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
UNCLASSIFIED ACT-25 DAAG29-78-C-0016
F/G 12/1 NL
LOWER BOUNDS FOR ON-LINE TWO-DIMENSIONAL PACKING ALGORITHMS

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

DAAG-29-78-C-0016

Joint Services Electronics Program

July 1980

Approved for public release; distribution unlimited

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

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.
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.

Reproduction in whole or in part is permitted for any purpose of the United States Government.

Approved for public release. Distribution unlimited.
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. Keauff
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
- never 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 $\text{OPT}(L)$ denote the height used by an optimal packing. As a measure of absolute worst-case performance, we study the ratio $\frac{A(L)}{\text{OPT}(L)}$; i.e., we consider bounds of the form $A(L) \leq \alpha \text{OPT}(L)$, where $\alpha$ 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 collateral 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 $p_1$, $p_3$, and $p_5$ are collateral at height 5. If $L_1 = (p_1, \ldots, p_i)$ and $L_2 = (p_1, \ldots, p_j)$ are two lists, then we write $L_1L_2$ to denote their concatenation $(p_1, \ldots, p_i, p_1, \ldots, p_j)$.

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 $\alpha$, there is a list $L$ such that $A(L) > \alpha \text{OPT}(L)$. 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 \text{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

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

**Proof:** Let $\delta$ and $\epsilon$ 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 $L$, $A(L) \leq (2 - \delta) \text{OPT}(L)$. In particular, we construct a list $L = L_1 L_2 L_3 L_4 L_5$ (with each list $L_k$ consisting of a single piece $p_i$) for which it cannot be the case that

$$A(L_1, \ldots, L_5) \leq (2 - \delta) \text{OPT}(L_1, \ldots, L_5)$$

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

$$\max \left( \frac{A(L_1)}{\text{OPT}(L_1)}, \frac{A(L_1, L_2)}{\text{OPT}(L_1, L_2)}, \frac{A(L_1, L_2, L_3)}{\text{OPT}(L_1, L_2, L_3)}, \frac{A(L_1, L_2, L_3, L_4)}{\text{OPT}(L_1, L_2, L_3, L_4)}, \frac{A(L_1, L_2, L_3, L_4, L_5)}{\text{OPT}(L_1, L_2, L_3, L_4, L_5)} \right) > 2 - \delta.$$  

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 $A$ must place $p_3$ at some height $h_3$ above the top of $p_2$.

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

$$A(L_1, \ldots, L_k) < 2 \text{OPT}(L_1, \ldots, L_k).$$
Letting \( y_i \) denote the height of piece \( p_i \), we have:

\[
\frac{A(L_1)}{OPT(L_1)} = \frac{h_1 + y_1}{y_1} < 2 \implies h_1 < y_1 = 1
\]

\[
\frac{A(L_2)}{OPT(L_2)} = \frac{h_1 + y_1 + h_2 + y_2}{y_1 + y_2} < 2
\]

\[\implies h_1 + h_2 < y_1 + y_2 = 1 + h_1 + \varepsilon\]

\[\implies h_2 < 1 + \varepsilon\]

\[
\frac{A(L_3L_2)}{OPT(L_3L_2)} = \frac{h_1 + y_1 + h_2 + y_3 + y_2}{y_2 + y_3} < 2
\]

\[\implies h_1 + h_2 + h_3 + y_1 < y_2 + y_3 = (h_1 + \varepsilon) + (1 + h_1 + h_2 + \varepsilon)\]

\[\implies h_3 < h_1 - 2\varepsilon < 1 + 2\varepsilon\]

So if piece \( p_4 \) has size \((3.1 + 2\varepsilon)\), 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 \((1.1 + h_1 + h_2 + h_3 + h_4 + 2\varepsilon)\) would then have to be placed above \( p_4 \), giving:

\[
A(L_4L_3L_2L_1) = h_1 + y_1 + h_2 + y_2 + h_3 + y_3 + h_4 + y_4 + y_5
\]

\[= h_1 + 1 + h_2 + y_2 + h_3 + (1 + h_1 + h_2 + \varepsilon) + h_4 + y_4 + y_5\]

\[= y_2 + y_4 + y_5 + (h_1 + \varepsilon) + (1 + 2\varepsilon) + (1 + h_1 + h_2 + h_3 + h_4 + 2\varepsilon) + h_2 - 4\varepsilon\]

\[= 2y_2 + y_4 + y_5 + h_2 - 4\varepsilon.\]

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

\[
A(L) = 2 OPT(L) + h_2 - 4\varepsilon > 2 OPT(L) - 8
\]

\[> (2 - 8) OPT(L)\]

thereby proving the theorem. \( \Box \)

