Recent Advances in Multi-dimensional Packing Problems

5

Recent Advances in Multi-Dimensional Packing Problems

Teodor Gabriel Crainic[1,2], Guido Perboli[1,3] and Roberto Tadei[3]
[1]*CIRRELT*
[2]*UQAM*
[3]*DAUIN, Politecnico di Torino*
[1,2]*Canada*
[3]*Italy*

1. Introduction

Packing problems have been much studied in the past decades due, in particular, to their wide range of applications in many settings of theoretical and practical interest, including packing/loading, scheduling, and routing. We focus on Multi-Dimensional Packing problems, which present specific methodological challenges while also being of particular interest to transportation and modern supply chains, due to the need to consolidate and optimize flows of freight and vehicles. The rich literature presents a plethora of problem variants, models, and solution methods. Yet, a general overview and synthesis of the field is missing, as we lack a general methodological framework able to efficiently address different problem variants, i.e., obtain good-quality solutions with limited computational efforts. Addressing these issues is the main goal of this chapter.

All Multi-Dimensional Packing problems display an identical structure, defining two sets of elements in one or more (usually two or three) geometric dimensions: 1) a set of large items, often called *containers*, *bins*, or *knapsacks* and 2) a set of small items, usually referred to simply as *items*. The goal is to select all or some of the items, group them into one or more subsets, and assign each of the resulting subsets to one of the bins, such that the *geometric* conditions hold, i.e., the group of items of each subset fits completely within the corresponding bin with no overlapping. Problem variants differ by the particular definition of their *packing* constraints (presence of guillotine cuts, balancing and stability of the packing, possible overlapping of certain items, forbidden rotations of the items, etc.) and objective function, going by the well-known names of Knapsack, Bin Packing, Strip Packing, Variable Sized Bin Packing, Container Packing, to name just a few (see Wäscher et al., 2007 for a tentative taxonomy of multi-dimensional packing problems). We focus in this chapter on *orthogonal packings*, i.e., items and bins are rectangular in two-dimensions (2D) and boxes in three-dimensions (3D), and items must be placed into bins with their sides parallel to the sides of the respective bin.

The aim of this chapter is twofold. We first survey the different approaches used to represent the packing of items into bins. The issue is common to all problem variants, the packing representation playing a central role in the efficiency of the solution methods. Second, we discuss the different solution approaches proposed for the main Multi-Dimensional Packing

considered, i.e., the items are characterized by mass and center of mass, and the center of mass of the overall packing must lie inside a given domain.

2.2 Packing rules for Multi-Dimensional Packing

All Multi-Dimensional Packing models present, at different degrees, two main challenges: the large number of variables and the high level of solution degeneracy. In fact, these models can be used only for small-sized instances (20 items in single-bin 3D problems). Rules were therefore introduced to define where to place additional items into a bin already holding some. All these methods deal with two different issues: reduce the computational effort and the data structure complexity needed to use the rule, and the possibility to introduce additional packing constraints, like fixing item positions and introducing guillotine cuts.

An approach often used for 2D-packing building consists in combining procedures designed for mono-dimensional problems, namely *shelf* (or *layer*) methods (Berkey & Wang, 1987; Bortfeldt & Winter, 2009; Chung et al., 1982). The items are first sorted and packed into "shelves" with sizes equal to the width of the bin. The problem then reduces to solving a mono-dimensional packing instance. Indeed, a 2D packing can be obtained by placing the shelves into the bin according to the solution of a mono-dimensional packing problem, where the size of the items equals the depth of the shelves and the size of the mono-dimensional bin equals the depth D of the two-dimensional one. The same approach can also be used to build 3D packings. Build first two-dimensional shelves by using any 2D algorithm and, then, arrange them into a three-dimensional bin by solving a mono-dimensional packing problem, where the size of the items equals the height of the shelves and the size of the bin equals the height H. When the 2D shelves are also built according to the shelf approach, the method is known as *wall-building* (George & Robinson, 1980; Pisinger, 2002). The drawback of the shelf approach is that it introduces guillotine cuts on the depth and height of the two and three-dimensional bins, respectively, leading to their underutilization. Figure 1 illustrates 2D and 3D packings obtained by means of the shelf approach.

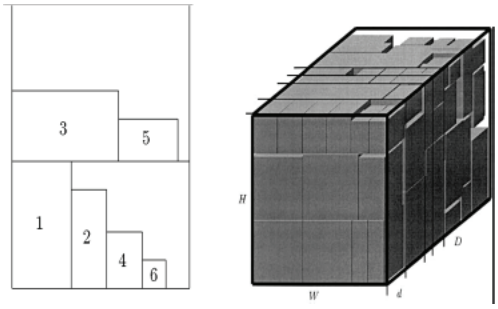

Fig. 1. Shelf Packings in 2D and 3D

A graph-theoretical approach for the characterization of Multi-Dimensional Packings was proposed by Fekete & Schepers (1997; 2004a). The authors considered the relative positions of the items in a feasible packing and defined a graph describing the item "overlapping" according to the projection of the items on each orthogonal axis. In this way, the authors were able to deal with classes of packings sharing a certain combinatorial structure, instead of having to consider one packing at a time. The packing classes are represented by a series of graphs, one for each axis. The graphs are proven to be interval graphs, i.e., a special and

well-studied class of graphs for which elegant and extremely efficient algorithms have been developed. More formally, let $G_d\,(V, E_d)$ be the graph associated to the d^{th} axis. Each vertex of $G_d\,(V, E_d)$ is associated to an item i in the bin and a non-oriented edge (i, j) between two items i and j exists if and only if their projections on axis d overlap (see Figure 2). The authors proved necessary conditions on the interval graphs to define a feasible packing. Combined to good heuristics for dismissing infeasible subsets of items, this characterization was used to develop a two-level tree search. According to computational results, mainly limited to 2D problems, this strategy outperformed previous methods. The method cannot handle additional constraints on the packing, however, such as fixing the position of one or more items. No direct comparison with the Branch & Bound of Martello et al. (2007) was performed. The link between guillotine cuts and interval graphs was analyzed by Perboli (2002). Recently, Joncour et al. (2010) introduced an efficient algorithm to manage the interval-graph structure by means of MPQ-trees, combinatorial structures introduced in Korte & Möhring (1989).

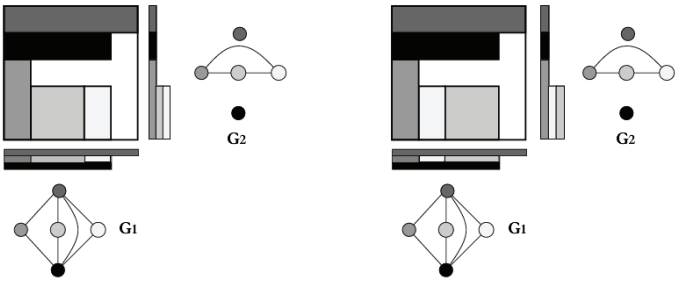

Fig. 2. Packings and Associated Interval Graphs

A similar approach to the one by Fekete & Schepers has been used by Imahori et al. (2003) to give a general representation for packing problems where the costs in the objective function depend on the location of the items. Instead of working on interval graphs, however, the authors directly deal with the sequencing of the items, i.e., which item is to be put before another into the packing. The decoding algorithm is managed by a dynamic programming method with pseudo-polynomial complexity and is used to derive Multi-Start and Iterated Local Search heuristics. As the representation addresses 2D packings, the computational experiments were limited to the Minimum Area Packing Problem, a variant of the 2D Container Loading problem where the minimal boxed envelope of the final packing must be considered (Murata et al., 1996).

