J. Opl Res. Soc. Vol. 38, No. 5, pp. 423–429, 1987
Printed in Great Britain.

0160-5682/87 $3.00 + 0.00

Two-Dimensional Finite Bin-Packing Algorithms

J. O. BERKEY and P. Y. WANG
Department of Computer Science, George Mason University, Fairfax, Virginia, USA

Given a set of rectangular pieces, the two-dimensional bin-packing problem is to place the pieces into an open-ended bin of infinite height such that the height of the resulting packing is minimized. In this paper we analyse the performance of two-dimensional bin-packing heuristics when applied to the special case of packing into finite bins. We develop new bin-packing heuristics by adapting the bottom-left packing method and the next-fit, first-fit and best-fit level-oriented packing heuristics to the finite-bin case. We present our implementation of these algorithms, and compare them to other finite-bin heuristics. Our computational results indicate that the heuristics presented in this paper are suitable for practical use, and behave in a manner which reflects the proven worst-case bounds for the two-dimensional open-ended bin-packing problem.

Key words: bin-packing, heuristics

INTRODUCTION

In a classical two-dimensional bin-packing problem, a finite set of rectangles is to be packed into a single open-ended bin so as to minimize the total height of the packing. Several well-known algorithms which solve this problem have been studied.[1] One variant of this packing problem that often occurs in operational research is the following two-dimensional problem.

A finite set of rectangular pieces is to be packed into bins or stock sheets having finite dimensions, and the number of bins used for the packing must be minimized. This is the case, for example, in the lumber industry when a customer order is to be cut from plywood panels and the trim waste must be minimized.

In this paper we present several heuristic algorithms which modify and extend some well-known bin-packing algorithms to the case where finite bins are to be packed. The results obtained by applying our algorithms to various test problems are also discussed. In these test problems, the sizes of the pieces to be packed and the size of the bins were varied. We analyse the resulting packings by relating them to known bin-packing performance measures, and we comment on any special considerations which were necessary when implementing the algorithms. Since the bin-packing problem is NP-hard, computational results of the kind presented in this paper are an important way to evaluate the heuristic solutions that must necessarily be used in industrial environments.

The heuristic algorithms we have developed are described as follows:

1. Finite next-fit (FNF)—a level-oriented next-fit heuristic that packs directly into finite bins.
2. Finite first-fit (FFF)—a level-oriented first-fit heuristic that packs directly into finite bins.
3. Finite best-strip (FBS)—a best-fit heuristic that is similar to hybrid first-fit,[2] a two-step algorithm in which the pieces are packed into an infinite height-strip and the strip is divided into blocks, which are then packed into finite bins.
4. Finite bottom-left (FBL)—a bottom-left approach[3] which packs directly into finite bins.

In addition, we also implemented and tested the hybrid first-fit (HFF) algorithm.

Each of the above algorithms is discussed in the remaining sections of this paper. We outline the basic approach used in each technique, and illustrate the various packings obtained by applying the methods to a sample packing problem.

To analyse the performance of these heuristics, we recorded the number of bins used when packing sets of randomly generated rectangles or 'pieces'. The pieces must be presorted by non-increasing height for the level-oriented algorithms, and presorted by non-increasing width for the finite bottom-left heuristic. This is a requirement of the algorithms upon which our methods

are based.[4,5] Our analysis is independent of the $0(n \log n)$ time needed for the preprocessing of the data.

THE ALGORITHMS

Each of the algorithms outlined below packs a set $P = \{P_1, P_2, \ldots, P_n\}$ of rectangular pieces into a minimum number of rectangular bins $B_1, B_2, \ldots, B_m$ of finite dimensions. Throughout the discussion, the following terms will be used to describe the state of the algorithm as it packs the list.

In *level-oriented packings*, the rectangular pieces are placed with their bottom edges on a level and as far left as possible in the bin. The first *level* in a bin is the bottom of the bin. Each subsequent level of the bin is defined by the height of the tallest piece placed on the previous level. Further, the *height of a level* is defined to be its distance from the bottom of the bin. The first level has height 0.

