Acta Informatica 9, 263-271 (1978)

Bin Packing: Maximizing the Number of Pieces Packed *

E.G. Coffman, Jr.[1]**, J. Y-T. Leung[2], and D.W. Ting[3]

[1] Department of Electrical Engineering and Computer Science, Columbia University, New York, NY 10027, USA
[2] Computer Science Department, Virginia Polytechnic University, Blacksburg, VA 24060, USA
[3] Bell Laboratories, Holmdel, NJ 07733, USA

Summary. We consider a variant of the classical one-dimensional bin-packing problem: The number of bins is fixed and the object is to maximize the number of pieces packed from some given set. Both problems have applications in processor and storage allocation in computer systems in addition to a broad application in operations research.

It can easily be shown that both problems are NP-complete; our approach will be to propose and analyze very fast heuristics. We consider a class of algorithms and bound the performance of an arbitrary algorithm in that class. Finally we propose an algorithm, the first-fit-increasing algorithm, and analyze its running time and relative performance.

I. Introduction

In the classical bin-packing problem [3, 4] one is concerned with minimizing the number of equal capacity bins necessary for the placement (storage) of a fixed set of pieces. A related problem is based on a fixed set of bins in which one attempts to maximize the number of pieces packed from some given set. It is this latter problem that we shall study in this paper. It is readily verified that both problems are NP-complete; as in [4] our approach will be to propose and analyze very fast heuristics. Although there is an apparent duality between the above problems, this view does not appear to carry one very far, as the sequel will indicate.

The problem we have posed is a fundamental one for which a broad application in operations research is easily envisioned. On the other hand, our immediate motivation stems from storage allocation in computer systems. Consider for example the problem of storing variable-length records in a multiple level storage system. Assume that the device at a given level is organized into logically disjoint "bins"; i.e. sectors, cylinders, tracks, etc. If we have no a priori way to distinguish

* This research was supported in part by NSF Grant No. 28290

** To whom offprint requests should be sent

0001-5903/78/0009/0263/$01.80

records on the basis of access probability, then maximizing the number of records stored at a given level (minimizing the number stored on slower devices) constitutes our particular bin packing problem and leads to minimum average access times.

In the next section we shall introduce notation and define classes of packing heuristics. In Section III, the main body of the paper, worst case bounds are derived for what we consider to be the simplest, reasonable packing algorithms. In Section IV other algorithms are considered and open problems discussed.

II. Definitions and Notation

Given a set of m equal capacity bins $B_1, \ldots, B_m$ and a set of pieces organized into a list $L=(p_1, p_2, \ldots, p_n)$, we consider the problem of packing into the bins a maximum subset of L such that no bin capacity is exceeded. Without loss of generality we assume unit bin capacities and hence the following constraint on piece sizes: $0<\text{size}(p)\leqq 1$ for all p.

It is intuitively clear that any algorithm having reasonable worst-case performance relative to an optimization algorithm must attempt to pack a maximum subset of smaller pieces. That is, if $\text{size}(p_1)\leqq\text{size}(p_2)\leqq\cdots\leqq\text{size}(p_n)$ then the algorithms to be considered are those which attempt to pack a maximum prefix of L into the given, fixed set of bins. With the above ordering of L it is obvious that for every sublist $L'\subseteq L$ that can be packed into m bins there is a prefix of L having at least as many pieces which can also be packed into the m bins. Thus, we shall restrict ourselves to algorithms which assume (or initially perform) an ordering of the list L such that $\text{size}(p_i)\leqq\text{size}(p_{i+1})$, $1\leqq i<n$. Figure 1 shows examples of packings that can be produced by such algorithms.

The symbol $P(L)$ denotes a given packing of some prefix of L into m bins, where m is understood. Frequently, L will also be understood by context, in which case the dependence on L may also be suppressed. In addition to its use as a bin name, B_i also denotes the set of pieces contained in the i^{th} bin according to a given packing. Thus, a packing can be represented as the corresponding sequence of bins $B_1, B_2, \ldots, B_m$. We define v_i as the *level* of bin B_i ($v_i=\sum_{p\in B_i}\text{size}(p)$), and $k_i=|B_i|$, the number of pieces in B_i.

