Final draft of paper that appeared in ALGORITHMICA 31:2 (2001), 115-138

Bounded Space On-Line Bin Packing: Best is Better than First

J. Csirik[1] and D. S. Johnson[2]

Abstract

We present a sequence of new linear-time, bounded-space, on-line bin packing algorithms, the K-Bounded Best Fit algorithms (BBF_K). They are based on the $\Theta(n \log n)$ Best Fit algorithm in much the same way as the Next-K Fit algorithms are based on the $\Theta(n \log n)$ First Fit algorithm. Unlike the Next-K Fit algorithms, whose asymptotic worst-case ratios approach the limiting value of 17/10 from above as $K \rightarrow \infty$ but never reach it, these new algorithms have worst-case ratio 17/10 for all $K \geq 2$. They also have substantially better average performance than their bounded-space competition, as we have determined based on extensive experimental results summarized here for instances with item sizes drawn independently and uniformly from intervals of the form $(0,u]$, $0 < u \leq 1$. Indeed, for each $u < 1$, it appears that there exists a fixed memory bound $K(u)$ such that $\mathrm{BBF}_{K(u)}$ obtains significantly better packings on average than does the First Fit algorithm, even though the latter requires unbounded storage and has a significantly greater running time. For $u = 1$, BBF_K can still outperform First Fit (and essentially equal Best Fit) if K is allowed to grow slowly. We provide both theoretical and experimental results concerning the growth rates required.

[1] Department of Computer Sciences, University of Szeged, Szeged, Hungary. Research partially supported by a Fulbright grant from the Council for International Exchange of Scholars, by the NSF Science and Technology Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), and by AT&T Labs -- Research.

[2] AT&T Labs – Research, Florham Park, NJ 07932.

A preliminary, shortened version of this paper appeared under the same title in the *Proceedings of the 2nd Annual ACM-SIAM Symposium on Discrete Algorithms*, 1991, pp. 309-319.

1. Introduction

In the one-dimensional bin packing problem, we are given a list of items $L = a_1, a_2, \cdots, a_n$, each item a_i with a positive (rational) *size* $s(a_i) \le 1$, and asked to find a *packing* of these items into a minimum number of unit-capacity bins B_k, $k \ge 1$. Since the problem of finding an optimal packing is NP-hard, research has concentrated on approximation algorithms that find near-optimal packings. In this paper, we shall direct attention to a specially restricted class of such algorithms: ones that are ''on-line'' and use ''bounded space.''

Definition. *A bin packing algorithm is* on-line *if it assigns items to bins in the order* $(a_1, a_2, \cdots)$*, with item* a_i *assigned solely on the basis of the sizes of the preceding items and the bins to which they were assigned, without reference to the sizes or number of remaining items.*

Note that in an on-line algorithm, we may assume without loss of generality that no item is placed in bin B_{k+1}, $k \ge 1$, until an item has been placed in bin B_k. Thus, if in the packing so far the non-empty bins are B_1 through B_k, the possibilities for the next item are B_1 through B_{k+1}, with bin B_{k+1} being referred to as the potential *new* bin. In the algorithms we consider in this paper, the possibilities are further limited by a classification of the bins as *open* and *closed*. Each bin B_i becomes open when it receives its first item, and remains so until it is declared closed. Only open bins (and the potential new bin) may receive items. Once a bin is closed, it may never receive an item again.

Definition. *An on-line bin packing algorithm uses* K-bounded space *if at no time during its operation does the number of open bins exceed* K.

This bounded-space restriction arises in many applications. Consider the problem of packing trucks at a loading dock that has positions for only K trucks (the open bins). If the next item to be packed does not fit in any of the trucks currently backed up to the dock, we will need a new truck, and if there are already K trucks present, one of these will have to drive away (be closed) to make room (and presumably to start making its deliveries). Alternatively consider a communication channel in which information moves in large fixed-size blocks. If these blocks are filled with smaller packets of various sizes that must be assigned to blocks as they arrive at the entrance to the channel, we have an on-line bin packing problem. If the buffer for the channel input is of bounded size, we have a bounded-space on-line bin packing problem.

