching at the layer level and branching at the strip level are interlaced. With each branching at the layer level some new layer variants of different depths (l_l) are constructed. Item dimensions of greater size and frequently-occuring item dimensions are used as appropriate layer depths. In order to limit the search effort, the number of layer depths considered for the next branching is restricted by a parameter *cut3*. At the strip level, the procedure is similar, and the number of strip widths used for the next branching is limited by a parameter *cut2*. For any given layer depth and strip width an instance

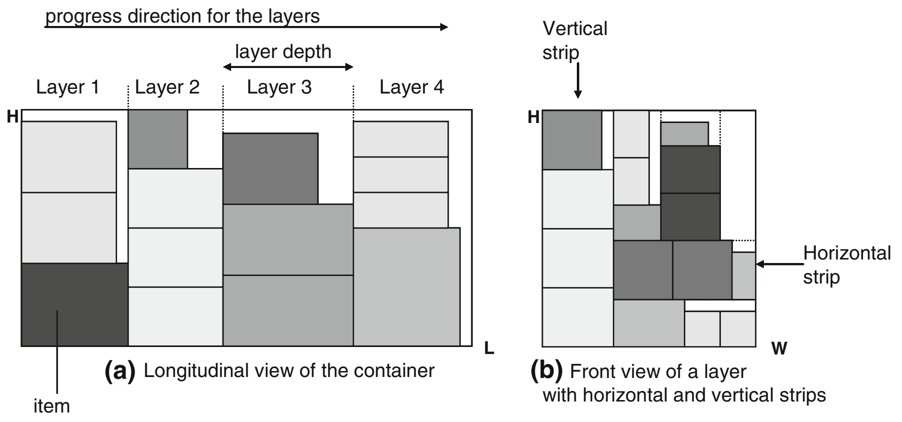
progress direction for the layers
layer depth
Layer 1
Layer 2
Layer 3
Layer 4
H
L
item
(a) Longitudinal view of the container
Vertical strip
H
W
Horizontal strip
(b) Front view of a layer with horizontal and vertical strips

Fig. 2 Structure of a solution generated by 3CLH

of the one-dimensional knapsack problem (1D-CLP) remains to be solved and this is done efficiently by an integrated 1D-CLP algorithm.

3.2 The 1D-BPP-algorithm (module 1BPH)

The 1D-BPP-algorithm is characterised here very briefly. A given 1D-BPP instance is calculated using the well-known "First Fit Decreasing" (FFD) heuristic as well as some stronger constructive and improvement heuristics based on FFD. The algorithm stops as soon as an optimal solution is found, which is verified by different lower bounds taken from the relevant literature. If no optimal solution has been found, a B&B-method is applied, performing a bin-oriented branching scheme. Different branching strategies are employed with regard to the selection of packing patterns tried per branching step. These strategies are tailored to different instance types where an instance type is characterised by the average relative item size. As an example, for so called triplet instances for which the average relative item size amounts to exactly one third of the bin size, a specific branching strategy is applied.

3.3 Generation of solutions for the 3D-BPP (module 3BP1S)

Figure 3 shows the pseudocode of module *3BP1S*. The heuristic follows an intuitive approach described, e.g., by Bischoff and Ratcliff (1995): a solution for the 3D-BPP is determined by iteratively filling new bins using the algorithm *3CLH* until all items of *J* are loaded. The input parameter *nbbest* designates a previously-found minimal number of bins for the given instance, *cut2* and *cut3* are parameters of the CLP-heuristic described above. *3BP1S* stops immediately without any new solution if it turns out that *nbbest* cannot be improved. Note that the parameters *cut2* and *cut3* are mainly influencing the search effort and the solution to be calculated.

After a new solution *sb* has been determined, it is assured that the layers are distributed in an optimal way among the bins. Generally, a redistribution of the layers could

```
procedure 3BP1S (in: problem data L, W, H, J; parameters cut3, cut2,
                 best no. of bins so far nbbest; out: 3D-BPP solution sb)
{initialise}
sb := ∅; {3D-BPP-solution}
nb := 0; {no of loaded bins}
R := J; {remaining items}
VR := total volume of R;
{find 3D-BPP-solution by iteratively loading single bins}
while R ≠ ∅ do
    if $nb + \lceil VR/(L \cdot W \cdot H) \rceil \geq nbbest$ then
        set sb := ∅; break; {no new best solution can be found}
    endif;
    calc. sol. s for CLP-inst. $I_{CL}$ := (L, W, H, R) using 3CLH and cut3, cut2;
    sb := sb ∪ s; {expand solution by a supplementary filled bin}
    nb := nb + 1;
    update remaining items R and appropriate volume VR;
endwhile;
{optimize 3D-BPP solution by redistributing the layers}
if sb ≠ ∅ and R = ∅ then {a new solution sb has been found}
    let {$l_i$} be the set of layer depths in solution sb;
    if $\sum_i l_i \leq (nb - 1)L$ then {no. of bins nb may possibly be improved}
        calc. sol. s1 for the 1D-BPP inst. given by L and {$l_i$} using 1BPH;
        if number of bins in solution s1 < nb then
            redistribute the layers in sb according to s1;
        endif;
    endif;
endif;
end.
```

