A Simple On-Line Bin-Packing Algorithm

C. C. LEE AND D. T. LEE

Northwestern University, Evanston, Illinois

Abstract. The one-dimensional on-line bin-packing problem is considered. A simple $O(1)$-space and $O(n)$-time algorithm, called HARMONIC$_M$, is presented. It is shown that this algorithm can achieve a worst-case performance ratio of less than 1.692, which is better than that of the $O(n)$-space and $O(n \log n)$-time algorithm FIRST FIT. Also shown is that 1.691... is a lower bound for *all* $O(1)$-space on-line bin-packing algorithms. Finally a revised version of HARMONIC$_M$, an $O(n)$-space and $O(n)$-time algorithm, is presented and is shown to have a worst-case performance ratio of less than 1.636.

Categories and Subject Descriptors: F2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms—*sequencing and scheduling*; G2.m [**Discrete Mathematics**]: Miscellaneous

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Bin packing, on line, suboptimal algorithms

1. *Introduction*

The classical (one-dimensional) bin-packing problem is to pack a list of items $L = (a_1, a_2, \ldots, a_n)$, where $a_i \in (0, 1]$ for all i, into a minimum number of bins each of unit capacity. This problem arises in a wide variety of contexts and has been studied extensively since the early 1970s. Since this problem has been shown to be NP-complete [3, 8], various heuristic algorithms with guaranteed bounds on performance have been proposed [5–7]. Summaries of these results can be found in the surveys by Garey and Johnson [4] and Coffman et al. [2].

Let L^* and $A(L)$ denote, respectively, the number of bins used by an optimum algorithm and the number of bins used by a heurustic algorithm A to pack the input list L. Then, the worst-case performance ratio of A, denoted by $r(A)$, is defined as $\lim_{L^* \to \infty} \sup_L [A(L)/L^*]$. This ratio is customarily used to evaluate the performance of a heuristic bin-packing algorithm.

In this paper we deal only with *on-line* bin packing for which items in the list must be processed in exactly the same order as they are given, one at a time. The on-line processing is difficult owing to unpredictable item sizes that may appear. In general, the performance of an on-line bin-packing algorithm is substantially affected by the permutation of items in a given list. The *NEXT-FIT* and *FIRST-FIT* are two well-known on-line bin-packing algorithms [2, 4]. It is not difficult to

The research of C. C. Lee was supported in part by the National Science Foundation under Grant ESC-8307264. The research of D. T. Lee was supported in part by the National Science Foundation under Grants MCS-8202359 and ECS-8121741.

Authors' address: Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60201.

prove that r(NEXT-FIT) = 2, while the proof of r(FIRST-FIT) = 1.7 is fairly involved [7].

A question was raised by Johnson [6] as to whether or not there exists a polynomial-time on-line algorithm A better than FIRST-FIT (i.e., with $r(A) < 1.7$) and was recently resolved by Yao [12] in the affirmative. To be precise, Yao has presented an algorithm, dubbed *REFINED FIRST-FIT*, with a worst-case performance ratio of $5/3 = 1.666\ldots$, which is the best ratio achieved by an on-line bin-packing algorithm known to date. In the same paper [12], Yao also established that any on-line algorithm A must have a worst-case performance ratio of at least 1.5. The bound was later improved to 1.536 ... by Liang [11] and Brown [1].

In addition to the performance ratio, we also consider the time and the space complexity of on-line bin-packing algorithms. We use $T_A(n)$ to denote the total time required by on-line bin-packing algorithm A to pack the list L whose size is n, and refer to algorithm A as a $T_A(n)$-time algorithm. Thus, NEXT-FIT is an $O(n)$-time algorithm, whereas FIRST-FIT and REFINED FIRST-FIT are both $O(n \log n)$-time algorithms. For convenience, a nonempty bin during the processing is called "filled" if it is not intended to pack any more items and "active" if it is. Using the uniform cost criterion for space, we need one storage location for each active bin. As soon as a bin becomes filled, it is among the output of the algorithm used, and we do not count its storage location in defining the space complexity of the algorithm. Specifically, we use $S_A(n)$ to denote the maximum number of storage locations (for active bins) needed by algorithm A during the processing of the list L whose size is n, and refer to algorithm A as an $S_A(n)$-space algorithm. For example, NEXT-FIT is an $O(1)$-space algorithm, since it involves only one active bin at all times, and FIRST-FIT is an $O(n)$-space algorithm, since in this case all nonempty bins remain active until the end of the processing.