Martello et al. (2000) defined *Corner Points* as the non-dominated locations where an item can be placed within an existing packing. In two dimensions, Corner Points are defined where the envelope of the items in the bin changes from vertical to horizontal (the large dots in Figure 3b). Corner Points on the three-dimensional envelope can be found applying the two-dimensional algorithm for each distinct value of the height of the bin defined by the lower and upper terminal lines of each item (large dots in Figure 3a). A Corner Point set can be computed in $O(n^2)$. Martello et al. (2000) used this idea to design a Branch & Bound algorithm to verify whether a given set of items can be packed into a bin or not. den Boef et al. (2005) showed that the algorithm to compute the Corner Points presented in Martello et al. (2000) may miss some feasible packings. Martello et al. (2007) addressed this issue by providing a new version of the procedure to compute the Corner Points, as well as an updated version of the related Branch & Bound algorithm.

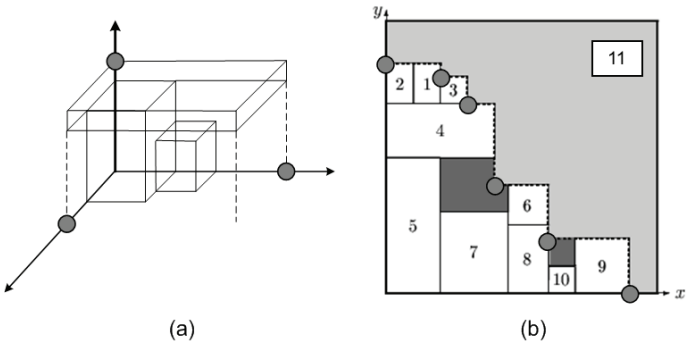

Fig. 3. Corner Points in 3D and 2D Packings

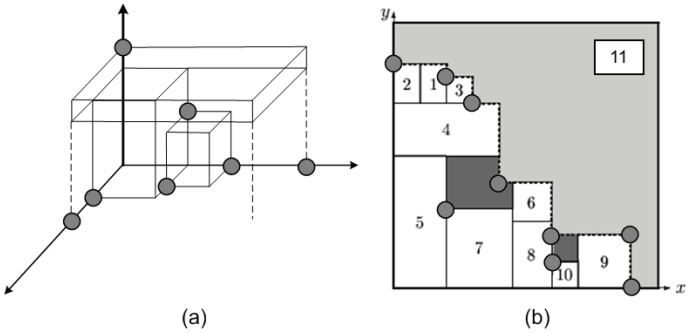

Fig. 4. Extreme Points in 3D and 2D Packings

Crainic et al. (2008) defined the Extreme Points, an extension of Corner Points providing a better exploitation of the bin volume by identifying additional points where an item can be added to an existing packing, as illustrated in Figure 4. In particular, while one cannot use Corner Points to add an item within the space left inside an existing packing, e.g., the dark gray regions in Figure 3b, Extreme Points provide this capability. Thus, for example, item 11 can be accommodated within the dark gray region on top of item 7 in Figure 4b, which is not possible with the Corner Points of Figure 3b. The Extreme Point idea was used by the authors to design new constructive heuristics based on the First Fit Decreasing and the Best Fit Decreasing heuristics for the mono-dimensional problem. Computational results showed that the proposed method outperformed all the other constructive heuristics for both 2D and 3D Bin Packing problems, and that it obtains, in negligible time, results comparable to those of the best existing meta-heuristics.

3. Multi-Dimensional Packing problems

We now turn to the main packing classes, 2D and 3D Bin Packing, 2D and 3D Knapsack, and 3D Container Loading. For each problem, we give its definition, the classification according to Wäscher et al. (2007), and a brief description of the state-of-the-art solution methods. Finally, a comparison of the computational results obtained with the state-of-the-art methods is presented. For this survey, we focus on meta-heuristic methods, as these methods are the most efficient way to solve real-sized instances while preserving sufficient accuracy.

The results of all the methods are taken from the literature. For the computational times, due to the variability of the different processors used in the computational tests, we adopted a Unified Computational Time (UCT) obtained considering a Pentium4 3000 MHz workstation as the reference machine and scaling the computational times according to the SPEC CPU2006 benchmarks published in SPEC (2006). Notice that, due to the limited amount of memory used by all the methods, this parameter does not effect the overall results.

3.1 Multi-dimensional bin packing problems

Given a set of box items $i \in I$, with sizes w_i, l_i, and h_i, and an unlimited number of bins of fixed sizes W, L, and H, the *Three-Dimensional Orthogonal Bin Packing* problem (*3D-BP*) consists in orthogonally packing the items into the minimum number of bins. We assume that the items cannot be rotated. According to the classification introduced by Wäscher et al. (2007), the problem is also known as the *Three-Dimensional Single Bin-Size Bin Packing Problem* (3D-SBSBPP). The *Two-Dimensional Orthogonal Bin Packing* problem (*2D-BP*) is the restriction of *3D-BP* in two dimensions.

TSPACK is the Tabu Search algorithm for the *2D-BP* developed by Lodi et al. (1999). This algorithm uses two simple constructive heuristics to pack items into bins. The Tabu Search only controls the movement of items between bins. Two neighborhoods are considered to try to relocate an item from the weakest bin (i.e., the bin that appears to be the easiest to empty) to another. Since the constructive heuristics produce guillotine packings, so does the overall algorithm. The algorithm is presently the best meta-heuristic for *2D-BP*, but it requires a computation effort of the order of 60 CPU seconds per instance to achieve these results.

The same authors presented a shelf-based heuristic for the *2D-BP*, called *Height first - Area second* (*HA*) (Lodi et al., 2004a). The algorithm chooses the best of two solutions. To obtain the first, items are partitioned into clusters according to their height and a series of layers are obtained from each cluster. The layers are then packed into bins by using the Branch-and-Bound approach by Martello & Toth (1990) for the *1D-BP* problem. The second solution is obtained by ordering the items by non-increasing area of their base and new layers are built. As previously, the layers are packed into bins by solving a *1D-BP* problem. The method is faster but less accurate than *TSPACK*.

The first exact method for the *3D-BP* was a two-level Branch-and-Bound algorithm proposed by Martello et al. (2000). The first level assigns items to bins. At each node of the first-level tree, a second level Branch-and-Bound is used to verify whether the items assigned to each bin can be packed into it. In the same paper, the authors introduced two constructive heuristics. The first, called *S-Pack*, is based on a layer-building principle derived from the shelf approach. The second, called *MPV-BS*, repeatedly fills one bin after the other by means of the Branch-and-Bound algorithm for the single bin presented by the authors in the same paper. The authors also gave the results of their method by limiting its computational effort to 1000 CPU seconds.