The Bottom-Leftmost algorithm \( [2] \) and the Split algorithm \( [7] \) both have a worst case performance of \( 3 OPT(L) \) 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 \((1 + \frac{\sqrt{6}}{3}) OPT(L) \) \( * \)

**Theorem 2:** For any on-line algorithm \( A \), there is a list \( L \) ordered by decreasing width such that

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

**Proof:** Let \( \varepsilon \) be fixed, \( 0 < \varepsilon < \frac{1}{24} \). Consider the list of rectangles \( L = L_1L_2L_3L_4 \) where

- \( L_1 \) consists of 8 pieces of size \((\frac{1}{2} - 3\varepsilon, 1)\).
- \( L_2 \) consists of 6 pieces of size \((1 + \varepsilon, -1 + \sqrt{6})\).

\( * \)This is an improvement over Stover'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,

$$\text{OPT}(L_1) = 1.$$  
$$\text{OPT}(L_1L_2) = 2.$$  
$$\text{OPT}(L_1L_2L_3) = \sqrt{3}.$$  
$$\text{OPT}(L_1L_2L_3L_4) = \text{OPT}(L) = 3.$$  

It is shown that any algorithm which packs each of the lists $L_1$, $L_1L_2$, $L_1L_2L_3$, and $L_1L_2L_3L_4 = L$, in such a way that

$$A(L_1) < (1 + \frac{\sqrt{3}}{3}) \text{OPT}(L_1),$$  
$$A(L_1L_2) < (1 + \frac{\sqrt{3}}{3}) \text{OPT}(L_1L_2),$$  
$$A(L_1L_2L_3) < (1 + \frac{\sqrt{3}}{3}) \text{OPT}(L_1L_2L_3),$$

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

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

and then obtain a contradiction, thereby proving the theorem.

We must first pack $L_1$. Since $\text{OPT}(L_1) = 1$, 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 $A(L_1) < (1 + \frac{\sqrt{3}}{3}) \text{OPT}(L_1)$. Thus, for sufficiently small $\delta_1 > 0$, all $L_1$ pieces are colateral 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 $24\epsilon$. 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 $\sqrt{3}$ in the bin. As above, in order to avoid violating $A(L_2) < (1 + \frac{\sqrt{3}}{3}) \text{OPT}(L_2)$, the $L_2$ pieces are colateral at height $\sqrt{3} - \delta_2$ in the bin, for any sufficiently small $\delta_2$ (see Figure 4b). In particular, it is not possible to pack two $L_2$ pieces on top of each other, because this would give

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

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

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

So at height $\sqrt{3} - \delta_2$, 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 $\sqrt{3}$. At least one of them must be above an $L_2$ or an $L_3$ piece, which gives

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

\[
A(L) \geq \frac{5}{3} \text{OPT}(L).
\]

**Proof:** Consider a bin of width 6. For \( 0 < \epsilon < \frac{3}{11} \), let the list \( L = L_1 \cup L_2 \cup L_3 \) 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

\[
\text{OPT}(L_1) = 1,
\]

\[
\text{OPT}(L_2) = 3,
\]

\[
\text{OPT}(L_1 \cup L_2 \cup L_3) = 6.
\]

Assume that

\[
\max \left( \frac{A(L_1)}{\text{OPT}(L_1)}, \frac{A(L_2)}{\text{OPT}(L_2)}, \frac{A(L)}{\text{OPT}(L)} \right) < \frac{5}{3}.
\]

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 \( 1-\epsilon \). Since no \( L_2 \) piece will fit below height 1, and yet all the \( L_2 \) pieces must pack below height 5 (since \( \text{OPT}(L_2) = 3 \)), there is not enough height for four \( L_2 \) pieces to fit above each other. Also, no three pieces of \( L_1 \) or \( L_2 \) 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 \( L_3 \) 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, \( A(L) \geq 10 = \frac{5}{3} \text{OPT}(L) \). □

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 \( 2 \text{OPT}(L) \). 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

\[
A(L) > (1.5-\delta) \text{OPT}(L).
\]

**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 \( 0 < \epsilon < \frac{3}{2} \). 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 \( A(L) < (1.5-\delta) \text{OPT}(L) \), they would have to be colateral at height \( \frac{1}{3} + \epsilon - \delta_1 \), for sufficiently small \( \delta_1 \). 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

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

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

$$A(L) \geq \frac{1 + \sqrt{\gamma}}{2} \cdot OPT(L) > 1.82 \cdot OPT(L).$$

**Proof:** Let $\epsilon$ be fixed, $0 < \epsilon < \frac{1}{4}$. For $k = \frac{1 + 2\sqrt{\gamma}}{3}$, 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

$$OPT(L_1) = 1,$$
$$OPT(L_2) = 2,$$
$$OPT(L_1L_2L_3) = \frac{k}{2} + 1$$
$$OPT(L) = k.$$

We shall show that the assumption

$$\max\left\{ \frac{A(L_1)}{OPT(L_1)}, \frac{A(L_2)}{OPT(L_2)}, \frac{A(L_1L_2)}{OPT(L_1L_2)}, \frac{A(L)}{OPT(L)} \right\} < \frac{1 + \sqrt{\gamma}}{2}$$

leads to a contradiction.

