Volume 26 (1), pp. 21–44
http://www.orssa.org.za

ORiON
ISSN 0529-191-X

Modified strip packing heuristics for the rectangular variable-sized bin packing problem

Frank G Ortmann* Jan H van Vuuren†

Received: 25 February 2010; Accepted: 19 April 2010

Abstract

Two packing problems are considered in this paper, namely the well-known *strip packing problem* (SPP) and the *variable-sized bin packing problem* (VSBPP). A total of 252 strip packing heuristics (and variations thereof) from the literature, as well as novel heuristics proposed by the authors, are compared statistically by means of 1 170 SPP benchmark instances in order to identify the best heuristics in various classes. A combination of new heuristics with a new sorting method yields the best results. These heuristics are combined with a previous heuristic for the VSBPP by the authors to find good feasible solutions to 1 357 VSBPP benchmark instances. This is the largest statistical comparison of algorithms for the SPP and the VSBPP to the best knowledge of the authors.

Key words: Packing problems, heuristics.

1 Introduction

While *cutting and packing* (C&P) problems have been studied for many years, *e.g.* the packing of animals, seafaring vessel, trains and vehicles, these problems have only become an active field of mathematical study since the 1939 landmark paper by Kantorovich [47] and papers by other early researchers in the mid-twentieth century, including those of Eisemann [25] in 1957 and Gilmore and Gomory [32–34] in the 1960s. In the early C&P literature, cutting problems were the most common type of C&P problems studied (Hixman [39] provides a detailed survey of early cutting stock problems). However, Dyckhoff [24, pp. 148–149] identified a strong relationship between cutting problems and packing problems due to the duality of solid objects and the space that they occupy. Cutting problems are typically characterised by the cutting of small items from large objects, while packing problems may be characterised by the packing of small items into large objects. Therefore, the packing of items into a bin may be considered as "cutting" away the empty space inside the bin, where the unused space is "trim loss." The literature on

*Corresponding author: Department of Logistics, University of Stellenbosch, Private Bag X1, Matieland, 7602, South Africa, email: frank.ortmann@gmail.com

†**(Fellow of the Operations Research Society of South Africa)**, Department of Logistics, University of Stellenbosch, Private Bag X1, Matieland, 7602, South Africa.

packing problems is vast and presented in some detail in papers on C&P typologies, such as those by Dyckhoff [24] and Wäscher *et al.* [69], and surveys such as those by Sweeney and Paternoster [64], Coffman *et al.* [17], and Lodi *et al.* [50, 53].

The aim in the so-called *strip packing problem* (SPP) is to pack small items into a bin of fixed width and infinite height such that the resulting packing height is a minimum. This problem has been studied extensively, with authors such as Coffman *et al.* [18], Berkey and Wang [7, p. 425], and Martello *et al.* [55] having proposed level-packing[1] heuristics for the SPP. Pseudolevel-packing heuristics for the same problem have been proposed by Lodi *et al.* [51,52], Bortfeldt [8], Ntene and Van Vuuren [57] and Ortmann *et al.* [59], while Sleator [63], Coffman *et al.* [18], Baker *et al.* [1,2], Golan [35], Chazelle [13] and Burke *et al.* [11, 12] have proposed plane-packing heuristics for the SPP.

The aim in the *variable-sized bin packing problem* (VSBPP) is to pack a set of items into a subset of bins (which may, or may not, all be of the same dimensions) in such a manner that the resulting total area of bins containing items is a minimum. The *single-sized bin packing problem* (SSBPP) is the special case of the VSBPP where the bins all have the same dimensions. Chung *et al.* [16] first proposed a heuristic approach towards solving the *two-dimensional* (2D) OG[2] SSBPP in 1982 by combining heuristics for two "well-studied packing problems" [16, p. 67], namely the *one-dimensional* (1D) bin packing problem and the 2D SPP. Bengtsson [6] proposed a heuristic for the 2D RF SSBPP that packs all items into bins, and then attempts to repack them into other bins until some stopping criterion is met. Frenk and Galambos [29] proposed a heuristic that packs items into bins in a *next-fit* manner in order to solve the 2D OG SSBPP. Berkey and Wang [7] proposed a number of heuristics for the 2D OG/OF SSBPP, making use of the *next-fit*, *first-fit* and *best-fit* packing principles, while also using the BLF algorithm by Chazelle [13] to fill bins. Lodi *et al.* [51, 52] proposed pseudolevel-packing heuristics for the 2D OG/OF/RG/RF SSBPP that allow items to be packed onto the floors or ceilings of levels in an attempt to save space when packed into a strip. The levels of the strip are then repacked into bins in a manner similar to that of the hybrid heuristic by Chung *et al.* [16]. The HBP algorithm proposed by Boschetti and Mingozzi [9] allows item rotation, with the items being packed in an order determined by a "price." This price may be adjusted after every packing iteration before the next packing iteration takes place, inducing a new order of items. This process continues until some time or iteration restriction is reached. El Hayek *et al.* [26] proposed a heuristic for the 2D RF SSBPP in which regions in a bin are defined by the location of the bin boundaries and the placement of items already packed into the bin. Items are packed into these regions in a best-fit manner, where the criterion for best-fit is a weighted sum of four properties.

Heuristics for the VSBPP have typically been confined to the 1D case. Friesen and

[1] Level-packing algorithms pack all items into horizontal levels such that the bottom edges of the items are adjacent to the floor of a level, while pseudolevel algorithms allow the items to be packed anywhere within the level. Plane-packing algorithms pack items into the strip without the constraint of packing items into levels. See Ortmann [58, p. 18] for further detail regarding these classes of heuristics.

[2] The abbreviation OG was proposed by Lodi *et al.* [49, 52] to denote the problem in which items may not be rotated (the oriented problem, hence the abbreviation "O") and in which a guillotine packing is required (hence the abbreviation "G"). The two other common abbreviations are "R" for the problem where items may be rotated and "F" for a free (non-guillotine) packing. These abbreviations are used throughout this paper.

Langston [30] proposed two strategies for this problem: one that packs the largest bins first in a first-fit manner before attempting to repack the items in these bins into smaller bins (called the FFDLR strategy), and another that shifts items to smaller bins under certain circumstances before the repacking takes place (called the FFDLS strategy). Chu and La [15] proposed four strategies for the 1D VSBPP that take into account the size of the bins and the absolute or relative waste remaining when an item has been packed. Kang and Park [46] combined the FFDLR strategy with the *first-fit decreasing* (FFD) and *best-fit decreasing* (BFD) algorithms to design the *iterative first-fit decreasing* and *iterative best-fit decreasing* heuristics which achieve an optimal packing if the sizes of items and bins are exactly divisible. The first heuristic for the 2D VSBPP was proposed by Ortmann *et al.* [59], and is a combination of strip packing algorithms, namely the hybrid approach to bin packing by Chung *et al.* [16] and the repacking strategy by Friesen and Langston [30]. While this approach may have been the first heuristic for the problem, Hopper and Turton [40, 41] used the *bottom-left fill* (BLF) algorithm [13] in combination with a number of metaheuristics to find solutions to the 2D RF VSBPP, Pisinger and Sigurd [60] proposed a branch-and-price algorithm to find exact solutions to the 2D VSBPP with variable bin costs, and Yanasse *et al.* [70] used a pattern-generation algorithm to find solutions to the related 2D multiple stock size stock cutting problem.

The objective in this paper is to perform a large-scale comparison of strip packing heuristics from the literature, and to compare the best of these algorithms with respect to their propensity of solving the VSBPP by means of a two-stage packing approach. To the best of our knowledge, this is the largest statistical comparison of strip packing heuristics to date. The remainder of this paper is organised as follows. Section 2 contains the details of the comparison of the various strip packing heuristics and Section 3 contains the results from the comparison of the heuristics when modified for the VSBPP. The paper closes with a few comments on the results obtained in Section 4.

2 The strip packing problem

In order to determine which strip packing heuristics may be suitable in an algorithmic approach towards solving the 2D VSBPP, a large-scale comparison of algorithms was performed. A total of 252 known or new heuristics and variations of heuristics were applied to the 1 170 benchmark instances listed in Table 1. These benchmark instances were sourced from a number of repositories, including Beasley's *OR-library* [5], Cui's *CutWeb* [20], the DEIS Operations Research Group's library of instances [22], the EURO Special Interest Group on Cutting and Packing (ESICUP) repository [27], Fekete and Van der Veen's *PackLib*^{2} [28], Hifi's *library of instances* [38], the test instances by Scheithauer *et al.* [62] and the repository for SPPs by Van Vuuren and Ortmann [66].

2.1 Level-packing algorithms

The level-packing algorithms which formed part of this study include the *next-fit decreasing height* (NFDH) [18], *first-fit decreasing height* (FFDH) [18], *best-fit decreasing height* (BFDH) [7, 19], *knapsack problem* (KP) [52] and JOIN [55] algorithms, as well as new 2D adaptations of the 1D *worst-fit decreasing* (WFD) [45] and *best two-fit* (B2F) [31] algorithms.