Faroe et al. (2003) presented a Guided Local Search (*GLS*) algorithm for the *3D-BP*. Starting with an upper bound on the number of bins obtained by a greedy heuristic, the algorithm iteratively decreases the number of bins, each time using *GLS* to search for a feasible packing. The process terminates when a given time limit has been reached or the upper bound matches a precomputed lower bound. Computational experiments were reported for 2 and

3-dimensional instances with up to 200 items. The results were satisfactory, but required a computational effort of the order of 1000 CPU seconds to be reached.

Crainic et al. (2008) defined the Extreme Points (*EPs*) and combined them to the well-known Best First Decreasing (*BFD*) heuristic for the *1D-BP*, producing the EP-BPH heuristic. Extending the EP-BPH to address the *3D-BP* proved far from trivial, however, as the ordering of items in higher dimensions may be affected by more than one attribute (e.g., volume, side area, width, length, and height of the items). Several sorting rules were tested and the best ones were combined into C-EPBFD, a composite heuristic based on EP-BPH. Extensive experimental results showed C-EPBFD requiring negligible computational efforts and outperforming both constructive heuristics for the *3D-BP* and more complex methods, e.g., the truncated Branch-and-Bound by Martello et al. (2000).

Crainic et al. (2009) proposed TS^2PACK, a two-level Tabu Search meta-heuristic for the *3D-BP*. The first level is a Tabu Search method that changes the assignment of items to bins. For each assignment, the items assigned to a bin are packed by means of the second-level Tabu Search, which makes use of the Interval Graph representation of the packing by Fekete & Schepers (2004a) to reduce the search space. The accuracy of the overall meta-heuristic is enhanced by the *k*-chain-move procedure, which increases the size of the neighborhoods without increasing the overall complexity of the algorithm. TS^2PACK currently obtains the best solutions for the *3D-BP*. Nevertheless, the method has a rather slow convergence rate, requiring 300 CPU seconds to find the best solution.

Finally, Perboli et al. (2011) introduced *GASP - Greedy Adaptive Search Procedure*, a meta-heuristic able to efficiently address two and three-dimensional multiple bin packing problems. GASP combines the simplicity of greedy algorithms with learning mechanisms aimed to guide the overall method towards good solutions. Extensive computational results showed that GASP is able to obtain state-of-the-art results for both *2D-BP* and *3D-BP* in negligible computational times.

We compare the different methods on standard benchmark instances, the results being taken from the literature.

We consider ten classes of instances from Berkey & Wang (1987) (Classes I-VI) and Martello & Vigo (1998) (Classes VII-X) for *2D-BP*. Each class is characterized by different distributions of the item sizes and considers a number of items equal to 20, 40, 60, 80, and 100. For each combination of class and instance size, 10 repetitions are considered.

We consider the instances of Martello et al. (2000) for the *3D-BP*. The instances are organized in six classes.

The bin size is $W = H = D = 100$ for Classes *I* to *III*, the items belonging to five types, ranging from small to large-sized. The five classes mix the item types in order to test different usage scenarios. Bin and item dimensions in Classes *IV* to *VI* vary according to the following rules:

- *Class IV*: $w_i, l_i, h_i \sim$ U[1,10] and $W = L = H = 10$;
- *Class V*: $w_i, l_i, h_i \sim$ U[1,35] and $W = L = H = 40$;
- *Class VI*: $w_i, l_i, h_i \sim$ U[1,100] and $W = L = H = 100$.

The number of items is fixed to 50, 100, 150, and 200 items for each class, and 10 instances are considered for each combination of class and cardinality of the item set.

The solution methods used for each problem variant and their experimental settings are:

- *2D-BP*
  - *TSPACK*: coded in C and run on a Silicon Graphics INDY R10000sc (195 MHz) with a time limit of 60 CPU seconds for each instance (Lodi et al., 1999);
  - *GASP*: coded in C++, runs were performed on a Pentium4 3 GHz workstation. The time limit was set to 3 seconds (Perboli et al., 2011).
- *3D-BP*
  - *GLS*: coded in C and run on a Digital workstation with a 500 MHz CPU. A time limit of 1000 CPU seconds was imposed for each instance (Faroe et al., 2003);
  - *MPV*: this is the truncated *Branch and Bound* proposed in Martello et al. (2000). It was coded in C and run on a Pentium4 with 3 GHz CPU with a time limit of 1000 CPU seconds per instance;
  - *TS*^{2}*PACK*: coded in C++ and run on a Pentium4 with 2 GHz CPU with a time limit of 300 CPU seconds per instance;
  - *GASP* coded in C++, runs were performed on a Pentium4 3 GHz workstation. The time limit was set to 5 seconds.

The results for *2D-BP* are summarized in Table 1. The instance type is given in the first column, while Columns 2, 3, and 4 present the results of *GASP*, *TSPACK*, and the best known solution taken from the literature (the optimal value in most cases), respectively. Notice that the best known solutions have been generally obtained by means of different exact methods and with a computational effort of several thousands of seconds. Finally, Columns 5 and 6 give the relative percentage gaps of *GASP* with respect to *TSPACK* and the best known solutions (a negative value means a better performance of *GASP*). All the time limits reported in the table are expressed in UCT.

GASP achieves better results than *TSPACK*, while reducing the computational effort by a factor of about 4. As the code of *TSPACK* is publicly available, we also run it for 30 UCT seconds but the results did not change. Moreover, *GASP* achieves results that are less than 1% from the overall optima.

Class	GASP 3 s	TSPACK 12 s	UB*	Gap TSPACK	Gap UB*
I	100.1	101.5	99.7	-1.40%	0.40%
II	12.9	13	12.4	-0.81%	4.03%
III	70.6	72.3	68.6	-2.48%	2.92%
IV	13	12.6	12.4	3.23%	4.84%
V	90.1	91.3	89.1	-1.35%	1.12%
VI	11.8	11.5	11.2	2.68%	5.36%
VII	83.1	84	82.7	-1.09%	0.48%
VIII	83.6	84.4	83	-0.96%	0.72%
IX	213	213.1	213	-0.05%	0.00%
X	51.4	51.8	50.4	-0.79%	1.98%
Total	729.6	735.5	722.5	-0.82%	0.98%

Table 1. *2D-BP* Comparative Results

The results of *3D-BP* are summarized in Table 2. Columns 1-3 give the instance type, bin dimension, and number of items, respectively, Column 4 presents the results of *GASP*, while

Columns 5-8 give the gaps of the solutions obtained by *GASP* relative to those of *MPV*, *GLS*, TS^2PACK, and the best lower bound available, respectively. The gaps were computed as $(mean_{GASP} - mean_o)/mean_o$, where, for a given set of instances, $mean_{GASP}$ and $mean_o$ are the mean values obtained by the *GASP* heuristic and the method compared to, respectively. A negative value means that *GASP* yields a better mean value. The last row displays the total number of bins used by GASP, computed as the sum of the values in the column, and the average of the mean gaps. As for the *2D-BP*, the time limits displayed are given in UCT.

