A 2.5 TIMES OPTIMAL ALGORITHM FOR PACKING IN TWO DIMENSIONS

Daniel D.K.D.B. SLEATOR
Computer Science Department, Stanford University, Stanford, CA 94305, U.S.A.

Received 6 January 1979; revised version received 13 November 1979

Bin packing, two dimensional bin packing

1. Introduction

We consider the following two dimensional bin packing problem: Given a set of rectangular pieces, find a way to pack the pieces into a bin of width 1, bounded below but not above, so as to minimize the height to which the pieces fill the bin. The pieces are not allowed to rotate (not even 90 degrees) or overlap. This problem was first considered in [1], which includes several applications of the problem. The most interesting one is the allocation of jobs to a shared storage multiprocessor system. In that application each rectangle represents a job whose memory and time requirements are known, and the horizontal dimension of the bin is memory, and the vertical dimension is time. A good way of packing the rectangles into the bin represents a way to run the jobs so that they all get done quickly.

It is easy to see that the problem of determining whether a given set of pieces will fit into a given height is NP-hard, because we can reduce any instance of the partition problem (problem SP11 in [3]) to an instance of this problem. Therefore we have tried to find algorithms that are fast and that give packings whose height is within a small constant factor of the height of an optimal packing. The same approach is taken in [1]. The authors of [1] give an algorithm that packs in $3H_{opt}$ where H_{opt} is the height of an optimal packing. In this note we present an algorithm that packs in $2.5H_{opt}$ and is easier to implement than the one given in [1].

2. The algorithm

Here is the packing algorithm, as illustrated in Fig. 1:

(1) Stack all the pieces of width greater than $\frac{1}{2}$ on top of one another in the bottom of the bin. Call the height to which they reach h_0, and the total area of these pieces A_0. All subsequent packing will occur above h_0.

(2) Sort the remaining pieces in order of decreasing height. The pieces will be placed into the bin in this order. Let h_1 be the height of the tallest of these pieces.

(3) Now place pieces from left to right with their bottoms along the line of height h_0 with the first piece adjacent to the left wall of the bin, and each subsequent piece adjacent to the one just packed. Continue until there is no more room or there are no pieces left to pack. Draw a vertical line that cuts the bin into two equal halves. Let d_1 be the height above h_0 of the highest point of any piece in the right half (or partially in the right half). Let A_1 be the area of

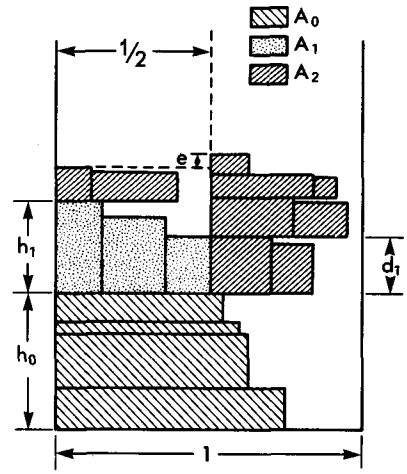

Fig. 1. The anatomy of a packing.

the intersection of the left half and those pieces packed in Step 3. Let A_2 be the area of the pieces not in A_0 or A_1, including those pieces that have yet to be packed. Thus the area of all the pieces is partitioned into 3 disjoints areas A_0, A_1, and A_2. Although A_0, A_1, and A_2 are numbers it will be convenient to be able to describe a piece by saying which of A_0, A_1, or A_2 it is 'in'. Notice that there may be one piece which is partly in A_1 and partly in A_2.

(4) Draw two horizontal line segments of length $\frac{1}{2}$, one across the left half and one across the right half of the bin as low as possible so that the segments do not divide any piece. Call these segments the left and right half baselines. All subsequent packing in the left and right halves will occur above these lines. Now, choose the half whose baseline is lower. Into this half bin pack one horizontal row of pieces from left to right along the baseline until there are no more pieces, or the next piece is too wide to fit in this row. Recall that we have already packed all the pieces of width greater than $\frac{1}{2}$, so this process must pack at least one piece.

(5) Repeat Step 4 until there are no more pieces.

3. The height bound

Theorem. Let H_{alg} be the height of a packing given by the algorithm above, H_{opt} be the height of an optimal packing and h_{tall} be the height of the tallest piece. Then we have

$$H_{alg} \leqslant 2H_{opt} + \tfrac{1}{2}h_{tall} .$$

Proof. Define S to be the sum of the heights of all of the rows of pieces in A_2. Let e be the difference in height between the left and right halves in the final packing. Note that $2h_0 + h_1 + S + e$ is exactly twice H_{alg}, the height of the packing.