In this paper we present a new on-line bin-packing algorithm, called *HARMONIC$_M$*, which is based on a special, nonuniform partitioning of the interval (0, 1] into M subintervals, where M is not to exceed 12 in practice. The algorithm uses no more than the constant M storage locations and runs in $O(n)$ time. The worst-case performance ratio is shown to be strictly better than that of FIRST-FIT for $M > 6$. Specifically, it is shown that $r(\text{HARMONIC}_M) = 1.692\ldots$ for $M = 12$, and that $\lim_{M\to\infty} r(\text{HARMONIC}_M) = 1.6910\ldots$. Moreover, we show that 1.6910 ... is the greatest lower bound for all $O(1)$-space on-line bin-packing algorithms. It can be shown [10] that the expected-case performance ratio of HARMONIC$_M$ is substantially better than that of NEXT-FIT when the item sizes are uniformly distributed. Finally, a slightly modified version of HARMONIC$_M$, called *REFINED-HARMONIC, is presented. It is an* $O(n)$-space and $O(n)$-time algorithm and its worst-case performance ratio is shown to be better than 1.636.

2. *The Algorithm HARMONIC$_M$*

Given a list $L = (a_1, a_2, \ldots, a_n)$, where $a_i \in (0, 1]$ for all i, we classify each item a_i according to the following "harmonic partitioning" of the unit interval (0, 1]: $(0, 1] = \bigcup_{k=1}^{M} I_k$, where $I_k = (1/(k+1), 1/k]$, $1 \le k < M$ and $I_M = (0, 1/M]$, where M is a positive integer. An item a_i will be called an I_k-piece if $a_i \in I_k$, $1 \le k \le M$. Similarly, the bins are also classified into M categories. A bin designated to pack I_k-pieces exclusively is called an I_k-bin. Note that each I_k-bin packs at most k I_k-pieces for $1 \le k < M$ and is referred to as being filled if it has exactly k I_k-pieces, and unfilled otherwise. Let m_k denote the number of I_k-bins used by the algorithm, $1 \le k \le M$. In the following algorithm, we keep at all times an active (unfilled) I_k-bin for each $1 \le k \le M$.

Algorithm HARMONIC$_M$

For $k := 1$ **to** M **do** $m_k \leftarrow 0$
For $i := 1$ **to** n **do**
  If a_i is an I_k-piece, $1 \le k < M$
  then begin place a_i into the I_k-bin
    if the I_k-bin is filled
    then $m_k \leftarrow m_k + 1$ and get a new I_k-bin
  end
  else (*Comment*: a_i is an I_M-piece)
    begin
      if there is room for a_i in the I_M-bin
      then pack it
      else $m_M \leftarrow m_M + 1$ and get a new I_M-bin
  end

This algorithm essentially performs item classification for each incoming item and then packs it by NEXT-FIT into a corresponding bin. Since item classification can be done in $O(\log M)$ time, and there are only M active bins at any time, the algorithm runs in $O(n \log M)$ time and uses M storage spaces for active bins. As is shown later, the worst-case performance ratio of the algorithm is insensitive to the partition number M; a reasonable and practical range of M is $3 \le M \le 12$. Therefore, M can be regarded as a constant, and hence we have an $O(1)$-space and $O(n)$-time algorithm. Another crucial advantage of this algorithm is that each filled I_k-bin, $1 \le k < M$, packs exactly k items, irrespective of the actual sizes of these items in the interval I_k. Also, except for I_M-bins, the number of bins used by HARMONIC$_M$ is completely independent of the arriving order of items. In other words, except for I_M-bins, any permutation of the items in L will result in the same number of bins used by HARMONIC$_M$. These advantages make the analysis of its worst-case performance relatively easy and is the motivation of doing the harmonic partitioning.

3. *Worst-Case Analysis of HARMONIC$_M$*

Let n_k denote the number of I_k-pieces in the list L, for $1 \le k \le M$, and s_M denote the total size of I_M-pieces, that is, $s_M = \sum_{a_i \in I_M} a_i$. The total number of bins used by HARMONIC$_M$, denoted HARMONIC$_M(L)$, is $\sum_{k=1}^{M} m_k$, where m_k is the number of I_k-bins used. It is easy to see that

$$m_1 = n_1 \tag{3.1}$$

and for $2 < k < M$,

$$m_k = \left\lceil \frac{n_k}{k} \right\rceil, \tag{3.2}$$

where $\lceil x \rceil$ denotes the smallest integer no less than x.

Since the size of each I_M-piece is at most $1/M$, each filled I_M-bin must be packed to a capacity greater than $(M-1)/M$. We thus have

$$m_M < \left\lceil \frac{s_M}{(M-1)/M} \right\rceil. \tag{3.3}$$

Combining (3.1)–(3.3) we have

$$\begin{aligned} \text{HARMONIC}_M(L) &= n_1 + \left\lceil \frac{n_2}{2} \right\rceil + \cdots + \left\lceil \frac{n_{M-1}}{M-1} \right\rceil + m_M \\ &< n_1 + \frac{n_2}{2} + \cdots + \frac{n_{M-1}}{M-1} + \frac{M s_M}{M-1} + (M-1). \end{aligned} \tag{3.4}$$