Since $OPT(L_1) = 1$, all $L_1$ pieces must be colateral at height $1 - \delta_1$ for sufficiently small $\delta_1$. 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 $\delta_2$; otherwise the above assumption would be violated, because the $L_2$ pieces would reach height $1 + \frac{k}{2} + \frac{k}{2}$, and

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

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

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

But having the $L_1$ 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

\[
\frac{A(L)}{\text{OPT}(L)} \approx \frac{1 + \frac{k}{2} + k}{k} = \frac{3k - 2}{2k} = \frac{1 + \sqrt{3}}{2}.
\]

So our assumption must be incorrect, which proves the desired result. \( \Box \)

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

\[
A(L) > \left( \frac{7}{4} - \delta \right) \text{OPT}(L).
\]

**Proof:** For fixed \( \epsilon \), \( 0 < \epsilon < \min \{48, \frac{1}{\delta} \} \), consider the list of squares \( L = L_1 L_2 L_3 \), 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

\[
\text{OPT}(L_1) = 1 - \epsilon,
\]

\[
\text{OPT}(L_2) = 2,
\]

\[
\text{OPT}(L) = 4.
\]

Once again, we prove that

\[
\max \left\{ \frac{A(L_1)}{\text{OPT}(L_1)}, \frac{A(L_2)}{\text{OPT}(L_2)}, \frac{A(L)}{\text{OPT}(L)} \right\} > \frac{7}{4} - \delta
\]

by assuming the contrary.

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

\[
\frac{A(L_2)}{\text{OPT}(L_2)} \approx \frac{(1-\epsilon)+2+2}{2} > \frac{7}{4} - \delta.
\]

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

\[
\frac{A(L)}{\text{OPT}(L)} \approx \frac{(3-\epsilon)+4}{4} = \frac{7-\epsilon}{4} > \frac{7}{4} - \delta. \quad \Box
\]

**Asymptotic Lower Bounds**

The lower bounds cited above are all bounds for absolute worst-case performance. If \( H, \alpha, \) and \( \beta \) are constants such that, for every list \( L \) with pieces of height at most \( H \), \( A(L) \leq \alpha \text{OPT}(L) + \beta \), then \( \alpha \) 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 $a$ and $\beta$ be constants such that for every list $L$ of squares of size at most $1/m$ ordered by decreasing size, $A(L) \leq a \text{OPT}(L) + \beta$. If $m>1$, then $a \geq \frac{m^2}{m^2-m+1}$. If $m=1$, then $a \geq \frac{\beta}{7}$.

**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 $\frac{1}{m+1} + me$ and $L_2$ contains $nm$ squares of size $\frac{1}{m+1} - e$. Note that $\text{OPT}(L_1) = \frac{n}{m}(-\frac{1}{m+1} + me)$ and $\text{OPT}(L_2) = n(-\frac{1}{m+1} + me)$. (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

$$A(L_1) \geq h_1 + h_2$$

$$\geq \lfloor n(\frac{1}{m+1} + me) - mh_2 \rfloor + h_2$$

$$= \frac{n}{m+1} + h_2(1-m) + nme.$$  

Thus,

$$A(L_1) \leq a \text{OPT}(L_1) + \beta$$

$$\frac{n}{m+1} - h_2(1-m) + nme \leq \frac{n}{m+1} - \frac{m}{m+1} + \alpha + \beta$$

$$h_2 \geq \frac{1 - \alpha}{m^2 - m} - \frac{\alpha \epsilon + \beta - m \epsilon}{m-1}$$

A slice containing $k \geq 1$ 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
\( n \left( \frac{1}{m+1} + m^2 \right) = \frac{nm}{m+1} - \epsilon \), and at most \( m+1 \) segments fit in a slice,

\[
A(L_1L_2) \geq h_1 + h_2 + \frac{1}{m+1} \left[ n \left( \frac{1}{m+1} + m^2 \right) + nm \left( \frac{1}{m+1} - \epsilon \right) - (m+1)h_1 - mh_2 \right]
\]

\[
= \frac{h_2}{m+1} + \frac{n}{m+1}
\]

Thus,

\[
A(L_1L_2) \leq \alpha \text{OPT}(L_1L_2) + \beta
\]

