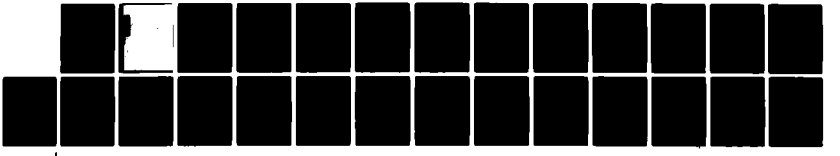
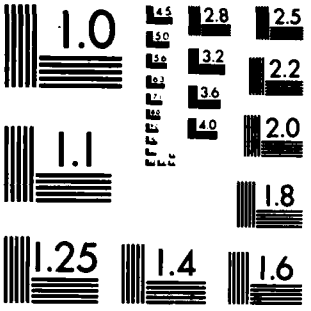
AD-A123 984 LOWER BOUNDS FOR ON-LINE TWO-DIMENSIONAL PACKING ALGORITHMS(U) ILLINOIS UNIV AT URBANA APPLIED COMPUTATION THEORY GROUP D J BROWN ET AL. JUL 80 1/1

UNCLASSIFIED ACT-25 DAAG29-78-C-0016 F/G 12/1 NL



END
DATE FILMED
-83
DTIC



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A123984	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) LOWER BOUNDS FOR ON-LINE TWO-DIMENSIONAL PACKING ALGORITHMS	5. TYPE OF REPORT & PERIOD COVERED Technical Report
	6. PERFORMING ORG. REPORT NUMBER (ACT-25); R-888 UILU-ENG 80-2220 ✓
7. AUTHOR(s) Donna J. Brown, Brenda S. Baker, Howard P. Katseff	8. CONTRACT OR GRANT NUMBER(s) DAAG-29-78-C-0016
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois at Urbana-Champaign Urbana, Illinois 61801	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program	12. REPORT DATE July 1980
	13. NUMBER OF PAGES 22
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

bin packing, scheduling, algorithms, rectangles, on-line, lower bounds

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Many problems, such as cutting stock problems and the scheduling of tasks with a shared resource, can be viewed as two-dimensional bin packing problems. Using the two-dimensional packing model of Baker, Coffman, and Rivest, a finite list L of rectangles is to be packed into a rectangular bin of finite width but infinite height, so as to minimize the total height used. An algorithm which packs the list in the order given without looking ahead or moving pieces already packed is called an on-line algorithm. Since the problem of finding an optimal packing is

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (cont.)

NP-hard, previous work has been directed at finding approximation algorithms. Most of the approximation algorithms which have been studied are on-line except that they require the list to have been previously sorted by height or width. This paper examines lower bounds for the worst-case performance of on-line algorithms for both non-preordered lists and for lists preordered by increasing or decreasing height or width.

UILU-ENG 80-2220

LOWER BOUNDS FOR ON-LINE TWO-DIMENSIONAL PACKING ALGORITHMS

by

Donna J. Brown, Brenda S. Baker, and Howard P. Katseff

This author's work was supported by the Joint Services Electronics Program (U.S. Army, U.S. Navy and U.S. Air Force) under Contract DAAG-29-78-C-0016.

Lower Bounds for On-Line Two-Dimensional Packing Algorithms

*Donna J. Brown**

Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61801

Brenda S. Baker

Bell Laboratories
Murray Hill, New Jersey 07974

Howard P. Katseff

Bell Laboratories
Holmdel, New Jersey 07733

ABSTRACT

Many problems, such as cutting stock problems and the scheduling of tasks with a shared resource, can be viewed as two-dimensional bin packing problems. Using the two-dimensional packing model of Baker, Coffman, and Rivest, a finite list L of rectangles is to be packed into a rectangular bin of finite width but infinite height, so as to minimize the total height used. An algorithm which packs the list in the order given without looking ahead or moving pieces already packed is called an *on-line* algorithm. Since the problem of finding an optimal packing is NP-hard, previous work has been directed at finding approximation algorithms. Most of the approximation algorithms which have been studied are on-line except that they require the list to have been previously sorted by height or width. This paper examines lower bounds for the worst-case performance of on-line algorithms for both non-preordered lists and for lists preordered by increasing or decreasing height or width.

Introduction

Two-dimensional packing problems arise in many contexts. For example, cutting stock problems involving rolls or sheets of material and the scheduling of tasks with a shared resource can be viewed as two-dimensional packing problems. In the model proposed by Baker, Coffman and Rivest [2], a finite list L of rectangles is to be packed into a rectangular bin of finite width but infinite height, in such a way as to minimize the maximum height used. The packed rectangles cannot overlap, nor can they be rotated. Since the problem of finding an optimal packing is NP-hard [2], several approximation algorithms have been studied [1,2,3,6,7,10]. Figure 1 illustrates possible packings of a list of five pieces, with sizes as specified. Notice that, for a computer scheduling application, the horizontal dimension represents core while the vertical dimension represents time.

A two-dimensional bin packing algorithm is said to be *on-line* if, given a list of rectangles $L = (p_1, \ldots, p_n)$, it

- packs the rectangles in the order given by L,
- packs each rectangle p_i without looking ahead at any p_j $(j > i)$, and
- nevers moves a rectangle already packed.

*This author's work was supported by the Joint Services Electronics Program (U.S. Army, U.S. Navy and U.S. Air Force) under Contract DAAG-29-78-C-0016.

Most of the algorithms which have been studied are designed to pack lists already sorted by decreasing or increasing height or width. Thus, some simple preordering is done before the actual on-line packing. For instance, the Split algorithm [7] is an on-line algorithm which requires that the list be ordered by decreasing width. Next-Fit and First-Fit Decreasing Height [6] are on-line algorithms which require that the list be first sorted by decreasing height. On the other hand, the Next-Fit and First-Fit Shelf algorithms [3] are on-line and do not require that the list be preordered.

This paper examines lower bounds for the performance of on-line packing algorithms for both non-preordered lists and for lists preordered by decreasing or increasing height or width. As a special case, lower bounds for packing squares in order of increasing or decreasing size are also investigated.

Absolute Lower Bounds

For any algorithm A, let $A(L)$ denote the height of the packing of L produced by A and let $\mathrm{OPT}(L)$ denote the height used by an optimal packing. As a measure of *absolute* worst-case performance, we study the ratio $\frac{A(L)}{\mathrm{OPT}(L)}$; i.e., we consider bounds of the form $A(L) \le \alpha OPT(L)$, where α is some constant.

A piece (rectangle) p_i is said to have *size* (x_i, y_i) if p_i has width x_i and height y_i. Pieces p_i and p_j are said to be *colateral* at height h from the bottom of the bin in a packing if a horizontal line at height h intersects both p_i and p_j. For instance, in Figure 1b pieces p_1, p_2, and p_5 are colateral at height 5. If $L_1 = (p_{i_1}, \ldots, p_{i_s})$ and $L_2 = (p_{j_1}, \ldots, p_{j_m})$ are two lists, then we write L_1L_2 to denote their *concatenation* $(p_{i_1}, \ldots, p_{i_s}, p_{j_1}, \ldots, p_{j_m})$.

When presented with lists which are not preordered appropriately, most of the algorithms which have been studied either are undefined or have performance which can be arbitrarily bad relative to an optimal packing, i.e. for any α, there is a list L such that $A(L) > \alpha OPT(L)$. The two exceptions are the Next-Fit and First-Fit Shelf algorithms of Baker and Schwartz [3]. Of these, the First-Fit Shelf algorithm performs better, with a worst-case performance of at most 6.99 $\mathrm{OPT}(L)$. We give here a corresponding lower bound of about 2; every on-line algorithm packs some list so badly that it comes arbitrarily close to doubling the height of an optimal packing. Thus, even for unpreordered lists, there may be room for substantial improvement over the performance of the First-Fit Shelf algorithm.