Fig. 3 Algorithm of heuristic *3BP1S*

achieve a saving of at least one bin. Note that the loading of single bins by *3CLH* is done in a greedy fashion and this in fact can lead to a suboptimal result for the 3D-BPP instance. Let $\{l_i\}$ be the set of layer depths in solution *sb*. If it seems possible to save a bin by redistributing layers, the 1D-BPP instance given by set $\{l_i\}$ and bin length L is solved by algorithm *1BPH*. If the 1D-BPP solution contains a lower number of bins compared to the solution *sb* calculated before, then the layers of *sb* are redistributed according to the 1D-BPP solution in order to realise the bin saving.

3.4 Main module of the 3D-BPP-heuristic (module BPHL)

Figure 4 describes the main module of the 3D-BPP heuristic. Given a problem instance, multiple solutions are generated by systematically altering the parameters *cut3* and *cut2* of the integrated 3D-CLP algorithm. For a specific pair of values (*cut3*, *cut2*) two solutions are generated by means of module *3BP1S* and the best solution so far *sbest* is updated if necessary. At the first call of *3BP1S*, the progress direction of the layers is set parallel to the container width W, i.e. the layers follow one after the other in direction of the container width (unlike the explanations in Sect. 3.1). At the second call of *3BP1S*, the progress direction of the layers is set parallel to the container length L. The switch of progress direction is implemented by permuting container length and container width when calling *3BP1S*. An incumbent best number of bins *nbbest* is passed to *3BP1S* in order to shorten the search where possible (see Sect. 3.3).

```
procedure BPHL (in: problem data L, W, H, J, parameters: cut3max, cut2max;
                out: best 3D-BPP-solution sbest)

nbbest := |J|;{initial value}
{generate solutions for different parameter sets}

for cut3 := 1 to cut3max by 1 do                        {no. of branches at layer level}
    for cut2 := cut3^2 to cut2max by cut3^2 do          {no. of branches at strip level}
        for sdir := 1 to 2 by 1 do                      {progress direction of the layers}
            if sdir = 1 then
                set LL := W; WW := L;                   {progress dir. is W}
            else
                set LL := L; WW := W;                   {progress dir. is L}
            endif;
            call 3BP1S(LL, WW, H, J, cut3, cut2, nbbest, sb);
            if  sb ≠ ∅ and no. of bins used in sb < nbbest then
                sbest := sb; nbbest := no. of bins used in sbest;
                sdirbest := sdir;
            endif;
        endfor sdir;
    endfor cut2;
endfor cut3;

{call postoptimisation procedure with current best solution}
call 3BPPO(L, W, H, J, cut3max, cut2max, sdirbest, sbest);
end.
```

Fig. 4 Algorithm of the main module *BPHL*

BPHL is tailored to large problem instances with 1,000 or more items. In order to determine an initial solution rapidly, the parameters are initially set to their minimal values ($cut3 = 1$, $cut2 = 1$). With the following calls, the parameters are incremented iteratively and the search effort increases. Parameter *cut3* is always incremented by 1, whereas *cut2* is raised by the amount $cut3^2$ per step. The search effort spent for a given parameter set is restricted by a time limit *t1*. Finally, the best solution so far is further processed by the heuristic *3BPPO*. The time effort of this post processing is limited by time limit *t2*.

3.5 Post processing of a 3D-BPP solution (module 3BPPO)

For each solution determined with heuristic *3BP1S*, there is a unique progress direction for the generated layers, given either by the container length or the container width. The post processing heuristic *3BPPO* aims at combining the two progress directions within a single solution. Therefore, some layers of a given *3BP1S*-solution are replaced with new layers following the alternative progress direction. Figure 5 shows the different steps of the post optimisation for a given example.

First, several layers with weakest volume utilisations are determined and removed from the solution; in the example the four weakest layers lie in distinct bins. In step 2, the remaining layers are redistributed among a minimal number of bins. The corresponding 1D-BPP instance is solved by module *1BPH*. In step 3, the heuristic checks whether additional items can be stowed into the containers. If this is the case, the remaining space is filled by algorithm *3CLH*. The progress direction of the additional layers runs orthogonally to the progress direction of the initial layers. In the fourth and last step additional bins are opened and filled with the CLP-algorithm until the