The last term $(M - 1)$ of (3.4) accounts for the total number of possibly unfilled bins at the end of the processing and can be ignored when asymptotic performance ratio is considered. Asymptotically, each I_k-piece costs HARMONIC$_M$ $1/k$ bin, for $1 \le k < M$, and an I_M-piece of size y will cost HARMONIC$_M$ no more than $My/(M-1)$ of a bin. We therefore define a *fraction function* g in (0, 1] as

$$g(x) = \frac{1}{k} \qquad \text{if} \quad x \in I_k \quad \text{and} \quad 1 \le k < M$$

$$= \frac{Mx}{M-1} \qquad \text{if} \quad x \in I_M. \tag{3.5}$$

Consider an optimal packing that uses L^* bins. Let the content of the ith bin be denoted as $(y_{i1}, y_{i2}, \ldots, y_{it_i})$, $t_i > 0$, with $y_{i1} \ge y_{i2} \ge \cdots \ge y_{it_i} > 0$. Since each bin has a maximum capacity 1, we have $y_{i1} + y_{i2} + \cdots + y_{it_i} \le 1$. Thus, the list $L = (a_1, a_2, \ldots, a_n)$ of items can be expressed as the concatenation of L^* lists, $(y_{i1}, y_{i2}, \ldots, y_{it_i})$, $i = 1, 2, \ldots, L^*$, each of which corresponds to the content of a bin in the optimal packing. In terms of the fraction function g, the number of bins used by HARMONIC$_M$, for a given M, to pack $(y_{i1}, y_{i2}, \ldots, y_{it_i})$ can be written as

$$G_{i,M} = \sum_{m=1}^{t_i} g(y_{im}), \qquad i = 1, 2, \ldots, L^*.$$

Let $S = \{(y_1, y_2, \ldots, y_t) \mid y_i > 0,\ 1 \le i \le t, \text{ and } \sum_{i=1}^{t} y_i \le 1\}$ be the set containing all possible partitions of 1, and let $\bar{G}_M = \sup_S \sum_{i=1}^{t} g(y_i)$. Thus, we have $G_{i,M} \le \bar{G}_M$ for all i and we can therefore rewrite (3.4) as

$$\begin{aligned} \text{HARMONIC}_M(L) &< \sum_{i=1}^{n} g(a_i) + (M-1) \\ &= \sum_{i=1}^{L^*} G_{i,M} + (M-1) \\ &\le (\bar{G}_M)L^* + (M-1). \end{aligned} \tag{3.6}$$

This implies that $\bar{G}_M$ is an upper bound for the worst-case performance ratio for HARMONIC$_M$ since

$$r(\text{HARMONIC}_M) = \lim_{L^* \to \infty} \sup_L \frac{\text{HARMONIC}_M(L)}{L^*} \le \bar{G}_M.$$

Therefore, we shall concentrate on finding the bound on the number $G_M = \sum_{i=1}^{t} g(y_i)$, given $\sum_{i=1}^{t} y_i \le 1$.

For $x \in I_k$, $1 \le k < M$, we have $x > 1/(k+1)$ and $g(x) = 1/k$ and, therefore, $g(x)/x = 1/(kx) < (k+1)/k$. Since $(k+1)/k$ is monotone decreasing with k, we have the following inequality:

$$\frac{g(x)}{x} \le \frac{k+1}{k} \qquad \text{if} \quad x < \frac{1}{k} \quad \text{and} \quad 1 \le k < M. \tag{3.7}$$

Define the sequence k_i, as follows:

$$k_1 = 1 \qquad \text{and} \qquad k_{i+1} = k_i(k_i + 1) \qquad \text{for} \quad i \ge 1. \tag{3.8}$$

It is not difficult to see that

$$\frac{1}{k_i} = 1 - \sum_{j=1}^{i-1} \frac{1}{k_j + 1} \qquad \text{for} \quad i \ge 2. \tag{3.9}$$

We now prove our main result.

THEOREM 1. *For $i > 1$, $k_i < M \le k_{i+1}$, where k_i is given by* (3.8),

$$r(HARMONIC_M) \le \bar{G}_M = \sum_{j=1}^{i} \frac{1}{k_j} + \frac{M}{k_{i+1}(M-1)}.$$

PROOF. Let $Q_M = \sum_{j=1}^{i} 1/k_j + M/(k_{i+1}(M-1))$. First, if $y_1 \notin I_1$, then $y_i < 1/2 = 1/k_2$ for $1 \le i \le t$. From (3.7), we have $g(y_i) \le 3y_i/2$ and therefore $G_M \le 3/2 \sum_{i=1}^{t} y_i \le 3/2 \le Q_M$ for $M > k_2$. Next let us assume that $y_1 \in I_{k_1}$, $y_2 \in I_{k_2}, \ldots, y_{j-1} \in I_{k_{j-1}}$, and $y_j \notin I_{k_j}$, for some $j \le i$. Then $\sum_{s=j}^{t} y_s \le 1/k_j$ and $y_s < 1/(k_j + 1)$ for all $s \ge j$. Thus, from (3.7) and (3.8), we have

$$\begin{aligned} G_M &\le \sum_{s=1}^{j-1} \frac{1}{k_s} + \sum_{s=j}^{t} \frac{y_s(k_j+2)}{k_j+1} \\ &= \sum_{s=1}^{j-1} \frac{1}{k_s} + \frac{1 + 1/(k_j+1)}{k_j} \\ &= \sum_{s=1}^{j+1} \frac{1}{k_s} \le \sum_{s=1}^{i+1} \frac{1}{k_s} < Q_M. \end{aligned}$$