We let $n_A(L)$ denote the number of pieces packed from L by algorithm A into m bins. L will be suppressed when clear by context. and $n_o(L)$ will denote the maximum number of pieces that can be packed, i.e., the number achieved by an optimization algorithm applied to L in m bins.

Suppose that for some list $L=(p_1, p_2, \ldots, p_n)$ and some $t\leqq n$ the pieces packed in $P(L)$ are $p_1, \ldots, p_t$. Suppose further that $t<n$ implies that for each i, $1\leqq i\leqq m$, $p_{t+1}>1-v_i$; i.e., no unpacked piece will fit into any bin. Then $P(L)$ will be called a *prefix* packing. Consistent with earlier assumptions, all algorithms that we consider produce prefix packings of given (ordered) lists.

As a specific algorithm consider the SPF (smallest-piece-first) rule which at each point in a left-to-right scan of L places the next (larger) piece into a lowest-level bin into which it will fit. If there is more than one lowest-level bin the piece is placed into that one having lowest index. When the algorithm first encounters a piece which will not fit into any bin, the algorithm terminates. Figure 1 shows an example of SPF packings.

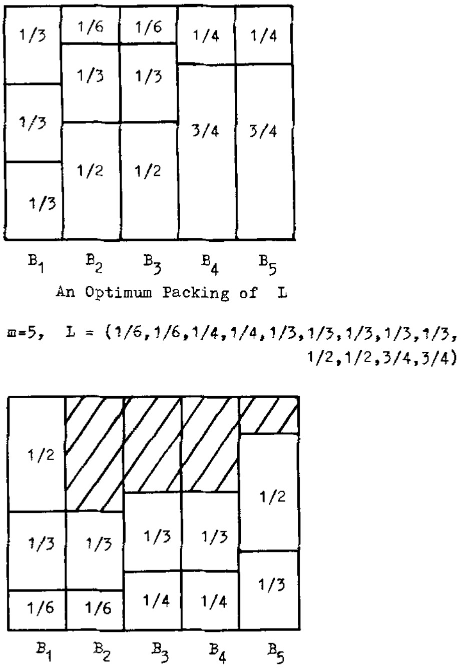

Fig. 1. Illustrative prefix packings

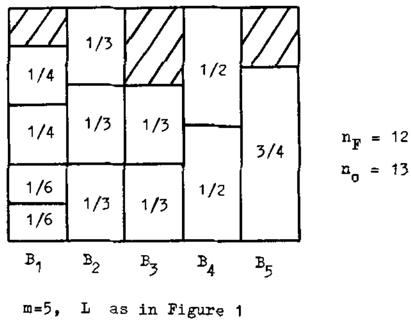

Fig. 2. An example FFI packing

The packing rule whose analysis is the principal contribution of the next section is called the FFI (first-fit-increasing) rule. As in the SPF rule the pieces are packed in the sequence $p_1, p_2, p_3, \ldots$, and the algorithm terminates when it first fails to pack a piece. However, with the FFI rule each successive piece is placed into the lowest indexed bin into which it will fit. Figure 2 gives the FFI packing for the list of Figure 1.

FFI packings clearly have a great deal of structure. In particular, suppose $B_1, \ldots, B_m$ is an FFI packing of L. Then no piece in $B_i, \ldots, B_m$ will fit into any of the bins $B_1, \ldots, B_{i-1}$. The cardinality of the bins is non-increasing ($k_{i+1} \leq k_i$, $1 \leq i \leq m-1$), and any sub-sequence of the bins $B_1, \ldots, B_m$ is a valid packing for a sub-sequence of L. In view of this structure the apparent difficulty of the FFI analysis may seem somewhat surprising.

III. Performance Bounds

In this section we bound the performance of an arbitrary algorithm producing prefix packings, and the performance of the FFI algorithm in particular.

Theorem 1. Let P be a prefix packing and let $k = \min\{k_i\}$ be the least number of pieces stored in any bin of P. Then if n_k denotes the number of pieces packed in P, we have

$$\frac{n_o}{n_k} \leqq \frac{k+1}{k} - \frac{1}{mk}. \tag{1}$$

Moreover, this bound is achievable for all $m \geqq 1$ and $k \geqq 1$.