The results indicate that *GASP* performs better than the truncated Branch & Bound and has a gap of only 0.9% with the best algorithm in the literature, with a negligible computational time: 5 CPU seconds compared to 1000 for *GLS* and 300 for TS^2PACK.

To further illustrate this efficiency, Table 3 displays the performance of *GASP* w.r.t. those of *GLS* and TS^2PACK, in comparable times (i.e., 60 CPU seconds on a Digital 500 workstation for *GLS*, equivalent to 18 UCT seconds, and 18 seconds for TS^2PACK). These results are impressive as *GASP* actually improves the solutions of both *GLS* and TS^2PACK up to 0.6% on average.

Class	Bins	n	GASP	MPV	GLS	TS2PACK	LB
			5 s	1000 s	300 s	300 s	
I	100	50	13.4	-1.47%	0.00%	0.00%	3.88%
		100	26.9	-1.47%	0.75%	0.75%	5.08%
		150	37	-3.14%	0.00%	0.00%	3.35%
		200	51.6	-1.34%	0.78%	0.98%	3.82%
II	100	50	29.4	0.00%	0.00%	0.00%	1.38%
		100	59	-0.17%	0.00%	0.17%	0.85%
		150	86.8	-0.46%	0.00%	0.00%	0.46%
		200	118.8	-0.59%	-0.17%	0.00%	0.42%
III	100	50	8.4	-8.70%	1.20%	1.20%	10.53%
		100	15.1	-13.71%	0.00%	-0.66%	7.86%
		150	20.6	-14.17%	1.98%	2.49%	9.57%
		200	27.7	-12.89%	1.84%	1.09%	6.54%
IV	10	50	9.9	1.02%	1.02%	1.02%	5.32%
		100	19.1	-1.55%	0.00%	0.00%	3.80%
		150	29.5	-0.34%	0.34%	1.03%	3.51%
		200	38	-0.52%	0.80%	0.80%	3.54%
V	40	50	7.5	-8.54%	1.35%	1.35%	10.29%
		100	12.7	-16.99%	3.25%	3.25%	10.43%
		150	16.6	-15.74%	5.06%	5.06%	15.28%
		200	24.2	-13.88%	2.98%	2.98%	6.61%
VI	100	50	9.3	-7.92%	1.09%	1.09%	6.90%
		100	19	-5.94%	0.53%	1.06%	3.26%
		150	24.8	-9.16%	3.77%	3.77%	10.22%
		200	31.1	-10.89%	4.01%	3.67%	10.28%
Total			736.4	-4.35%	0.85%	0.90%	3.89%

Table 2. *3D-BP* Comparative Results

Class	Bins	n	GASP 5 s	GLS 18 s	TS2PACK 18 s	LB
I	100	50	13.4	0.00%	0.00%	3.88%
		100	26.9	0.00%	-0.37%	5.08%
		150	37	-1.33%	-1.86%	3.35%
		200	51.6	-2.27%	-2.64%	3.82%
IV	100	50	29.4	0.00%	0.00%	1.38%
		100	59	0.00%	-0.34%	0.85%
		150	86.8	-0.34%	-0.57%	0.46%
		200	118.8	-0.92%	-0.34%	0.42%
V	100	50	8.4	1.20%	1.20%	10.53%
		100	15.1	0.00%	-1.95%	7.86%
		150	20.6	-0.48%	-1.44%	9.57%
		200	27.7	-0.36%	-1.07%	6.54%
VI	10	50	9.9	1.02%	0.00%	5.32%
		100	19.1	-1.04%	-2.05%	3.80%
		150	29.5	0.00%	0.34%	3.51%
		200	38	-1.30%	-1.81%	3.54%
VII	40	50	7.5	1.35%	1.35%	10.29%
		100	12.7	3.25%	3.25%	10.43%
		150	16.6	5.06%	3.75%	15.28%
		200	24.2	-0.82%	-2.42%	6.61%
VIII	100	50	9.3	1.09%	1.09%	6.90%
		100	19	0.53%	-1.04%	3.26%
		150	24.8	1.22%	0.81%	10.22%
		200	31.1	1.63%	0.97%	10.28%
Total			736.4	-0.23%	-0.57%	3.89%

Table 3. *3D-BP* Comparison of State-of-the-Art Methods in Comparable Computational Times

3.2 Multi-dimensional knapsack problems

Given a set of rectangular items $i = 1, \ldots, n$ with sizes w_i and l_i and a profit p_i, and a bin of fixed dimensions W and L, the *Two-Dimensional Orthogonal Knapsack* problem (*2D-KP*) consists in orthogonally packing a subset of the items into the bin to maximize the sum of the profits of the loaded items. Most algorithms present in the literature assume the items cannot be rotated.

The natural extension to the three-dimensional case is called *Three-Dimensional Orthogonal Knapsack* problem (*3D-KP*). The variant studied in the literature considers explicitly the rotation of the items.

The restriction of *2D-KP* to the case where the item profits are equal to the rectangle areas is also known in the literature as the *Cutting Stock* problem. According to the classification of Wäscher et al. (2007), *2D-KP* can be characterized as a Two-Dimensional Single Large Object Placement Problem (2D-SLOPP) and *3D-KP* as Three-Dimensional Single Large Object Placement Problem (3D-SLOPP).

Integer Programming formulations for the *2D-KP* were presented by Beasley (1985a), Hadjiconstantinou & Christofides (1995), and Boschetti et al. (2002), among others. Fekete & Schepers (1997; 2004b) addressed 2D and 3D Knapsack problems using an advanced graph representation within a Branch-and-Bound algorithm, which assigned items to the bin without specifying their position. Pisinger & Sigurd (2007) proposed a Branch-and-Cut approach for the *2D-KP*, in which a one-dimensional knapsack problem selects the most profitable items whose overall area does not exceed the area of the bin. A two-dimensional packing problem in decision form is then solved through constraint programming to check the feasibility of loading the selected items. Caprara & Monaci (2004) also proposed a Branch-and-Bound algorithm for the *2D-KP*, where items are assigned to the bin without specifying their positions, the feasibility check being performed afterwords through an enumeration scheme.

Many authors presented heuristic procedures for the *2D-KP*. Lai & Chan (1997a;b) developed two meta-heuristics based on Simulated Annealing and evolutionary principles. The former proceeds in three steps: splitting the master surface into sub-areas that can be used for packing, placing the items according to a fitting heuristic, and finally a classical search procedure based on moving the items and a cooling scheme. The evolutionary strategy includes hill-climbing and mutation procedures. Both meta-heuristics were tested on randomly generated instances as well as on real world problems with the objective of minimizing the wasted material. Leung et al. (2001) proposed a Genetic Search approach and a Simulated Annealing meta-heuristic. The authors hybridized the Genetic Search with a simple on-line bottom-left heuristic that packed the items as down and as left as possible. An extensive study of different crossover operators is presented but no detailed computational results are given. The meta-heuristics proposed by Lai & Chan (1997a) and Leung et al. (2001) cannot produce all the feasible cutting patterns. Liu & Teng (1999) used a different heuristic, denoted the *improved BL-algorithm*, to overcome this issue by, first, placing the first item in the bottom left-hand corner of the master surface, and then inserting all the other items starting from the top right-hand corner of the surface and then shifting them alternatively left and down until no further shifting is possible. Unfortunately, no computational results are given for problem instances drawn from the literature, making impossible a direct comparison with other heuristics.