Finally, suppose that $y_1 \in I_{k_1}, y_2 \in I_{k_2}, \ldots, y_i \in I_{k_i}$. Then we have $\sum_{s=k_i+1}^{t} y_s < 1/k_{i+1}$, and since $M \le k_{i+1}$, $y_s \in I_M$ for $s > k_i$. Thus, $G_M = \sum_{j=1}^{i} 1/k_j + M/(M-1) \cdot \sum_{s=k_i+1}^{t} y_s < Q_M$. In addition, it is easy to verify that $\lim_{\epsilon \to 0} G_M = Q_M$ if $y_j = 1/(k_j + 1) + \epsilon$ for $1 \le j \le i$ and $\sum_{j=i+1}^{t} y_s = 1/(k_{j+1}) - \epsilon$. We conclude that $\bar{G}_M = Q_M$. Q.E.D.

COROLLARY 1

$$\begin{aligned} \lim_{M \to \infty} r(HARMONIC_M) \le \bar{G}_\infty &= \sum_{j=1}^{\infty} \frac{1}{k_j} \\ &= 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{42} + \frac{1}{42 \cdot 43} + \cdots \\ &= 1.6910\ldots. \end{aligned}$$

The numerical values of the performance bound $\bar{G}_M$ for various M are listed in Table I. In view of Corollary 1 and $\bar{G}_{12} = 1.6926\ldots$ (see Table I), little improvement on the performance is achieved beyond $M = 12$. Since a smaller M is more desirable from a practical viewpoint, the region $3 \le M \le 12$ is recommended for practical use of HARMONIC$_M$. When $M = k_{i+1}$, $i \ge 2$, we prove in the following theorem that the bound for r(HARMONIC$_M$) in Theorem 1 is tight.

THEOREM 2. *For $M = k_{i+1}$, $i \ge 2$, $r(HARMONIC_M) = \bar{G}_M = \sum_{j=1}^{i} 1/k_j + 1/(M-1)$.*

PROOF. Consider items of $i + 2$ sizes: $a_j = 1/(k_j + 1) + 1/c$, $1 \le j \le i$, $a_{i+1} = 1/k_{i+1} - i/c$, and $a_{i+2} = 1/d$, where c and d are sufficiently large integers. Note that $\sum_{j=1}^{i+1} a_j = 1$ and both a_{i+1} and a_{i+2} are I_M-pieces. Assume that the list L consists of X a_j's, for each $1 \le j \le i + 1$, and $Y = X/(M-1)$ a_{i+2}'s and that the arrival pattern of the a_{i+1}'s and a_{i+2}'s is a sequence of $(M-1)$ a_{i+1}'s followed by one a_{i+2}, which repeats Y times. Thus, algorithm HARMONIC$_M$ packs exactly $(M-1)$ a_{i+1}'s and one a_{i+2} into one bin and uses a total of HARMONIC$_M(L) = \sum_{j=1}^{i}$

TABLE I. THE WORST-CASE PERFORMANCE BOUNDS FOR ALGORITHM HARMONIC$_M$ WITH PARTITION NUMBER M

M	$\bar{G}_M$
3	1.75
4	1.7222...
5	1.7083...
6	1.7
7	1.6944...
8	1.6938...
9	1.6934...
10	1.6931...
11	1.6928...
12	1.6926...
42	1.6910...
∞	1.6910...

$X/k_j + X/(M-1)$ bins, whereas an optimal algorithm uses $L^* = X + X/((M-1)d)$ bins. Therefore,

$$\frac{\text{HARMONIC}_M(L)}{L^*} = \frac{\sum_{j=1}^{i} 1/k_j + 1/(M-1)}{1 + 1/((M-1)d)}$$

and the claim follows since d can be arbitrarily large. Q.E.D.

4. *The Asymptotic Optimality of HARMONIC$_M$*

In comparison with the lower bound 1.536 ... for the worst-case ratio of all on-line bin-packing algorithms established by Liang [11] and Brown [1], the number $\bar{G}_\infty$ of Corollary 1 is still about 10 percent larger. However, we show that *no* $O(1)$-space on-line bin-packing algorithm can do better than $\bar{G}_\infty$.