Since over half of the area of the bin below height h_0 is filled with pieces we have

$$\tfrac{1}{2}h_0 \leqslant A_0 .$$

We claim (and will prove below) that

$$S \leqslant 4A_2 + d_1 .$$

Combining the statements above we get

$$2H_{alg} = 2h_0 + h_1 + S + e \leqslant 4A_0 + h_1 + 4A_2 + d_1 + e .$$

Since the pieces were packed in order of decreasing height, all the pieces in A_1 have height at least d_1. Either d_1 is not zero, in which case the widths of all pieces in A_1 add to exactly $\frac{1}{2}$, or d_1 is zero, in which case we must have run out of pieces in Step 3 before any piece crossed the center line. In either case we know that

$$\tfrac{1}{2}d_1 \leqslant A_1 .$$

The two statements above imply that

$$2H_{alg} \leqslant 4(A_0 + A_1 + A_2) + h_1 + e - d_1 .$$

Since an optimal packing has height H_{opt} we know that the area of all the pieces cannot exceed H_{opt}, i.e., $A_0 + A_1 + A_2 \leqslant H_{opt}$. Substituting this into the above expression we find

$$2H_{alg} \leqslant 4H_{opt} + h_1 + e - d_1 .$$

If the height of the right column has never exceeded that of the left, then the height of the entire packing is $h_0 + h_1$. But $h_0 \leqslant H_{opt}$, since no two pieces in A_0 can be packed on the same level in any packing. We also have $h_1 \leqslant H_{opt}$, so in this case the height of the packing is bounded by $2H_{opt}$.

If the height of the right column has ever exceeded that of the left, then e is bounded by the height of the tallest row packed in A_2, i.e., $e \leqslant d_1$. Combining this with the last line above and the fact that $h_1 \leqslant h_{tall}$ we get the desired result:

$$H_{alg} \leqslant 2H_{opt} + \tfrac{1}{2}h_{tall} .$$

We will now prove the claim made above. The pieces of A_2 are to be packed into a bin of height S and width $\frac{1}{2}$ in order of decreasing height, "row by row" in the manner described above. We will show that $S \leqslant 4A_2 + d_1$. This result is very similar to one found in [4].

Let the first row packed be called p_1, the next one packed p_2 ... and the last one packed p_n. Let the height of p_i be d_i. (This is consistent with our definition of d_1.) Let R_i be the d_i by $\frac{1}{2}$ rectangle into which the pieces of p_i are packed.

We now partition R_i into three disjoint parts a_i, b_i and c_i, as illustrated in Fig. 2:

a_i = that portion of R_i covered by pieces.

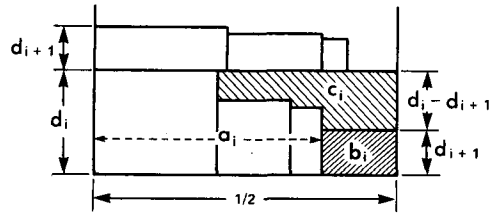

Fig. 2. How we divide up R_i.

b_i = { a rectangular subregion of R_i bounded on the left by the right side of the rightmost piece placed in R_i, on the right by the right wall of R_i, and on the bottom by the bottom of R_i, and on the top by the horizontal line of height d_{i+1} above the bottom of R_i. If $i = n$, then b_i is empty.

c_i = that part of R_i not contained in a_i or b_i.

Again we will be sloppy and allow the use of symbols a_i, b_i and c_i to represent both a region in the bin and the area of that region.

Clearly $A_2 = \cup_i a_i$. Let $B = \cup_i b_i$ and $C = \cup_i c_i$.

Since A_2, B, and C are disjoint and cover the $\frac{1}{2}$ by S rectangle we have

$$\tfrac{1}{2}S = A_2 + B + C .$$

Now, $b_i \leqslant a_{i+1}$ because the first piece in a_{i+1} is wider and just as high as b_i. Therefore $B \leqslant A_2$.

Consider C. The height of c_i is $d_i - d_{i+1}$ for $i \leqslant n-1$ and the height of c_n is d_n. So

$$\sum_i \text{height of } c_i = d_n + \sum_{1 \leqslant i \leqslant n-1} (d_i - d_{i+1}) = d_1 .$$

Thus the pieces of C can be placed into a $\frac{1}{2}$ by d_1 rectangle without overlapping, so $C \leqslant \frac{1}{2}d_1$.

Combining the above three relations we get:

$$\tfrac{1}{2}S \leqslant A_2 + A_2 + \tfrac{1}{2}d_1 ,$$

which implies the desired result:

$$S \leqslant 4A_2 + d_1 .$$

Since $h_{tall} \leqslant H_{opt}$ we can summarize the above result as $H_{alg} \leqslant 2.5H_{opt}$.

The worst case bound of $2.5H_{opt}$ given above is in fact tight. In other words there exists a sequence of sets of pieces $\{S_k\}$ such that the ratio of the height of the packing given by the algorithm to the height of an optimal packing goes to 2.5 as $k \to \infty$. We define one such sequence of sets.