Beasley (2004) proposed an innovative population-based meta-heuristic for a new nonlinear formulations of the problem. Boolean variables were used to indicate whether an item is cut from the master surface or not, two other integer variables representing the coordinates of the center of the item cut. This formulation leads to a three-dimensional solution encoding used to create the individuals of the population, which was evolved through crossover and mutation. Infeasible solutions were penalized during the fitness-evaluation step. Computational results were presented for a number of standard problems from the literature, as well as for a number of large randomly generated problems.

These results were improved by Hadjiconstantinou & Iori (2007), who proposed a Genetic Search meta-heuristic hybridized with a greedy procedure, where the items can be placed on the point maximizing the so-called touching perimeter, i.e., the fraction of the perimeter of an item to be added to the bin touching either the edges of the existing packing, or the edges of the surface of the bin. Computational results showed that this method outperformed the algorithm by Beasley (2004) from both the computational and the solution-quality points of view.

In the same year, Alvarez-Valdes et al. (2007) introduced another meta-heuristic that improved the results of Beasley (2004). The algorithm is a Tabu Search meta-heuristic implementing interesting moves able to compact the packing, thus reducing the wasted space. Moreover, the algorithm is able to deal with additional constraints, e.g., the presence of different types of items and lower bounds on the number of items to be loaded for each type.

The best results for *2D-KP* are due to Bortfeldt & Winter (2009) who proposed a Genetic Algorithm for guillotine packings. The algorithm is also able to deal with item rotations and was tested on a wide series of instances with and without guillotine cuts. Their results improved both Hadjiconstantinou & Iori (2007) and Alvarez-Valdes et al. (2007), but with a significant computational effort.

The first contribution to 3D Knapsack problems is to be found in Egeblad & Pisinger (2009), where the authors proposed an exact model and heuristics for 2D and 3D Knapsack problems. The model cannot be used to derive both lower and upper bounds, however, whilst the heuristic manages instances up to 60 items for the 3D case.

Finally, Perboli (2011) extended the GASP algorithm (Perboli et al., 2011) for the Multi-Dimensional Bin Packing problem. The algorithm shares the same structure and packing representation by means of *EP*s with the original method. Compared to the version for Multi-Dimensional Bin Packing Problem, this variant of the algorithm incorporates a long-term memory mechanism, which adapts the search according to the number of times an item is loaded in a solution considered during the search. The method achieves state-of-the-art results for both *2D-KP* and *3D-KP*, with or without rotation, within negligible computational times.

We used two set on instances to test *2D-KP*. A set of 38 small-size instances, with less than 100 items and known optimal solutions (except gcut13), was built as follows:

- Twelve instances (ngcut01 to ngcut12) from Beasley (1985b), available from the ORLIB-Library (Beasley, 1990);
- Four instances (hccut03, hccut08, hccut11, hccut12) from Hadjiconstantinou & Christofides (1995);
- Five instances (okp01 to okp05) from Fekete & Schepers (2004a);
- Three instances (cgcut01 to cgcut03) from Christofides & Whitlock (1977);
- Thirteen instances (gcut01 to gcut13) from Beasley (1985a), available from the ORLIB-Library (Beasley, 1990);
- One instance (wang20) from Wang (1983).

A second set contains large-size instances with up to 4000 items from the sets ngcutfs01, ngcutfs02, and ngcutfs03, randomly generated by Beasley (2004) similarly to the procedure of Fekete & Schepers (2004a). All these instances are available from the ORLIB-Library Beasley (1990). The large-size test set works with a bin of size $[100, 100]$ and is composed of instances of Type I, II, and III, according to the criteria used for the random generation of items. For each of the three types, $m = 7$ item subtypes and $Q = 3$ items for each item subtype were considered and 10 random instances were built for each combination of m and Q. The complete set is thus made up of 630 instances with up to 4000 items. Both sets refer to the *2D-KP* problem with fixed rotation, i.e., items cannot be rotated.

For the *3D-KP*, we consider the instances presented in http://www.diku.dk/hjemmesider/ansatte/pisinger/codes.html, together with the corresponding results.

The instances were obtained as follows:

- *number of items*: $n \in \{20, 40, 60\}$
- *item generation strategy*: $t \in \{C, R\}$, where:
  - C alias *clustered*, because the instance consists of only 20 items, which are duplicated appropriately;
  - R alias *random*, because the instance consists of independently generated items
- *bin size*: $p \in \{50, 90\}$, expressed as a percentage of the total volume of the items
- *item attributes*:
  - *size*: $s_i = (w_i, d_i, h_i)$, which must belong to one among the following *geometric classes* (see Egeblad & Pisinger (2009)):
    - Cubes (C). The items are cubic and their sizes are defined as $w_i \in [1, 100]$, $d_i = w_i$, $h_i = w_i$;
    - Diverse (D). The sizes of the items are randomly chosen in the following ranges $w_i \in [1, 50]$, $d_i \in [1, 50]$, $h_i \in [1, 50]$;
    - Long (L). The sizes of the items are randomly chosen in the following ranges $w_i \in [1, 200/3]$, $d_i \in [50, 100]$, $h_i \in [1, 200/3]$;
    - Uniform (U). The sizes of the items are randomly chosen in the following ranges $w_i \in [50, 100]$, $d_i \in [50, 100]$, $h_i \in [50, 100]$;
  - *profit*: $p_i = 200 + w_i\, d_i\, h_i$.

The combination of all the values gives 60 instances for each set. In this case, the items can be rotated.

The different algorithms compared and the test environments are:

- *2D-KP*
  - H_B: heuristic by Beasley (2004), coded in FORTRAN and run on a Silicon Graphics O2 workstation with a R10000 225 MHz processor;
  - H_{HI}: heuristic by Hadjiconstantinou & Iori (2007), coded in FORTRAN and run on a Pentium IV 1700 MHz;
  - H_{AP}: heuristic by Alvarez-Valdes et al. (2007), coded in C++ and run on a Pentium III at 800 MHz;
  - H_{BW}: heuristic by Bortfeldt & Winter (2009), coded in C and run on an Intel PC 3 GHz, Dual Core;
  - GASP coded in C++ and run on a Pentium4 3000 MHz workstation
- *3D-KP*
  - H_{EP}: heuristic by Egeblad & Pisinger (2009), coded in C++ and run on an AMD Athlon 64 3800+ processor;
  - GASP coded in C++ and run on a Pentium4 3 GHz workstation.

We report only aggregated results for *2D-KP* on the small-size instances of the first set. The best results of H_B present an average gap of 1.24% from the optimal solutions, while H_{HI} solves all the instances to the optimum, except gcut13 (for which no optimal solution is known in the literature) and gcut02. H_{AP} finds all the known optima (and the best known value for gcut13), while H_{BW} fails only in instance wang20. GASP solves to optimality all instances for which the optimum is available and determines the best known value for gcut13 with a mean computational effort less than 5 seconds.