Let $x_1 = 1/2 + e$, $x_2 = 1/3 + e$, and $x_3 = 1/6 - 2e$, where e is a sufficiently small positive number. Consider a list L containing $n/3$ x_1's, followed by $n/3$ x_2's, and then $n/3$ x_3's. With no exception, any on-line bin-packing algorithm must use $n/3$ bins to finish packing x_1 pieces. Under the assumption that the maximum number T of storage locations for active bins allowed is a constant (i.e., $O(1)$-space) we can only keep T of these $n/3$ bins for further packing. Therefore, *any* on-line algorithm requires more than $(n/3 - 2T)/2$ bins to pack all x_2's. Since no more than T nonempty bins can be kept, *any* algorithm requires more than $(n/3 - 6T)/6$ bins to pack all x_3's. As a result, *any* on-line algorithm requires more than $(1 + 1/2 + 1/6)n/3 - 2T$ bins to pack L. Since $a_1 + a_2 + a_3 = 1$, an optimum packing requires only $n/3$ bins. Therefore, the worst-case performance ratio is always larger than $1 + 1/2 + 1/6$. In general, let $x_j = 1/(k_j + 1) + e$ for $1 \le j \le i - 1$ and $x_i = 1/k_i - (i-1)e$, where e is a sufficiently small positive number. It follows from (3.9) that $\sum_{j=1}^{i} x_j = 1$. If the list L contains n/i x_j's for each $1 \le j \le i$ and smaller pieces always come after larger ones, then it is easily derived that *any* on-line algorithm requires more than $\sum_{j=1}^{i} (n/i - k_j T)/k_j$ bins to pack L. In comparison with optimum packing, which requires only n/i bins, it follows that any on-line bin-packing algorithm cannot have a worst-case performance ratio better than $\sum_{j=1}^{i} 1/k_j$. Since i can be arbitrarily large, we arrive at the following conclusion.

THEOREM 3. *Let A be any $O(1)$-space on-line bin-packing algorithm. Then $r(A) \ge \bar{G}_\infty = 1.6910\ldots$.*

COROLLARY 2. *The number* $\bar{G}_\infty = 1.6910\ldots$ *is the greatest lower bound for the worst-case performance ratio of any* $O(1)$*-space on-line bin-packing algorithm.*

PROOF. Corollary 1 implies that, for any $\delta > 0$, there exists an integer N such that $|r(\text{HARMONIC}_M) - \bar{G}_\infty| < \delta$ for all $M > N$. The corollary holds since HARMONIC_M is an $O(1)$-space algorithm and $\bar{G}_\infty$ is a lower bound for $r(\text{HARMONIC}_M)$.

As a result, if only a constant amount of temporary space is allowed, the algorithm HARMONIC_M is asymptotically optimal as $M \to \infty$. Q.E.D.

5. *REFINED-HARMONIC*

In this section, we modify, at the expense of storage space, the algorithm HARMONIC_M to improve the worst-case performance ratio to 1.6359 ..., which is better than the ratio 1.666... of REFINED-FIRST-FIT [12]. Indeed, the algorithm REFINED-HARMONIC to be presented is an $O(n)$-space and $O(n)$-time algorithm.

In algorithm HARMONIC_M, and I_1-piece whose size is slightly over 1/2 occupies one bin alone; this seems too wasteful. If this category of I_1-pieces can be properly packed with certain smaller pieces, an improvement on the worst-case performance can be expected. To achieve this, the unit interval is now partitioned into $M + 2$ subintervals as follows: $(0, 1] = (\bigcup_{k=1}^{M} J_k) \cup J_a \cup J_b$, where $J_1 = (59/96, 1]$, $J_a = (1/2, 59/96]$, $J_2 = (37/96, 1/2]$, $J_b = (1/3, 37/96]$, $J_k = (1/(k+1), 1/k]$, for $k = 3, 4, \ldots, M-1$, and $J_M = (0, 1/M]$, where M is chosen to be 20. In comparison with the harmonic partition of Section 2, it is seen that $J_k = I_k$ for $3 \le k \le M$ and that $I_1 = J_1 \cup J_a$ and $I_2 = J_2 \cup J_b$. The packing strategy for J_k-pieces, $1 \le k \le M$, will be the same as that for I_k-pieces in HARMONIC_M but J_a- and J_b-pieces will be "mixed." In other words, the intention is to pack small I_1-pieces (i.e., J_a-pieces) with small I_2-pieces (i.e., J_b-pieces) in some predetermined manner. Note that the sum of a J_a-piece and a J_b-piece never exceeds unity.

As before, a bin used to pack J_k-pieces exclusively is called a *J_k-bin*, $1 \le k \le M$; each J_k-bin can pack exactly k J_k-pieces, for $1 \le k < M$, and a J_M-bin can pack at least to a capacity $(M-1)/M$. There are four possibilities for a bin used to pack J_a-pieces and J_b-pieces:

(1) A J_a-bin contains a J_a-piece only.
(2) A J_b-bin contains a J_b-piece only.
(3) A J_{ab}-bin contains one J_a-piece and one J_b-piece.
(4) A J_{bb}-bin contains two J_b-pieces.