Proof. Since the total capacity is m, an optimization algorithm can not pack more than $m-1$ (necessarily larger) pieces beyond those packed by a prefix algorithm. Thus, letting $d = n_o - n_k$ we have $d \leqq m-1$. By definition of k we have $n_k \geqq km$, and hence $n_o/n_k = 1 + (n_o - n_k)/n_k \leqq 1 + (m-1)/mk = (k+1)/k - 1/mk$.

To verify that (1) is achievable one uses the example $n = m(k+1) - 1$ and $\text{size}(p_i) = 1/mk$, $1 \leqq i \leqq mk$, and $\text{size}(p_i) = 1$, $mk+1 \leqq i \leqq m(k+1)-1$. □

Since $n_o/n_k > 1$ implies $k \geqq 1$, we see from (1) that $2 - 1/m$ is a best bound, as a function only of m. It is also readily verified that $2 - 1/m$ is a best bound as a function of both m and the maximum piece size. Note also from the above example that (1) must be a best bound on n_o/n_{SPF}.

We turn now to bounds on the performance of the FFI rule. First, it will be convenient to introduce the following notation. We let P_F denote an FFI packing, and n_F the number of pieces in P_F. We define P_o and n_o similarly for an optimum packing. In the remainder of this section B_i and k_i will always refer to an FFI packing; B_i^o and k_i^o will refer to a corresponding optimum packing. We define the index r as the largest integer such that $k_r > k_m = \min\{k_i\}$ in an FFI packing. We shall continue with the notation $d = n_o - n_F$.

Theorem 2. For any list L packed into any number, m, of bins we have

$$\frac{n_o}{n_F} \leqq \frac{4}{3}. \tag{2}$$

Moreover, for every even m there exists a list which achieves the bound.

Proof. The proof is based on the following four claims. The first follows from simple capacity arguments.

Claim 1. For any list packed into m bins we must have $d \leqq r k_m$.

Proof. The pieces in $P_o - P_F$ are all at least as large as those in P_F, and no piece in $\bigcup_{r+2 \leqq i \leqq m} B_i \cup (P_o - P_F)$ will fit into a bin along with all of the pieces in B_{r+1}. Thus, it follows that no $k_m + 1$ pieces in $S \equiv \bigcup_{r+1 \leqq i \leqq m} B_i \cup (P_o - P_F)$ will fit into a single bin. Hence, since $|S| = d + (m-r)\,k_m$ and $|S| \leqq m k_m$ must hold, we have $d + (m-r)\,k_m \leqq m k_m$ and hence $d \leqq r k_m$. □

By definition of the FFI packing the smallest piece in B_{i+1} must be larger than the unused capacity in B_i. This simple property is instrumental in the proof of

Claim 2. In an FFI packing let $x_j = \sum_{i=1}^{j} v_i$. If for some $k \geqq 1$ we have $|B_{i+1}| = k$ and $v_i \leqq k/(k+1)$, then $v_j > k/(k+1)$, $1 \leqq j < i$, $v_i + v_{i+1} > 2k/(k+1)$ and $x_{i+1} > (i+1)\,k/(k+1)$.

Proof. From the properties of FFI packings, if $v_i \leqq k/(k+1)$ then $|B_{i+1}|=k$ implies $v_i+v_{i+1}>v_i+k(1-v_i)=k-(k-1)v_i$, and hence $v_i+v_{i+1}>k-k(k-1)/(k+1)=2k(k+1)$. Clearly, the smallest piece in B_i has a size no greater than $v_i/k \leqq 1/(k+1)$. From the FFI rule it must therefore be true that $(1-v_j)<1/(k+1)$, $1 \leqq j<i$, and hence $v_j>k/(k+1)$, $1 \leqq j<i$. From this last observation and the inequality $v_i+v_{i+1}>2k/(k+1)$ we obtain $x_{i+1}>(i+1)\,k/(k+1)$ directly. □

Using Claim 2 we may next prove

Claim 3. For a given list L packed into m bins suppose $d>0$, and let

$$x'_j=\sum_{i=j}^{m} v_i+\sum_{p\in P_o-P_F} \text{size}(p).$$

Then for all $s=r, r+1, \ldots, m$ we have $x'_s>(m-s+1)\,k_m/(k_m+1)+d/(k_m+1)$.