THEOREM 1: Let A be an on-line algorithm. For any $\delta > 0$, there is a list L for which

$$A(L) > (2-\delta)\ \mathrm{OPT}(L).$$

Proof: Let δ and ϵ be fixed, with $0 < \epsilon < \delta/4$, and suppose that the bin has width 3. We obtain a contradiction by assuming that, for every list L, $A(L) \le (2-\delta)\ \mathrm{OPT}(L)$. In particular, we construct a list $L = L_1L_2L_3L_4L_5$ (with each list L_i consisting of a single piece p_i) for which it cannot be the case that

$$A(L_1, \ldots, L_k) \le (2-\delta)\ \mathrm{OPT}(L_1, \ldots, L_k)$$

for each k, $1 \le k \le 5$. In other words,

$$\max\left\{\frac{A(L_1)}{\mathrm{OPT}(L_1)}, \frac{A(L_1L_2)}{\mathrm{OPT}(L_1L_2)}, \frac{A(L_1L_2L_3)}{\mathrm{OPT}(L_1L_2L_3)}, \frac{A(L_1L_2L_3L_4)}{\mathrm{OPT}(L_1L_2L_3L_4)}, \frac{A(L)}{\mathrm{OPT}(L)}\right\} > 2-\delta.$$

Let L_1 consist of a piece p_1 of size $(1, 1)$. The algorithm A packs p_1 at some height h_1, as indicated in Figure 2a. Let the next piece, p_2, have size $(3, h_1 + \epsilon)$. Clearly, p_2 must be placed above p_1. Let h_2 denote the difference in height between the top of p_1 and the bottom of p_2. If the next piece, p_3, has size $(1, 1 + h_1 + h_2 + \epsilon)$, then p_3 is too tall to fit below p_2 and so A must place p_3 at some height h_3 above the top of p_2.

Assume that, for $1 \le k \le 3$,

$$A(L_1 \ldots L_k) < 2\ \mathrm{OPT}(L_1 \ldots L_k).$$

Letting y_i denote the height of piece p_i, we have:

$$\frac{A(L_1)}{\mathrm{OPT}(L_1)} = \frac{h_1+y_1}{y_1} < 2 \Rightarrow h_1 < y_1 = 1$$

$$\frac{A(L_1L_2)}{\mathrm{OPT}(L_1L_2)} = \frac{h_1+y_1+h_2+y_2}{y_1+y_2} < 2$$

$$\Rightarrow h_1+h_2 < y_1+y_2 = 1+h_1+\epsilon$$

$$\Rightarrow h_2 < 1+\epsilon$$

$$\frac{A(L_1L_2L_3)}{\mathrm{OPT}(L_1L_2L_3)} = \frac{h_1+y_1+h_2+y_2+h_3+y_3}{y_2+y_3} < 2$$

$$\Rightarrow h_1+h_2+h_3+y_1 < y_2+y_3 = (h_1+\epsilon)+(1+h_1+h_2+\epsilon)$$

$$\Rightarrow h_3 < h_1+2\epsilon < 1+2\epsilon$$

So if piece p_4 has size $(3, 1+2\epsilon)$, then $y_4 > \max\{h_1, h_2, h_3\}$, and p_4 will be placed with its bottom at some height h_4 above the top of p_3. A piece p_5 of size $(1, 1+h_1+h_2+h_3+h_4+2\epsilon)$ would then have to be placed above p_4, giving:

$$\begin{aligned}
&A(L_1L_2L_3L_4L_5) \\
&\quad \geq h_1+y_1+h_2+y_2+h_3+y_3+h_4+y_4+y_5 \\
&\quad = h_1+1+h_2+y_2+h_3+(1+h_1+h_2+\epsilon)+h_4+y_4+y_5 \\
&\quad = y_2+y_4+y_5+(h_1+\epsilon)+(1+2\epsilon)+(1+h_1+h_2+h_3+h_4+2\epsilon)+h_2-4\epsilon \\
&\quad = 2[y_2+y_4+y_5]+h_2-4\epsilon.
\end{aligned}$$

Noting that $\mathrm{OPT}(L) = y_2+y_4+y_5 > 1$ (see Figure 2b), we have

$$\begin{aligned}
A(L) &\geq 2\ \mathrm{OPT}(L)+h_2-4\epsilon \\
&> 2\ \mathrm{OPT}(L)-8 \\
&> (2-8)\ \mathrm{OPT}(L)
\end{aligned}$$

thereby proving the theorem. □

The Bottom-Leftmost algorithm [2] and the Split algorithm [7] both have a worst case performance of 3 $\mathrm{OPT}(L)$ for lists ordered by decreasing width. The following result shows that every on-line algorithm which packs pieces ordered by decreasing width has a worst case bound of at least $(1+\frac{\sqrt{6}}{3})\mathrm{OPT}(L)$.*

THEOREM 2: For any on-line algorithm A, there is a list L ordered by decreasing width such that

$$A(L) \geq (1+\frac{\sqrt{6}}{3})\ \mathrm{OPT}(L) > 1.81\ \mathrm{OPT}(L).$$

Proof: Let ϵ be fixed, $0<\epsilon<\frac{1}{24}$. Consider the list of rectangles $L = L_1L_2L_3L_4$ where

L_1 consists of 8 pieces of size $(\frac{3}{2}-3\epsilon, 1)$,

L_2 consists of 6 pieces of size $(1+\epsilon, -1+\sqrt{6}\ )$,

*This is an improvement over Storer's result of approximately 1.78 [11].

L_3 consists of 3 pieces of size (1,2),

L_4 consists of 3 pieces of size (1,3).

Note that L is ordered by decreasing width.

Figures 3a,b.c,d give optimal packings of lists L_1, L_1L_2, $L_1L_2L_3$, and $L_1L_2L_3L_4 = L$, respectively, for a bin of width 12. Therefore,

$$\mathrm{OPT}(L_1) = 1,$$

$$\mathrm{OPT}(L_1L_2) = 2,$$

$$\mathrm{OPT}(L_1L_2L_3) = \sqrt{6},$$

$$\mathrm{OPT}(L_1L_2L_3L_4) = \mathrm{OPT}(L) = 3.$$

It is shown that any algorithm which packs each of the lists L_1, L_1L_2, $L_1L_2L_3$ in such a way that

$$A(L_1) < (1+\frac{\sqrt{6}}{3})\ \mathrm{OPT}(L_1),$$

$$A(L_1L_2) < (1+\frac{\sqrt{6}}{3})\ \mathrm{OPT}(L_1L_2),$$

$$A(L_1L_2L_3) < (1+\frac{\sqrt{6}}{3})\ \mathrm{OPT}(L_1L_2L_3),$$

will necessarily lead to a packing of list $L_1L_2L_3L_4 = L$ for which $A(L) \geq (1+\frac{\sqrt{6}}{3})\ \mathrm{OPT}(L)$. In other words, we assume that

$$\max\left\{\frac{A(L_1)}{\mathrm{OPT}(L_1)}, \frac{A(L_1L_2)}{\mathrm{OPT}(L_1L_2)}, \frac{A(L_1L_2L_3)}{\mathrm{OPT}(L_1L_2L_3)}, \frac{A(L)}{\mathrm{OPT}(L)}\right\} < 1+\frac{\sqrt{6}}{3}$$

and then obtain a contradiction, thereby proving the theorem.

We must first pack L_1. Since $\mathrm{OPT}(L_1)=1$, it is clear that the bottom of every L_1 piece must be strictly below height 1, or else we would violate our assumption that $A(L_1) < (1+\frac{\sqrt{6}}{3})\ \mathrm{OPT}(L_1)$. Thus, for sufficiently small $\delta_1>0$, all L_1 pieces are colateral in the bin at height $1-\delta_1$ (see Figure 4a). Since the bin is filled to a width of $12-24\epsilon$ at height $1-\delta_1$, the total remaining unfilled space is only 24ϵ. None of the remaining pieces of L will be able to fit below height 1.