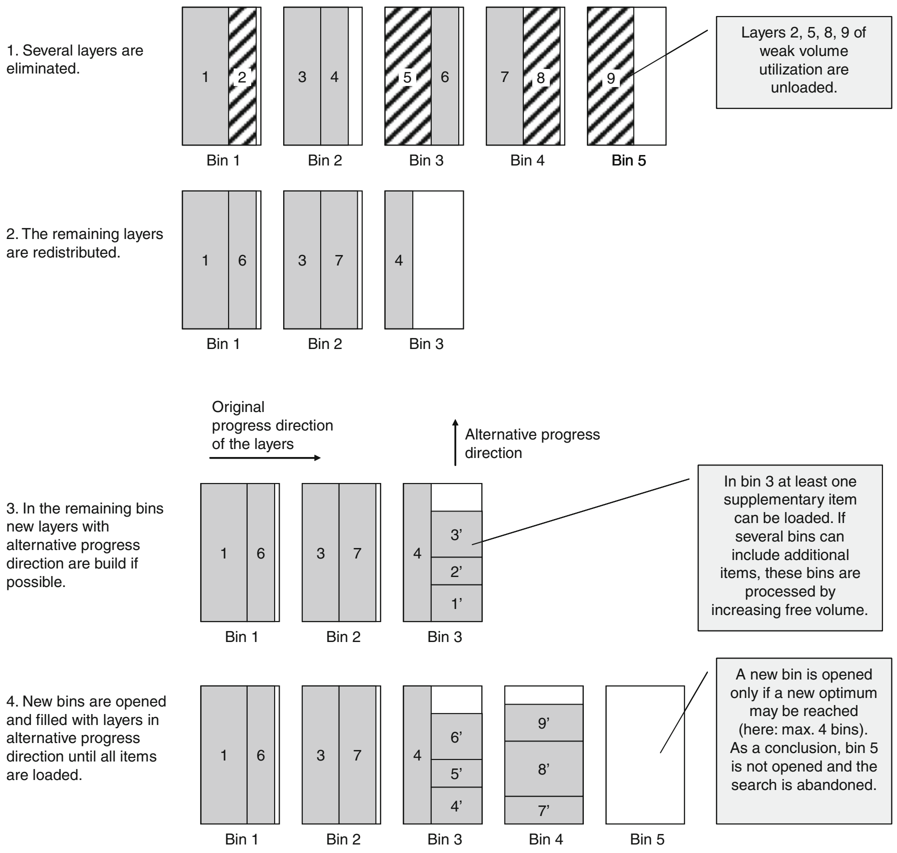
1. Several layers are eliminated.
1
2
3
4
5
6
7
8
9
Bin 1
Bin 2
Bin 3
Bin 4
Bin 5
Layers 2, 5, 8, 9 of weak volume utilization are unloaded.
2. The remaining layers are redistributed.
Original progress direction of the layers
Alternative progress direction
3. In the remaining bins new layers with alternative progress direction are build if possible.
3'
2'
1'
In bin 3 at least one supplementary item can be loaded. If several bins can include additional items, these bins are processed by increasing free volume.
4. New bins are opened and filled with layers in alternative progress direction until all items are loaded.
6'
5'
4'
9'
8'
7'
A new bin is opened only if a new optimum may be reached (here: max. 4 bins). As a conclusion, bin 5 is not opened and the search is abandoned.

Fig. 5 Four steps of post optimisation for a given number of layers to be unloaded

complete item stock is packed. The progress direction of the new layers is chosen orthogonally to the progress direction of the initial layers.

Figure 6 specifies the post optimisation in detail. In the main loop the variable k designates the number of layers to be unloaded. k is incremented from 1 to $nl-1$, where nl stands for the total number of layers in the input solution sb. For a given value k, the four steps of Fig. 5 are executed.

At the beginning, the search effort is relatively small as the number of items to reload is small. Effort grows continuously when k is incremented. The post optimisation is stopped when time limit $t2$ is reached.

Heuristic *3BPPO* is time consuming because of the iterative increase of resolved layers. Therefore the post optimisation is only called once to process the formerly determined best solution.

Finally, it is proven that the guillotine cut constraint is respected by the proposed heuristic. The guillotine cut constraint is met at the layer level through the integrated CLP-heuristic (cf. Fig. 2). A complete solution observes this constraint because the layers can be isolated by cuts running parallel to the container sides. Furthermore, using

```
procedure 3BPPO (in: problem data $L$, $W$, $H$, $J$, param. cut3, cut2,
                 progress direction sdir, inout: solution sb)
if sdir = 1 then   LL := W; WW := L;   {progress dir. of sb is container width W}
           else    LL := L; WW := W;   {progress dir. of sb is container length L}
endif;
sbold := sb;
let ($l_1$, $l_2$,...,$l_{nl}$) be the series of layer depths in sbold, sorted by ascending volume
    utilisation;
for k := 1 to nl–1 do
        {step 1: eliminate some layers}
        unload the first k layers of sbold and
            set R := set of items in layers 1,...,k of sbold;
        {step 2: redistribute kept layers}
        solve the 1D-BPP instance (LL, {$l_j$ | j = k+1,...,nl}) using 1BPH and
            let s1 be the solution including $nb_0$ bins;
        generate 3D-BPP partial solution sbnew by combining
            layers k+1,...,nl of sbold in $nb_0$ 3D-bins according to solution s1;
        sort the bins in sbnew by ascending free distance in progress dir. sdir;
        {step 3: complete non-empty bins}
        for each bin ib in sbnew do
          if free distance lfree(ib) $\geq$ minimal item dimension in R then
            {generate layers in bin ib with progress direction orthogonal to sdir}
            calculate solution s for CLP instance $I_{CL}$ = (WW, lfree(ib), H, R) using
              3CLH and parameters cut3, cut2;
            complete the arrangement in bin ib by arrangement s;
            remove items used by s from R;
          endif;
        endfor;
        {step 4: fill additional bins with layers orthogonal to direction sdir}
        call 3BP1S(WW, LL, H, R, cut3, cut2, no. of bins(sb) – $nb_0$, sbr);
        if solution sbr $\neq \emptyset$ then
          sbnew := sbnew $\cup$ sbr;
          if no. of bins(sbnew) < no. of bins(sb) then sb := sbnew; endif;
        endif;
endfor;
end.
```

Fig. 6 Algorithm *3BPPO*

alternative progress directions for layers does not violate the guillotine cut constraint (see Fig. 5).