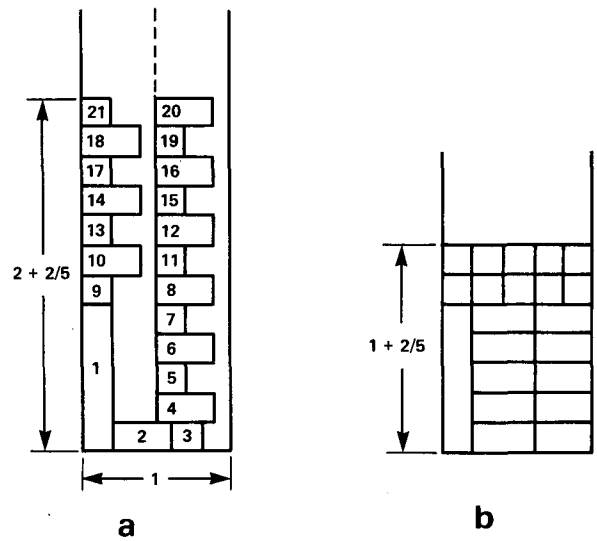

Fig. 3. (a) How our algorithm might pack the pieces of S_5; (b) an optimal packing of S_5.

Let S_k consist of the following pieces:

S_k = { 1 piece 1 unit high and 1/k units wide, 2k pieces 1/k units high and $\frac{1}{2} - \frac{1}{2}(1/k)$ units wide, 2k pieces 1/k units high and 1/k units wide.

Fig. 3 shows how these pieces would be packed by our algorithm and also how they are packed optimally.

The packing given by our algorithm will have height $1 + (1/k)\lceil \frac{1}{2}(3k-1) \rceil$ units. (Assuming that pieces of equal height are sorted in an order which makes the algorithm do badly. This assumption is reasonable since it is possible to force equal height pieces to come out in any order we want simply by adding or subtracting an infinitesimal amount to their heights, and have the resulting packing still have essentially the same height.) But an optimal packing will have height $1 + 2/k$ units. Clearly the ratio of the height achieved by our algorithm to the height of the optimal packing goes to 2.5 as $k \to \infty$.

The time used by the algorithm to place each piece is constant, therefore the dominant term in the time complexity is the sorting step. Thus the algorithm is O(n log n) time if there are n pieces.

4. Generalizations

The idea behind the algorithm which makes it work is this: It is possible to pack pieces wider than $\frac{1}{2}$ in an efficient way separately from the others. Once we have eliminated the pieces wider than $\frac{1}{2}$, we can pack in the left and right half of the bin separately and prevent the height difference among pieces from wasting space all the way across the bin. It is conceivable that we could improve the packing algorithm by extending this idea, and packing all pieces wider than 1/n first, and then partition the bin into n sub-bins each 1/n units in width. The problem with this approach is that it doesn't seem possible to pack the pieces wider than 1/n efficiently enough to give an improved algorithm for any $n > 2$.

However, if we assume that all rectangles have width not exceeding 1/n, then we can use an algorithm analogous to the one given above: After packing one row of pieces across the bin, divide the upper part of the bin into n parts and proceed to pack rows of pieces into the lowest part. This modified algorithm will have a better worst case bound than the one given in this paper. In fact the height of a packing given by this modified algorithm will not exceed $2H_{opt} + h_{tall}/n$.

It has been suggested that the algorithm might work better if it attempted to put pieces into gaps below the level that is currently being filled. This method might significantly improve the average behavior of the algorithm, but we have been unable to show that any such method would improve the worst case behavior. The reason this is hard appears to be that even though the area left empty by our algorithm may be large, it may be broken up into a large number of small regions into which it is difficult to fit things.

5. Asymptotic behavior

An alternative method of measuring the performance of a packing algorithm is to consider the ratio of the height of the packing generated to the height of an optimal packing for sets of pieces in which the height of the tallest piece is small compared to the height of an optimal packing. The asymptotic ratio of our algorithm is 2, since $2H_{opt} + \frac{1}{2}h_{tall} \to 2H_{opt}$ as $h_{tall} \to 0$.

One of the algorithms given in [2] asymptotically packs in height $1.5H_{opt}$, which is considerably better than our algorithm. Unfortunately this algorithm has a worst case that is worse than our algorithm. However if for a given set of pieces we run both the algorithm in [2] and our algorithm, and choose the packing with lowest height, we in effect have a single algorithm with good asymptotic behavior and good worst case behavior.

References

[1] B.S. Baker, E.G. Coffman Jr. and R.L. Rivest, Orthogonal packings in two dimensions, SIAM J. Comput., to appear.

[2] E.G. Coffman Jr., M.R. Garey, D.S. Johnson and R.E. Tarjan, Performance bounds for level-oriented two-dimensional packing algorithms, SIAM J. Comput., to appear.

[3] M.R. Garey and D.S. Johnson, A Guide to the Theory of NP-Completeness (Freeman, San Francisco, CA).

[4] I. Golan, unpublished manuscript.