Authors	Year	Reference	Number	Optimal
Christofides & Whitlock	1977	[14]	3	1 Known
Bengtsson	1982	[6]	10	All Known
Beasley	1985	[3]	13	2 Known
Beasley	1985	[4]	12	All Known
Berkey & Wang	1987	[7]	300	None known
Jakobs	1996	[44]	2	Both Known
Dagli & Poshyanonda	1997	[21]	11	None Known
Martello & Vigo	1998	[56]	200	None Known
Ratanapan & Dagli	1998	[61]	1	Not Known
Hifi	1998	[36]	25	10 Known
Hifi	1999	[37]	9	None Known
Burke & Kendall	1999	[10]	1	Known
Hopper & Turton	2000	[40, 42]	21	All Known
Hopper & Turton	2000	[40, 43]	70	All Known
Wang & Valenzuela	2001	[68]	480	All Known
Burke, Kendall & Whitwell	2004	[11]	12	All Known
Total			1 170	621 Known

Table 1: *Benchmark problem instances used to evaluate the strip packing algorithms cited in §2. Ten of the benchmark sets [3, 4, 6, 7, 14, 21, 36, 37, 56, 61] were randomly generated subject to certain area and dimensional constraints, but not from an initial rectangle in the same manner that the the others [11, 40, 42–44, 68] were generated (which allows one to deduce an optimal packing). Known optimal solutions to some of these instances are due to Martello et al. [55] and Kenmochi et al. [48].*

The *worst-fit decreasing height* (WFDH) algorithm for the 2D SPP packs items into levels in a manner that leaves a maximum residual horizontal space (the BFDH algorithm packs items so that the residual horizontal space is a minimum). The 2D adaptation of the B2F algorithm packs items into a level until no further items fit, and then attempts to replace the last item packed (called the *incumbent*) with two items that either have a greater combined width (denoted by B2FW), or a greater combined area (denoted by B2FA) and fit into the remaining space, onto the floor of the level. Searching the entire list of unpacked items for replacements may prove impractical for large problem instances; instead the algorithm restricts the search space to $k-1$ items ahead of the current item under consideration. In particular, the B2FA$_n$DH algorithm allows all items ahead of the current item to be investigated for the best pairing according to their combined areas, while the B2FW$_2$DH algorithm only allows items adjacent to one another in an ordered list to be considered for replacing the incumbent according to their combined width. While the items are typically sorted according to decreasing height for these algorithms, any ties may be resolved by sorting items of equal height according to decreasing width (denoted by DHDW) or according to increasing width (denoted by DHIW), as previously described by Ntene and Van Vuuren [57].

2.2 Pseudolevel-packing algorithms

Two classes of pseudolevel algorithms are included in this comparison study, namely those that yield guillotine feasible layouts and those that do not. The guillotine pseudolevel algorithms that form part of this study include the oriented guillotine *floor-ceiling* ($\mathrm{FC_{OG}}$) [51, 52], *modified best-fit decreasing height* (BFDH*) [8], *size-alternating stack* (SAS), *modified SAS* (SASm) [58,59] and *best-fit with stacking* (BFS) algorithms [58,59], as well as a novel *stack level* (SL) algorithm.

The SL algorithm was developed in order to combine the stacking ability of the BFS algorithm with the idea of joining items of similar height [55] in order to establish a wider platform on which items may be stacked, thereby stacking short but wide items onto tall but narrow items. The algorithm begins by sorting all items according to decreasing height, resolving any ties by sorting them according to decreasing width, and then initialising the first level with the first item in the list. Items are then packed in a best-fit manner unless the item that follows in the list is of the same height (or of heights within a percentage δ of one another). If the heights are similar, then the two items are packed adjacent to each other, and this process continues until the next item is not of similar height, or insufficient space remains on the level. Once the process is terminated, a region of height equal to the difference between the level height and the height of the first item within the height range, and of width equal to the combined widths of these adjacent items, is defined within which items may further be stacked.

The free-packing pseudolevel algorithms that form part of this comparison study include the oriented free-packing *floor-ceiling* ($\mathrm{FC_{OF}}$) [51, 52], *stack ceiling* (SC) [59] and *stack ceiling with re-sorting* (SCR) [59] algorithms. Some variations, with respect to the sorting of items in the FC and BFDH* algorithms are also included (instead of only sorting items according to decreasing height, items of equal height are sorted according to increasing or decreasing width).

2.3 Plane-packing algorithms

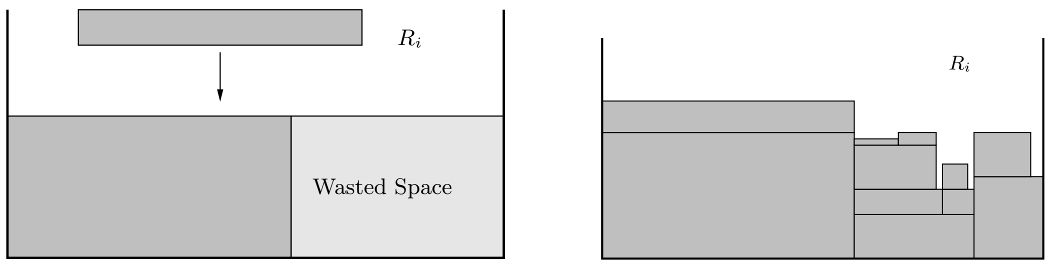
A large number of plane-packing algorithms form part of this study, including Sleator's algorithm [63], the *split-fit* (SF) algorithm [18], the *bottom-up left-justified* (BL) algorithm [2], the *split* (SP) algorithm [35], the *mixed* (M) algorithm [35], the *up-down* (UD) algorithm [1], Chazelle's BLF algorithm [13], the *guillotine cutting stock* (GCS) algorithm[3] [54] and the *best-fit* (BF) algorithm[4] [11]. Two novel modifications are also proposed for the SP algorithm.

The SP algorithm sorts items according to decreasing width and packs them into certain regions that have formed due to the items that have previously been packed. In Figure 1(a) a region R_i has been generated by the packing of an item, after which another item of the same width is to be packed. In the original algorithm the space to the right of the packed item would be wasted, while the modifications attempt to fill this space by the same procedure used in the BFS and SL algorithms, namely to stack items onto floor-packed items, taking the ceiling to which the items may be stacked as the height of the topmost edge of the item already packed (see Figure 1(b)). The resulting layout

[3]This algorithm was originally proposed to solve the 2D single stock size stock cutting problem, but is included as an example of a plane-packing heuristic that yields a guillotine layout.

[4]There are three variations of the BF algorithm by Burke *et al.* [11]: the *left-most* (BFLM), the *tallest neighbour* (BFTN) and the *shortest neighbour* (BFSN) algorithms. Each involves a different policy on item location above a skyline segment.

remains guillotine feasible. The free-packing variation allows items to drop lower if there is sufficient space, which requires a search involving the locations of all packed items in order to prevent overlaps. Ntene and Van Vuuren [57] have shown how the BF algorithm (originally designed for the 2D RF SPP) may be used to solve the 2D OF SPP. By modifying the BF algorithm to pack the first item in a list that fits into a skyline segment, instead of packing the widest item that fits into the space, the modified algorithm may find other solutions than the oriented version of the original algorithm when the sorting of items is not performed according to decreasing width (then it is the same as the oriented original). These modified algorithms are denoted by BFmLM, BFmTN and BFmSN for the left-most, tallest neighbour and shortest neighbour variations, respectively.



(a) A large space may remain empty if the SP algorithm is used to pack consecutive items with a combined width larger than $w(R_i)$.

(b) The resulting SPm algorithms attempt to fill this space with smaller items before the next item is packed.

Figure 1: *An illustration of the proposed modification to the SP algorithm. An attempt is made to pack smaller items adjacent to an item before another item is packed above it in region R_i.*

2.4 A new overarching classification for strip packing heuristics

Studying the large number of heuristics cited above has naturally led to the identification of two overarching classes of algorithms, namely *sorting-dependent* and *sorting-independent* algorithms. The class of sorting-dependent algorithms includes all level and pseudolevel algorithms forming part of this study, Sleator's algorithm, and the SF, SP, M and UD algorithms. These are called sorting-dependent algorithms because either their packing efficiency depends heavily on the order in which items are sorted,[5] or because the items are arranged into subsets according to their dimensions and these subsets are sorted in a specific manner.[6] The class of sorting-independent algorithms includes the BL, BLF, GCS and modified BF algorithms. These algorithms may be presented with a list of items in any order without affecting their packing efficiencies, on expectation. For example, MacLeod

[5]Consider any level-packing algorithm. If the items are not sorted according to decreasing height (DH), then the results will most likely be worse than they could have been had the items been sorted according to decreasing height. Every time an item is taller than the item preceding it, a new level will have to be initialised which makes it very likely that the resulting strip height will be larger than if the taller item had been packed first. The SP algorithm, for example, depends heavily on the fact that all unpacked items are no wider than those that have been packed. The algorithm does not allow for the packing of items wider than those that have already been packed.

[6]Consider the M algorithm in which items are allocated to one of five subsets, each of which is sorted in its own manner. If the items are not sorted in the correct manner, then the resulting regions may not have the correct size, shape or location for the generation of regions which are to be filled by items from other subsets.

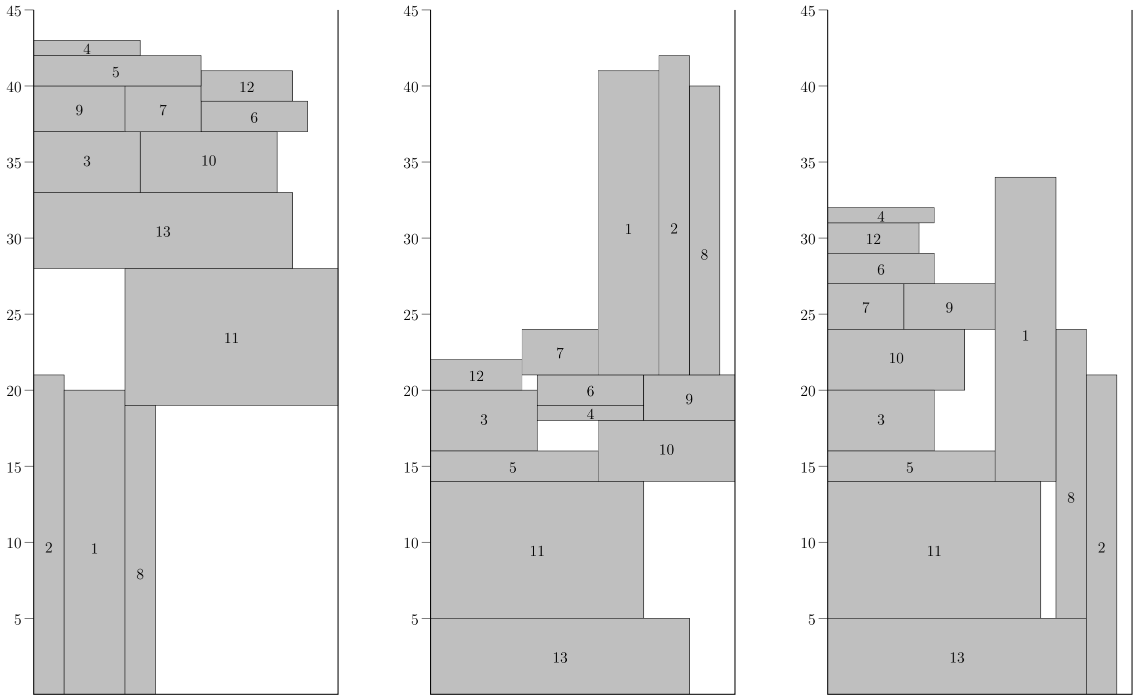
et al. [54] pass their GCS algorithm the same packing list many times, each time sorted in a different random manner, and keep the best solution. In fact, sorting-independent algorithms, such as the BL and BLF heuristics, have been incorporated into metaheuristic solution approaches, where the metaheuristic part of the algorithm determines the order in which items are arranged (see Hopper and Turton [40, 41]).