4 Numerical test

The heuristic *BPHL* was implemented in C and tested using a standard PC (1.86 GHz Intel and 500 MB RAM). In the following the benchmark instances used for the test are introduced before the test results are presented and analysed.

4.1 Problem instances

On the one hand, BPHL shall be applied to 740 well-known benchmark instances of limited problem size (max. 200 items). On the other hand, 1,800 new BPP instances with larger item stocks will be introduced.

Among the 740 problem instances from literature, 300 2D instances were proposed by Berkey and Wang (1987); they are denoted here as BW instances. Another 200 2D instances were introduced by Martello and Vigo (1998) (MV instances). The given 500 2D instances are grouped in 10 instance classes of 50 instances each. In each class the item numbers range between 20 and 100 items per instance. A detailed description of these instances can also be found in, e.g., Lodi et al. (1999a).

240 3D instances were designed by Martello et al. (2000) (MPV instances) and they are grouped in 6 classes having numbers 1, 4, 5, 6, 7 and 8; note that further original classes (2, 3, 9) are usually *not* considered in the literature (cf. Faroe et al. 2003). Each of the 6 classes comprises 40 instances with identical container dimensions, but different item numbers (50, 100, 150, 200 items per problem).

In order to test the ability of BPHL to deal with very large item stocks, 1,800 new BPP instances shall be derived from the well-known CLP benchmark instances proposed by Bischoff and Ratcliff (1995) (short: BR). These BR-instances are grouped in 15 test cases with 100 instances each. All instances of any test case have the same number of distinct item types. The item sets of the test cases vary from weakly to strongly heterogeneous. The bin dimensions always correspond to the internal dimensions of a standard 20-foot-container.

The new benchmark instances should meet the following conditions:

– 2D and 3D-instances are to be provided.
– Only large instances are requested with approximately 1,000 items or more.
– Item sets with different average volumes (compared to the container volume) should be offered.
– The item sets should vary from weakly to strongly heterogeneous.
– The material lower bound for the number of bins *L0* (also called continuous lower bound) should have a uniform value for greater subsets of instances in order to facilitate the comparison of results. The bound *L0* is defined as ⌈total item volume / container volume⌉

The generation of problem instances is first described for the 3D-case:

Container size: Three container sizes are considered, where the container volume represents 100, 25 and 15% of the volume of a 20-foot-container. The sizes are referred to in the following as "100%", "25%" and "15%". All internal dimensions of the 20-foot-container are reduced proportionally for the variants "25%" and "15%". The item dimensions are adopted unchanged from the original instances. This ensures that different scales of items are generated in comparison with the container size.

Item types: Two modes of item type selection (short: selection modes) are distinguished. In mode 1, all item types of the original BR instance are adopted to generate one new BPP instance. In mode 2, all item types of 10 subsequent problem instances are aggregated to a new instance (e.g., the item types of instances 1 to 10 of a test case are taken for one BPP instance). By this the number of item types per problem is multiplied by a factor 10, so that the instances created with selection mode 2 present between 30 and 1,000 types of items and offer a much greater level of heterogeneity compared to instances of selection mode 1.

Numbers of items: having determined the container and the item types, the numbers of items of the different types are set in a way that the lower bound *L0* will be: *L0* = 10 for container variant 100%, *L0* = 40 for container variant "25%" and *L0* = 100 for container variant "15%". The numbers of items per type are determined at random.

Table 2 Denomination of 12 new test cases for the BPP

Test cases (150 instances each)	Container volume	100 (%)	25 (%)	15 (%)
	L0	10	40	100
3D	Selection mode 1	MB01	MB03	MB05
	Selection mode 2	MB02	MB04	MB06
2D	Selection mode 1	MB07	MB09	MB11
	Selection mode 2	MB08	MB10	MB12

10 3D-BPP instances are generated for each combination of container size (“100%”,“25%”,“15%”), mode of item type selection (1 or 2) and BR-test case (1 to 15). In total, $3\times2\times15\times10=900$ 3D-BPP instances are created. 2D-instances are generated in an analogous fashion. However, the container height and one dimension of the original item types (selected at random) are set to 1. Again, 900 2D-instances are generated in total. Table 2 proposes a denomination for the 12 new test cases with 150 instances each. Unlike the original BR-instances, a test case represents a specific combination of dimensionality, mode of item type selection and container size. Depending on the problem size, it can be stated that almost every instance (except of ten instances) does include 1,000 or more items. The average number of items per problem instance over the 1,800 instances is 1,566. 2D-instances present on average 1,646 rectangles, 3D-instances present on average 1,486 boxes. The smallest instance has 792, the largest 4,232 items. The 1,800 BPP-instances can be downloaded from the website www.fernuni-hagen.de/WINF (see area “Download”, File MB_BPP_instances.zip).