In many of the algorithms, a *current-bin list* is maintained. This is a list of all the possible bins into which the next piece can be packed. Finally, we shall refer to an *optimal packing* of a set of pieces as the minimum number of bins, N^*, into which the pieces could be packed.

FINITE NEXT-FIT

In the traditional next-fit packing technique, only one bin is held in the current-bin list. When the next piece to be packed will not fit into the only bin in this list (the current bin), that bin is removed from the list and a new empty bin is added to it. The newly created bin becomes the current bin. The finite next-fit algorithm uses this method and the level-oriented packing approach. A piece is placed on the current level of the current bin if it will fit. If not, an attempt is made to start a new level in the current bin. If the height of the piece is less than or equal to the height of the bin minus the height of the current level, the current level is 'closed off', and the piece is packed in the new level created. When the next piece to be packed cannot be packed on a new level in the current bin, the current-bin list is updated to contain a new bin.

The finite next-fit algorithm has the best time complexity of the algorithms tested. It runs in $0(n)$ time, and could be implemented in 0(1) space. The pieces, presorted by decreasing height, are packed sequentially, and the storage for each previous bin could be released when a new bin is started. The algorithm utilizes an 'on-line' approach, which is frequently employed in industrial settings.

In Figure 1, the finite next-fit packing of the following list P of piece widths and heights is shown. Let $P = \{P_i\}$, with $n = 7$ and $P_1 = (5 \times 6)$, $P_2 = (8 \times 5)$, $P_3 = (5 \times 4)$, $P_4 = (4 \times 3)$, $P_5 = (9 \times 3)$, $P_6 = (1 \times 2)$ and $P_7 = (4 \times 1)$. The dimensions of the bin used for packing this list are 10×10.

FINITE FIRST-FIT

In a first-fit packing, all of the bins that have been created are retained on the current-bin list. Finite first-fit packs a piece by beginning at the first bin created and checking each of its levels

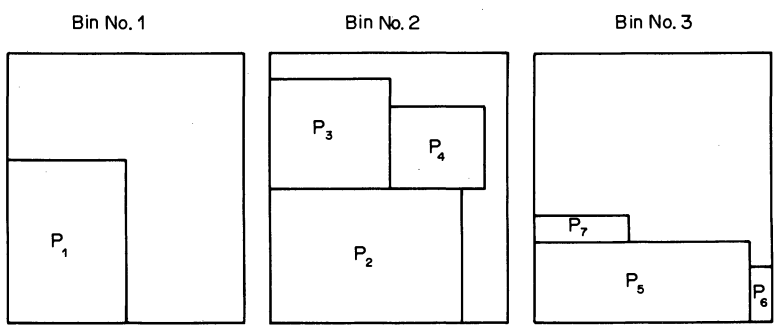

FIG. 1. *A finite next-fit packing.*

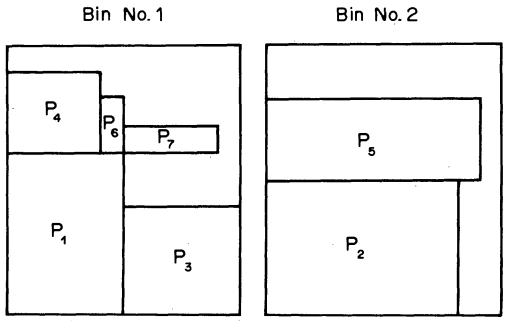

FIG. 2. *A finite first-fit packing.*

in turn from the bottom of the bin upwards to see if the piece can be placed there. If it cannot be packed on any level in this bin, the next bin is searched or a new bin is created if there are no more bins in the current-bin list. Each piece will be packed into the lowest level of the first bin in which it fits. The current-bin list is maintained by a linked-list data structure so that a sequential search can be easily implemented.

In Figure 2, the packing produced by applying FFF to the list P is shown.

The time complexity for FFF is $0(n^2)$. In the worst case, all levels in every bin will have to be searched each time a piece is packed. As expected, our test results showed that FFF became very slow as the number of bins increased. For large packings it required up to 152 times as much CPU time as did FNF, HFF or FBS.