The fact that the class of sorting-independent algorithms does not require a specific sorting of items allows one to experiment with the order in which the items are sorted. A packing list sorted according to decreasing height may result in a packing that is sparse below a single wide item (particularly for the BL algorithm which does not allow items to be packed into holes in the same manner as the BLF and GCS algorithms). On the other hand, a packing list sorted by decreasing width (DW) may yield a packing with a single, pronounced vertical spike, which may have been avoided had the item been packed earlier. An attempt is made to clarify this observation in Figure 2. Potential problems that may be encountered when sorting according to decreasing height or decreasing width are shown in Figures 2(a) and 2(b), respectively, when the BL algorithm is used to pack a set of items. An attempt at finding a solution to this problem is shown in Figure 2(c). In order to achieve this result the items are first sorted according to DW, resolving ties according to DH (denoted by DWDH), and then partitioned into two groups: those items, $\mathcal{W}$, that are wider than half the strip width, and the remaining items, $\mathcal{N}$. The items in $\mathcal{N}$ are then sorted according to DHDW. This sorting approach is denoted by ½WDWDH, where the fraction denotes the fraction of the strip width at the splitting point, *i.e.* the width at which the two lists $\mathcal{W}$ and $\mathcal{N}$ are separated. In this example, the items that are wider than half the strip width remain sorted according to DWDH, and the remaining items are sorted according to DHDW. A natural modification would be to split the list according to the number of items to be packed. For example, one may want to sort the widest half of the items according to DWDH, and the remaining items according to DHDW. This strategy is denoted by ½RDWDH.

2.5 Methodology of algorithmic comparison

In order to determine whether the algorithms cited in §2.1–2.3 yield results that are significantly different from one another, the nonparametric Friedman test (as recommended by Demšar [23]) was employed, followed by a Nemenyi test (the post-hoc test recommended by Demšar [23] for these data) if the null hypothesis (that all algorithms in a comparison set yield similar results) was rejected. A nonparametric test was used, because the packing heights achieved by the heuristics relative to the optimal packing heights (or their best known lower bounds) were not normally distributed (as evidenced by a box plot of the benchmark packing data in Figures 3–6). The performance ranks of the algorithms were calculated in such a manner that the algorithm with the lowest packing height was awarded a rank of 1, while algorithms yielding the same packing heights were awarded the average of the ranks[7] that they would have been awarded had the results not been equal. The Nemenyi test determines whether algorithms are significantly different by finding a *critical distance* (CD) between ranks — if the difference between two ranks is greater

[7] Consider three algorithms that achieved packing heights of 10, 11 and 11, respectively. They would be awarded the ranks 1, 2.5 and 2.5, respectively.



(a) DH ($H = 43$) (b) DW ($H = 42$) (c) ½WDWDH ($H = 34$)

Figure 2: *An illustration of the working of a new sorting method for the class if sorting-independent algorithms; in this case the BL algorithm.*

than the CD, then the difference is significant; otherwise there is insufficient evidence to distinguish between the two algorithms. All significance tests reported in this paper were performed at a confidence level of 95%.

2.6 Strip packing algorithmic result comparison

We compare the ratios of the packing heights of the various algorithms to the optimal packing heights (or appropriate lower bounds) in this section. We adopt a divide-and-conquer approach, comparing the results of the level heuristics in four comparison sets, those of the pseudolevel algorithms in two comparison sets and finally those of the plane algorithms in eight comparison sets.

2.6.1 Results of the level-packing algorithms

The NFDH, FFDH, BFDH and WFDH algorithms and their variations were the first set of algorithms to be compared (called the *LP-1 comparison set*). The BFDHDW algorithm yields the best mean rank of 3.56 when applied to the 1 170 SPP benchmarks listed in Table 1, with the BFDH and FFDHDW algorithms yielding mean ranks of 3.78 and 3.92, respectively. The CD for the twelve algorithms and 1 170 benchmark instances is 0.49, suggesting that the best three algorithms are not significantly different from one another. However, the BFDHDW algorithm is significantly better than the nine remaining algorithms in the LP-1 comparison set.

The KP algorithm was compared to a time-restricted version (denoted by KP_{TR}) that allows a maximum time of one second to find a (possibly approximate) solution to the knapsack problem on each level, forming the *LP-2 comparison set*. If the allotted time is exceeded, then the best solution found by the solver, or from a heuristic solution, is used to fill the level. It was found that the KP_{TR}DHDW algorithm is significantly worse than the KP algorithm, but that the KP_{TR}DH and KP_{TR}DHIW are not, and they are significantly faster.

The JOIN algorithms which join items vertically and horizontally were compared for the DH, DHDW and DHIW sorting methods (in the algorithm joining items horizontally if they have a similar height), and for the DW, DWDH and DWIH sorting methods (in the algorithm that joins items vertically if they have a similar width), for height/width difference allowances of 0%, 5%, 10% and 15%, resulting in a total of 24 algorithms, forming the *LP-3 comparison set*. The $JOIN_0$DHDW, $JOIN_0$DW and $JOIN_0$DHIW algorithms yield mean ranks of 6.95, 7.07 and 7.97, respectively, and are not significantly different according to the Nemenyi test which requires a CD of 1.06 between ranks. However, these three algorithms are all significantly better than the remaining algorithms in the LP-3 comparison set.

The B2FA and B2FW algorithms were compared for the DH, DHDW and DHIW sorting variations and for the search spaces $k \in \{n, 2, 4, 6, 8, 10\}$, forming the *LP-4 comparison set*. The $B2FA_{10}$DHDW algorithm yields the lowest mean rank, followed by the $B2FA_n$DHDW and other $B2FA_k$DHDW ($k \in \{2, 4, 6, 8\}$) algorithms which, combined with the $B2FA_{10}$DH algorithm, are all not significantly different. However the $B2FA_{10}$DHDW algorithm is significantly better than the remaining algorithms in the LP-4 comparison set.

The best algorithms from each of the comparison sets LP-1, LP-2, LP-3 and LP-4 are compared to one another in Figure 3 and in Table 2. The BFDHDW algorithm yields the lowest mean rank, but the results obtained via the $B2FA_{10}$DHDW algorithm are not significantly different according to the Nemenyi test (which requires a CD of 0.14 for the four algorithms and 1 170 benchmark instances). However, these two algorithms are both significantly better than the KP_{TR}DHIW and $JOIN_0$DHDW algorithms.

2.6.2 Results of the pseudolevel-packing algorithms

The pseudolevel-packing algorithms that guarantee a guillotine layout considered in this study include the DH, DHDW and DHIW variations of the FC_{OG} and BFDH* algorithms, and the SAS, SASm, BFS and SL_δ (where $\delta \in \{0, 5, 10, 15\}$) algorithms, forming *comparison set PLP-1*. The SL_5 algorithm yields the lowest mean rank of 4.96 over the 1 170 benchmark instances, with the FC_{OG}DHDW, SL_{10} and SL_0 algorithms yielding mean ranks of 5.16, 5.33 and 5.39, respectively. The Nemenyi test suggests a CD of 0.53 for 13 algorithms and 1 170 benchmark instances, which means that these four algorithms are not significantly different. However, the SL_5 algorithm is significantly better (with respect to packing height) than the remaining 9 algorithms in the set. The SASm algorithm is the fastest[8] algorithm in the set (mean times of 1.06/1.17 seconds for "nice"/"pathological" in-

[8] All computations were performed on a Windows XP personal computer with a 3.0 GHz Intel Core 2 Duo CPU and 4 GB RAM.

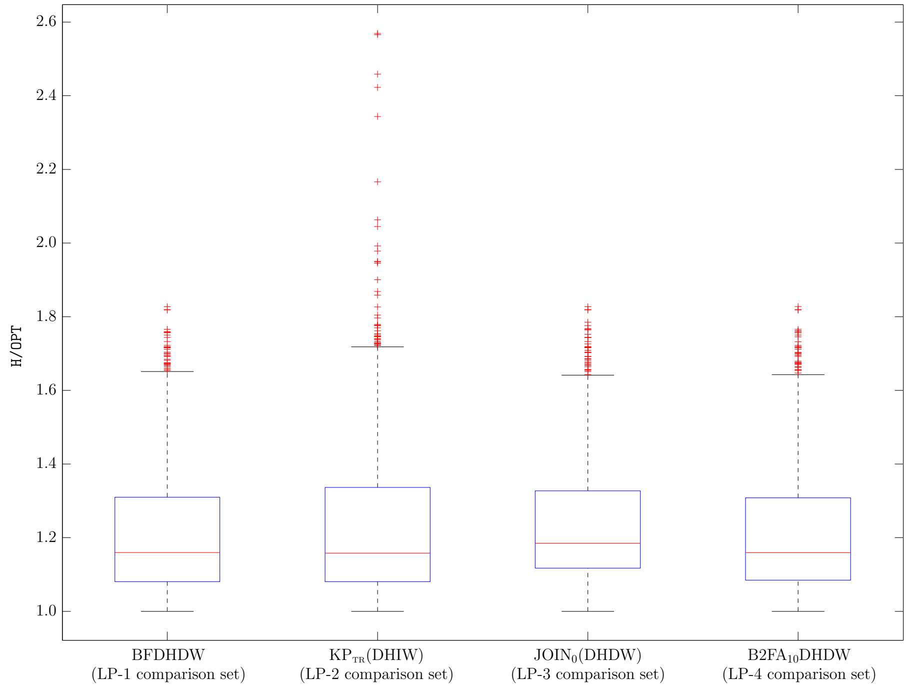


Figure 3: *Box plots of the results achieved by the best level-packing algorithm from each of the comparison sets LP-1, LP-2, LP-3 and LP-4 when applied to the* 1 170 *SPP benchmark instances listed in Table 1.*

stances of 5 000 items by Wang and Valenzuela [68]), while the BFDH* algorithms require the longest execution time for 5 000-item benchmark instances (mean times of 5.99/5.34 seconds for the "nice"/"pathological" instances).