The first bounded-space on-line bin packing algorithms to be studied were the *Next-K Fit* (NF_K) algorithms, introduced in [12,13]. In Next-K Fit, if B_J is the current non-empty bin of highest index, then the set of open bins is $\{B_j: \max(1, J-K+1) \le j \le J\}$, and the item a_i to be packed is placed in the lowest-indexed bin among these into which it will fit. If no open bin has room for a_i, then it is placed in B_{J+1} (a new bin), and if $J \ge K$, bin B_{J-K+1} is closed. (Next-1 Fit is often referred to simply as ''Next Fit'' or ''NF'' for short.) These algorithms can be viewed as bounded-space restrictions of the familiar (unbounded-space) *First Fit* (FF) algorithm, which is equivalent to Next-∞ Fit. Not surprisingly, their worst-case ratios approach that of First Fit as $K \to \infty$.

To make this claim formally, we need some additional definitions. Let $A(L)$ be the number of bins used in the packing of L generated by algorithm A and $OPT(L)$ be the optimum number of bins.

Definition. $R_n[A] = max\left\{ \frac{A(L)}{OPT(L)} : OPT(L) = n \right\}$.

Definition. *The* asymptotic worst-case ratio *for algorithm A is* $R^\infty[A] = \lim\sup_{n\to\infty} R_n[A]$.

The precise values of $R^\infty[A]$ have been obtained for both FF and the NF_K algorithms:

Theorem [14]. $R^\infty[FF] = 1.7$.

Theorem [8,21]. *For* $K \geq 2$, $R^\infty[\mathrm{NF}_K] = 1.7 + \frac{3}{10(K-1)}$.

Csirik and Imreh [8] first proved the general lower bound for NF_K and the upper bound for NF_2; Mao [21] proved the general upper bound. (For $K = 1$, the result $R^\infty[NF] = 2$ is trivial to obtain, and was first reported in [12,13].) Note that in recompense for the degraded worst-case ratios of these algorithms, we obtain substantially faster running times, $O(n\log K)$ or simply $O(n)$ for each fixed K, in comparison to the $\Theta(n\log n)$ required by FF.

These are not the best possible bounded-space on-line algorithms, however. Even the 1.7 worst-case ratio for the (unbounded-space) FF algorithm can in fact be beaten (slightly). This was first observed by Lee and Lee [19], who devised a series of K-bounded-space ''Harmonic'' algorithms H_K whose asymptotic worst-case ratios approach a limiting constant $h_\infty \approx 1.69103...$ In algorithm H_K, the input items are divided into K different classes as they arrive, with items in each class packed separately according to Next Fit. The classification is by item size, with the size ranges being $(1/j, 1/(j-1)]$, $2 \leq j \leq K$, and $(0, 1/K]$. The worst-case behavior of the algorithms H_K is intimately connected with the sequence $<k_i> = 1,2,6,42,\cdots$ where $k_1 = 1$ and $k_{i+1} = k_i(k_i+1)$, $i > 1$. Let $h_j = \Sigma_{i=1}^{j}(1/k_i)$. Note that $h_1 = 1$, $h_2 = 3/2$, $h_3 = 5/3$, $h_4 = 1.6904..$, and $h_\infty = \lim_{j\to\infty} h_j = 1.69103...$

Theorem [19]. *Suppose* $k_i < K \leq k_{i+1}$ *for some* $i \geq 1$. *Then*

$$h_i + \frac{1}{k_{i+1} - 1} \leq R^\infty[\mathrm{H}_K] \leq h_i + \frac{K}{k_{i+1}(K-1)}$$

Note that the upper and lower bounds coincide when $K = k_{i+1}$, so for these choices of K the precise value of $R^\infty[\mathrm{H}_K]$ is known. The precise value can also be determined when $K = k_{i+1}+1$, as in this case it is easy to construct instances that show that the upper bound given by the theorem is tight. For $K = 4,5$, the precise values were recently derived by van Vliet [29]. For all other values of K, however, the question of the precise value of $R^\infty[\mathrm{H}_K]$ remains open, although we can construct examples improving the lower bounds of the theorem for many values of K.