Now each piece of L_2 must be placed with its bottom at or above height 1 and will therefore reach a height of at least $\sqrt{6}$ in the bin. As above, in order to avoid violating $A(L_1L_2) < (1+\frac{\sqrt{6}}{3})\mathrm{OPT}(L_1L_2)$, the L_2 pieces are colateral at height $\sqrt{6}-\delta_2$ in the bin, for any sufficiently small δ_2 (see Figure 4b). In particular, it is not possible to pack two L_2 pieces on top of each other, because this would give

$$\frac{A(L_1L_2)}{\mathrm{OPT}(L_1L_2)} \geq \frac{1+2(-1+\sqrt{6})}{2} > 1+\frac{\sqrt{6}}{3}.$$

Similarly, no L_3 piece can be placed on top of an L_2 piece because we would have

$$\frac{A(L_1L_2L_3)}{\mathrm{OPT}(L_1L_2L_3)} \geq \frac{1+(-1+\sqrt{6})+2}{\sqrt{6}} = 1+\frac{\sqrt{6}}{3}.$$

So at height $\sqrt{6}-\delta_2$, the three L_3 pieces are colateral with the L_2 pieces, filling the bin to a width of $9+6\epsilon$. Thus, it is not possible to pack all of the L_4 pieces below height $\sqrt{6}$. At least one of them must be above an L_2 or an L_3 piece, which gives

$$\frac{A(L)}{\mathrm{OPT}(L)} \geq \frac{1+(-1+\sqrt{6})+3}{3} = 1+\frac{\sqrt{6}}{3}.$$

This contradicts our assumption, proving the desired result. □

The First-Fit Decreasing Height algorithm [6] does somewhat better than the above algorithms which use decreasing width; its performance is at most 2.7 OPT(L). The following theorem gives a corresponding lower bound of $\frac{5}{3}$.

THEOREM 3: For any on-line algorithm A, there is a list L ordered by decreasing height such that

$$A(L) \geq \frac{5}{3}\ \mathrm{OPT}(L).$$

Proof: Consider a bin of width 6. For $0<\epsilon<\frac{2}{11}$, let the list $L=L_1L_2L_3$ be defined as follows:

L_1 consists of 6 pieces of size $(1-2\epsilon, 1)$,

L_2 consists of 6 pieces of size $(2+\epsilon, 1)$,

L_3 consists of 6 pieces of size $(3+\epsilon, 1)$.

Observing Figure 5a, it is easy to verify that

$$\mathrm{OPT}(L_1) = 1,$$

$$\mathrm{OPT}(L_1L_2) = 3,$$

$$\mathrm{OPT}(L_1L_2L_3) = 6.$$

Assume that

$$\max\left\{\frac{A(L_1)}{\mathrm{OPT}(L_1)}, \frac{A(L_1L_2)}{\mathrm{OPT}(L_1L_2)}, \frac{A(L)}{\mathrm{OPT}(L)}\right\} < \frac{5}{3}.$$

Then, in order to avoid violating this assumption, the *bottom* of every L_1 piece must be strictly below height 1; i.e., for sufficiently small $\delta>0$, all L_1 pieces are colateral at height $1-\delta$. Since no L_2 piece will fit below height 1, and yet all the L_2 pieces must pack below height 5 (since $\mathrm{OPT}(L_1L_2) = 3$), there is not enough height for four L_2 pieces to fit above each other. Also, no three pieces of L_2 or L_3 can be colateral. Thus, there is no way to leave space for an L_3 piece below height 4, and an algorithm A can do no better than to pack L_2 as shown in Figure 5b. But this forces all the pieces in L_3 to be at or above height 4 and, since no two L_3 pieces can be colateral, $A(L) \geq 10 = \frac{5}{3}\ \mathrm{OPT}(L)$. □

Some algorithms perform better for squares than for rectangles. The Bottom-Leftmost algorithm [2] and the Next-Fit and First-Fit Decreasing Height algorithms [6] pack squares in order of decreasing size with performance no worse than 2 OPT(L). This performance is not bad in light of the following theorem.

THEOREM 4: Let A be any on-line algorithm. For any $\delta>0$, there is a list L of squares ordered by decreasing size such that

$$A(L) > (1.5-\delta)\ \mathrm{OPT}(L).$$

Proof: This proof uses a list L consisting of two squares of size $\frac{1}{3}+\epsilon$ and four squares of size $\frac{1}{3}-\epsilon$, where $0<\epsilon<\frac{2}{3}\delta$. An optimal packing into a bin of width 1, illustrated in Figure 6a, has height $\frac{2}{3}$. For L ordered by decreasing size, the two $\frac{1}{3}+\epsilon$ squares must be packed first. In order to achieve $A(L) < (1.5-\delta)\ \mathrm{OPT}(L)$, they would have to be colateral at height $\frac{1}{3}+\epsilon-\delta_1$, for sufficiently small δ_1. Since this fills the bin to a width of $\frac{2}{3}+2\epsilon$, there is not enough space left for a third piece at height $\frac{1}{3}+\epsilon-\delta_1$. Thus, all four of the $\frac{1}{3}-\epsilon$ squares must be placed with their bottoms at height at least $\frac{1}{3}+\epsilon$. Because no four of the squares can be colateral, the best any on-line algorithm can do

is to have $A(L) = 1-\epsilon$, as illustrated in Figure 6b. This gives

$$\frac{A(L)}{\mathrm{OPT}(L)} \geq \frac{1-\epsilon}{\frac{2}{3}} = \frac{3}{2}-\frac{3}{2}\epsilon > \frac{3}{2}-\delta. \qquad \square$$

Most of the algorithms thus far proposed have used lists ordered by decreasing width or height. An obvious alternative would be to pack pieces in order of increasing width or height. The lower bound in this case is somewhat higher than the other lower bounds presented here for preordered lists.

THEOREM 5: For any on-line algorithm A, there is a list L ordered by both increasing width and increasing height such that

$$A(L) \geq \frac{1+\sqrt{7}}{2}\,\mathrm{OPT}(L) > 1.82\,\mathrm{OPT}(L).$$

Proof: Let ϵ be fixed, $0<\epsilon<\frac{1}{4}$. For $k=\frac{4+2\sqrt{7}}{3}$, consider the list of pieces $L=L_1L_2L_3L_4$, where

L_1 consists of 4 pieces of size $(1-\epsilon, 1)$,

L_2 consists of 2 pieces of size $(1, \frac{k}{2})$,

L_3 consists of 1 piece of size $(1, k-1)$,

L_4 consists of 1 piece of size $(1+\epsilon, k)$.

An optimal packing of L into a bin of width 4 is illustrated in Figure 7a. Notice that

$$\mathrm{OPT}(L_1) = 1,$$

$$\mathrm{OPT}(L_1L_2) = 2,$$

$$\mathrm{OPT}(L_1L_2L_3) = \frac{k}{2}+1$$

$$\mathrm{OPT}(L) = k.$$

We shall show that the assumption

$$\max\left\{\frac{A(L_1)}{\mathrm{OPT}(L_1)}, \frac{A(L_1L_2)}{\mathrm{OPT}(L_1L_2)}, \frac{A(L_1L_2L_3)}{\mathrm{OPT}(L_1L_2L_3)}, \frac{A(L)}{\mathrm{OPT}(L)}\right\} < \frac{1+\sqrt{7}}{2}$$

leads to a contradiction.

Since $\mathrm{OPT}(L_1)=1$, all L_1 pieces must be colateral at height $1-\delta_1$ for sufficiently small δ_1. So at height $1-\delta_1$, the bin is filled to a width of $4-4\epsilon$, which forces all remaining pieces to have their bottoms at height at least 1 (see Figure 7b). Thus, the L_2 pieces must be colateral at height $1+\frac{k}{2}-\delta_2$, for sufficiently small δ_2; otherwise the above assumption would be violated, because the L_2 pieces would reach height $1+\frac{k}{2}+\frac{k}{2}$, and

$$\frac{A(L_1L_2)}{\mathrm{OPT}(L_1L_2)} \geq \frac{1+\frac{k}{2}+\frac{k}{2}}{2} = \frac{1+k}{2} > \frac{1+\sqrt{7}}{2}.$$

In fact the L_3 piece must also be colateral with the L_2 pieces at height $1+\frac{k}{2}-\delta_2$, or else

$$\frac{A(L_1L_2L_3)}{\mathrm{OPT}(L_1L_2L_3)} \geq \frac{1+\frac{k}{2}+(k-1)}{\frac{k}{2}+1} = \frac{3k}{k+2} = \frac{1+\sqrt{7}}{2}.$$

But having the L_2 and L_3 pieces all colateral at height $1+\frac{k}{2}-\delta_2$ means that there is not enough

width left to fit the L_4 piece also at this height. This forces

$$\frac{A(L)}{\mathrm{OPT}(L)} \geq \frac{1+\frac{k}{2}+k}{k} = \frac{3k+2}{2k} = \frac{1+\sqrt{7}}{2}.$$

So our assumption must be incorrect, which proves the desired result. □

Similarly, the lower bound for squares preordered by increasing size is higher than for squares preordered by decreasing size.