The free-packing pseudolevel algorithms include the DH, DHDW and DHIW variations of the FC_{OF} algorithm, and the SC and SCR algorithms, forming *comparison set PLP-2*. The SC algorithm yields a mean rank of 2.56, followed by the FC_{OF}DHDW algorithm with a mean rank of 2.87 and the SCR and FC_{OF}DH algorithms with mean ranks of 3.09 and 3.10, respectively. The Nemenyi test requires a CD of 0.18 for five algorithms and 1 170 benchmark instances, suggesting that the SC algorithm is significantly better than the FC_{OF}DHDW algorithm which, in turn, is significantly better than the other algorithms in this comparison set. The SC algorithm is the fastest in this set (significantly so) requiring a mean time of 2.45/2.82 seconds to find solutions to the 5 000-item sets of "nice"/"pathological" items, while the FC_{OF}DHDW algorithm required 4.81/5.64 seconds and the SCR algorithm (the slowest in the set) required 4.96/6.47 seconds for the same problem instances.

2.6.3 Results of the plane-packing algorithms

The set of free-packing, sorting-dependent plane algorithms includes the DH, DHDW and DHIW sorting variations of the Sleator, modified SP (SPmF), M and UD algorithms, forming *comparison set PP-1*. The SPmF(DHDW) algorithm yields the lowest mean rank of 3.97, followed by the M algorithm with a mean rank of 4.03 and the DH and DHIW

	BFDHDW	KP_{TR}DHIW	$JOIN_0$(DHDW)	$B2FA_{10}$DHDW
Low. Q. `H/OPT`	108.1%	108.1%	111.7%	108.5%
Med. `H/OPT`	116.0%	115.8%	118.5%	115.9%
Up. Q. `H/OPT`	131.0%	133.6%	132.7%	130.8%
IQR	22.9%	25.5%	20.9%	22.3%
Max. `H/OPT`	182.7%	256.9%	182.7%	182.8%
Mean Rank	2.02 (1)	2.84 (3)	3.05 (4)	2.09 (2)
Nem. Class	C	B	A	C
Nice 5 000 t	2.3164	76.475	2.3166	2.2636
Path 5 000 t	2.2992	81.418	2.2797	2.2107

Table 2: *A summary of the results achieved by the best level-packing algorithms cited in this paper when applied to the* 1 170 *strip packing benchmark problem instances listed in Table 1. The row labelled* 'Median `H/OPT`' *contains the median packing height for all benchmark instances as a percentage of the optimum packing height, or its lower bound if the optimum is not known. The row labelled* 'Low. Q. `H/OPT`' *contains the value of the lower quartile, the row labelled* 'Up. Q. `H/OPT`' *contains the values of the upper quartile and the interquartile range (in the row labelled* 'IQR'*) is the difference between the two. The row labelled* 'Max. `H/OPT`' *contains the worst result achieved by the algorithms for all benchmark instances. The row labelled* 'Nem. Class' *contains results obtained by means of a Nemenyi test. Algorithms in the same group (indicated by alphabetic letters) do not produce results that are significantly different. The row labelled* 'Mean Rank' *contains the mean ranks achieved by the algorithms (a rank of 1 indicates that the algorithm packed to the lowest height for an instance), with their ranks shown in parentheses. If algorithms yielded the same packing height for an instance, the mean of the ranks that would have been awarded had these ranks been different was used in the analysis. The rows labelled* 'Nice 5 000 t' *and* 'Path 5 000 t' *show the mean solution time (in seconds) required for instances of* 5 000 *items (for the "nice" and "pathological" benchmark problem instances [68]).*

variations of the SPmF algorithm, yielding mean ranks of 4.07 and 4.18, respectively. The Nemenyi test requires a CD of 0.31 for significance, meaning that these four algorithms are not significantly different. However, they are significantly better than the Sleator and UD algorithms. The drawback of the SPmF algorithms is the time they require to find solutions to large problems — the SPmF(DHDW) algorithm requires a mean time of 197/870 seconds to find solutions to the 5 000-item "nice"/"pathological" benchmark instances, compared to the 4.45/4.38 seconds required by the M algorithm.

The sorting-dependent, guillotine-packing plane algorithms include the DH, DHDW and DHIW versions of the SF algorithm, and the DW, DWDH and DWIH versions of the SP and modified SP (SPmG) algorithms, forming *comparison set PP-2*. The SPmG algorithms are significantly better than the SF algorithms which, in turn, are significantly better than the SP algorithms. The SPmG(DWDH), SPmG(DW) and SPmG(DWIH) algorithms achieve mean ranks of 3.20, 3.41 and 3.49, compared to the mean ranks of 4.28, 4.37 and 4.51 for the SF(DHDW), SF(DH) and SF(DHIW) algorithms, respectively (the Nemenyi CD is 0.35 when comparing nine algorithms over 1 170 benchmark instances). However, the SPmG(DWDH) algorithm requires a mean time of 3.16/3.49 seconds to solve the 5 000-item instances, compared to the SF(DHDW) algorithm's 2.56/2.55 seconds and the SP(DWDH) algorithm's 2.33/2.34 seconds. The SL_5 algorithm yields better results than the SPmG(DWDH) algorithm in a time similar to that of the SP(DWDH) algorithm; hence algorithms from this comparison set are not used in further comparisons.

The sorting-independent algorithms were tested according to 23 sorting methods, namely the DH, DHDW, DHIW, DW, DWDH, DWIH, DA, DADH, DADW, xWDWDH and xRDWDH methods, where $x \in \{2/3, 3/5, 11/20, 1/2, 9/20, 2/5, 1/3\}$. The new xWDWDH sortings yield the best mean ranks for the BL algorithm (*comparison set PP-3*), with the ½WDWDH sorting yielding the lowest mean rank (the mean rank of 7.45 is not significantly different to the mean ranks of 7.63 and 7.73 for the 11/20WDWDH and 3/5WDWDH variations due to a Nemenyi CD of 1.01) and the variations that sort according to decreasing item height yielding the lowest solution times (means of 8.73/9.70 seconds for 5 000-item benchmark instances by the DHDW variation compared with 11.16/12.15 seconds required by the ½WDWDH variation).

The BLF2/5WDWDH algorithm yields the best mean rank of 8.09 (followed by 8.18 for the 9/20WDWDH variation and 8.26 for the ½WDWDH variation — the Nemenyi CD is again 1.01) for the set of BLF algorithms (*comparison set PP-4*) and it belongs to the subset of fast algorithms as it requires 27.8/33.8 seconds to solve 5 000-item problem instances compared with times of 34.4/193.9 seconds for the DWIH variation (which yields a mean rank of 18.14).

The ½WDWDH sorting method yields the best mean rank for the GCS algorithms (*comparison set PP-5*) with a mean rank of 8.48 (compared to the mean ranks of 8.56 and 8.75 for the 9/20WDWDH and 11/20WDWDH variations, respectively, with a Nemenyi CD of 1.01), but required 2 021/694 seconds to find solutions to problems with 2 000 items.

The 1/3WDWDH sorting method yields the best mean rank in the set of BFmLM algorithms (comparison set PP-6) with a mean rank of 9.70 (compared to the mean ranks of 10.05 and 10.08 for the 2/5WDWDH and DADW variations, respectively, for a Nemenyi CD of 1.06), but the DADW method proved faster, requiring a mean time of 2.34/2.49 seconds *versus* 4.84/4.80 seconds for the 2/5WDWDH method to find solutions to 5 000-item benchmark instances. The same sorting methods yield the best (with respect to packing height) and fastest solutions when used by the BFmTN algorithm (*comparison set PP-7*), with the 1/3WDWDH, 2/5WDWDH and 9/20WDWDH variations yielding the mean ranks of 9.11, 9.49 and 9.77, respectively (the Nemenyi CD is 1.06). However, the DADW sorting method proved most effective when used by the BFmSN algorithm (*comparison set PP-8*) and was fast, being significantly slower than only the oriented version of the original algorithm. It yields a mean rank of 10.15, compared to 10.16 and 10.20 for the DADH and DA variations, respectively. The packing results achieved by the best of these algorithms are shown in Figure 4 and Table 3.

2.6.4 Overall appraisal of strip packing heuristics

When comparing the algorithms in Figure 4 it is immediately obvious that the SASm, M and BL algorithms do not yield results as good as those of the other algorithms, and that the best of the BFmSN algorithms is not as good as the best of the BFmLM or BFmTN algorithms. Of the guillotine algorithms, the GCS½WDWDH algorithm yields the best mean rank, but it is not significantly better than the SL_5 algorithm according to the Nemenyi test and it is significantly slower. The SASm algorithm may not yield very good solutions when compared to many of the other algorithms in this set, but it does require the lowest execution time to find feasible solutions. The pseudolevel-

	SASm	SL_5	SC	M	BL(DHDW)	BL(½W)	BLF(⅖W)
Low. Q. `H/OPT`	109.3%	106.1%	105.8%	110.2%	111.3%	109.6%	103.9%
Med. `H/OPT`	114.4%	109.5%	109.0%	118.8%	116.7%	114.4%	107.5%
Up. Q. `H/OPT`	123.1%	116.2%	114.7%	137.0%	123.5%	119.5%	113.2%
IQR	13.77%	10.09%	8.98%	26.81%	12.19%	9.84%	9.28%
Max. `H/OPT`	195.1%	169.2%	153.2%	234.9%	168.3%	151.7%	151.7%
Mean Rank	9.78 (11)	6.72 (8)	6.03 (6)	10.57 (13)	10.24 (12)	9.11 (10)	4.43 (3)
Sig. Class	B	E	FG	A	AB	C	H
Nice 2 000 t	0.1369	0.2672	0.2791	0.4843	1.2473	1.5018	4.1914
Path 2 000 t	0.1518	0.2705	0.3479	0.4825	1.3395	1.6000	4.7251