FINITE BEST-STRIP

The finite best-strip algorithm first packs the list P into levels in an open-ended strip, using a best-fit packing approach to select the level for packing the next piece. The strip is stratified into blocks, where the width of each block is the width of the strip, and the height of the block corresponds to the height of the level resulting from the initial packing. The resulting blocks are packed into finite bins, again using a best-fit heuristic. In the finite best-strip approach, the level (bin) that is selected for the next piece (block) is the level (bin) whose resulting packing has the minimum remaining horizontal area. In this way, an attempt is made to make maximum use of the remaining space. The hybrid first-fit algorithm uses a similar, but simpler approach, which is examined in a later section. The HFF method was also implemented and tested against its proven bound,[2] and served as a means of comparison for FBS.

The blocks created in the first phase of the procedure are maintained in a binary search-tree data structure. This yields an $0(n \log n)$ time complexity for FBS. The space complexity of the heuristic is $0(n)$.

For small packings, FBS ran rather slowly in comparison with FNF, HFF and even FFF, but as the number of pieces to be packed increased, the efficiency of the binary search-tree data structure became apparent, and the time complexity was much better than FFF, HFF and FBL.

Figure 3 illustrates the finite best-strip packing of the list P.

HYBRID FIRST-FIT

The hybrid first-fit[2] algorithm uses the level-oriented first-fit decreasing height (FFDH) heuristic to pack the pieces into an open-ended strip whose width is equal to the width of the finite bins. Blocks, corresponding to the levels of the strip, are formed as before when the pieces are packed. The block's width is the width of the strip, and its height is the height of the first piece placed on that level. A piece is packed into the lowest level (block) in which it will fit. When all the pieces have been packed in the strip, the blocks are packed into finite bins, using a one-dimensional first-fit packing heuristic. Recall that this is similar to our finite best-strip algorithm. Because we implemented the HFF algorithm with a linked-list data structure, its time complexity was $0(n^2)$.

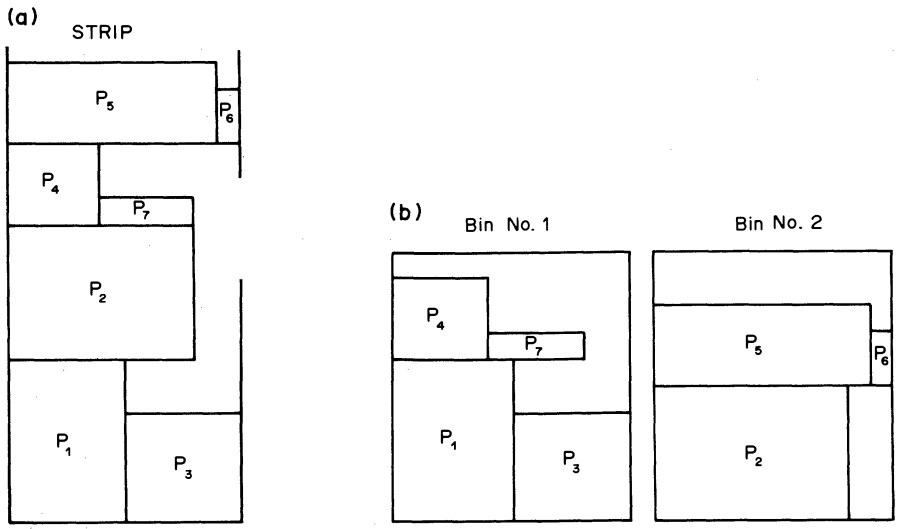

FIG. 3. (a) *A best-fit strip packing.* (b) *A finite best-strip packing.*

The space complexity of HFF is $0(n)$. A more detailed description of the algorithm may be found in the referenced paper.

The hybrid first-fit packing of the list P is shown in Figure 4.