4.2 Test results

To compute the 740 benchmark instances from literature the heuristic BPHL was parameterized as follows: $t1=3s$, $t2=3s$, $cut3max=2$, $cut2max=8$. An overall time limit of 3 (1.86 GHz-) seconds was applied. Results for the 2D and 3D instances are reported in Tables 3 and 4, respectively.

Both tables are designed in a similar manner:

– In the left part the computed instance sets are indicated by their dimension and class name (BWi, MVi or MPVi). Additionally, the mean continuous lower bound *L0* is given per class.
– In the right part results for the heuristic BPHL and for some other methods from literature are presented. Only mean values over all instances of a class are given; *nb* means the number of required bins while nb_{rel} stands for the ratio *nb*/*L0*. For each compared method the addressed BPP type (OF, RG, cf. Sect. 1) is indicated. The time limit (*tl*) used is given where applicable and the types of the compared methods are also indicated (TS, GLS etc.).

Taking the 2D instances the method BPHL achieves slightly better results than the method(s) from Lodi et al. (1998) that address the same BPP type RG. The chosen time limit of 3 s shows the speed of the proposed heuristic. In average, the best

Table 3 Test results of heuristic *BPHL* for 500 2D benchmark problems

Dim	Test class	*L0*	Lodi et al. (1998)			BPHL	
			RG			RG	
			FC_{RG}	UB_F	TS		
			nb_{rel}	nb_{rel}	nb_{rel}	*nb*	nb_{rel}
2D	BW01	18.54	1.09	1.09	1.07	19.56	1.06
	BW02	2.48	1.04	1.06	1.04	1.50	1.01
	BW03	12.58	1.19	1.19	1.14	14.02	1.12
	BW04	2.38	1.05	1.07	1.05	2.44	1.02
	BW05	15.72	1.15	1.15	1.13	17.60	1.12
	BW06	2.16	1.10	1.10	1.10	2.20	1.03
Average BW		8.98	1.10	1.11	1.09	9.72	1.06
2D	MV07	14.38	1.18	1.18	1.16	15.68	1.10
	MV08	14.42	1.17	1.17	1.17	15.64	1.09
	MV09	27.42	1.54	1.54	1.54	42.42	1.54
	MV10	9.52	1.09	1.09	1.08	10.20	1.08
Average MV		16.44	1.25	1.25	1.24	20.99	1.20
Average BW/MV		11.96	1.16	1.17	1.15	14.23	1.12

Table 4 Test results of heuristic *BPHL* for 240 3D benchmark problems

Dim	Test class	*L0*(RG)	Lodi et al. (2002)			Faroe et al. (2003)			BPHL
			OF			OF			RG
			TS			GLS			
			$tl = 60$	$tl = 150$	$tl = 1,000$	$tl = 60$	$tl = 150$	$tl = 1,000$	$tl = 3$
			nb	*nb*	*nb*	*nb*	*nb*	*nb*	*nb*
3D	MPV1	23.78	32.43	32.23	31.98	32.65	32.35	32.08	28.88
	MPV4	38.73	73.50	73.50	73.50	73.85	73.75	73.55	73.30
	MPV5	14.00	18.05	18.00	17.85	17.98	17.80	17.70	17.45
	MPV6	21.20	24.28	24.10	24.03	24.28	24.08	24.00	23.20
	MPV7	11.93	15.30	15.18	15.00	15.10	14.90	14.75	14.55
	MPV8	16.45	20.98	20.93	20.65	20.80	20.58	20.48	19.83
Average		21.01	30.75	30.65	30.50	30.78	30.58	30.43	29.53

BPHL-solution was reached after only 0.1 s. From Table 4 it can be stated that BPHL achieves slightly better bin numbers than the compared metaheuristics from other authors. However, a direct comparison is not possible, as these methods consider fixed orientations of all items (BPP type OF) while BPHL allows rotating the items. Therefore, this comparison should be interpreted with care. However, the time limit of 3 s

Table 5 Test results of heuristic *BPHL* for 1,800 new benchmark problems (part 1)