THEOREM 6: For any on-line algorithm A, and any $\delta>0$, there is a list L of squares ordered by increasing size such that

$$A(L) > (\tfrac{7}{4}-\delta)\ \mathrm{OPT}(L).$$

Proof: For fixed ϵ, $0<\epsilon<\min\{4\delta, \frac{1}{6}\}$, consider the list of squares $L = L_1L_2L_3$, where

L_1 consists of 7 squares of size $1-\epsilon$,

L_2 consists of 2 squares of size 2,

L_3 consists of 1 square of size 4.

Figure 9a illustrates an optimal packing of L into a bin of width $8-\epsilon$, and

$$\mathrm{OPT}(L_1) = 1-\epsilon,$$
$$\mathrm{OPT}(L_1L_2) = 2,$$
$$\mathrm{OPT}(L) = 4.$$

Once again, we prove that

$$\max\left\{\frac{A(L_1)}{\mathrm{OPT}(L_1)}, \frac{A(L_1L_2)}{\mathrm{OPT}(L_1L_2)}, \frac{A(L)}{\mathrm{OPT}(L)}\right\} > \frac{7}{4}-\delta$$

by assuming the contrary.

In order for $\frac{A(L_1)}{\mathrm{OPT}(L_1)} < \frac{7}{4}$, it must be the case that all L_1 pieces are colateral at height $1-\epsilon-\delta_1$, for sufficiently small δ_1. Thus each L_2 square must have its bottom at height at least $1-\epsilon$. For sufficiently small δ_2, the L_2 pieces must be colateral at height $3-\epsilon-\delta_2$, or else we would have

$$\frac{A(L_1L_2)}{\mathrm{OPT}(L_1L_2)} \geq \frac{(1-\epsilon)+2+2}{2} > \frac{7}{4}-\delta.$$

This means that the bin is filled to width 4 at height $3-\epsilon-\delta_2$ (see Figure 8b), and so the square of size 4 must be packed above an L_2 square, giving

$$\frac{A(L)}{\mathrm{OPT}(L)} \geq \frac{(3-\epsilon)+4}{4} = \frac{7-\epsilon}{4} > \frac{7}{4}-\delta. \qquad \square$$

Asymptotic Lower Bounds

The lower bounds cited above are all bounds for absolute worst-case performance. If H, α, and β are constants such that, for every list L with pieces of height at most H, $A(L)\leq\alpha\ \mathrm{OPT}(L)+\beta$, then α is called an *asymptotic* worst case bound. The absolute worst case bound seems to be a better measure of performance when the number of rectangles to be packed is small, whereas the asymptotic bound is a better measure when the number of rectangles is large.

In this section we shall need the following definition. If horizontal lines are drawn across the bin through the top and bottom of each piece, as illustrated in Figure 9, the region between two

successive horizontal lines is called a *slice*.

The results of Brown [4] and Liang [9] for one-dimensional bin packing can be interpreted in two dimensions to give the following result.

THEOREM 7: Any on-line algorithm which packs rectangles in order of increasing or decreasing height or increasing width has an asymptotic bound of at least 1.536.

The First-Fit Decreasing Height algorithm has an asymptotic worst-case bound of 1.7 [6], which is not much worse than 1.536. If the widest rectangle packed has width at most $1/m$ times the bin width, where m is a positive integer greater than 1, then its asymptotic worst-case bound is $(m+1)/m$ [6]. Thus, the narrower the pieces are with respect to the width of the bin, the better the algorithm performs. Note that for $m=2$, the asymptotic bound is 1.5, which is better than the lower bound of 1.536 for $m=1$.

For on-line algorithms without preordering, the asymptotic worst-case bound must also be at least 1.536. By picking a parameter appropriately, the asymptotic performance of the First-Fit Shelf algorithm can be made arbitrarily close to 1.7 [3], again not much worse than the lower bound of 1.536.

Coffman [5] showed that for on-line algorithms which pack squares in order of decreasing size, the asymptotic worst-case bound is at least 8/7. The Up-Down algorithm packs squares ordered by decreasing size with an asymptotic worst-case bound of 1.25 [1], not much worse than 8/7. The following theorem generalizes Coffman's result based on the maximum width of the squares.

THEOREM 8: Consider any on-line algorithm A and a bin of width 1. Let m be a positive integer. Let α and β be constants such that for every list L of squares of size at most $1/m$ ordered by decreasing size, $A(L) \le \alpha \mathrm{OPT}(L)+\beta$. If $m>1$, then $\alpha \ge \frac{m^3}{m^3-m+1}$. If $m=1$, then $\alpha \ge \frac{8}{7}$.

Proof: Let m be an integer greater than 1, and let n be a positive integer divisible by m. Consider the list $L=L_1L_2$, where L_1 contains n squares of size $\frac{1}{m+1}+m\epsilon$ and L_2 contains nm squares of size $\frac{1}{m+1}-\epsilon$. Note that $\mathrm{OPT}(L_1) = \frac{n}{m}(\frac{1}{m+1}+m\epsilon)$ and $\mathrm{OPT}(L_1L_2) < n(\frac{1}{m+1}+m\epsilon)$. (See Figure 10.)

L_1 is packed first. Let h_1 be the total height of slices containing exactly one segment of a square of L_1, and let h_2 be the total height of slices with at least two segments of squares of L_1 (see Figure 9). Then

$$\begin{aligned} A(L_1) &\ge h_1+h_2 \\ &\ge [n(\frac{1}{m+1}+m\epsilon)-mh_2]+h_2 \\ &= \frac{n}{m+1}+h_2(1-m)+mn\epsilon. \end{aligned}$$

Thus,

$$A(L_1) \le \alpha \mathrm{OPT}(L_1)+\beta$$

$$\frac{n}{m+1}+h_2(1-m)+mn\epsilon \le \frac{n}{m}(\frac{1}{m+1}+m\epsilon)\alpha+\beta$$

$$\frac{h_2}{n} \ge \frac{1}{m^2-1}-\frac{\alpha}{m^3-m}-\frac{\alpha\epsilon+\frac{\beta}{n}-m\epsilon}{m-1}.$$

A slice containing $k>1$ segments of squares of L_1 can contain at most $m-k$ segments of squares of L_2. Therefore, after packing L_1 and L_2 the total height of pieces packed in the slices composing h_1 and h_2 is at most $(m-1)h_1+mh_2$. Since the total height of squares in L_1 and L_2 is

$n(\frac{1}{m+1}+m\epsilon)+nm(\frac{1}{m+1}-\epsilon)$, and at most $m+1$ segments fit in a slice,

$$A(L_1L_2) \geq h_1+h_2+\frac{1}{m+1}[n(\frac{1}{m+1}+m\epsilon)+nm(\frac{1}{m+1}-\epsilon)-(m+1)h_1-mh_2]$$

$$= \frac{h_2}{m+1}+\frac{n}{m+1}$$

Thus,

$$A(L_1L_2) \leq \alpha OPT(L_1L_2)+\beta$$

$$\frac{h_2}{m+1}+\frac{n}{m+1} < \alpha[\frac{n}{m+1}+nm\epsilon]+\beta.$$

$$\alpha > \frac{\frac{h_2}{n}+1-\frac{\beta}{n}(m+1)}{1+m(m+1)\epsilon}$$

Substituting in for $\frac{h_2}{n}$,

$$\alpha > \frac{\frac{m^2}{m^2-1}-\frac{\alpha}{m^3-m}-\frac{\alpha\epsilon+\frac{\beta}{n}-m\epsilon}{m-1}-\frac{\beta}{n}(m+1)}{1+m(m+1)\epsilon}$$

$$\alpha > \frac{\frac{m^2}{m+1}-\frac{\beta}{n}m^2+m\epsilon}{\frac{m^3-m+1}{m(m+1)}+(m^3-m+1)\epsilon}$$

Choosing n sufficiently large,

$$\alpha > \frac{\frac{m^2}{m+1}}{\frac{m^3-m+1}{m(m+1)}}-O(\epsilon) = \frac{m^3}{m^3-m+1}-O(\epsilon)$$

Thus, for any $\delta>0$, a list of squares ordered by decreasing size, with each piece of size at most $\frac{1}{m}$, can be found such that for any on-line algorithm A, $A(L) \leq \alpha OPT(L)+\beta$ implies $\alpha > \frac{m^3}{m^3-m+1}-\delta$.