	H_B	H_{HI}	H_{AP}	H_{BW}	GASP (best)	GASP (average)
Type I	1.64	1.24	0.95	1.03	0.98	1.01
	558.00	46.41	10.00	3600.00	4.15	4.82
Type II	1.70	1.32	1.06	1.09	1.06	1.07
	668.00	48.18	12.16	3600.00	3.77	3.68
Type III	1.66	1.35	0.94	0.95	0.94	1.01
	685.00	63.80	16.61	3600.00	5.39	6.25
Average	1.67	1.30	0.98	1.02	0.99	1.03
	637.00	52.79	12.92	3600.00	4.44	4.92

Table 4. *2D-KP* Comparative Results

A more challenging comparison can be made considering the large-size instances. The results are summarized in Table 4, which displays the mean values of the results obtained by the three heuristics by instance type. Columns 1 to 4 report the results of H_B, H_{HI}, H_{AV}, and H_{BW}, respectively, while the remaining columns display the the best and average solution values obtained by GASP over the 10 repetitions. We report the percentage gaps from the best known results and the Unified Computational Times. Notice that for the computational times of H_{BW}, the authors provided only the time limit and not the computational times they needed to reach their best results.

According to these results, the best heuristic is still H_{AP} with a mean optimality gap of 0.98%. H_B is no longer competitive, while H_{BW} has competitive results, but with a heavy computational effort. We notice that H_{BW} is still the best heuristic for the variant of the problem where only guillotine cuts are considered. GASP performs practically as well as H_{AP} with a much smaller computational effort, actually reducing computing times by up to two orders of magnitude compared to the other methods. GASP also shows high performance stability when the random seeds are varied.

The results of the comparison for *3D-KP* are presented in Table 5 aggregated by instance type. We display the mean percentage gaps with respect to the upper bound obtained by solving the mono-dimensional knapsack problem with the items of the *3D-KP* instance with item weights equal the their respective volumes and a knapsack maximum weight equal to the volume of the 3D knapsack. This bound is known to be quite poor in quality, but it is the current best for 3D knapsack where items can be rotated. As for *2D-KP*, two results are provided for GASP: the best and mean over 10 runs, respectively. Computational times were fixed to 120 seconds for H_{EP} and 10 seconds for GASP.

These results indicate that GASP performs better than H_{EP}, both in quality and computational efficiency, even when the mean GASP value over 10 random runs is considered. The small gap between best and average results also shows that GASP is stable with respect to random repetitions.

3.3 Container loading problem

Given a set of box items $i = 1, ..., n$ with sizes w_i, l_i, and h_i and a container of fixed dimensions W, L, and H, the *Three-Dimensional Container Loading* problem (*3D-CLP*) consists in orthogonally packing a subset of the items into the container maximizing the used fraction of

n	H_{EP}	GASP (best)	GASP (average)
20	19.7	15.2	15.3
40	19.3	18.0	18.3
60	15.7	14.8	15.1
Total	18.2	16	16.2

Table 5. *3D-KP* Comparative Results

its volume. Usually, items can be rotated, but some restrictions on the feasible rotations can be present. According to Wäscher et al. (2007), the *3D-CLP* can be classified as Three-Dimensional Rectangular Single Large Object Placement problem (3D-SLOPP).

The problem arises in important practical cases, particularly in logistics and distribution where containers, trucks, rail cars, etc. must be loaded with freight. It can be seen as a special case of the Three-Dimensional Knapsack Problem, where the profits of the items are their volumes. Yet, due to the large number of items that can be loaded into a container and to the fact that the item profit is linked to the item size, the methods developed for the Multi-Dimensional Knapsack Problem fail in most cases of interest. Consequently, the *3D-CLP* has been studied as a separate problem in the literature.

The first heuristic for the *3D-CLP* was proposed by George & Robinson (1980). The authors developed a wall-building procedure, which was later improved by several authors, e.g., Bischoff & Marriot (1990) and, more recently, Moura & Oliveira (2005). A different approach, based on column generation, was proposed by Gehring & Bortfeldt (1997), which provided the starting point of a series of meta-heuristics by the same authors: Tabu Search (Bortfeldt & Gehring, 1998), a hybrid algorithm (Bortfeldt & Gehring, 2001), as well as their best algorithm, a parallel hybrid local search combining Simulated Annealing and Tabu Search (Mack et al., 2004). Parreño et al. (2008) presented a reactive GRASP, which uses a constructive-block heuristic similar, for its usage of the space, to the Residual-Space idea of Crainic et al. (2008). The authors improved their GRASP in Parreño et al. (2010). The method implemented the same constructive heuristic presented in Parreño et al. (2008), but introduced five types of neighborhoods, mixed in a VNS-based meta-heuristic, which yielded the best results in the literature. Pisinger (2002) also presented a wall-building approach yielding interesting results, but tested on a different set of instances than the other contributions, making a direct comparisons difficult. Finally, because *3D-CLP* is a special case of *3D-KP*, any available code for *3D-KP* with rotations can be used as well. Consequently, we also consider the GASP version developed for Multi-Dimensional Knapsack problems presented in Perboli (2011).

The experiments were performed using the standard benchmark instances for the *3D-CLP* generated by Bischoff & Ratcliff (1995). The whole set of instances is made up of 14 classes, namely BR1 to BR14, of 100 instances each, but only sets BR1 to BR7 were tested on all the algorithms presented previously. The number of box types increases from 3 in BR1 to 20 in BR7, thus covering a wide range of situations. The number of items of each type decreases from an average of 50.2 items per type in BR1 to 5.60 in BR7. For each item type, the maximum number of items available is known. The total volume of the items is on average 99.5% of the capacity of the container.

We compare the results of RG_{PAO}, the Reactive GRASP by Parreño et al. (2008), TS_{BG}, the Tabu Search by Bortfeldt & Gehring (1998), and H_{MO}, the GRASP approach by Moura & Oliveira (2005) truncated after 5000 and 50000 iterations (indicated as 5k and 50k, respectively).

Instance set	TS_{BG}	H_{MO}	RG_{PAO} 5k	RG_{PAO} 50k	VNS_{PAO}	GASP
BR1	93.23	93.26	93.27	93.66	94.93	93.32
BR2	93.27	93.56	93.38	93.90	95.16	93.78
BR3	92.86	93.71	93.39	94.00	94.99	93.65
BR4	92.40	93.30	93.16	93.80	94.71	93.70
BR5	91.61	92.78	92.89	93.49	94.33	92.00
BR6	90.86	92.20	92.62	93.22	94.04	92.05
BR7	89.65	91.20	91.86	92.94	93.53	93.42
Mean volume	91.98	92.86	92.94	93.57	94.53	93.13
Mean time	38	205	8	77	28	10
Mean UCT	9	126	10	99	36	10

Table 6. *3D-CLP* Comparative Results

Algorithm TS_{BG} run on a Pentium II 400 MHz, using a mean computation effort of 316 seconds, H_{MO} on a Pentium IV at 2.4 GHz, with an average time of 69 seconds, while RG_{PAO} run on a 1.5 GHz Pentium mobile with a mean effort of 7.83 seconds. GASP was coded in C++ and run on a Pentium4 3000 MHz workstation. For each instance, 10 repetitions were performed changing the seed of the random generator, with a time limit of 10 seconds.