Proof. Since $\text{size}(p)>1/(k_m+1)$ for each of the d pieces in P_o-P_F, the result is manifest when $v_i>k_m/(k_m+1)$, $i=s, s+1, \ldots, m$. Moreover, from Claim 2 and the additional fact that $k_i=k_m$, $r+1 \leqq i \leqq m$, we know that only B_r or B_{r+1}, but not both, can possibly have a level exceeding $k_m/(k_m+1)$. Thus, the result is immediate when $s>r+1$. Using Claim 2, the result still follows easily in the remaining cases, except when $r=m-1$ and $v_m \leqq k_m/(k_m+1)$. In this last case we use the argument of Claim 2 and write

$$x'_s>v_m+d(1-v_m)+\begin{cases} k_m/(k_m+1), & s=m-1 \\ 0, & s=m, \end{cases}$$

On using $d>0$ we get

$$x'_s>k_m/(k_m+1)+d/(k_m+1)+\begin{cases} k_m/(k_m+1), & s=m-1 \\ 0, & s=m \end{cases}$$

which accounts for the claim when $s=m-1$ and $s=m$. □

Note that $x_i+x'_{i+1} \leqq m$ must hold, since the cumulative size of the pieces in P_o can not exceed the capacity, m, of m bins. A key result for this and the following theorem is given next.

Claim 4. Suppose list L is packed into m bins such that

$$n_o/n_F>f(k_m)\equiv(k_m+1)^2/(k_m^2+k_m+1).$$

Suppose further that there is no shorter list $L' \subset L$ for which a packing into $m' \leqq m$ bins is such that $n'_o/n'_F>f(k_{m'})$. Then we must have $k_i^o>k_m$, $1 \leqq i \leqq m$, and either $n_o \geqq (k_m+2)\,m-k_m$, or $r \geqq m-k_m+1$ (and hence $n_F \geqq m k_m+r \geqq (k_m+1)\,m-(k_m-1)$).

Proof. In P_o suppose $k_i^o \leqq k_m$ and let S be the set of k_i^o largest pieces in P_o. It is easily seen that the packings P'_o and P'_F of the list $L'=L-S$ into $m'=m-1$ bins provide a smaller example for which $n'_o/n'_F>f(k_{m'})$ – a contradiction.

For the second part, suppose both $n_0<(k_m+2)\,m-k_m$ and $r<m-k_m+1$. Note that these inequalities and $k_i^o>k_m$, $1 \leqq i \leqq m$, imply that P_o has at least k_m+1 bins with exactly k_m+1 pieces, and P_F has at least k_m bins with exactly k_m pieces. It is not difficult to verify that the packings P'_o and P'_F of the list $L'=L-B_r$ into

$m' = m - (k_m + 1)$ bins again provide us with a smaller example for which $n'_o/n'_F > f(k_{m'})$. □

We may now proceed with a proof of the theorem. We distinguish three principal cases.

Case 1 ($k_m \geqq 3$). Since $n_F \geqq 3m$ we have immediately from $d \leqq m-1$, $n_o/n_F = 1 + d/n_F \leqq 1 + (m-1)/3m < \frac{4}{3}$, $m \geqq 1$.

Case 2 ($k_m = 2$). Suppose L is such that $n_o/n_F > \frac{4}{3} > (k_m+1)^2/(k_m^2 + k_m + 1) = \frac{9}{7}$. If $n_o < 4m-2$ and $r < m-1$ then from the arguments in Claim 4 there must be a shorter list, L', violating $\frac{4}{3}$ in $m' < m$ bins. Moreover, we can not, according to Case 1, assume that the packing of L' is such that $k_{m'} \geqq 3$. Thus, if we assume, as we may, that L is the shortest list for which $k_m = 2$, then we require that either $n_0 \geqq 4m-2$ or $r \geqq m-1$ and hence $n_F \geqq 3m-1$. But if $n_o \geqq 4m-2$ then for all $m \geqq 1$, $n_o/n_F = n_o/(n_o - d) \leqq (4m-2)/((4m-2)-(m-1)) < \frac{4}{3}$, and if $n_F \geqq 3m-1$ then for all $m \geqq 1$, $n_o/n_F = 1 + d/n_F \leqq 1 + (m-1)/(3m-1) < \frac{4}{3}$. We obtain a contradiction in either case.