Note that for lists of squares of size at most 1, the asymptotic bound must be at least as large as for lists of squares of size at most 1/2. Therefore, for $m=1$, $\alpha \geq \frac{2^3}{2^3-2+1}=8/7$. □

The following result extends the lower bound of 1.536 for one-dimensional on-line algorithms [4,9] to two-dimensional algorithms which pack squares ordered by increasing size.

THEOREM 9: For any on-line algorithm, the asymptotic worst-case bound when packing squares ordered by increasing size is at least 1.536.

Proof: It is sufficient to make some straightforward modifications to the proof of Brown [4] that in the one-dimensional case, every on-line algorithm has an asymptotic bound greater than 1.536. Intuitively, wherever the one-dimensional proof requires summing over bins, this proof sums over slices of varying heights.

Define the sequence of integers $\{a_n\}$, for $n \geq 1$, by

$$a_1 = 2$$

$$a_{n+1} = 1+\prod_{i=1}^{n} a_i$$

Define

$$R_t = \frac{\sum_{i=1}^{t} \frac{i}{a_i - 1}}{\sum_{i=1}^{t} \frac{1}{a_i - 1}}. \tag{1}$$

Let $\delta > 0$ and for any positive integer $t \geq 3$, choose ϵ such that $0 < \epsilon < \min\{\frac{1}{a_t(a_t-1)(t-1)}, \frac{\delta}{t\, R_t a_{t-1}}\}$. Let r be a multiple of $(a_{t-1}-1)$. Consider the list of squares $L = L_1 L_2 \ldots L_t$, where L_1 consists of $(a_t - 1)r$ squares of size $p_1 = \frac{1}{a_t - 1} - (t-1)\epsilon$ and L_i, $2 \leq i \leq t$, consists of $r a_{t+1-i}$ squares of size $p_i = \frac{1}{a_{t+1-i}} + \epsilon$. Then, for $1 \leq k \leq t$,

$$OPT(L_1 L_2 \ldots L_k) \leq \frac{r}{a_{t+1-k} - 1} + r a_{t-1}\epsilon. \tag{2}$$

Let S be the set of all slices in the packing after $L_1 L_2 \ldots L_{t-1}$ has been packed. A slice $s \in S$ intersects $m_i(s)$ squares of size p_i. For $1 \leq i \leq t-1$, the set α_i is defined to consist of those slices in S which are at least half full and in which the smallest piece has size p_i. Similarly, we define β_i to be those slices in S which are less than half full and in which the smallest piece has size p_i. Let $h(\alpha_i)$ ($h(\beta_i)$) represent the total height of slices in α_i (β_i). For $1 \leq k \leq t-1$

$$A(L_1 L_2 \ldots L_k) = \sum_{i=1}^{k} (h(\alpha_i) + h(\beta_i)) \tag{3}$$

and

$$A(L_1 L_2 \ldots L_t) \geq r + \sum_{i=1}^{t} h(\alpha_i). \tag{4}$$

Assume that

$$\max_{1 \leq k \leq t} \{\frac{A(L_1 L_2 \ldots L_k)}{OPT(L_1 L_2 \ldots L_k)}\} < R_t - \delta. \tag{5}$$

It follows from (3), (4), and (5) that for $1 \leq k \leq t-1$,

$$OPT(L_1 L_2 \ldots L_k)(R_t - \delta) > \sum_{i=1}^{k} (h(\alpha_i) + h(\beta_i)) \tag{6}$$

and

$$OPT(L_1 L_2 \ldots L_t)(R_t - \delta) > r + \sum_{i=1}^{t} h(\alpha_i). \tag{7}$$

Because there are $r a_{t+1-i}$ squares of size p_i $(2 \leq i \leq t)$,

$$[\frac{1}{a_{t+1-i}} + \epsilon] r a_{t+1-i} = \sum_{s \in S} m_i(s) h(s) \tag{8}$$

where $h(s)$ represents the height of slice s. Summing inequalities (6) and (7) and using (2) and (8) gives

$$(R_t - \delta) \sum_{k=1}^{t} [\frac{r}{a_{t+1-k} - 1} + r a_{t-1}\epsilon] - \sum_{i=2}^{t} \frac{i}{a_i - 1} r a_{t+1-i} [\frac{1}{a_{t+1-i}} + \epsilon] \tag{9}$$

$$> \sum_{k=1}^{t-1} \sum_{i=1}^{k} [h(\alpha_i) + h(\beta_i)] + r + \sum_{i=1}^{t} h(\alpha_i) - \sum_{i=2}^{t} \frac{i}{a_i - 1} \sum_{s \in S} [m_{t+1-i}(s)] h(s).$$

By (1) and the choice of $\epsilon < \frac{\delta}{t R_t a_{t-1}}$, the left hand side is less than

$$R_t \sum_{k=1}^{t} \frac{r}{a_{t+1-k}-1} - \sum_{i=2}^{t} \frac{i}{a_i-1} r = r. \tag{10}$$

Combining (9) and (10)

$$\sum_{s \in S} h(s) \sum_{i=2}^{t} \frac{i}{a_i-1} m_{t+1-i}(s) > \sum_{j=1}^{t-1} [(j+1)h(\alpha_{t-j}) + jh(\beta_{t-j})]. \tag{11}$$

At this point, it is possible to apply Brown's original proof [4] which shows that (11) leads to a contradiction for $\epsilon < \frac{1}{a_t(a_t-1)(t-1)}$. We conclude that the assumption in (5) is incorrect, and the asymptotic bound is at least $R_t > 1.536$ for $t \geq 5$. □

Conclusions

The lower bounds show the extent to which it might be possible to improve on the current packing algorithms. They suggest that decreasing height and width are likely to yield better algorithms than increasing height or width.

In order to improve performance beyond the lower bounds presented here, it would be necessary either to violate the on-line conditions or to try other orderings of the lists. Sleator [8] describes an algorithm which achieves an absolute worst case bound of 2.5 by first packing pieces at least half as wide as the bin, and then packing the remaining pieces in order of decreasing height. Coffman, Garey, Johnson and Tarjan [6] have investigated the Split-Fit algorithm which has an asymptotic bound of 1.5. It groups pieces by width and then orders each group by decreasing height, and is not on-line since it requires moving rectangles around. More recently, Baker, Brown and Katseff [1] have proposed the Up-Down algorithm which groups pieces by width and orders each group by decreasing height or width, but is on-line and has an asymptotic bound of 1.25. By the result of Brown cited earlier, it is substantially better than any on-line algorithm which packs solely by increasing or decreasing height or by increasing width.

Note that the proofs of Theorems 3 and 4 use pieces which are all of the same height. Thus, these results also apply to algorithms for one-dimensional bin packing. Theorems 3 and 4 give absolute lower bounds of 5/3 and 3/2 for lists ordered by increasing size and decreasing size, respectively. Removing the epsilons from the heights in the proof of Theorem 8 gives an asymptotic one-dimensional lower bound of $\frac{m^3}{m^3-m+1}$ for pieces of size at most $1/m$ ordered by decreasing size.

References

[1] B.S. Baker, D.J. Brown, and H.P. Katseff, A 5/4 Algorithm for Two-Dimensional Bin Packing, in preparation.

[2] B.S. Baker, E.G. Coffman, Jr., and R.L. Rivest, Orthogonal Packings in Two Dimensions, *SIAM J. Comp*, to appear.

[3] B.S. Baker and J.S. Schwarz, Shelf Algorithms for Two-Dimensional Packing Problems, Proceedings of the 1979 Conference on Information Sciences and Systems, Baltimore (1979).

[4] D.J. Brown, On-Line One-Dimensional Bin Packing Algorithms, technical report ACT-19, Coordinated Science Laboratory, University of Illinois (1979).

[5] E.G. Coffman, Jr., personal communication (1978).

[6] E.G. Coffman, M.R. Garey, D.S. Johnson and R.E. Tarjan, Performance Bounds for Level-Oriented Two-Dimensional Packing Algorithms, to appear in *SIAM J. Comp*.