	GCS(½W)	BFmLM(DADW)	BFmLM(⅓W)	BFmTN(DADW)	BFmTN(⅓W)	BFmSN(DADW)
Low. Q. `H/OPT`	104.9%	106.0%	104.6%	104.6%	104.0%	108.1%
Med. `H/OPT`	109.7%	109.5%	109.4%	107.9%	107.2%	112.5%
Up. Q. `H/OPT`	116.5%	113.9%	114.0%	111.6%	111.2%	117.1%
IQR	11.59%	7.88%	9.46%	7.03%	7.19%	8.92%
Max. `H/OPT`	151.7%	171.3%	152.2%	162.4%	152.2%	171.3%
Mean Rank	6.40 (7)	5.99 (5)	5.55 (4)	4.39 (2)	3.71 (1)	8.09 (9)
Nem. Class	EF	FG	G	H	I	D
Nice 2 000 t	2 021.3	0.2736	0.5338	0.2758	0.5343	0.2767
Path 2 000 t	694.25	0.2918	0.5396	0.2933	0.5397	0.2935

Table 3: *A summary of the results achieved by the best algorithms from the class of pseudolevel and plane packing algorithms when applied to the 1 170 SPP benchmark instances listed in Table 1. The headings (x*W*) (where x is a fraction) are abbreviations of xWDWDH. The row labelled* 'Median `H/OPT`' *contains the median packing height for all benchmark instances listed in Table 1 as a percentage of the optimal packing height, or its best lower bound if the optimum is not known. The row labelled* 'Low. Q. `H/OPT`' *contains the value of the lower quartile, the row labelled* 'Up. Q. `H/OPT`' *contains the values of the upper quartile and the interquartile range (in the row labelled* 'IQR'*) is the difference between the two. The row labelled* 'Max. `H/OPT`' *contains the worst result achieved by the algorithms for all benchmark instances. The row labelled* 'Nem. Class' *contains results obtained by means of a Nemenyi test. Algorithms in the same group (indicated by alphabetic letters) do not produce results that are significantly different at a 95% level of confidence. The row labelled* 'Mean Rank' *contains the mean ranks achieved by the algorithms in this set (a rank of 1 indicates that the algorithm packed to the lowest height for an instance), with their ranks shown in parentheses. If algorithms yielded the same packing height for an instance, the mean of the ranks that would have been awarded was used in the analysis. The rows labelled* 'Nice 2 000 t' *and* 'Path 2 000 t' *show the mean solution time (in seconds) required for instances of 2 000 items (for the "nice" and "pathological" benchmark problem instances [68]).*

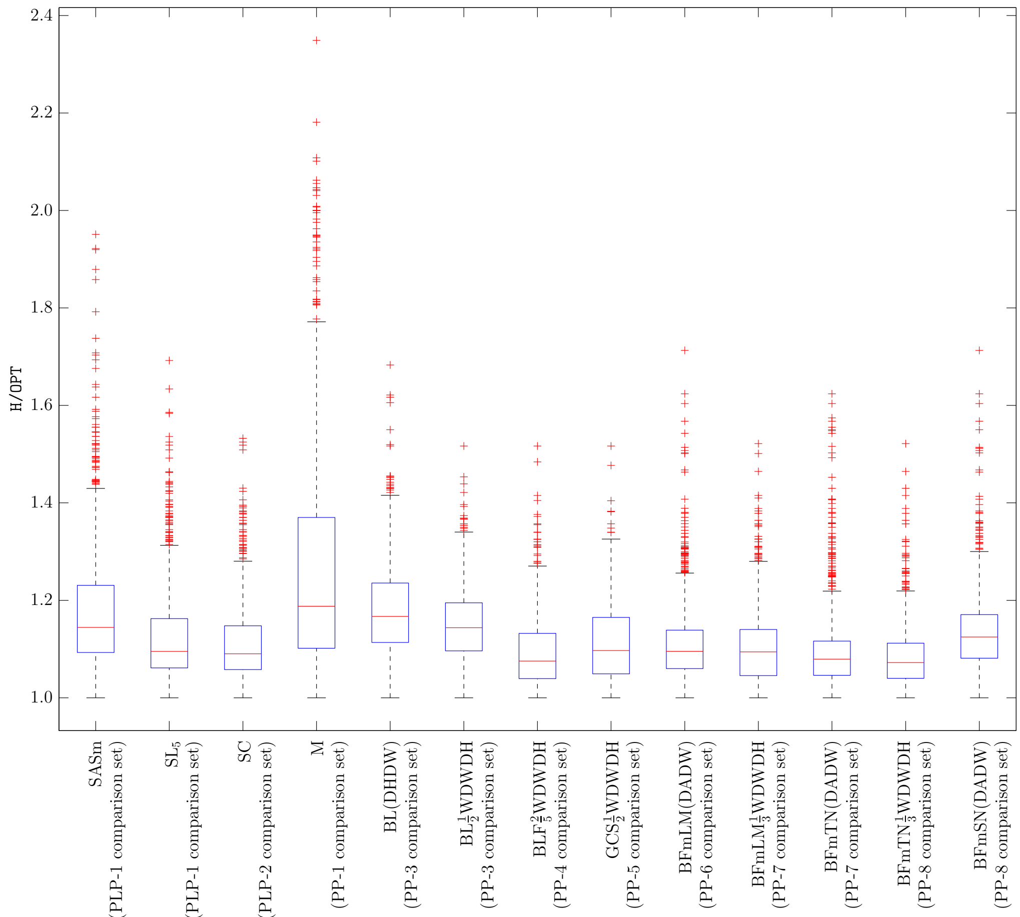


Figure 4: *Box plots of the distribution of results achieved by the best pseudolevel and plane algorithms from each comparison set when applied to the 1 170 SPP benchmark instances listed in Table 1.*

packing SC algorithm yields solutions that are not significantly different to those of the plane-packing GCS½WDWDH, BFmLM(DADW) and BFmLM⅓WDWDH algorithms, but results that are significantly better than the plane-packing M, BL and BFmSN algorithms. The BFmTN⅓WDWDH algorithm yields the lowest mean rank and is significantly better than the second-ranked BFmTN(DADW) algorithm, which is equivalent to the BLF⅖WDWDH algorithm in terms of packing height, but much faster. These algorithms are followed by the BFmLM algorithms in terms of mean ranks. It is clear that using the xWDWDH sorting method typically results in slower algorithms than do the previously used sorting methods (those sorting items according to height, width or area), but in some cases yields solutions of significantly higher quality.

3 The variable-sized bin packing problem

Now that suitable strip packing algorithms have been found which may be combined with the 2SVSBP algorithm of Ortmann *et al.* [59], the combination of these strip packing algorithms with the 2SVSBP algorithm may be compared on benchmarks for the VSBPP. For this comparison we made use of the multiple stock size stock cutting problem instance described by Wang [67, p. 585], and the algorithmically-generated benchmark instances by Hopper and Turton [40, 42], Pisinger and Sigurd [60] and Ortmann *et al.* [59]. Out of interest, the algorithms were also applied to the benchmark instances by Berkey and Wang [7] and Martello and Vigo [56] for the 2D SSBPP, giving rise to a total of 857 benchmark instances for the VSBPP and 500 instances for the SSBPP.

3.1 The 2SVSBP algorithm

The 2SVSBP algorithm [58, 59] begins by packing all items into a strip by means of a level or pseudolevel strip packing algorithm. The bins are then sorted according to decreasing area and the levels of the packing are packed into the largest bin in the set. When no further levels fit into the bin, then the levels are packed into the next bin in the list. If the new bin width is different to the width of the previous bin, then a new strip packing is performed, with the width of the strip taken as the width of the empty bin. Once all items have been packed, the bin containing the smallest area of items is selected for repacking, and the smallest empty bin of area no less than the area of the items is selected as the target. The items are packed into a strip of the same width as the empty bin. If the strip height is no larger than the height of the empty bin, then the items may be repacked into it. However, if the strip height is larger than the bin height, then the previous (possibly larger) bin in the list is selected as a target. This process continues until the items are repacked, or the target bin is the same bin as the one containing the items. When an attempt to repack a bin has been completed, the bin with the next smallest area of items is selected for repacking. This process continues until attempts have been made to repack all bins.

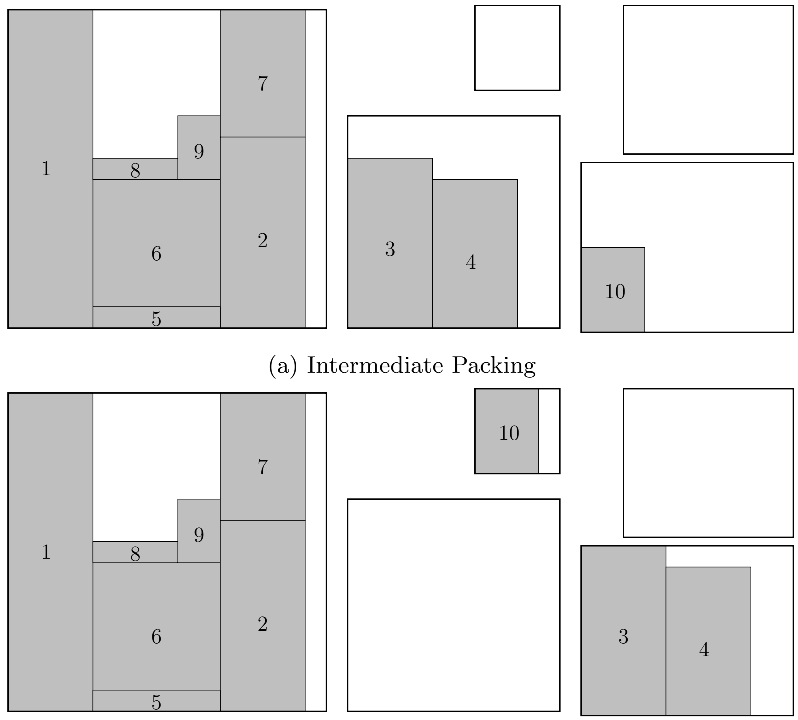
Figure 5 contains an example of how the repacking stage of the algorithm may improve utilisation. The items are first packed into a strip by means of the SAS algorithm [57] and then packed into the bins as shown in Figure 5(a). The item in the third-largest bin may be repacked into the smallest bin and the items in the second-largest bin may be repacked into the bin that was rendered empty by the previous packing. The items in the largest bin cannot be repacked into any of the remaining empty bins and the resulting packing is shown in Figure 5(b).