FINITE BOTTOM-LEFT

The last two algorithms which we developed and tested were based on the bottom-left packing approach. The finite bottom-left algorithm searches each bin in the current-bin list for the lowest and left-most position at which the next piece in P can be packed. This is referred to as bottom-left stability. If there is no bin in which the piece can be placed, a new bin is created.

Each bin is represented as a linked list of vertices. These vertices define the boundary of the bin. As the bin is packed, the boundary is updated to represent the remaining space in the bin. An area of the bin that becomes isolated as a result of packing the pieces against all edges of its boundary is called a *subhole* of the bin. A search for a possible packing location in a bin requires that all subholes of that bin be searched.

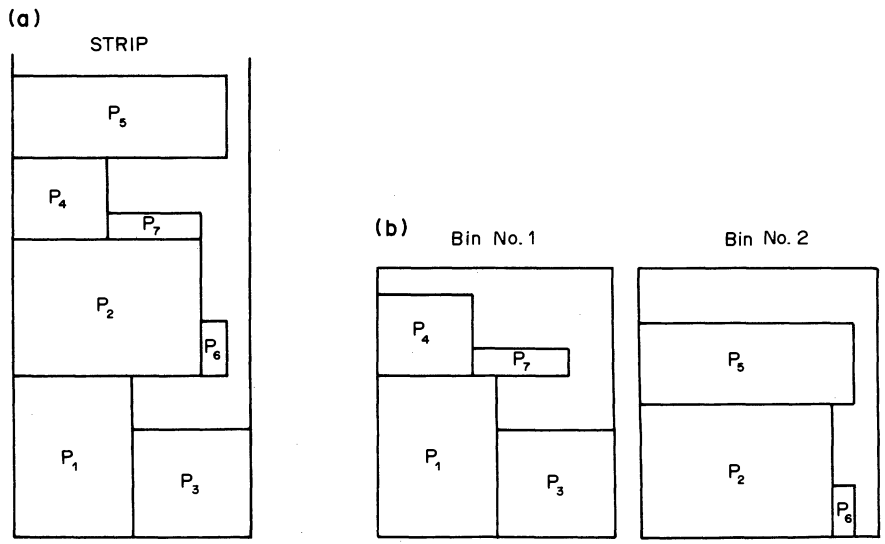

FIG. 4. (a) *A first-strip packing.* (b) *A hybrid first-fit packing.*

Chazelle has showed that his bottom-left bin-packing implementation, as applied to the open-ended packing problem, does an exhaustive search of all subholes in $0(n^2)$ time and uses $0(n)$ space. In his implementation, a 'false' top is added to the open-ended bin to complete the boundary. In essence this creates a finite bin. We adapted this approach and added the capability of creating additional finite bins. A single linked-list was used to maintain the pointers to all of the active subholes in all of the bins.

In our quantitative study of this algorithm, we found that the space used to maintain the boundary of each subhole created by the FBL packing was excessive. It was not unusual to find 30 or more subholes in each bin when packing 100 randomly generated pieces into 100×100 bins. A practical consideration of this algorithm would be to limit the number of active subholes that are maintained. For the purpose of this research, we searched all the subholes in each bin for every piece packed. A Pascal version of the FBL algorithm was run on a CDC Cyber 730. Using the allotted user-memory of 376,500 words, we were only able to pack data sets of limited size. For pieces P_i with maximum height or width of 100, the largest data set we could pack was about 90 pieces. When we packed lists containing pieces whose maximum dimension was 10, the number of pieces we could pack increased to about 175. This can be explained by the fact that as the number of possible piece dimensions decreases, the boundaries of the subholes created are less irregular and can be maintained with fewer memory nodes.

In an effort to increase the number of pieces that could be packed by the bottom-left algorithm for our computational study, we designed a next bottom-left heuristic, which also packed in the typical bottom-left manner. In this case, however, the generation of a new bin for packing meant that all the subholes from the previous bin were discarded. Thus only one bin was active at a time. We were then able to pack larger sets of pieces.

A finite bottom-left packing of the list P is shown in Figure 5, and the next bottom-left packing for the list appears in Figure 6.