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

\[
\alpha > \frac{h_2 - \beta (m+1)}{n} \quad \frac{n}{1+m(m+1)e}
\]

Substituting in for \( \frac{h_2}{n} \),

\[
\alpha > \frac{m^2 - \beta}{m^2 - m - m - 1} - \frac{\beta}{n} \quad \frac{m-1}{n} \quad \frac{1}{1+m(m+1)e}
\]

\[
\alpha > \frac{m^2 - \beta}{m^2 - m + 1} + \frac{(m^2 - m + 1)e}{m(m+1)}
\]

Choosing \( n \) sufficiently large,

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

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 \( A \), \( A(L) \leq \alpha \text{OPT}(L) + \beta \) implies

\[
A > \frac{m^2}{m^2 - m + 1} - \delta.
\]

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 \( m=1 \), \( \alpha \geq \frac{2^2}{2^2 - 2 + 1} = \delta/7 \). \( \Box \)

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

\[ R_t = \frac{\sum_{i=1}^{t} i}{\sum_{i=1}^{t} a_i - 1}. \]  

(1)

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

\[ \text{OPT}(L_1 L_2 \ldots L_k) \leq \frac{r}{a_{t-1} - 1} \epsilon. \]  

(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 \( \alpha_i \) 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 \( \beta_i \) 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 \( \alpha_i \) (\( \beta_i \)). For \( 1 \leq k \leq t - 1 \)

\[ A(L_1 L_2 \ldots L_k) = \sum_{i=1}^{k} (h(\alpha_i) - h(\beta_i)). \]  

(3)

and

\[ A(L_1 L_2 \ldots L_t) = r + \sum_{i=1}^{t} h(\alpha_i). \]  

(4)

Assume that

\[ \max_{1 \leq i \leq t} \left\{ \frac{A(L_1 L_2 \ldots L_i)}{\text{OPT}(L_1 L_2 \ldots L_i)} \right\} < R_t - \delta. \]  

(5)

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

\[ \text{OPT}(L_1 L_2 \ldots L_k)(R_t - \delta) > \sum_{i=1}^{k} (h(\alpha_i) - h(\beta_i)) \]  

(6)

and

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

(7)

Because there are \( ra_{t-1} \) squares of size \( p_i, (2 \leq i \leq t) \),

\[ \left( \frac{1}{a_{t-1} - 1} + \epsilon \right) [ra_{t-1}] = \sum_{s \in S} m_i(s) h(s) \]  

(8)

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

\[ (R_t - \delta) \sum_{i=1}^{t} \left( \frac{r}{a_{t-1} - 1} + \epsilon \right) [ra_{t-1}] - \sum_{i=2}^{t} [ra_{t-1}] \left( \frac{1}{a_{t-1} - 1} + \epsilon \right) \]

\[ > \sum_{i=1}^{t} \sum_{s \in S} [h(\alpha_i) - h(\beta_i)] + r + \sum_{i=1}^{t} h(\alpha_i) - \sum_{i=2}^{t} [ra_{t-1}] \sum_{s \in S} m_i(s) h(s). \]  

(9)

By (1) and the choice of \( \epsilon < \frac{\delta}{tr_p} \), the left hand side is less than
Combining (9) and (10)

\[ \frac{R_i}{\sum_{i=1}^{n} \frac{r}{a_i} - 1} - \frac{i}{\sum_{i=1}^{n} \frac{r}{a_i}} = r. \]

(10)

At this point, it is possible to apply Brown's original proof [4] which shows that (11) leads to a contradiction for \( e < \frac{1}{a_i(a_i-1)(i-1)} \). We conclude that the assumption in (5) is incorrect, and the asymptotic bound is at least \( R_i > 1.536 \) for \( i \geq 5 \). 

\[ \sum_{i \leq n} h(s) \sum_{j=1}^{i-1} m_{a_i} = \sum_{j=1}^{(j+1)h(a_i-1)+jh(\beta_{i-1})}. \]

(11)

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 \( \frac{m^2}{m^2 - m + 1} \) for pieces of size at most \( 1/m \) ordered by decreasing size.

References


(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 $x_1$: 3 8 11 4 1
and height $y_1$: 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.
An optimal packing of $L_1$.

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

(c) An optimal packing of $L_1 L_2 L_3$.

(d) An optimal packing of $L_1 L_2 L_3 L_4 = L$.

Figure 3. Optimal packings of sublists in Theorem 2.
Figure 4. Packing list $L$ of Theorem 2.

(a) A packing of $L_1$.

(b) A packing of $L$ by an algorithm $A$. 
(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.
Figure 8. Packing list $L$ of Theorem 6.
Figure 9. Division of a packing into slices.
Figure 10. Optimal packings of sublists in Theorem 8, for \( m = 2 \) and \( n = 4 \).