Case 3 ($k_m = 1$). Suppose we have a shortest list L such that $k_m = 1$ and $n_o/n_F > \frac{4}{3}$. We consider two sub-cases based on the level of B_r.

Case 3a ($v_r > \frac{2}{3} = (k_m+1)/(k_m+2)$). Since the cumulative size of the pieces in P_o must not exceed the total capacity m, we must have $m \geqq x_r + x'_{r+1}$ (see Claims 2 and 3). From $v_r > \frac{2}{3}$ and Claims 2 and 3 we get $x_r > 2r/3$ and $x'_{r+1} > (m-r)/2 + d/2$. Hence, $m > 2r/3 + (m-r+d)/2$. On using $d \leqq r k_m = r$ from Claim 1, we obtain $d < 3m/4$ or $d \leqq (3m-1)/4$.

Next, we get a lower bound on n_F. First, we note that for $k_m = 1$, $r \leqq m - k_m$ is always true. Hence, if $n_o < m(k_m+2) - k_m = 3m-1$ we can always find a shorter list L' violating the $\frac{4}{3}$ bound according to the transformation in Claim 4. Moreover, since we have already shown that $n_o/n_F \leqq \frac{4}{3}$ for all lists such that $k_m \geqq 2$, the shorter list would have to be such that $k_{m'} = 1$. By our assumptions no such list can exist, and therefore $n_o \geqq 3m-1$ must hold.

Using these last two bounds on d and n_o we have

$$n_o/n_F = n_o/(n_o - d) \leqq (3m-1)/(3m-1) - (3m-1)/4) = \tfrac{4}{3},$$

the desired contradiction.

Case 3b ($v_r \leqq \frac{2}{3}$). In this case every piece in $B_1, \ldots, B_{r-1}$ has a size no greater than $\frac{1}{3}$. Hence, $k_i \geqq 3$, $1 \leqq i \leqq r-1$, and a count of the pieces in P_F must give $n_F \geqq 3(r-1) + 2 + (m-r) = m + 2r - 1$. On applying Claim 1 we obtain $n_o/n_F = 1 + d/n_F \leqq 1 + r/(m+2r-1) \leqq 1 + (m-1)/(3m-3) = \frac{4}{3}$. This contradiction completes the proof of the bound $\frac{4}{3}$.

To show that no constant smaller than $\frac{4}{3}$ will suffice for all m, we consider the examples where m is even, $n = 2m$, and the piece sizes are given by $\text{size}(p_i) = \frac{1}{2} - \varepsilon$, $1 \leqq i \leqq m$, and $\text{size}(p_i) = \frac{1}{2} + \varepsilon$, $m+1 \leqq i \leqq 2m$, where $0 < \varepsilon < \frac{1}{6}$. It is readily verified that $n_o = 2m$, $n_F = 3m/2$ and hence $n_o/n_F = \frac{4}{3}$. Figure 3 with $k_m = 1$ shows the general case. □