QUANTITATIVE RESULTS

We studied the performance of the above algorithms by varying both the bin size and the size of the pieces to be packed. The widths and heights of the test pieces were randomly generated

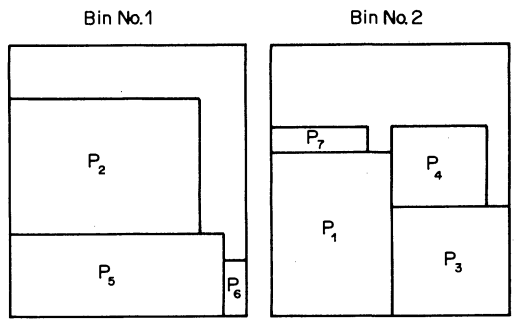

FIG. 5. *A finite bottom-left packing.*

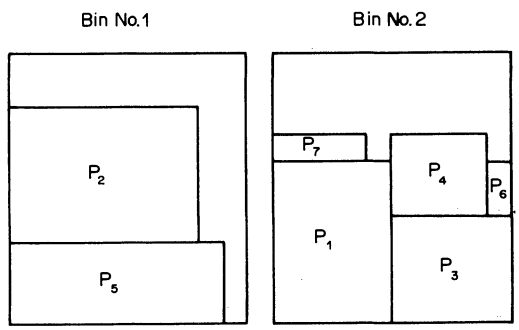

FIG. 6. *A next-bottom-left packing.*

TABLE 1

	Average number of bins packed					
	$1 \leq$ height ≤ 10 $1 \leq$ width ≤ 10		$1 \leq$ height ≤ 35 $1 \leq$ width ≤ 35		$1 \leq$ height ≤ 100 $1 \leq$ width ≤ 100	
Algorithm	Bin size 10×10	Bin size 30×30	Bin size 40×40	Bin size 100×100	Bin size 100×100	Bin size 300×300
Finite next-fit	43	4	40	5	41	4
Finite first-fit	32	4	27	5	29	3
Finite best-strip	32	4	26	5	29	3
Finite bottom-left	36	5	29	6	32	4
Next-bottom-left	45	5	40	6	41	4
Hybrid first-fit	32	4	26	5	29	3
Lower bound for N^*	28	4	23	4	24	3

$n = 100$.

integral numbers which were taken from the same uniform distribution. The total area of the pieces in a list P was also calculated, and served as a bound on the optimal number of bins needed for packing P. The quantitative results for bin sizes 10×10, 40×40 and 100×100 are presented in Table 1.

Observation 1

Finite first-fit, finite best-strip and hybrid first-fit used the least number of bins. Finite bottom-left resulted in a somewhat less efficient packing, and the two next-fit heuristics, finite next-fit and next bottom-left, used the greatest number of bins. As we packed data sets with more pieces ($n = 1000$ and $n = 5000$), finite best-strip became about 10% more efficient than finite first-fit and hybrid first-fit.

Observation 2

As n increased, the average number of pieces packed per bin appeared to approach a constant for each of the packing algorithms, thus establishing their stability. These constants are shown in Table 2. On the average, an upper bound on the number of pieces that could be packed per bin was estimated to be 3.65.

Observation 3

Several worst-case bounds have been proved for the two-dimensional open-ended bin-packing problem.[2,3,4,6] In particular, the following asymptotic bounds are known. The height of the packing obtained by the next-fit level-oriented algorithm for open-ended bins is bounded by $2 * \text{OPT}$, where OPT is the minimum height of all possible packings of the given list P. The height of the first-fit level-oriented packing of a list is bounded by $1.7 * \text{OPT}$. For the bottom-left packing of a list preordered by decreasing widths, an asymptotic bound of $2 * \text{OPT}$ is known on the height of the BL packing. To our knowledge, the only published result for the finite-bin case is an asymptotic upper bound of the hybrid first-fit algorithm: $2.125\,N^* + 5$.[2] The average ratios of (the number of bins packed)/N^* which we observed are shown in Table 2.