Dim	Test case	*L0*	BPHL						BPHL-		
			nb (bins)	*gap* (%)	*vutil* (%)	*n_opt*	*n_opt* (%)	*t_tot* (s)	*nb* (bins)	*gap* (%)	*vutil* (%)
3D	MB01	10	11.0	10.40	86.33	11	7.3	125.5	11.1	10.87	85.97
	MB02		10.7	6.67	88.99	50	33.3	99.1	10.7	7.13	88.60
	MB03	40	46.9	17.17	84.29	0	0.0	101.2	47.8	19.52	82.63
	MB04		44.0	10.10	89.71	0	0.0	97.1	45.1	12.68	87.65
	MB05	100	130.4	30.41	76.33	0	0.0	123.2	133.9	33.86	74.37
	MB06		122.1	22.11	81.51	0	0.0	122.6	124.4	24.38	80.02
Average 3D			–	16.14	84.53	–	6.8	111.5	–	18.07	83.21
2D	MB07	10	10.2	2.00	93.28	120	80.0	26.7	10.2	2.00	93.28
	MB08		10.2	1.53	93.40	127	84.7	22.5	10.2	1.73	93.21
	MB09	40	41.4	3.53	95.38	4	2.7	118.1	41.5	3.87	95.07
	MB10		40.9	2.17	96.61	27	18.0	106.5	41.0	2.42	96.38
	MB11	100	111.1	11.07	89.57	0	0.0	122.5	111.9	11.87	88.92
	MB12		104.7	4.71	95.03	0	0.0	122.9	105.4	5.38	94.43
Average 2D			–	4.17	93.88	–	30.9	86.5	–	4.54	93.55
Average			–	10.16	89.20	–	18.8	99.0	–	11.31	88.38

applied to BPHL is very low, and the best BPHL results are reached on average after only 0.6 s, which again proves the efficiency of the proposed approach.

For the new benchmark instances introduced above, all test results were calculated using the following parameter set that has been extracted from a smaller pretest: $t1 = 60$ s, $t2 = 60$ s, $cut3max = 4$, $cut2max = 8$. Tables 5 and 6 summarise the results of the heuristic for all 1,800 problem instances:

- Each row represents average values over all 150 instances of the respective test case (cf. 4.1).
- The three first columns indicate the dimensionality, the name of the test case and the constant value of *L0*.
- "*BPHL*" stands for the full variant of the heuristic including post optimisation, whereas "*BPHL-*" represents the reduced variant without post optimisation.
- The average number of required bins (*nb*), the relative deviation from the lower bound *gap* (given by (*nb-L0)/L0* in %) and the average volume utilisation of the bins *vutil* (in %) are indicated per test case for both variants of the heuristic. For the full version *BPHL,* the number and percentage of instances (*n_opt*) are shown, for which an optimal solution with *L0* bins was reached. The total calculation time (*t_tot*) is specified in (1.86 GHz-) seconds.
- The columns "Effects of *3BPPO*" resume the differences of the results between *BPHL* und *BPHL-*. In addition, *n_i1* gives the number of instances per test case where at least one bin was saved by post optimisation (*3BPPO*). Note that negative signs in columns *nb* and *gap* indicate improved results.

Table 6 Test results of heuristic *BPHL* for 1,800 new benchmark problems (part 2)

Dim	Test case	*L0*	Effects of 3BPPO				Effects of 1BPH		
			nb (bins)	*gap* (%)	*vutil* (%)	*n_i1*	*t_1BPH* (s)	*t_1BPH* (%)	*n_i2*
3D	MB01	10	−0.05	−0.47	0.36	7	0.9	0.7	3
	MB02		−0.05	−0.47	0.39	7	1.2	1.2	0
	MB03	40	−0.94	−2.35	1.66	97	2.2	2.2	13
	MB04		−1.03	−2.58	2.06	137	1.9	1.9	0
	MB05	100	−3.45	−3.45	1.97	132	3.2	2.6	12
	MB06		−2.27	−2.27	1.49	147	2.6	2.1	6
Average 3D			−1.30	−1.93	1.32	87.8	2.0	1.8	5.7
2D	MB07	10	0.00	0.00	0.00	0	0.1	0.5	2
	MB08		−0.02	−0.20	0.18	3	0.1	0.6	0
	MB09	40	−0.13	−0.33	0.31	16	2.3	1.9	21
	MB10		−0.10	−0.25	0.24	15	4.2	3.9	11
	MB11	100	−0.80	−0.80	0.64	78	1.4	1.1	23
	MB12		−0.67	−0.67	0.60	83	1.2	1.0	10
Average 2D			−0.29	−0.37	0.33	32.5	1.5	1.5	11.2
Average			−0.79	−1.15	0.82	60.2	1.8	1.7	8.4

– The columns "Effects of *1BPH*" indicate the total calculation time of module *1BPH* as an absolute value (seconds) and as a percentage (% of the total calculation time). Furthermore, *n_i2* gives the number of instances per test case where *1BPH* leads to a bin reduction.

The test results are analysed under three aspects. Firstly, the overall performance of the heuristic is evaluated. Secondly, some of the components of the heuristic are assessed separately. Thirdly, the relation between the instance parameters and the yielded solution quality is investigated.

For the 2D-instances good results were achieved as the average volume utilisation of approximately 94% and the average gap to the rather weak lower bound *L0* of ca. 4.2% reveal. Nearly one in three instances was solved to proven optimality while the percentage of (globally) optimal solutions lies over 80% for the 2D-instances with the largest container size "100%" (test cases MB07, MB08).