Theorem 3. For a given list L packed into m bins we have

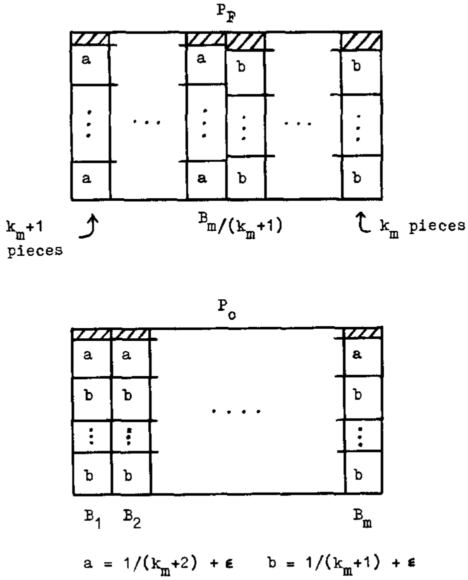

Fig. 3. An example achieving $(k_m+1)^2/(k_m^2+k_m+1)$

$$n_o \leqq \frac{m(k_m+1)}{mk_m+1} n_F, \qquad m \leqq k_m+1, \tag{3}$$

$$n_o \leqq \frac{(k_m+1)^2}{k_m^2+k_m+1} n_F+1, \qquad m > k_m+1. \tag{4}$$

The bound in (3) is best in the sense that there are examples for all $1 \leqq m \leqq k_m+1$ such that (3) is achieved. The bound in (4) is asymptotically best in the sense that there exist examples for every multiple of $m \geqq k_m+1$ such that the coefficient of n_F is equal to n_o/n_F.

Proof. From Claim 1 we must have $r \geqq 1$ in order for $d > 0$, and hence $n_o/n_F > 1$. By definition of r we have $n_F \geqq mk_m + r$. Thus, using $d \leqq m-1$ we get $n_o/n_F = 1 + d/n_F \leqq 1 + (m-1)/(mk_m+r) < m(k_m+1)/(mk_m+1)$. The examples, $n = m(k_m+1)$, $\text{size}(p_i) = 1/(k_m+2)+\varepsilon$, $1 \leqq i \leqq k_m+1$, and $\text{size}(p_i) = 1/(k_m+1)+\varepsilon$, $k_m+2 \leqq i \leqq m(k_m+1)$, are readily seen to achieve (3) for all $m \leqq k_m+1$ and $0 < \varepsilon < 1/(k_m+1)^2(k_m+2)$.

For $m \geqq k_m+2$ we may restrict ourselves to $k_m \geqq 2$, for Theorem 2 establishes (4) when $k_m = 1$. The pieces packed in P_o can not have a cumulative size exceeding the total capacity, m. Thus, from the definitions of x_j and x_j' in Claims 2 and 3 we have $m \geqq x_{r-1} + x_r'$. Now if $v_r \leqq \frac{k_m+1}{k_m+2}$ then using Claim 2 we have $v_{r-1} > \frac{k_m+1}{k_m+2}$. Hence, using Claims 2 and 3

$$m > \frac{k_m+1}{k_m+2}(r-1) + \frac{k_m}{k_m+1}(m-r+1) + \frac{d}{k_m+1}$$

from which one derives

$$d \leqq m - r/(k_m+2).$$

If $v_r > \frac{k_m+1}{k_m+2}$ then we must have from Claims 2 and 3

$$m > \frac{k_m+1}{k_m+2} r + (m-r)\frac{k_m}{k_m+1} + \frac{d}{k_m+1} > \frac{k_m+1}{k_m+2}(r-1) + (m-r+1)\frac{k_m}{k_m+1} + \frac{d}{k_m+1}$$

so that we must still have $d \leqq m - r/(k_m+2)$.

Next, suppose L is the shortest list for which (4) is violated for some m and k_m. According to Claim 4 we need only consider the following two cases.

Case 1 ($n_o \geqq (k_m+2)m - k_m$). In this case we have from $d \leqq m - r/(k_m+2)$ and using $r \geqq d/k_m$ from Claim 1, $d \leqq m k_m(k_m+2)/(k_m+1)^2$. Hence, we have

$$\frac{n_o}{n_F} = \frac{n_o}{n_o - d} \leqq \frac{(k_m+2)m - k_m}{(k_m+2)m - k_m - m k_m(k_m+2)/(k_m+1)^2}$$

from which (4) follows routinely for all $m \geqq k_m+2$, $k_m \geqq 2$.

Case 2 ($r \geqq m - k_m + 1$). From $d \leqq m - r/(k_m+2)$ we have

$$n_o/n_F = 1 + d/n_F \leqq 1 + (m - r/(k_m+2))/(m k_m + r),$$

whereupon substitution of $r \geqq m - k_m + 1$ gives after some manipulation

$$\frac{n_o}{n_F} \leqq 1 + \frac{(k_m+1)m + (k_m-1)}{(k_m+2)((k_m+1)m - (k_m-1))}$$

from which (4) again follows routinely for all $m \geqq k_m+2$ and $k_m \geqq 2$. This contradiction completes the proof of (4).