Comparative performance results are reported in Table 6 as average values of the solutions obtained on the 100 instances. Column 1 displays the instance set, while Columns 2 to 4 display the results of TS_{BG}, H_{MO}, RG_{PAO}, VNS_{PAO}, and GASP, respectively. The last three rows display the mean used container volume (in %), the mean computational time as reported in the literature, and the mean Unified Computational Time (no ratio is available for the computer used by TS_{BG}). The results indicate that, without any particular adaptation, GASP compares advantageously in solution quality with previous state-of-the-art algorithms for the *3D-CLP* problem, while significantly reducing the computational effort. In particular, GASP is on average more effective than H_{MO} and RG_{PAO}, requiring some 10 times less computational effort than RG_{PAO}, which leads in solution quality by a very narrow margin.

4. General remarks

Many solution methods were proposed for Multi-Dimensional Packing problems, the two methodological frameworks emerging as the most used being Tabu Search and Genetic Algorithms. The latter needs usually an hybridization with specific optimization procedures managing the peculiarities of each problem setting.

Most methods in the literature generally aim for better solution quality without much care to how general or flexible it is. To evaluate this aspect, we adopt the two additional performance criteria stated by Cordeau et al. (2002) for evaluating Vehicle Routing heuristics, namely *simplicity* and *flexibility*. The former relates to ease of understanding and coding of an algorithm, while the latter focuses on the possibility to easily introduce additional constraints.

From the simplicity point of view, *TSPACK* (Lodi et al., 2004b) and GASP (Perboli et al., 2011) stand out due to their simple structures and ease of implementation. Notice though, that, similarly to many packing meta-heuristics, *TSPACK* mixes in the neighborhood structure the issues of packing feasibility, which follows from the packing representation, and its optimality, which relates to the particular problem settings. This greatly reduces the generality of the method. This contrasts with the modular structure of GASP, which let the authors to

successfully address different packing problems (Bin Packing, Knapsack, Container Loading, etc.). On the other hand, methods such as TS^2PACK (Crainic et al., 2009) present an interesting structure, but the quite complicated packing representation given by the interval graphs make them harder to understand and manage.

Evaluating flexibility, we see methods, e.g., $TSPACK$ for the Multi-Dimensional Bin Packing, and H_{BW} (Bortfeldt & Winter, 2009) and H_{EP} (Egeblad & Pisinger, 2009) for the Multi-Dimensional Knapsack Problem, which have been successfully tested on different variants of the same problem, showing a good flexibility at this level. On the other hand, up to now, GASP is the only method that has been satisfactory tested on different packing problem classes.

Turning to the packing representation, the most elegant approach is the interval graph representation by Fekete & Schepers (1997). Unfortunately, it is also the less flexible when one has to deal with additional constraints like items in fixed positions and guillotine cuts (Perboli, 2002). Moreover, its performances strongly depends on the data structures used to update the representation. Corner Points by Martello et al. (2000) are probably the easiest to understand and implement. The Extreme Points by Crainic et al. (2008) offer a better exploitation of the bin volume and represent a good compromise between simplicity and accuracy.

The last point to focus on is the public availability of these methods. For most of them, the code is not public. To our best knowledge, the only methods that are publicly available are $TSPACK$, the Branch & Bound by Martello et al. (2000) and the heuristic for container loading by Pisinger (2002). All these codes can be downloaded from the web sites of the authors.

5. Conclusions

In this chapter we presented a detailed and up-to-date survey of solution methods for Multi-Dimensional Packing problems. We first focused on the common issue of packing problems, i.e., the representation of the packing. We then considered the main Multi-Dimensional Packing problems and discussed the efficiency and accuracy of the available solution methods.

We identified for each problem setting the methods that perform best. We also observed that most methods are tailored for one problem setting only. The only method that emerges as a general framework is GASP, which has been successfully applied to different variants of the problems presented in this chapter.

6. Acknowledgments

While working on this project, T.G. Crainic was the NSERC Industrial Research Chair on Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway. Partial funding was provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs.

This project has been partially supported by the Ministero dell'Istruzione, Università e Ricerca (MIUR) (Italian Ministry of University and Research), under the Progetto di Ricerca di Interesse Nazionale (PRIN - Research Project of National Interest), 2009: "Models and algorithms for the Optimization in Logistics".

7. References

Alvarez-Valdes, R., Parreno, F. & Tamarit, J.-M. (2007). A tabu search algorithm for a two-dimensional non-guillotine cutting problem, *European Journal of Operational Research* 183: 1167–1182.

Baldacci, R. & Boschetti, M. A. (2007). A cutting plane approach for the two-dimensional orthogonal non guillotine cutting stock problem, *European Journal of Operational Research* 183: 1136–1149.

Baldi, M., Perboli, G. & Tadei, R. (2011). The three-dimensional knapsack problem with balancing constraints, *Publication CIRRELT-2011-51*, Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport, Université de Montréal, Montréal, Canada.

Beasley, J. E. (1990). Or-library: distributing test problems by electronic mail, *Journal of the Operational Research Society* 41: 1069–1072.

Beasley, J. E. (1985a). Algorithms for two-dimensional unconstrained guillotine cutting, *Journal of the Operational Research Society* 36: 297–306.

Beasley, J. E. (1985b). An exact two-dimensional non-guillotine cutting stock tree search procedure, *Operations Research* 33: 49–64.

Beasley, J. E. (2004). A population heuristic for constrained two-dimensional non-guillotine cutting, *European Journal of Operational Research* 156: 601–627.

Berkey, J. O. & Wang, P. Y. (1987). Two dimensional finite bin packing algorithms, *Journal of the Operational Research Society* 38: 423–429.

Bischoff, E. E. & Marriot, M. D. (1990). A comparative evaluation of heuristics for container loading, *European Journal of Operational Research* 44: 267–276.

Bischoff, E. E. & Ratcliff, M. S. W. (1995). Issues in the development of approaches to container loading, *Omega* 23: 377–390.

Bortfeldt, A. & Gehring, H. (1998). A tabu search algorithm for weakly heterogeneous container loading problems, *OR Spectrum* 20: 237–250.

Bortfeldt, A. & Gehring, H. (2001). A hybrid genetic algorithm for the container loading problem, *European Journal of Operational Research* 131: 143–161.

Bortfeldt, A. & Winter, T. (2009). A genetic algorithm for the two-dimensional knapsack problem with rectangular pieces, *International Transactions in Operational Research* 16: 685–713.

Boschetti, M. A., Hadjiconstantinou, E. & Mingozzi, A. (2002). New upper bounds for the twodimensional orthogonal cutting stock problem, *IMA Journal of Management Mathematics* 13: 95–119.

Caprara, A. & Monaci, M. (2004). On the 2-dimensional knapsack problem, *Operations Research Letters* 32: 5–14.