For convenience, a bin is classified as a $J_{b'}$-bin, instead of a J_b-bin, if it contains a J_b-piece only, but is designated to become a J_{bb}-bin as soon as the next J_b-piece is received. In REFINED-HARMONIC, the number of $J_{b'}$-bin never exceeds one. We use N_a, N_b, N_{ab}, N_{bb}, and $N_{b'}$ to count the numbers of J_a-bins, J_b-bins, J_{ab}-bins, J_{bb}-bins, and $J_{b'}$-bins, respectively, during the processing. Also, let $N_c = N_b + N_{ab}$. Given the list $L = (a_1, a_2, \ldots, a_n)$, we now describe the algorithm.

Algorithm REFINED-HARMONIC

1. $N_a = N_b = N_{ab} = N_{bb} = N_{b'} = N_c = 0$
2. **If** a_i is a J_k-piece, $1 \le k \le M$
   then use algorithm HARMONIC_M to pack it

3. **else if** a_i is a J_a-piece
   then if $N_b \neq 0$, **then** pack a_i into *any* J_b-bin; $N_b \leftarrow N_b - 1$; $N_{ab} \leftarrow N_{ab} + 1$
   else place a_i in a new (empty) bin; $N_a \leftarrow N_a + 1$
4. **else if** a_i is a J_b-piece
   then if $N_{b'} = 1$
   then pack a_i into the $J_{b'}$-bin; $N_{b'} \leftarrow 0$; $N_{bb} \leftarrow N_{bb} + 1$
5. **else if** $N_{bb} \leq 3N_c$
   then place a_i in a new bin and designate it as a $J_{b'}$-bin; $N_{b'} \leftarrow 1$
   else if $N_a \neq 0$
   then pack a_i into *any* J_a-bin; $N_a \leftarrow N_a - 1$; $N_{ab} \leftarrow N_{ab} + 1$; $N_c \leftarrow N_c + 1$
   else place a_i in a new bin; $N_b \leftarrow N_b + 1$; $N_c \leftarrow N_c + 1$

end

It is obvious that each item takes $O(1)$ time to pack and REFINED-HARMONIC is therefore an $O(n)$-time algorithm. However, the algorithm has to store all J_a-bins and J_b-bins during the processing and thus needs $O(n)$-space.

6. *Worst-Case Analysis of REFINED-HARMONIC*

For $1 \leq k \leq M$, let n_k be the number of J_k-pieces in L and m_k the total number of J_k-bins used by REFINED-HARMONIC. Also, let s_M be the total size of all J_M-pieces in L. Since algorithm HARMONIC$_M$ is used to pack these pieces and bins, we have from Section 2 that $m_k = \lceil n_k/k \rceil < n_k/k + 1$, for $1 \leq k \leq M - 1$ and $m_M < Ms_M/(M - 1) + 1$. Let n_a and n_b denote, respectively, the numbers of J_a-pieces and J_b-pieces in the list L. Let m_a, m_b, m_{ab}, m_{bb}, $m_{b'}$, and m_c be the final values (i.e., the values at the end of processing) of N_a, N_b, N_{ab}, N_{bb}, $N_{b'}$, and N_c, respectively. Note that $m_c = m_b + m_{ab}$ and

$$n_b = 2m_{bb} + m_c \tag{6.1a}$$

and

$$n_a = m_a + m_{ab}. \tag{6.1b}$$

Essentially, the algorithm is designed to yield a ratio of m_{bb} to m_c equal to 3 asymptotically (step 5). Specifically, the algorithm maintains $3(N_c - 1) < N_{bb} < 3(N_c + 1)$. Thus we have

$$-3 \leq m_{bb} - 3m_c \leq 3. \tag{6.2}$$

It follows from (6.1a) and (6.2) that

$$\frac{3n_b}{7} - \frac{3}{7} \leq m_{bb} \leq \frac{3n_b}{7} + \frac{3}{7} \tag{6.3}$$

and that

$$\frac{n_b}{7} - \frac{6}{7} \leq m_c \leq \frac{n_b}{7} + \frac{6}{7}. \tag{6.4}$$

It is also clear that the algorithm ensures that a J_a-bin and a J_b-bin never coexist. In other words, we have $N_aN_b = 0$ all the time and thus

$$m_am_b = 0. \tag{6.5}$$

The total number of bins used by REFINED-HARMONIC to pack L is given by

$$\begin{aligned}\text{REFINED-HARMONIC}(L) &= \sum_{k=1}^{M} m_k + m_a + m_{bb} + m_c + m_b, \\ &< \sum_{k=1}^{M-1} \frac{n_k}{k} + \frac{Ms_M}{M-1} + (M-1) \\ &\quad + m_a + m_{bb} + m_c + 1. \end{aligned} \tag{6.6}$$

To proceed further, we need to distinguish two cases based on (6.5).