In our study of finite packing algorithms, we were interested in obtaining information about the expected behaviour of our heuristics. In all of our test packings for the finite-bin problem, we observed a behaviour that was reflected by the worst case bounds for the open-ended packing problem.

TABLE 2

Algorithm	Average number of pieces packed per bin	Average number of bins packed N^*
Finite next-fit	2.23	1.700
Finite first-fit	3.45	1.064
Finite best-strip	3.53	1.040
Finite bottom-left	3.03	1.333
Next-bottom-left	2.25	1.742
Hybrid first-fit	3.48	1.050

Observation 4

Bengtsson[7] has proposed an heuristic solution to the problem of packing sheets with unsorted rectangular pieces. He uses an iterative procedure that rearranges the initial packing until no further improvement in packing efficiency is possible. Because only fully packed sheets (bins) are included in the final packing in this algorithm, a number of pieces may not be packed. This contrasts with our algorithms, which always pack all of the pieces. Our algorithms compared favourably with Bengtsson's when they were run on the sets of test pieces presented in the referenced paper. As in Bengtsson's results, we found that the overall bin utilization of our finite bin heuristics increased consistently as the number of pieces to be packed increased.

Observation 5

For each test case, the time complexity of the algorithm under consideration was verified computationally. Finite next-fit ran the fastest in all of our tests. In comparison, finite first-fit and hybrid first-fit were very slow for large n, which is typical of a first-fit packing. Finite best-strip was slower than finite first-fit and hybrid first-fit for small n, but as n became large, the difference between the speed of the binary tree-search used in finite best-strip and the sequential search used in finite first-fit and hybrid first-fit became significant, and finite best-strip was much more time-efficient. The bottom-left packings, finite bottom-left and next-bottom-left, consumed the most time but appeared to be within the polynomial bound claimed by Chazelle.

CONCLUSIONS

Finite next-fit, finite first-fit, hybrid first-fit and finite best-strip extend the classical bin-packing heuristics to the case of finite bins in a straightforward manner. The best-fit heuristic is superior in both packing efficiency and time consumption for large data sets. Bottom-left packing methods, while intuitively auspicious, do not perform as well as other heuristics.

Next-fit algorithms are appropriate for interactive use in industrial settings where the input pieces are presorted. Here an interaction would be possible between the user and a program executing a next-fit heuristic. This is an advantage in that it allows for the graphical display of the complete packing of each bin as it progresses. In an application such as stock cutting, either finite next-fit or next-bottom-left would allow for the cutting of a stock piece and the simultaneous determination of the packing to be used for the next stock piece.

We have shown that bin-packing heuristics for the case of finite bins are easily implemented. The heuristics presented in this paper are suitable for practical use, and behave in a manner which reflects the proven worst-case bounds for the two-dimensional open-ended bin-packing problem.

REFERENCES

1. E. G. Coffman Jr, M. R. Garey and D. S. Johnson (1984) Approximation algorithms for bin-packing—an updated survey. In *Algorithm Design for Computer System Design* (G. Ausiello, M. Lucertini and P. Serafini, Eds). Springer, New York.
2. F. R. K. Chung, M. R. Garey and D. J. Johnson (1982) On packing two-dimensional bins. *SIAM J. Alg. Disc. Meth.* **3,** 66–76.
3. B. S. Baker, E. G. Coffman Jr and R. L. Rivest (1980) Orthogonal packings in two dimensions. *SIAM J. Comput.* **9,** 846–855.
4. B. Chazelle (1983) The bottom-left bin-packing heuristic: an efficient implementation. *IEEE Trans. Comput.* **C-32,** 697–707.
5. E. G. Coffman Jr, M. R. Garey, D. S. Johnson and R. E. Tarjan (1980) Performance bounds for level-oriented two dimensional packing algorithms. *SIAM J. Comput.* **9,** 808–826.
6. D. S. Johnson, A. Demars, J. D. Ullman, M. R. Garey and R. L. Graham (1974) Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.* **3,** 295–325.
7. B. E. Bengtsson (1982) Packing rectangular pieces—a heuristic approach. *Comput. J.* **25,** 353–357.