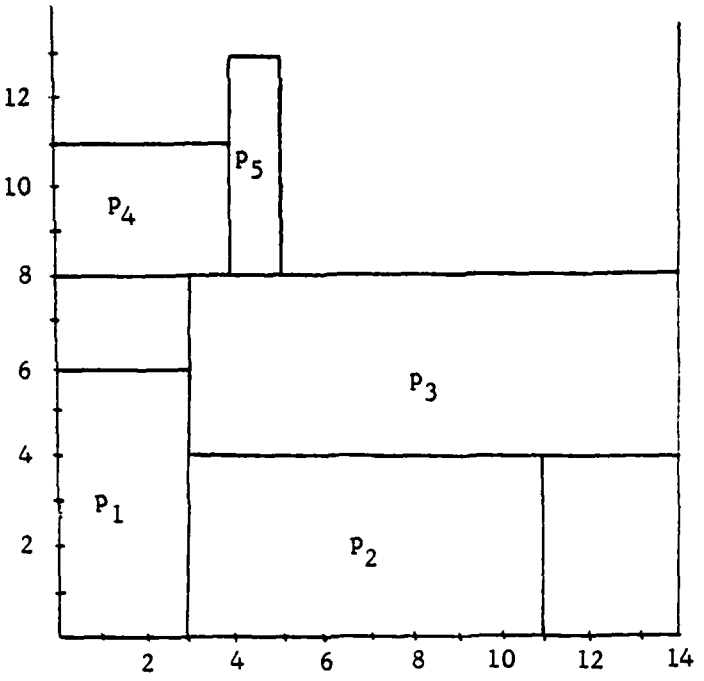
[7] I. Golan, Orthogonal Oriented Algorithms for Packing in Two Dimensions, draft (1978).

[8] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, and R.L. Graham, Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms, *SIAM J. Comp 3,4* (1974), 299-326.

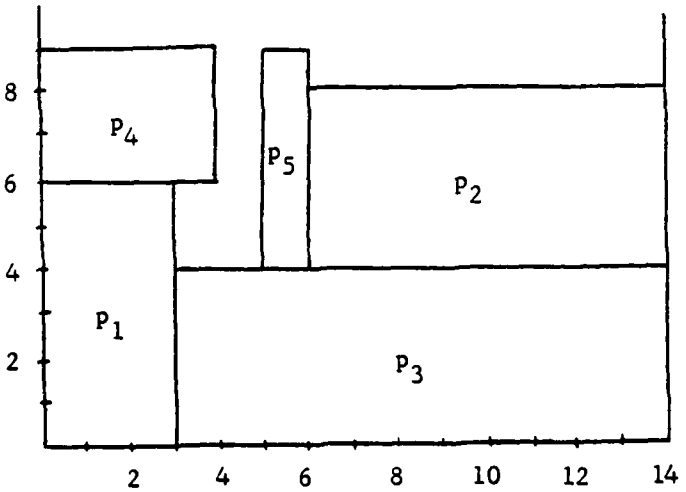
[9] F.M. Liang, A Lower Bound for On-line Bin Packing, *Info. Proc. Letters 10,2* (1980), 76-79.

[10] D.D.K.D.B. Sleator, A 2.5 Times Optimal Algorithm for Packing in Two Dimensions, *Info. Proc. Letters 10,1* (1980), 37-40.

[11] J.A. Storer, personal communication.

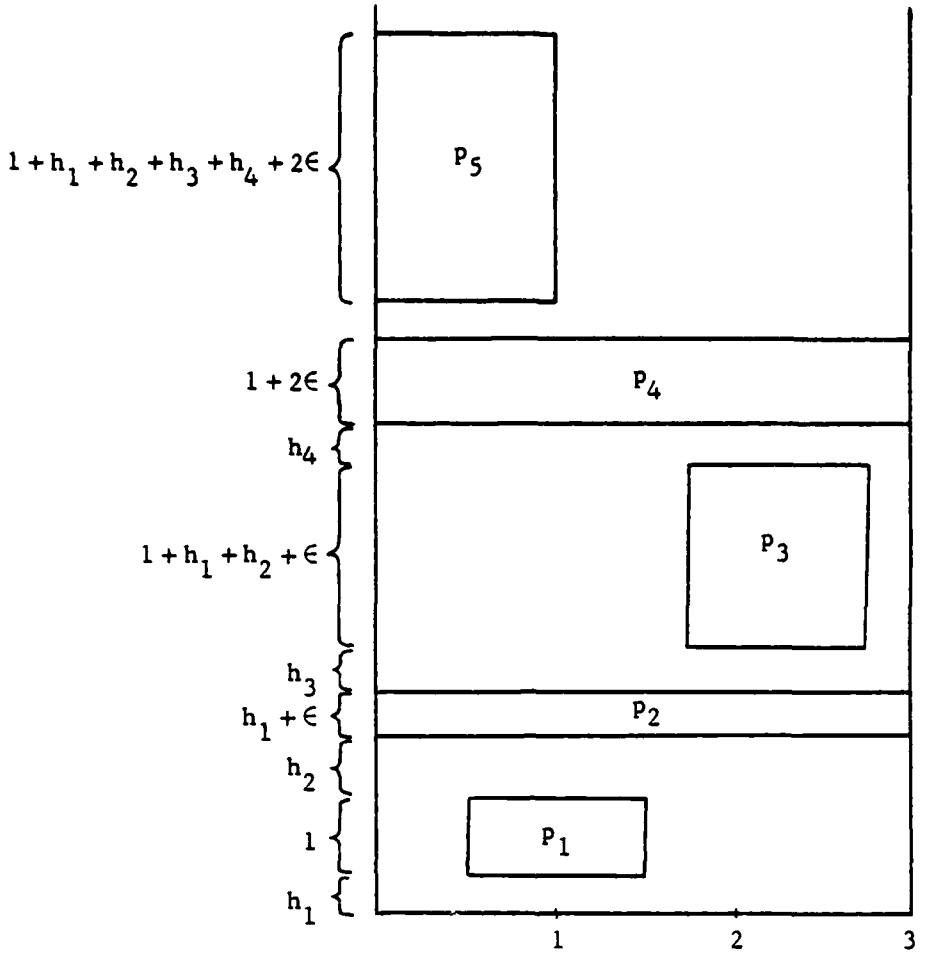


(a) One possible packing of list L.

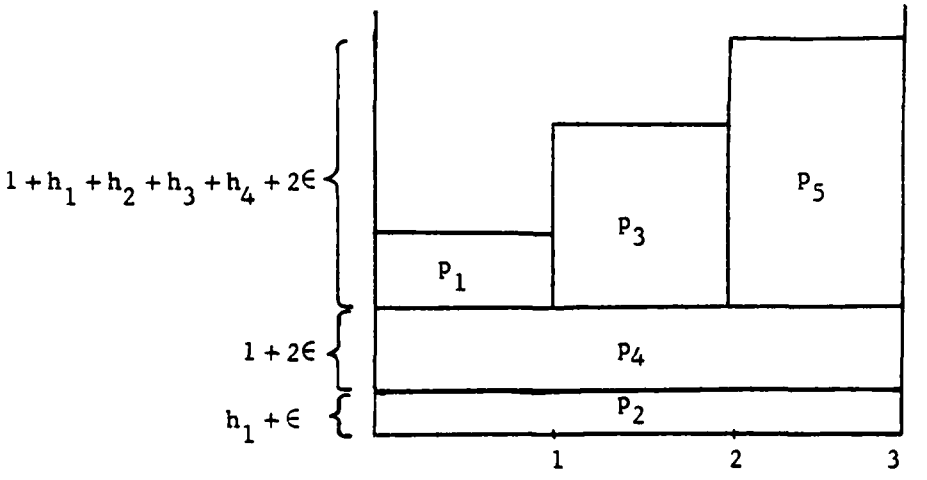


(b) An optimal packing of list L.

Figure 1. Packing list L = $(P_1, P_2, P_3, P_4, P_5)$
with width x_i: 3 8 11 4 1
and height y_i: 6 4 4 3 5

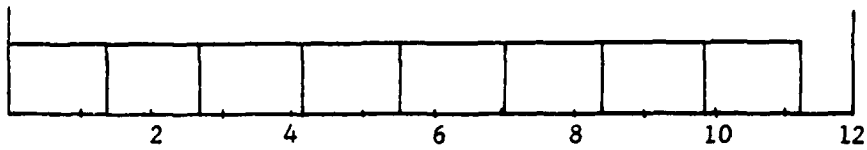


(a) A packing of L by an algorithm A.