Case 1. $m_a = 0$. In this case, it follows from (6.3), (6.4), and (6.6) that

$$\begin{aligned}&\text{REFINED-HARMONIC}(L) \\ &\quad < \sum_{k=1}^{M-1} \frac{n_k}{k} + \frac{Ms_M}{M-1} + (M-1) + \frac{4n_b}{7} + \frac{9}{7} + 1. \end{aligned} \tag{6.7}$$

In parallel with the fraction function g of Section 2, we define h_1 as

$$\begin{aligned} h_1(x) &= \frac{1}{k} = g(x) && \text{if } x \in J_k \quad \text{and} \quad 1 \le k \le M-1 \\ &= \frac{Mx}{M-1} = g(x) && \text{if } x \in J_M \\ &= \frac{4}{7} && \text{if } x \in J_b \\ &= 0 && \text{if } x \in J_a, \end{aligned} \tag{6.8}$$

which represents the fraction of a bin costed by an item of size x in REFINED-HARMONIC.

Case 2. $m_b = 0$. In this case, we have $m_c = m_{ab}$. Therefore, it follows from (6.1b), (6.3), and (6.6) that

$$\begin{aligned}&\text{REFINED-HARMONIC}(L) \\ &\quad < \sum_{k=1}^{M-1} \frac{n_k}{k} + \frac{Ms_M}{M-1} + (M-1) + n_a + \frac{3n_b}{7} + \frac{3}{7} + 1. \end{aligned} \tag{6.9}$$

The corresponding fraction function is thus given by

$$\begin{aligned} h_2(x) &= \frac{1}{k} = g(x) && \text{if } x \in J_k \quad \text{and} \quad 1 \le k \le M-1 \\ &= \frac{Mx}{M-1} = g(x) && \text{if } x \in J_M \\ &= \frac{3}{7} && \text{if } x \in J_b \\ &= 1 && \text{if } x \in J_a. \end{aligned}$$

Having defined h_1 and h_2, we are ready to study the worst-case performance ratio of REFINED-HARMONIC. As before, let $(y_1, y_2, \ldots, y_t)$ be the content of an arbitrary bin in an optimum packing, where $y_1 \ge y_2 \ge \cdots \ge y_t$. Let H_j denote the supremum of $\sum_{i=1}^{t} h_j(y_i)$ for all $y_1, y_2, \ldots, y_t$, $j = 1, 2$. Then, using the same

arguments of Section 3, we have

$$r(\text{REFINED-HARMONIC}) < \max\{H_1, H_2\}. \tag{6.10}$$

LEMMA 1. $H_1 < 1.63$.

PROOF. It is easy to verify that $h_1(x)/x \le 96/59$ for $x \notin J_b$. This implies that $\sum_{i=1}^{t} h_1(y_i) \le 96/59 = 1.627\ldots$ if there is no J_b-piece among the y's. Now assume that there is at least one J_b-piece in $\{y_1, y_2, \ldots, y_t\}$. Then we have four possible cases as follows:

Case 1. $y_1 \in J_1$. Then $y_2 \in J_b$ and $y_3 + \cdots + y_t < 5/96 \in J_{20} = J_M$. Thus, we have $\sum_{i=1}^{t} h_1(y_i) < 1 + 4/7 + 5/96 \cdot 20/19 = 1.626\ldots$.

Case 2. $y_1 \in J_a$. Then, $h_1(y_1) = 0$ and $y_2 + y_3 + \cdots + y_t < 1/2$. Since $h_1(x)/x \le 12/7$ for all $x \in (0, 1]$, we have $\sum_{i=1}^{t} h_1(y_i) < 1/2 \cdot 12/7 < 1$.

Case 3. $y_1 \in J_2$. Then $y_2 \in J_b$ and $y_3 + \cdots + y_t < 27/96$. Since $h_1(x) = g(x)$ for $x < 27/96$ and since (3.7) implies that $g(x)/x \le 6/5$ for $x < 27/96 < 1/5$, we have $\sum_{i=1}^{t} h_1(y_i) < 1/2 + 4/7 + 27/96 \cdot 6/5 = 1.4089\ldots$.

Case 4. $y_1 \in J_b$. If $y_2 \in J_b$, then $y_3 + \cdots + y_t < 1/3$ and we have $\sum_{i=1}^{t} h_1(y_i) < 4/7 + 4/7 + 1/3 \cdot 4/3 < 1.6$, since $h_1(x)/x = g(x)/x < 4/3$ for $x < 1/3$. If $y_2 \notin J_b$, then $y_2 < 1/3$ and $\sum_{i=1}^{t} h_1(y_i) < 4/7 + 2/3 \cdot 4/3 < 1.5$. Q.E.D.

LEMMA 2. $H_2 < 373/228 = 1.6359\ldots$.

PROOF. It is easy to verify that $h_2(x)/x \le 96/59$ for $x \notin J_a$. This implies that $\sum_{i=1}^{t} h_2(y_i) \le 96/59 = 1.627\ldots$ if $y_1 \notin J_a$. Assume that $y_1 \in J_a$. Then we have four possible cases as follows:

Case 1. $y_2 \in J_2$. Then $y_3 + \cdots + y_t \le 11/96 \in J_8$. Since $h_2(x)/x \le 9/8$ for $x < 11/96 < 1/8$, we have $\sum_{i=1}^{t} h_2(y_i) < 1 + 1/2 + 11/96 \cdot 9/8 < 1.63$.