Christofides, N. & Whitlock, C. (1977). An algorithm for two-dimensional cutting problems, *Operations Research* 25: 30–44.

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

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y. & Semet, F. (2002). A guide to vehicle routing heuristics, *Journal of the Operational Research Society* 53(5): 512–522.

Crainic, T., Perboli, G. & Tadei, R. (2008). Extreme point-based heuristics for three-dimensional bin packing, *INFORMS Journal on Computing* 20: 368–384.

Crainic, T., Perboli, G. & Tadei, R. (2009). TS2PACK: A two-level tabu search for the three-dimensional bin packing problem, *European Journal of Operational Research* 195: 744–760.

den Boef, E., Korst, J., Martello, S., Pisinger, D. & Vigo, D. (2005). Erratum to "the three-dimensional bin packing problem": Robot-packable and orthogonal variants of packing problems, *Operations Research* 53(4): 735–736.

Egeblad, J. & Pisinger, D. (2009). Heuristic approaches for the two- and three-dimensional knapsack packing problem, *Computers & Operations Research* 36: 1026–1049.

Faroe, O., Pisinger, D. & Zachariasen, M. (2003). Guided local search for the three-dimensional bin packing problem, *INFORMS Journal on Computing* 15(3): 267–283.

Fekete, S. P. & Schepers, J. (1997). A new exact algorithm for general orthogonal d-dimensional knapsack problems, *ESA '97, Springer Lecture Notes in Computer Science* 1284: 144–156.

Fekete, S. P. & Schepers, J. (2004a). A combinatorial characterization of higher-dimensional orthogonal packing, *Mathematics of Operations Research* 29(2): 353–368.

Fekete, S. P. & Schepers, J. (2004b). A general framework for bounds for higher-dimensional orthogonal packing problems, *Mathematical Methods of Operations Research* 60(2): 311–329.

Gehring, H. & Bortfeldt, A. (1997). A genetic algorithm for solving the container loading problem, *International Transactions in Operational Research* 4: 401–418.

George, J. A. & Robinson, D. F. (1980). A heuristic for packing boxes into a container, *Computers & Operations Research* 7(3): 147–156.

Gilmore, P. C. & Gomory, R. E. (1965). Multistage cutting problems of two and more dimensions, *Operations Research* 13: 94–119.

Hadjiconstantinou, E. & Christofides, N. (1995). An exact algorithm for general, orthogonal, two-dimensional knapsack problems, *European Journal of Operational Research* 83(1): 39–56.

Hadjiconstantinou, E. & Iori, M. (2007). A hybrid genetic algorithm for the two-dimensional single large object placement problem, *European Journal of Operational Research* 183: 1150–1166.

Imahori, S., M.Yagiura & Ibaraki, T. (2003). Local search algorithms for the rectangle packing problem with general spatial costs, *Mathematical Programming, Series B* 97: 543–569.

Joncour, C., Pêcher, A. & Valicov, P. (2010). MPQ-trees for orthogonal packing problem, *Electronic Notes on Discrete Mathematics* 36: 423–429.

Korte, N. & Möhring, R. (1989). An incremental linear-time algorithm for recognizing interval graphs, *SIAM J. Comput.* 18: 68–81.

Lai, K. K. & Chan, J. W. M. (1997a). Developing a simulated annealing algorithm for the cutting stock problem, *Computers & Industrial Engineering* 32: 115–127.

Lai, K. K. & Chan, J. W. M. (1997b). An evolutionary algorithm for the rectangular cutting stock problem, *International Journal of Industrial Engineering* 4: 130–139.

Leung, T. W., Yung, C. H. & Troutt, M. D. (2001). Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem, *Computers & Industrial Engineering* 40: 201–214.

Liu, D. & Teng, H. (1999). An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles, *European Journal of Operational Research* 112: 413–420.

Lodi, A., Martello, S. & Monaci, M. (2002). Two-dimensional packing problems: a survey, *European Journal of Operational Research* 141: 241–252.

Lodi, A., Martello, S. & Vigo, D. (1999). Heuristic and metaheuristic approaches for a class of two–dimensional bin packing problems, *INFORMS Journal on Computing* 11: 345–357.

Lodi, A., Martello, S. & Vigo, D. (2004a). Models and bounds for two–dimensional level packing problems, *Journal of Combinatorial Optimization* 8: 363–379.

Lodi, A., Martello, S. & Vigo, D. (2004b). Tspack: A unified tabu search code for multi-dimensional bin packing problems, *Annals of Operations Research* 131: 203–213.

Mack, D., Bortfeldt, A. & Gehring, H. (2004). A parallel hybrid local search algorithm for the container loading problem, *International Transactions in Operational Research* 11: 511–533.

Martello, S., Pisinger, D. & Vigo, D. (2000). The three-dimensional bin packing problem, *Operations Research* 48(2): 256–267.

Martello, S., Pisinger, D., Vigo, D., den Boef, E. & Korst, J. (2007). Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem, *ACM Transactions on Mathematical Software* 33, Article No 7: 1–12.

Martello, S. & Toth, P. (1990). *Knapsack Problems - Algorithms and computer implementations*, John Wiley & Sons, Chichester, UK.

Martello, S. & Vigo, D. (1998). Exact solution of the finite two dimensional bin packing problem, *Management Science* 44(44): 388–399.

Moura, A. & Oliveira, J. F. (2005). A grasp approach to the container loading problem, *IEEE Intelligent Systems* 20: 50–57.

Murata, H., Fujiyoshi, K., Nakatake, S. & Kajitani, Y. (1996). Vlsi module placement based on rectangle-packing by the sequence-pair, *IEEE Trans. Comput. Aided Des.* 15: 1518–1524.

Parreño, F., Alvarez-Valdes, R. & Oliveira, J. F. (2008). A maximal-space algorithm for the container loading problem, *INFORMS Journal on Computing* 20(3): 412–422.

Parreño, F., Alvarez-Valdes, R. & Oliveira, J. F. (2010). Neighborhood structures for the container loading problem: a VNS implementation, *Journal of Heuristics* 16: 1–22.

Perboli, G. (2002). *Bounds and heuristics for the Packing Problems*, PhD thesis, Politecnico di Torino, Turin, Italy.

Perboli, G. (2011). An efficient metaheuristic for multi-dimensional knapsack problems, *DAUIN Tech. Rep.*, Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy.

Perboli, G., Crainic, T. G. & Tadei, R. (2011). An efficient metaheuristic for multi-dimensional multi-container packing, *Proceedings of seventh annual IEEE Conference on Automation Science and Engineering (IEEE CASE 2011)*, pp. 1–6.

Pisinger, D. (2002). Heuristics for the container loading problem, *European Journal of Operational Research* 141: 382–392.

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

SPEC (2006). SPEC CPU2006 benchmarks. http://www.spec.org/cpu2006/results/. URL: *http://www.spec.org/cpu2006/results/*

Wang, P. Y. (1983). Two algorithms for constrained two-dimensional cutting stock problems, *Operations Research* 31: 573–586.

Wäscher, G., Haussner, H. & Schumann, H. (2007). An improved typology of cutting and packing problems, *European Journal of Operational Research* 183: 1109–1130.