For the 3D-instances the results provide a more differentiated picture. Again, for the 3D-instances with non-reduced 20-foot-containers (test cases MB01, MB02) a satisfactory solution quality was reached since the optimum is missed on average by one bin or less and at least 20% of these 300 instances were solved to optimality. However, much larger *gap* values are observed for the smaller "25%" and "15%" container sizes (see above for definition). These results also appear plausible to a certain extent if one bears in mind that as a consequence of the container reduction the items become more and more "bulky". Moreover, the volume utilisation is certainly negatively affected by the fact that the heuristic respects the guillotine cutting constraint (C2) in any case.

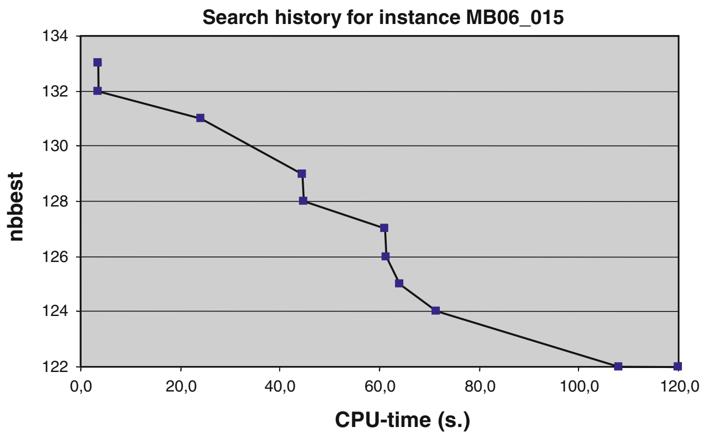

Fig. 7 Search history of *BPHL* for the instance MB06_015

If the calculation times are compared to those reported for BPP methods in the relevant literature (e.g., Faroe et al. 2003), it can be stated that the time effort spent with heuristic *BPHL* is rather small. All in all, the heuristic *BPHL* seems to be capable of producing solutions of acceptable quality for large scale 2D and 3D bin packing instances with 1,000 or more items in short calculation time of few minutes.

Module *1BPH* aims to reduce the number of bins by a redistribution of layers. In 101 of the 1,800 instances (5.6%), this measure is successful at least once during the search process. The effort is very small with 1.8 s per instance (1.7% of the average calculation time).

The post processing module *3BPPO* reduces the gap to bound *L0* by 1.15%-points on average and enhances the volume utilisation by 0.82%-points over all 1,800 problems. The post processing takes 40 to 60 s per instance, so that an omission of post processing can reduce the calculation time by nearly 50% to about 60 s without affecting the solution quality significantly. This underlines the capability of *BPHL* to produce acceptable results for large scale problems in a very short time.

Figure 7 shows the search progress for instance MB06_015. After a few seconds, an initial solution with 133 bins is determined, which can be improved rapidly by a redistribution of layers using module *1BPH*. During the first 60 s the parameters *cut2* and *cut3* are incremented successively and a solution with 128 bins is found. After 60 s, the post-processing iteratively unloads the weakest layers and creates layers in the alternative progress direction, achieving a decrease to 122 bins after 105 s.

Finally, the influence of the instance parameters to the solution quality is analysed. For 2D-instances, the determined volume utilisations are much higher on average than for the 3D-instances and consequently the *L0* gaps are lower. This is in line with the results reported in the relevant literature: good 2D-algorithms often yield volume utilisations of 98% and more, whereas reported filling rates for 3D-algorithms are generally significantly lower. This seems plausible, as for each additional dimension it becomes more difficult to avoid holes in the packing plan, particularly if supplementary geometric constraints are to be respected.

As indicated above, a clear correlation can also be observed between the averaged relative item sizes (relative to container size) and the average gap to the *L0* bound. As already explained this trend seems plausible since for increasing relative item sizes the items become more and more "bulky" thus requiring a larger number of bins. The modules for layer reallocation (*1BPH)* and post optimisation (*3BPPO)* are most successful at problems with higher relative item sizes.

Finally, the results for instances generated with mode 2 of item type selection are superior on average to the corresponding results for instances of selection mode 1 (cf. Sect. 4.1). The heterogeneity of item sets is much higher for selection mode 2. So this is not an intuitive result, as CLP-algorithms generally yield higher volume utilisations for problem instances of lower heterogeneity (cf., e.g., Moura and Oliveira 2005). The heterogenity of item sets does affect the achieved volume utilisation at least in two ways. On the one hand, a greater number of items per type allows for building compact gapless arrangements of items, i.e. less heterogeneous box sets are advantageous in this regard. On the other hand, a larger heterogeneity of items yields a greater richness of packing patterns. A possible explanation for the observed trend could be that the second effect dominates the first one for the given instances. In fact, it can be observed that even for problems of selection type 1, there are often not enough items per type to build larger gapless arrangements. Of course, this reasoning has to be verified again after further results are available for the new benchmark instances.