3.2 Selection of algorithms for comparison purposes

The best of each class of level-packing algorithms, when incorporated into the 2SVSBP algorithm, was compared by means of the so-called packing *utilisation*[9] and *fitness*[10] scores. The BFDHDW, KP_{TR}DHDW, JOIN_0DHDW, B2FA_{10}DHDW and B2FW_2DHDW algorithms yield the best mean ranks in their respective classes when they are compared with

[9] The utilisation μ of a packing is the total area of the items to be packed divided by the sum of the areas of the bins that eventually contain items.

[10] The fitness ν of a solution to the VSBPP, as proposed by Hopper [40], is a measure that aims to reward algorithms for dense packing of bins. This allows one to distinguish between algorithms when their utilisations are equal for all solutions. A solution in which most bins are densely packed and one bin is not, would typically achieve a higher fitness score than an algorithm that packs bins less densely. The fitness



(a) Intermediate Packing

(b) Final Packing

Figure 5: *Results obtained by the two-stage algorithm for the VSBPP (2SVSBP-SAS), using the SAS algorithm for strip packing.*

respect to the 857 benchmark instances. A Nemenyi test (requiring a CD of 0.27) on their utilisations show that the JOIN algorithm, with a mean rank of 3.86, is significantly worse than the B2FW algorithm (which yields a mean rank of 2.99) which, in turn, is significantly worse than the BFDHDW, KP_{TR}DHDW and B2FA_{10}DHDW algorithms which yield mean ranks of 2.68, 2.72 and 2.74, respectively. The final three algorithms are not significantly different when compared with respect to mean ranks of bin utilisation. However, if the fitness scores are used, then the BFDHDW algorithm (with a mean rank of 2.46) is significantly better than the KP_{TR}DHDW and B2FA_{10}DHDW algorithms, which achieve mean ranks of 2.69 and 2.74, respectively.

Comparing the guillotine pseudolevel-packing algorithms shows that the SASm algorithm is significantly worse (with a mean rank of 3.98) than the FC_{OG}DHDW algorithm (which yields the best mean rank of 2.70), the BFDH*(DW) algorithm (which yields the fourth lowest mean rank of 2.86), the BFS algorithm (which yields the third lowest mean rank of 2.74) and the SL_5 algorithm (which yields the second lowest mean rank of 2.71), but

of a solution is defined as

$$\nu = \frac{\sum_{i=1}^{M}\left(\frac{A(\mathcal{I}^{\mathcal{B}_i})}{A(\mathcal{B}_i)}\right)^k}{M},$$

where $A\left(\mathcal{I}^{\mathcal{B}_i}\right)$ denotes the total area of the items packed into bin $\mathcal{B}_i$, where $A\left(\mathcal{B}_i\right)$ denotes the area of bin $\mathcal{B}_i$, and where M is the number of bins that contain items in the solution. Typically $k = 2$.

these algorithms are not significantly different from one another due to a Nemenyi CD of 0.21 (the significance results, but not the ranks, are the same for the algorithms when applied to the SSBPP). By using the fitness score to rank the algorithms, the Nemenyi test suggests that the BFDH*(DW) algorithm (with a mean rank of 2.93) is significantly worse than the FC_{OG}DHDW, BFS and SL_5 algorithms which achieve mean ranks of 2.56, 2.67 and 2.65, respectively. The free-packing pseudolevel algorithms, when combined with the 2SVSBP algorithm, yield results that are significantly different according to the Friedman test when applied to the fitness score, and the Nemenyi test suggests that the FC_{OF}DHDW algorithm (with a mean rank of 1.92) is significantly better than the SCR algorithm (which yields a mean rank of 2.08), but the SC algorithm (with a mean rank of 2.00) cannot be distinguished from either algorithm due to a Nemenyi CD of 0.11.

3.3 Adaptation of plane-packing algorithm

The 2SVSBP algorithm was initially designed to make use of available level and pseudolevel algorithms to pack items into bins in a generic manner that does not require each algorithm to be reimplemented for the 2D VSBPP. However, the best of the strip packing algorithms, the BFmTN heuristic, requires a new procedure for incorporating the repacking strategy of the 2SVSBP algorithm. The structure of this algorithm is very similar to that of the 2SVSBP algorithm, but instead of performing a strip packing with the unpacked items, these items are packed directly into the first bin. The procedure that performs this task is very similar to the BFmTN algorithm for the SPP, with the difference that an item is only packed if the space between the skyline onto which an item is to be packed and the top-most edge of the bin is no smaller than the height of the item. If no item is found that fits onto a skyline segment, this segment is raised to the height of the lowest neighbouring segment in the hope that a wider item which is short enough to fit into the bin may fit onto the wider segment. The bin is filled until none of the unpacked items fit into the remaining space. If the new xWDWDH sorting method is employed, the items may have to be re-sorted if the bin width changes. An attempt is made to repack the bins (after all items have been packed for the first time) in the same manner as for the original 2SVSBP algorithm.

3.4 Variable-sized bin packing algorithmic result comparison

It turns out that the BFmTN(DA) algorithm for the VSBPP yields the lowest mean rank and is significantly better than the BFmTN⅓WDWDH algorithm. The first problem instance by Wang [67] is the largest in terms of the number of items to be packed and the BFmTN(DA) algorithm requires 4.87 seconds to find a solution for this instance compared to the 143 seconds required by the BFmTN⅓WDWDH algorithm and the 129 seconds required by the BFmTN⅓RDWDH algorithm. The two new sorting methods are likely to yield slow solution procedures due to the re-sorting required for each bin packing. The results in Figure 6 and Table 4 show that the modified BFmTN(DA) algorithm typically finds better solutions (significantly better according to the Nemenyi test which requires a CD of 0.31 when comparing seven algorithms over 857 benchmark instances) than the level and pseudolevel algorithms — an expected result considering that the BFmTN(DA) algorithm is not constrained by a rule requiring it to pack items into levels. It is also faster than the other algorithms which is likely an artifact of the level and pseudolevel SPP algorithms being called by a generic algorithm, while the BFmTN(DA) algorithm is more closely integrated with the bin packing and repacking procedures. The results show that the pseudolevel-packing SASm algorithm, while fast, yields results that are significantly worse than the level-packing BFDHDW algorithm. The BFDHDW algorithm is also significantly worse than the remaining pseudolevel algorithms. The Nemenyi test was unable to distinguish between the remaining pseudolevel-packing algorithms, and the mean ranks for the guillotine $\mathrm{FC_{OG}DHDW}$ and $\mathrm{SL_5}$ algorithms were better than for the free-packing SC algorithm — an unexpected result, even if the difference in ranks is very small. The results for the SSBPP (shown in Figure 5) suggest that there is no significant difference between the BFDHDW algorithm and the pseudolevel-packing algorithms (excluding the SASm algorithm, which is significantly worse). The BFmTN(DA) algorithm is the best by a large margin.

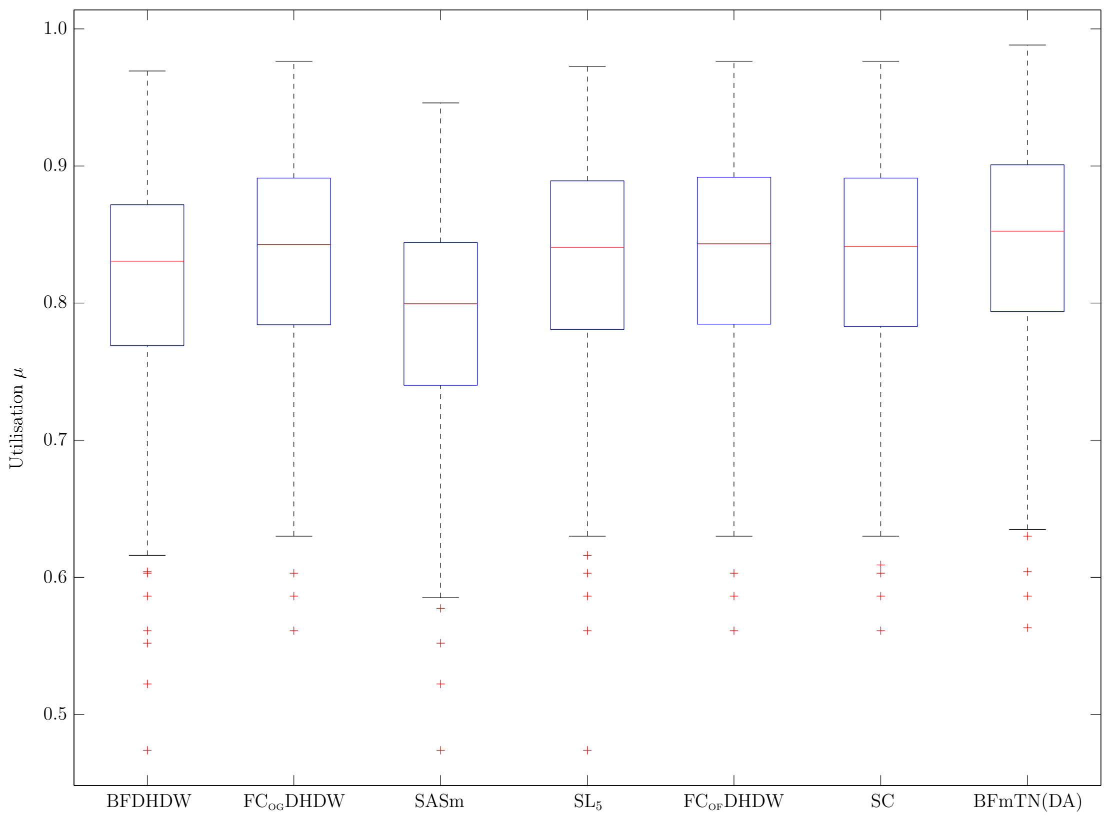


Figure 6: *Box plot of the utilisations achieved by the best heuristics for the VSBPP when applied to the 857 VSBPP benchmark instances.*