To show that $n_o/n_F = (k_m+1)^2/(k_m^2+k_m+1)$ can be achieved we consider any m a multiple of k_m+1, $n = m(k_m+1)$, and the piece sizes $\text{size}(p_i) = 1/(k_m+2)+\varepsilon$, $1 \leqq i \leqq m$, and $\text{size}(p_i) = 1/(k_m+1)+\varepsilon$, $m+1 \leqq i \leqq n$, for any $0 < \varepsilon < 1/(k_m+1)^2(k_m+2)$. Figure 3 pictures the general example. □

Theorem 3 may be easily re-stated as a function of the maximum piece-size in P_F, for if $\max\{\text{size}(p_i)\} \leqq 1/k$, then $k_m \geqq k$ and the bounds of Theorem 3 may be used with k_m replaced by k. The statement in Theorem 3 is more informative, however, since it is clearly possible that $k_m \geqq k$, even though $\max\{\text{size}(p_i)\} > 1/k$.

IV. Discussion

Note that Theorem 3 reveals the not unexpected result that as k_m increases n_o/n_F approaches unity approximately as $1 + 1/k_m$.

At moderate costs in complexity it is not difficult to fix up the FFI algorithm so that worst-case performance is likely to be improved. However, the new algorithms normally become very hard to analyze. A good example, motivated by classical bin-packing algorithms [4], is constructed as follows from the so-called first-fit-decreasing (FFD) rule.

Given a list $L=(p_1, \ldots, p_n)$ in non-decreasing order of piece-size, the FFD rule first finds the maximum-length prefix $L^{(1)}=(p_1, \ldots, p_{t_1}) \subseteq L$ such that $\sum_{i=1}^{t_1} \text{size}(p_i) \leqq m$. The algorithm then packs $L^{(1)}$ into as many, say m', bins as required, by scanning right-to-left, and placing the next smaller piece into that bin with lowest index into which it will fit. The algorithm terminates successfully if $L^{(1)}$ has been packed into $m' \leqq m$ bins. Otherwise, the algorithm constructs $L^{(2)} \subset L^{(1)}$ by discarding the largest piece in $L^{(1)}$, and then proceeds as above to pack $L^{(2)}$. This process is repeated until for some j, $L^{(j)}$ has been packed into $m' \leqq m$ bins.

The above use of the FFD rule is obviously more time-consuming than the FFI rule. In particular, there are examples showing that the FFD rule can require as many as m passes. (Consider the example list of $2m$ pieces each of size $\frac{1}{2}+\varepsilon$ for some small $\varepsilon > 0$.) Thus, the worst-case time complexities of the FFI and FFD algorithms are respectively $O(n \log_2 n)$ and at least $O(n \log_2 n + mn \log_2 m)$.

These observations may be of little moment, however, if the expected performance of the FFD rule is significantly better than that of the FFI rule. In this regard we have been able to prove only that the FFD rule always packs at least as many pieces as the FFI rule (and hence the bounds of Theorem 2 and 3 also apply to FFD packings[1]). On the other hand, we have been unable to come up with examples violating the inequality $n_o \leqq \frac{8}{7} n_{\text{FFD}} + 1$. In view of the difficulties experienced with earlier analyses of the FFD rule in connection with other bin-packing problems [1-4], the prospects for proving the $\frac{8}{7}$ asymptotic bound would not appear very encouraging.

References

1. Coffman, E.G., Garey, M.R., Johnson, D.S.: An application of bin-packing to multiprocessor scheduling. SIAM J. Comput. (to appear)
2. Coffman, E.G., Jr., Leung, J.Y-T., Ting, D.: Bin-packing: Maximizing the number of pieces packed. Technical Report, Computer Science Dept., The Pennsylvania State Univ., 1976
3. Graham, R.L.: Bounds on the performance of scheduling algorithms. In: Computer and job-shop scheduling theory (E.G. Coffman, ed.). New York:Wiley 1976
4. Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L.: Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J. Comput. **3** 299–326 (1974)

Received August 11, 1976

[1] From Theorem 2 and the example in Figure 3 it follows also that $1 \leqq n_{\text{FFD}}/n_{\text{FFI}} \leqq \frac{4}{3}$ and both bounds are achievable