Case 2. $y_2 \in J_b$. Then $y_3 + \cdots + y_t < 1/6$. Since $h_2(x)/x \le 7/6$, for $x < 1/6$, we have $\sum_{i=1}^{t} h_2(y_i) < 1 + 3/7 + 1/6 \cdot 7/6 < 1.63$.

Case 3. $y_2 \in J_3$. Then $y_3 + \cdots + y_t < 1/4$. If $y_3 \in J_4$, then $y_4 + \cdots + y_t < 1/20 \in J_M$ and $\sum_{i=1}^{t} h_2(y_i) < 1 + 1/3 + 1/4 + 1/20 \cdot 20/19 = 373/228$. If $y_3 \notin J_4$, then $y_3 < 1/5$ and $\sum_{i=1}^{t} h_2(y_i) < 1 + 1/3 + 1/4 \cdot 6/5 = 1.6333\ldots$.

Case 4. $y_2 < 1/4$. Then since $h_2(x)/x < 5/4$ for $x < 1/4$, we have $\sum_{i=1}^{t} h_2(y_i) < 1 + 1/2 \cdot 5/4 = 1.625$. Q.E.D.

From these two lemmas, we have the following theorem.

THEOREM 4. *The worst-case performance ratio of REFINED-HARMONIC*

$$r(\textit{REFINED-HARMONIC}) \le \frac{373}{228} \ (= 1.6359\ldots) \qquad \text{for} \quad M = 20.$$

7. *Concluding Remarks*

We have presented an $O(1)$-space and $O(n)$-time on-line bin-packing algorithm whose performance is far better than that of NEXT-FIT. In particular, for $M > 6$, HARMONIC$_M$ has a worst-case performance ratio strictly less than 1.7, which is better than FIRST-FIT–an $O(n \log n)$-time algorithm. Also we have shown that $G_\infty (= 1.6910\ldots)$ is the greatest lower bound for the worst-case performance ratio for any $O(1)$-space on-line algorithm.

We have also presented an $O(n)$-space and $O(n)$-time algorithm and shown that its worst-case performance ratio is less than 1.636, which is better than any on-line bin-packing algorithm known to date.

As a possible direction of future research, one may try to modify HARMONIC$_M$ in a more sophisticated manner [9] so as to improve its worst-case performance ratio, possibly at the expense of time-complexity. Finally, we note that it is plausible to extend the idea to bin-packing problems in higher dimensions.

ACKNOWLEDGMENTS. The authors wish to thank Professor A. C. Yao and an anonymous referee for their comments on an earlier draft of the paper. Their suggestions have helped improve the presentation of the paper.

REFERENCES

1. BROWN, D. J. A lower bound for on-line one-dimensional bin packing algorithms. Tech. Rep. No. R-864, Coordinated Sci. Lab., Univ. of Illinois, Urbana, Ill., 1979.
2. COFFMAN, E. G., JR., GAREY, M. R., AND JOHNSON, D. S. Approximation algorithms for bin-packing—An updated survey. Bell Laboratories, Murray Hill, N.J., Oct. 1983.
3. GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman & Co., San Francisco, Calif., 1979.
4. GAREY, M. R., AND JOHNSON, D. S. Approximation algorithms for bin packing problems: A survey. In *Analysis and Design of Algorithms in Combinatorial Optimization*, G. Ausiello and M. Lucertini, Eds. Springer-Verlag, New York, 1981.
5. JOHNSON, D. S. Near optimal bin packing algorithms. Ph.D. dissertation, MIT, Cambridge, Mass., June 1973.
6. JOHNSON, D. S. Fast algorithms for bin packing. *J. Comput. Syst. Sci. 8* (1974), 272–314.
7. JOHNSON, D. S., DEMERS, A., ULLMAN, J. D., GAREY, M. R., AND GRAHAM, R. L. Worst-case performance bounds for simple one-dimensional bin packing algorithms. *SIAM J. Comput. 3* (1974), 299–325.
8. KARP, R. M. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller and J. M. Thatcher, Eds. Plenum Press, New York, 1972, pp. 85–103.
9. LEE, C. C., AND LEE, D. T. A new algorithm for on-line bin packing. Tech. Rep. No. 83-03-FC-02, Dept. of Electrical Engineering and Computer Science, Northwestern Univ., Evanston, Ill., Nov. 1983.
10. LEE, C. C., AND LEE, D. T. Robust on-line bin packing algorithms. Submitted for publication.
11. LIANG, F. M. A lower bound for on-line bin packing. *Inf. Proc. Lett. 10* (1980), 76–79.
12. YAO, A. C. New algorithms for bin packing. *J. ACM. 27* (1980), 207–227.

RECEIVED MAY 1983; REVISED OCTOBER 1984; ACCEPTED JANUARY 1985