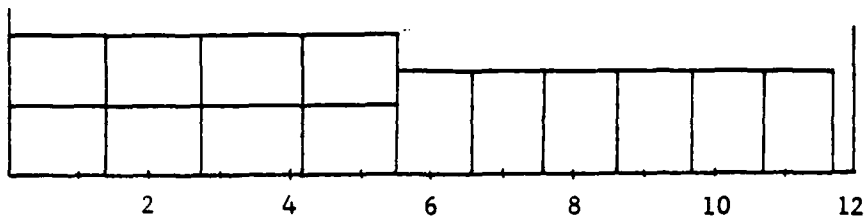


(b) An optimal packing of L.

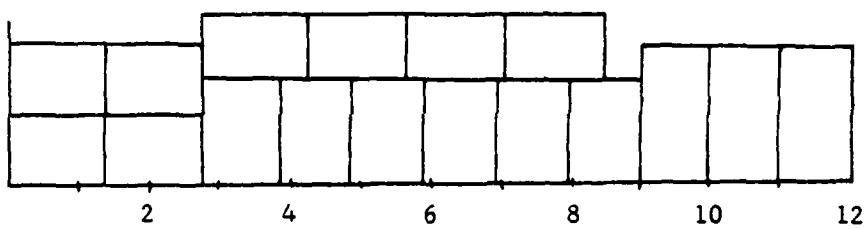
Figure 2. Packing list L of Theorem 1.



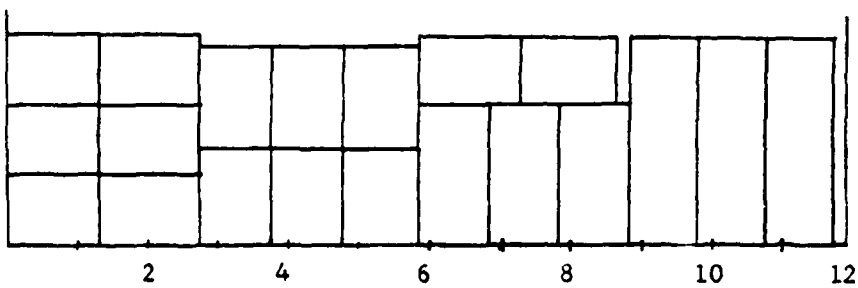
(a) An optimal packing of L_1.



(b) An optimal packing of L_1L_2.

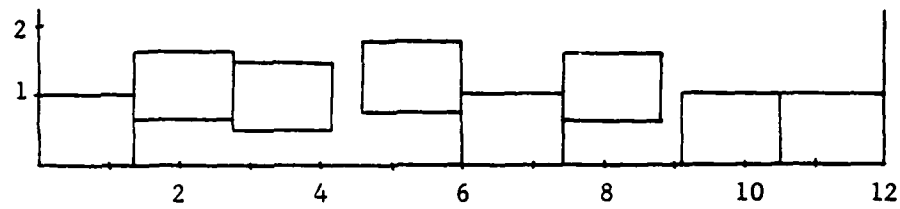


(c) An optimal packing of $L_1L_2L_3$.

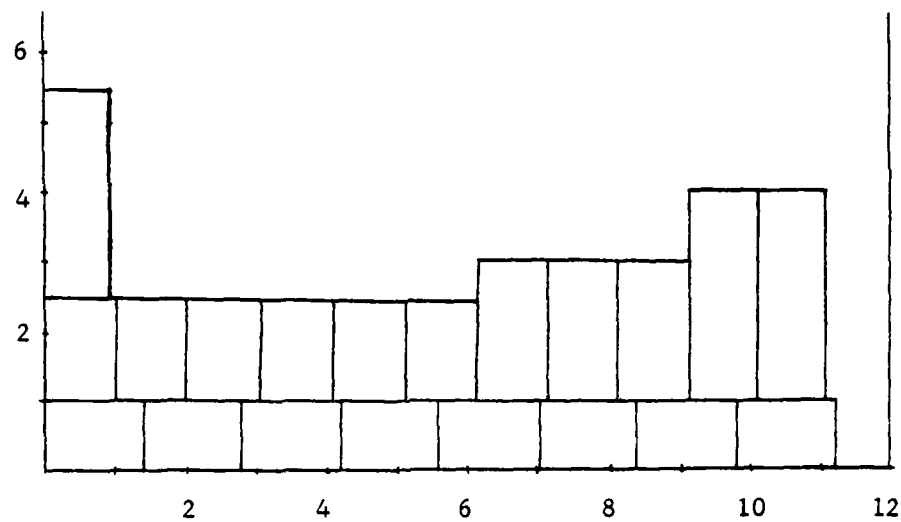


(d) An optimal packing of $L_1L_2L_3L_4 = L$.

Figure 3. Optimal packings of sublists in Theorem 2.

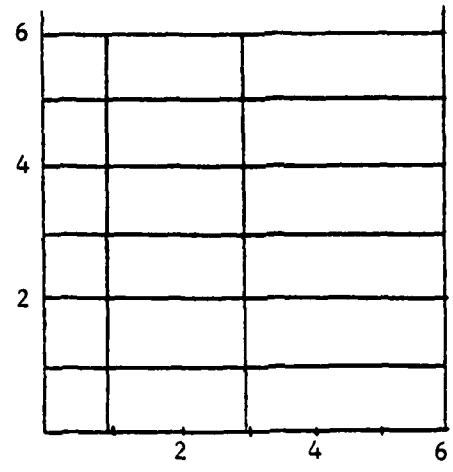


(a) A packing of L_1.

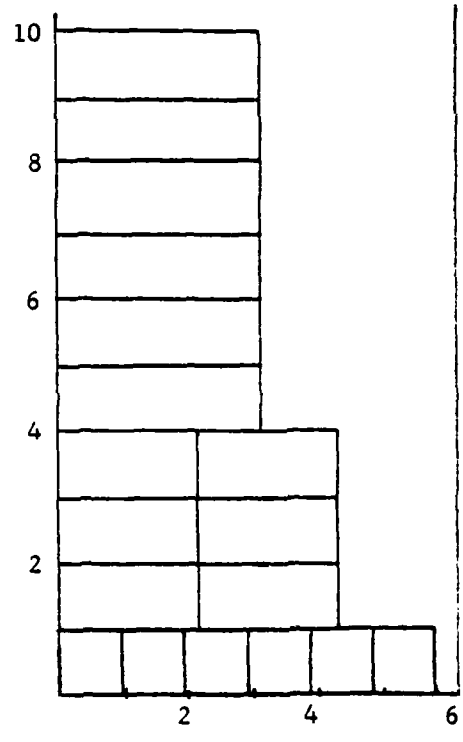


(b) A packing of L by an algorithm A.

Figure 4. Packing list L of Theorem 2.

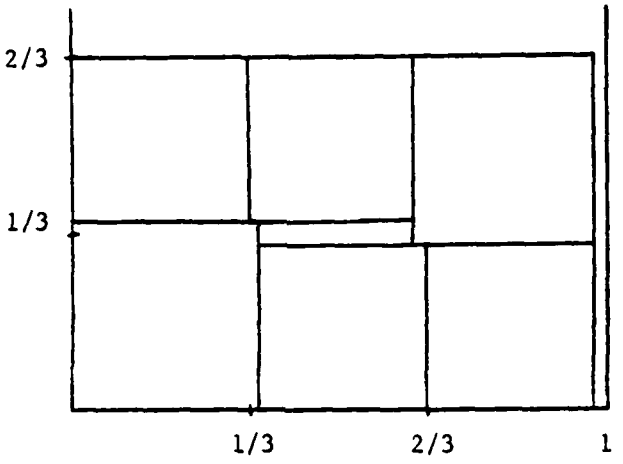


(a) An optimal packing of L.