5 Summary

This article presents a heuristic for the bin packing problem (BPP) in two and three spatial dimensions with a single container type and rectangular shaped packing pieces. The heuristic is based upon a method for the 3D-container loading problem (CLP) following a layer building approach and an algorithm for the one-dimensional BPP. The proposed heuristic can be characterised as an improvement heuristic. In order to prove the effectiveness of the algorithm for large scale problem instances, 1,800 new benchmark instances with more than 1,500 items on average were introduced. The dimensionality, the relative size of the packing pieces and the heterogeneity of the item sets are varied systematically. The test showed the capability of the heuristic of calculating solutions of generally acceptable quality for large scale instances in short computation times of one or two minutes. Moreover, competitive results could be achieved for 740 well-known 2D and 3D benchmark instances of smaller size (up to 200 items) in less than 1 s. To the best knowledge of the authors multi-dimensional bin packing problems of this extreme size have not been dealt with in the relevant literature so far. An empirical study is analysing the dependency of the solution quality on instance features as average item size and heterogeneity of item sets. The consideration of further packing constraints remains a subject for further research.

Acknowledgments The authors wish to thank Mr. Jens Hochmuth for his valuable assistance in the technical implementation of the algorithm.

References

Bengtsson B-E (1982) Packing rectangular pieces—a heuristic approach. Comput J 25:353–357

Berkey JO, Wang PY (1987) Two dimensional finite bin packing algorithms. J Oper Res Soc 38:423–429

Bischoff EE, Ratcliff MSW (1995) Issues in the development of approaches to container loading. Omega 23:377–390

Bortfeldt A (2000) Eine Heuristik für multiple containerladeprobleme. OR Spectr 22:239–261

Bortfeldt A (2005) A hybrid procedure for the one-dimensional bin packing problem. Presentation on the 2th ESICUP-Meeting, Southampton 2005 Siehe. http://paginas.fe.up.pt/~esicup/

Bortfeldt A (2007) Verfahren für das eindimensionale Bin Packing Problem. Vortrag in der Forschungswerkstatt der Fakultät Wirtschaftswiss. der FernUniversität in Hagen, März 2007. S. http://www.fernuni-hagen.de/WINF

Coffman EG Jr, Csirik J, Johnson DS, Woeginger GJ (2004) An Introduction to Bin Packing. Bibliography. See. http://www.inf.u-szeged.hu/~csirik

Crainic TG, Perboli G, Tadei R (2008) Extreme point-based Heuristics for three-dimensional bin packing. INFORMS J Comput 20:368–384

Davies AP, Bischoff EE (1998) Weight distribution considerations in container loading. In: European business management school. University of Wales, Swansea, Technical report

de Castro Silva JL, Soma NY, Maculan N (2003) A greedy search for the tree-dimensional bin packing problem: the packing stability case. Int Trans Oper Res 10:141–153

Dell'Amico M, Martello S, Vigo D (2002) A lower bound for the non-oriented two-dimensional bin packing problem. Discrete Appl Math 118:13–24

El Bouri A, Popplewell N, Balakrishnan S, Alfa A (1994) A search based heuristic for the two-dimensional bin-packing problem. INFOR 32:265–274

Eley M (2003) A bottleneck assignment approach to the multiple container loading problem. OR Spectr 25:45–60

Faroe O, Pisinger D, Zachariasen M (2003) Guided local search for the three-dimensional bin-packing problem. INFORMS J Comput 15:267–283

Glover F, Kochenberger G (2003) Handbook of metaheuristics. Kluwer Academic Publishers, Dordrecht

Ivancic N, Mathur K, Mohanty BB (1989) An integer-programming based heuristic approach to the three-dimensional packing problem. J Manuf Oper Manag 2:268–298

Lodi A, Martello S, Vigo D (1998) Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem. In: Voss S, Martello S, Osman IH, Roucairol C (eds) Meta-heuristics: advances and trends in local search paradigms for optimisation. Kluwer Academic Publishers, Boston pp 125–139

Lodi A, Martello S, Vigo D (1999a) Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. INFORMS J Comput 11:345–357

Lodi A, Martello S, Vigo D (1999b) Approximation algorithms for the oriented two-dimensional bin packing problem. Eur J Oper Res 112:158–166

Lodi A, Martello S, Vigo D (2002) Heuristic algorithms for the three-dimensional bin packing problem. Eur J Oper Res 141:410–420

Lodi A, Martello S, Vigo D (2004) TSpack: a unified tabu search code for multi-dimensional bin packing problems. Ann Oper Res 131:203–213

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

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

Moura A, Oliveira JF (2005) A GRASP approach to the container-loading problem. IEEE Intell Syst 20:50–57

Mumford-Valenzuela CL, Vick J, Wang PY (2003) Heuristics for large strip packing problems with guillotine patterns: an empirical study. In: Metaheuristics: computer-decision making. Kluwer Academic Publishers B.V., pp 510–522

Pisinger D (2002) Heuristics for the container loading problem. Eur J Oper Res 141:143–153

Pisinger D, Sigurd M (2007) Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. INFORMS J Comput 19:36–51

Terno J, Scheithauer G, Sommerweiß U, Riehme J (2000) An efficient approach for the multi-pallet loading problem. Eur J Oper Res 123:372–381

Wäscher G, Haußner H, Schumann H (2007) An improved typology of cutting and packing problems. Eur J Oper Res 183:1109–1130