4 Conclusion

In this paper a total of 252 SPP heuristics (or variations of thereof) were tested on a total of 1 170 benchmark instances; to the best knowledge of the authors this is the largest comparison of SPP heuristics performed to date. The results were subjected to nonparametric

	BFDHDW	FC_{OG}DHDW	SASm	SL_5	FC_{OF}DHDW	SC	BFmTN(DA)
Min. μ	47.4%	56.1%	47.4%	47.4%	56.1%	56.1%	56.3%
Low. Q. μ	76.9%	78.4%	74.0%	78.1%	78.5%	78.3%	79.4%
Med. μ	83.1%	84.3%	80.0%	84.1%	84.3%	84.1%	85.3%
Up. Q. μ	87.2%	89.1%	84.4%	88.9%	89.2%	89.1%	90.1%
Max. μ	96.9%	97.6%	94.6%	97.3%	97.6%	97.6%	98.8%
IQR	10.3%	10.7%	10.4%	10.8%	10.7%	10.8%	10.7%
Wang P1 t (s)	5.9768	16.082	5.4528	5.9386	16.328	5.8691	4.8691
Total Bins	9 306	9 248	9 642	9 259	9 246	9 280	9 158
Repacked Bins	1 503	1 538	1 840	1 561	1 535	1 610	1 541
% Repacked	16.2%	16.6%	19.1%	16.9%	16.6%	17.3%	16.8%
Stationary Bins	7 803	7 710	7 802	7 698	7 711	7 670	7 617
Mean μ Rank	4.45 (6)	3.71 (3)	5.49 (7)	3.74 (4)	3.69 (2)	3.77 (5)	3.15 (1)
Nem. μ Class	B	C	A	C	C	C	D
Mean ν Rank	4.86 (6)	3.65 (3)	5.76 (7)	3.75 (5)	3.53 (2)	3.73 (4)	2.71 (1)
Nem. ν Class	B	C	A	C	C	C	D

Table 4: *Overview of the results achieved by a selection of algorithms for the VSBPP. The BFDHDW algorithm was the best of the level-packing algorithms, the FC_{OG}DHDW algorithm yielded the best mean rank in the set if guillotine pseudolevel algorithms, while the SASm algorithm yielded the fastest results and the SL_5 algorithm yielded a good balance of speed and packing density. The FC_{OF}DHDW algorithm yielded the lowest mean rank for free-packing pseudolevel algorithms, while the SC algorithm yielded the fastest results in the same comparison set. The row labelled* 'Min. μ' *contains the minimum bin utilisation over the 857 MBSBP benchmark instances, while the rows labelled* 'Low. Q. μ', 'Med. μ', 'Up. Q. μ', 'Max. μ' *and* 'IQR' *contain the lower quartile, median, upper quartile, maximum and interquartile range of the results for the instances, respectively. The row labelled* 'Wang P1 t (s)' *contains the time taken (in seconds) for the algorithms to complete the packing of the first problem by Wang [67], the largest benchmark instance. The row labelled* 'Total Bins' *contains the total number bins used over all MBSBP benchmark instances, the row labelled* 'Repacked Bins' *documents the total number of bins that were repacked during the repacking phase of the 2SVSBP algorithm, the row labelled* '% Repacked' *indicates what percentage of the total number of bins this repack value is, and the row labelled* 'Stationary Bins' *lists the number of bins that were not repacked. The row labelled* 'Mean μ Rank' *documents the mean ranks of the algorithms when applied to the utilisation (the ranks are given in parentheses), while the row labelled* 'Nem. μ Class' *shows which algorithms are not significantly different according to the Nemenyi test [23] by assigning them the same letter. The same tests are performed for the fitness ν in the two rows that follow.*

	BFDHDW	FC_{OG}DHDW	SASm	SL_5	FC_{OF}DHDW	SC	BFmTN(DA)
Mean p Rank	3.97 (5)	3.83 (3)	5.19 (7)	3.89 (4)	3.82 (2)	4.02 (6)	3.28 (1)
Nem. p Class	B	B	A	B	B	B	C
100 t (ms)	1.8920	4.2915	1.6787	2.0243	3.9288	2.1285	1.6618
BW 1	20.62 (3.97)	20.60 (3.90)	21.24 (5.74)	20.62 (3.96)	20.60 (3.90)	20.62 (3.95)	20.20 (2.58)
BW 2	2.64 (4.09)	2.60 (3.95)	2.70 (4.30)	2.60 (3.95)	2.60 (3.95)	2.60 (3.95)	2.56 (3.81)
BW 3	14.72 (3.52)	14.72 (3.52)	16.26 (6.74)	14.80 (3.75)	14.72 (3.52)	14.86 (3.93)	14.54 (3.02)
BW 4	2.60 (4.09)	2.56 (3.95)	2.70 (4.44)	2.56 (3.95)	2.54 (3.88)	2.54 (3.88)	2.52 (3.81)
BW 5	18.70 (3.76)	18.70 (3.76)	19.66 (6.03)	18.70 (3.76)	18.70 (3.76)	18.88 (4.28)	18.34 (2.65)
BW 6	2.36 (3.98)	2.36 (3.98)	2.44 (4.26)	2.36 (3.98)	2.36 (3.98)	2.36 (3.98)	2.32 (3.84)
MV 7	17.18 (4.24)	17.10 (3.96)	17.22 (4.36)	17.12 (4.03)	17.10 (3.96)	17.18 (4.24)	16.88 (3.21)
MV 8	17.52 (3.93)	17.50 (3.86)	18.08 (5.47)	17.54 (3.99)	17.50 (3.86)	17.52 (3.93)	17.22 (2.96)
MV 9	42.78 (3.85)	42.78 (3.85)	43.00 (4.62)	42.78 (3.85)	42.78 (3.85)	42.90 (4.27)	42.74 (3.71)
MV 10	10.74 (4.27)	10.52 (3.57)	11.42 (5.98)	10.54 (3.63)	10.52 (3.57)	10.58 (3.75)	10.42 (3.23)
Total Bins	7 493	7 472	7 736	7 481	7 471	7 502	7 387

Table 5: *Algorithmic results for a selection of SSBPP algorithms with respect to various sets of benchmark instances. The BFDHDW algorithm was the best of the level-packing algorithms, the FC_{OG}DHDW algorithm yielded the best mean rank in the set if guillotine pseudolevel algorithms, while the SASm algorithm yielded the fastest results and the SL_5 algorithm yielded a good balance of speed and packing density. The FC_{OF}DHDW algorithm yielded the lowest mean rank for free-packing pseudolevel algorithms, while the SC algorithm yielded the fastest results in the same comparison set. The row labelled* 'Mean p Rank' *shows the mean rank over the 500 benchmark instances in terms of the number of bins packed, while the row* 'Nem. p Class' *shows which algorithms are not significantly different by placing them in the same class, indicated by a letter. Finally, the row labelled* '100 t (ms)' *shows the mean time (in milliseconds) that the algorithms required to solve the SSBPP benchmark instances with 100 items. The results below these rows are the mean numbers of bins for each problem class.*

statistical tests in an attempt to compare the algorithms in an unbiased fashion, at a 95% level of confidence.

A new strip packing heuristic was also proposed for the 2D OG SPP, namely the SL algorithm. This novel heuristic outperformed other guillotine-packing algorithms that pack items into levels in terms of packing height and, in many cases, execution time.

Two new sorting methods were proposed for the class of sorting-independent SPP algorithms and the xWDWDH method was shown to yield the best results in terms of packing height for many of the algorithms. This included modified versions of the best-fit algorithm by Burke *et al.* [11] which were shown to yield better results than the BLF algorithm, one of the "most documented heuristic approaches" for the SPP [11, p. 656].

It was also shown how all of the above algorithms may be combined with the 2SVSBP algorithm by Ortmann *et al.* [59] in order to find good solutions to the 2D OF VSBPP. These solutions may be used as initial solutions to metaheuristics designed to improve on the packing density.

Finally, we remark that the differences between the solution qualities of the algorithms are not as marked for the VSBPP as for the SPP.

Acknowledgements

Work towards this paper was financially supported by the South African National Research Foundation (NRF) in the form of bursaries (GUN 2072815 and GUN 66976) awarded to the first author, and a research grant (GUN 2072999) awarded to the second author.

References

[1] Baker BS, Brown DJ & Katseff HP, 1981, *A 5/4 algorithm for two-dimensional packing*, Journal of Algorithms, **2(4)**, pp. 348–368.

[2] Baker BS, Coffman EG & Rivest RL, 1980, *Orthogonal packings in two dimensions*, SIAM Journal on Computing, **9(4)**, pp. 846–855.

[3] Beasley JE, 1985, *Algorithms for unconstrained two-dimensional guillotine cutting*, Journal of the Operational Research Society, **36(4)**, pp. 297–306.

[4] Beasley JE, 1985, *An exact two-dimensional non-guillotine cutting tree search procedure*, Operations Research, **33(1)**, pp. 49–64.

[5] Beasley JE, 2009, *OR-Library*, [Online], [Cited August 25th, 2009], Available from `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`

[6] Bengtsson BE, 1982, *Packing rectangular pieces — A heuristic approach*, The Computer Journal, **25(3)**, pp. 353–357.

[7] Berkey JO & Wang PY, 1987, *Two-dimensional finite bin-packing algorithms*, Journal of the Operational Research Society, **38(5)**, pp. 423–429.

[8] Bortfeldt A, 2006, *A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces*, European Journal of Operational Research, **172(3)**, pp. 814–837.

[9] Boschetti MA & Mingozzi A, 2003, *The two-dimensional finite bin packing problem. Part II: New lower and upper bounds*, 4OR (Quarterly Journal of the Belgian, French and Italian Operations Research Societies), **1(2)**, pp. 135–147.

[10] Burke E & Kendall G, 1999, *Applying simulated annealing and the no fit polygon to the nesting problem*, pp. 27–30 in *Proceedings of the WMC '99*, World Manufacturing Congress, Durham.

[11] Burke EK, Kendall G & Whitwell G, 2004, *A new placement heuristic for the orthogonal stock-cutting problem*, Operations Research, **52(4)**, pp. 655–671.

[12] Burke EK, Kendall G & Whitwell G, 2006, *A new placement heuristic for the orthogonal stock-cutting problem*, (Unpublished) Technical Report NOTTCS-TR-2006-3, School of Computer Science and Information Technology, University of Nottingham, Nottingham.