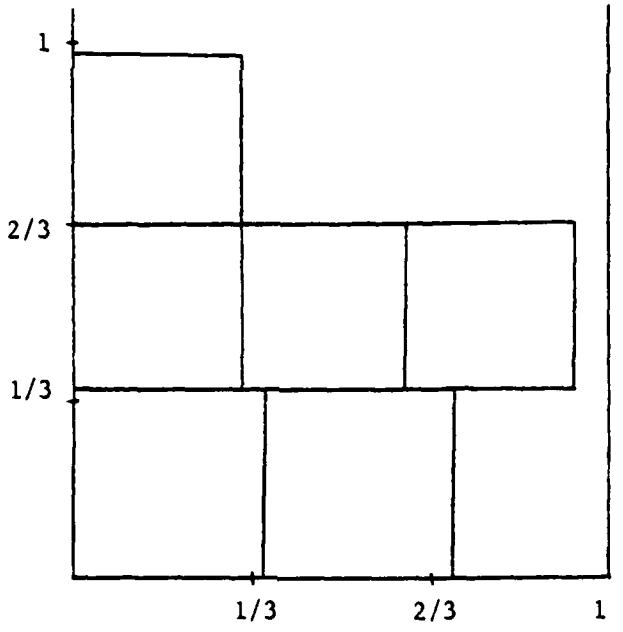


(b) A packing of L by an algorithm A.

Figure 5. Packing list L of Theorem 3.

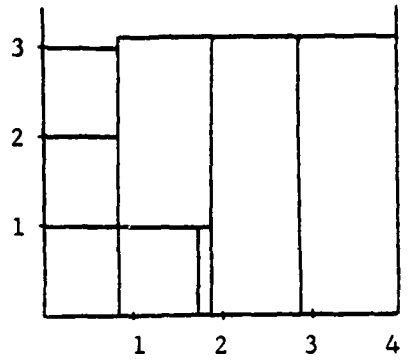


(a) An optimal packing of L.

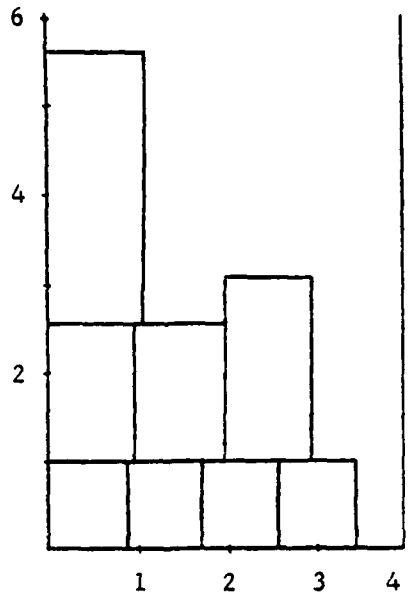


(b) A packing of L by an algorithm A.

Figure 6. Packing list L of Theorem 4.

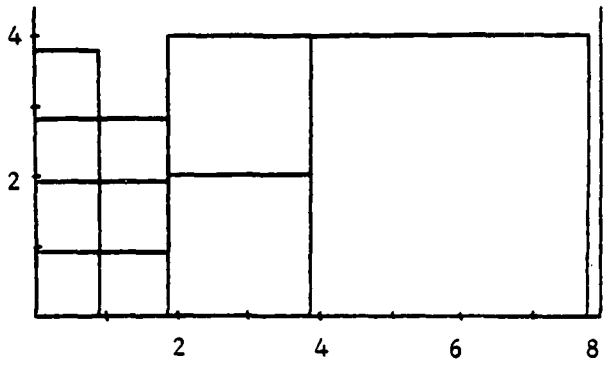


(a) An optimal packing of L.

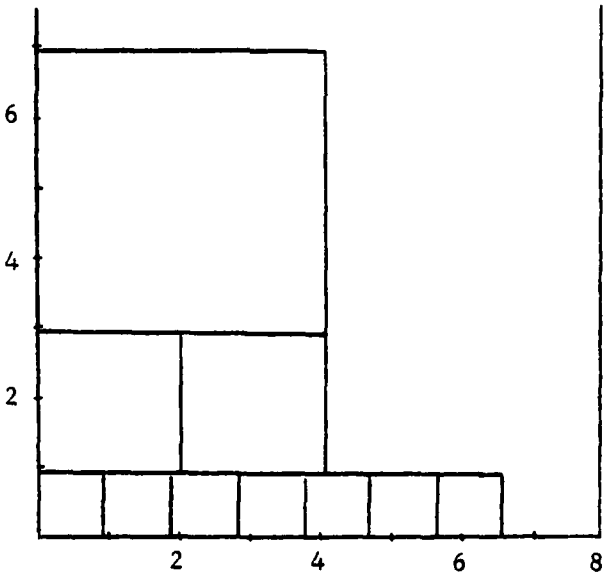


(b) A packing of L by an algorithm A.

Figure 7. Packing list L of Theorem 5.



(a) An optimal packing of L.



(b) A packing of L by an algorithm A.

Figure 8. Packing list L of Theorem 6.

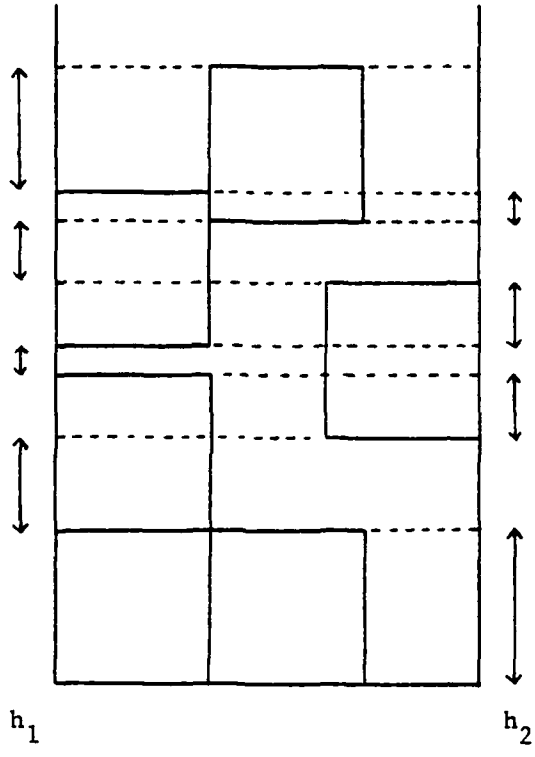
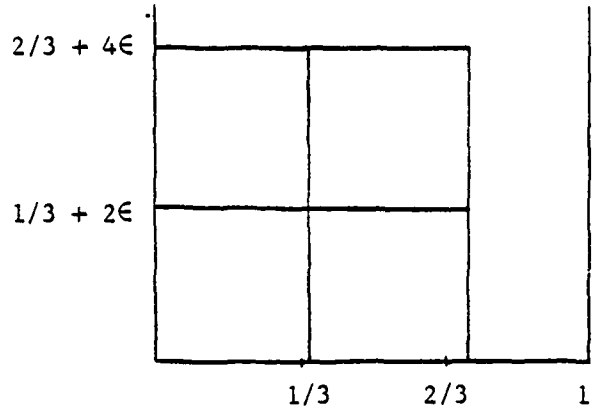
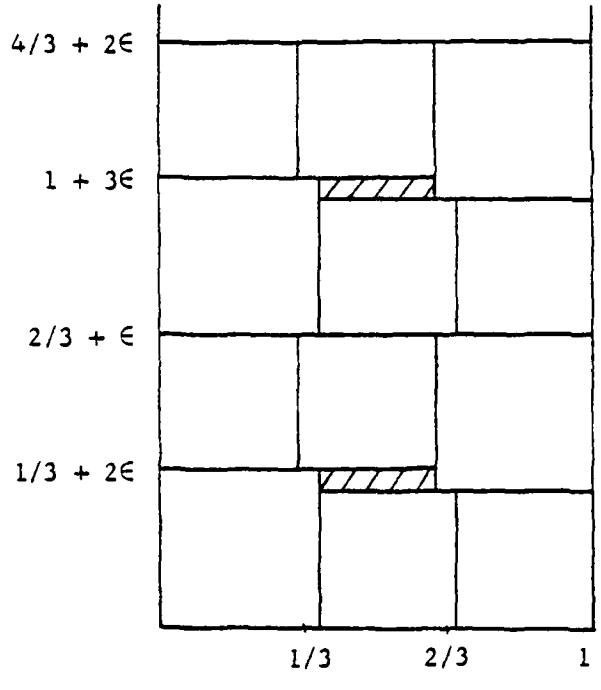


Figure 9. Division of a packing into slices.



(a) An optimal packing of L_1.



(b) An optimal packing of $L_1 L_2$.

Figure 10. Optimal packings of sublists in Theorem 8, for m = 2 and n = 4.