[13] Chazelle B, 1983, *The bottom-left bin packing heuristic: An efficient implementation*, IEEE Transactions on Computers, **32(8)**, pp. 697–707.

[14] Christofides N & Whitlock C, 1977, *An algorithm for two-dimensional cutting problems*, Operations Research, **25(1)**, pp. 31–44.

[15] Chu C & La R, 2001, *Variable-sized bin packing: Tight absolute worst-case performance ratios for four approximation algorithms*, SIAM Journal on Computing, **30(6)**, pp. 2069–2083.

[16] Chung FRK, Garey MR & Johnson DS, 1982, *On packing two-dimensional bins*, SIAM Journal on Algebraic and Discrete Mathematics, **3(1)**, pp. 66–76.

[17] Coffman EG, Garey MR & Johnson DS, 1996, *Approximation algorithms for bin packing: A survey*, pp. 46–93 in Hochbaum DS (Ed), *Approximation algorithms for NP-hard problems*, PWS Publishing Company, Boston (MA).

[18] Coffman EG, Garey MR, Johnson DS & Tarjan RE, 1980, *Performance bounds for level-oriented two dimensional packing algorithms*, SIAM Journal on Computing, **9(4)**, pp. 808–826.

[19] Coffman EG & Shor PW, 1990, *Average-case analysis of cutting and packing in two dimensions*, European Journal of Operational Research, **44(2)**, pp. 134–144.

[20] Cui Y, 2009, *CutWeb*, [Online], [Cited August 25$^{\text{th}}$, 2009], Available from `http://www.gxnu.edu.cn/Personal/ydcui/English/index.htm`

[21] Dagli CH & Poshyanonda P, 1997, *New approaches to nesting rectangular patterns*, Journal of Intelligent Manufacturing, **8(3)**, pp. 177–190.

[22] DEIS — Operations Research Group, 2004, *Library of instances*, [Online], [Cited August 25$^{\text{th}}$, 2009], Available from `http://www.or.deis.unibo.it/research.html`

[23] Demšar J, 2006, *Statistical comparisons of classifiers over multiple data sets*, Journal of Machine Learning, **7**, pp. 1–30.

[24] Dyckhoff H, 1990, *A topology of cutting and packing problems*, European Journal of Operational Research, **44(2)**, pp. 145–159.

[25] Eisemann K, 1957, *The trim problem*, Management Science, **3(3)**, pp. 279–284.

[26] El Hayek J, Moukrim A & Negre S, 2008, *New resolution algorithm and pretreatments for the two-dimensional bin-packing problem*, Computers and Operations Research, **35(10)**, pp. 3184–3201.

[27] ESICUP, 2007, *Listing gallery: Data Sets 2D — Rectangular*, [Online], [Cited October 29$^{\text{th}}$, 2007], Available from `http://paginas.fe.up.pt/~esicup/tiki-list_file_gallery.php?galleryId=3`

[28] Fekete SP & Van der Veen J, 2009, *Instances PackLib^2*, [Online], [Cited April 15$^{\text{th}}$, 2008], Available from `http://mo.math.nat.tu-bs.de/packlib/instances.shtml`

[29] Frenk JBG & Galambos G, 1987, *Hybrid next-fit algorithm for the two-dimensional rectangle bin-packing problem*, Computing, **39(3)**, pp. 201–217.

[30] Friesen DK & Langston MA, 1986, *Variable sized bin packing*, SIAM Journal on Computing, **15(1)**, pp. 222–230.

[31] Friesen DK & Langston MA, 1991, *Analysis of a compound bin packing algorithm*, SIAM Journal on Discrete Mathematics, **4(1)**, pp. 61–79.

[32] Gilmore PC & Gomory RE, 1961, *A linear programming approach to the cutting-stock problem*, Operations Research, **9(6)**, pp. 849–859.

[33] Gilmore PC & Gomory RE, 1963, *A linear programming approach to the cutting-stock problem — Part II*, Operations Research, **11(6)**, pp. 863–888.

[34] Gilmore PC & Gomory RE, 1965, *Multistage cutting stock problems of two and more dimensions*, Operations Research, **13(1)**, pp. 94–120.

[35] Golan I, 1981, *Performance bounds for orthogonal oriented two-dimensional packing algorithms*, SIAM Journal on Computing, **10(3)**, pp. 571–582.

[36] Hifi M, 1998, *Exact algorithms for the guillotine strip cutting/packing problem*, Computers and Operations Research, **25(11)**, pp. 925–940.

[37] Hifi M, 1999, *The strip cutting/packing problem: Incremental substrip algorithms-based heuristics*, Pesquisa Operacional, **19(2)**, pp. 169–188.

[38] Hifi M, 2003, *Library of instances*, [Online], [Cited on August 25th, 2009], Available from `ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/OR-Benchmark.html`

[39] Hinxman AI, 1980, *The trim-loss and assortment problems: A survey*, European Journal of Operational Research, **5(1)**, pp. 8–18.

[40] Hopper E, 2000, *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods*, PhD Dissertation, University of Wales, Cardiff.

[41] Hopper E & Turton BCH, 2001, *An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem*, European Journal of Operational Research, **128(1)**, pp. 34–57.

[42] Hopper E & Turton BCH, 2001, *A review of the application of meta-heuristic algorithms to 2D strip packing problems*, Artificial Intelligence Review, **16(4)**, pp. 257–300.

[43] Hopper E & Turton BCH, 2002, *Problem generators for rectangular packing problems*, Studia Informatica Universalis, **2(1)**, pp. 123–136.

[44] Jakobs S, 1996, *On genetic algorithms for the packing of polygons*, European Journal of Operational Research, **88(1)**, pp. 165–181.

[45] Johnson DS, 1974, *Fast algorithms for bin packing*, Journal of Computer and System Sciences, **8(3)**, pp. 272–314.

[46] Kang J & Park S, 2003, *Algorithms for the variable sized bin packing problem*, European Journal of Operational Research, **147(2)**, pp. 365–372.

[47] Kantorovich LV, 1960, *Mathematical methods for organising planning production* (translated from a report in Russian, dated 1939), Management Science, **6(4)**, pp. 366–422.

[48] Kenmochi M, Imamichi T, Nonobe K, Yagiura M & Nagamochi H, 2009, *Exact algorithms for the two-dimensional strip packing problem with and without rotations*, European Journal of Operational Research, **198(1)**, pp. 73–83.

[49] Lodi A, 1999, *Algorithms for two-dimensional bin packing and assignment problems*, PhD Dissertation, Università di Bologna, Bologna.

[50] Lodi A, Martello S & Monaci M, 2002, *Two-dimensional packing problems: A survey*, European Journal of Operational Research, **141(2)**, pp. 241–252.

[51] Lodi A, Martello S & Vigo D, 1998, *Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem*, pp. 125–139 in Voß S, Martello S, Osman IH & Roucairol C (Eds), *Meta-heuristics: Advances and trends in local search paradigms for optimization*, Kluwer Academic Publishers, Boston (MA).

[52] 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(4)**, pp. 345–357.

[53] Lodi A, Martello S & Vigo D, 2002, *Recent advances on two-dimensional bin packing problems*, Discrete Applied Mathematics, **123(1)**, pp. 379–396.

[54] MacLeod B, Moll R, Girkar M & Hanifi N, 1993, *An algorithm for the 2D guillotine cutting stock problem*, European Journal of Operational Research, **68(3)**, pp. 400–412.

[55] Martello S, Monaci M & Vigo D, 2003, *An exact approach to the strip-packing problem*, INFORMS Journal on Computing, **15(3)**, pp. 310–319.

[56] Martello S & Vigo D, 1998, *Exact solution of the two-dimensional finite bin packing problem*, Management Science, **44(3)**, pp. 388–399.

[57] Ntene N & Van Vuuren JH, 2009, *A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem*, Discrete Optimization, **6(2)**, pp. 174–188.

[58] Ortmann FG, 2010, *Heuristics for offline rectangular packing problems*, PhD Dissertation, Stellenbosch University, Stellenbosch.

[59] Ortmann FG, Ntene N & Van Vuuren JH, 2010, *New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems*, European Journal of Operational Research, **203(2)**, pp. 306–315.

[60] Pisinger D & Sigurd M, 2005, *The two-dimensional bin packing problem with variable bin sizes and costs*, Discrete Optimization, **2(2)**, pp. 154–167.

[61] Ratanapan K & Dagli CH, 1998, *An object-based evolutionary algorithm: The nesting solution*, pp. 581–586 in *Proceedings of the International Conference on Evolutionary Computation*, IEEE, Piscataway (NJ).

[62] Scheithauer G, Riehme J, Rietz J & Belov G, 2006, *Test instances & results*, [Online], [Cited August 25th, 2009], Available from `http://www.math.tu-dresden.de/~capad/`

[63] Sleator DDKDB, 1980, *A 2.5 times optimal algorithm for packing in two dimensions*, Information Processing Letters, **10(1)**, pp. 37–40.

[64] Sweeney PE & Paternoster ER, 1992, *Cutting and packing problems: A categorised, application-oriented research bibliography*, Journal of the Operational Research Society, **43(7)**, pp. 691–706.

[65] Tukey JW, 1977, *Exploratory data analysis*, Addison–Wesley Publishing Company, Manila.

[66] Van Vuuren JH & Ortmann FG, 2009, *Benchmark instances for the 2D rectangular strip packing problem*, [Cited August 25th, 2009], Available from `http://www.vuuren.co.za/` → *Benchmarks*

[67] Wang PY, 1983, *Two algorithms for constrained two-dimensional cutting stock problems*, Operations Research, **31(3)**, pp. 573–586.

[68] Wang PY & Valenzuela CL, 2001, *Data set generation for rectangular placement problems*, European Journal of Operational Research, **134(2)**, pp. 378–391.

[69] Wäscher G, Haußner H & Schumann H, 2007, *An improved typology of cutting and packing problems*, European Journal of Operational Research, **183(3)**, pp. 1109–1130.

[70] Yanasse HH, Zinober ASI & Harris RG, 1991, *Two-dimensional cutting stock with multiple stock sizes*, Journal of the Operational Research Society, **42(8)**, pp. 673–683.