The current best lower bounds for $R^\infty[\mathrm{H}_K]$, $2 \leq K \leq 10$, are given in the third column of Table 1 below. The entries in this table are rounded to four decimal places, but for those K with matching upper and lower bound entries in the table, the two bounds are in fact equal. Table 1 also includes entries for a variant on the H_K algorithms introduced by Woeginger in [30]. These *Simplified Harmonic* algorithms (SH_K) use a more effective partition of the items and obtain the same worst-case behavior as the H_K using only $O(\log\log K)$ space.

The questions of the difference between the upper and lower bounds for H_K and of the amount by which SH_K improves over H_K become more and more academic as K increases and all these bounds approach the same limiting value of 1.69103... This limiting value takes on

added significance in light of a second result of Lee and Lee, which implies that in a sense the Harmonic (and Simplified Harmonic) algorithms are the best possible bounded-space on-line algorithms.

Theorem [19]. *For any integer $K > 0$ and any K-bounded-space on-line bin packing algorithm A, $R^{\infty}[A] \geq h_{\infty}$.*

Note that this result is in contrast to the result for general (unbounded space) on-line algorithms, where asymptotic worst-case ratios as good as 1.5887 have been obtained [25] and where the best lower bound known is 1.5400 [28].

The drawback to the Harmonic (and Simplified Harmonic) algorithms is that they perform much more poorly on average than do the NF_K. Let L_n be a random n-item list with item sizes chosen independently from a uniform distribution on [0,1]. Then the average value of $\mathrm{NF}_K(L_n)/OPT(L_n)$ empirically approaches 1 as $K \rightarrow \infty$, whereas the expected ratio for the H_K approaches 1.28986... [7,18], and it is reported in [30] that computational experiments suggest that SH_K performs just as poorly. The question thus arises whether there exist bounded-space on-line algorithms that have better worst-case ratios than the NF_K algorithms *and* have at least as good average-case performance. In this paper we answer this question in the affirmative, presenting a sequence of algorithms that is provably better in the worst case and is experimentally observed to be better on average as well.

We shall call these algorithms the *K-Bounded Best Fit* algorithms (BBF_K), as they are derived from the familiar (unbounded) Best Fit (BF) algorithm in much the same way that the NF_K algorithms were derived from First Fit. As with NF_K, algorithm BBF_K runs in time $O(n \log K)$, whereas BF requires time $\Theta(n \log n)$. Algorithm BBF_K differs from NF_K, however, both in the way it chooses the bin into which to pack the current item and in the way it chooses the bin to close when a new bin must be started. Let us denote by $s(B)$ the sum of the sizes of the items currently in bin B. In the packing rule for BBF_K, a_i is placed in the open bin with maximum value of $s(B)$ among those open bins B with $s(B) + s(a_i) \leq 1$. Ties are broken in favor of the bin with lower index. (Recall that in NF_K we simply chose the lowest indexed open bin with $s(B) + s(a_i) \leq 1$.) If no open bin has enough space, a new bin is started. If in this case there are already K open bins, we first close the currently open bin B with maximum value of $s(B)$, again breaking ties in favor of the lower index. (Recall that in NF_K we simply closed the lowest indexed open bin.)

Note that the Best Fit algorithm can be viewed simply as BBF_{∞}, for which, as with FF, we have $R^{\infty}[BF] = 1.7$ [14]. Our main result is the following:

Theorem 1. *For all $K \geq 2$, $R^{\infty}[\mathrm{BBF}_K] = 1.7$.*

This is a surprising result because the original proof that $R^{\infty}[BF] = 1.7$ relied heavily on the fact that in BF all bins remain open and available for future items. In fact, as Theorem 1 indicates, only *two* open bins are necessary. Observe that this implies that, although the algorithms BBF_K asymptotically lose out to the Harmonic algorithms in worst-case performance, they outperform the H_K for small values of K. Table 1 compares the values of $R^{\infty}[A]$ for BBF_K, NF_K, H_K, and SH_K when $2 \leq K \leq 10$. (For $K = 1$ all three algorithms yield identical packings and have asymptotic worst-case ratios of 2.) Note that BBF_K strictly dominates H_K and SH_K for $2 \leq K \leq 4$, and is at least as good for all $K \leq 5$.

K	NF_K	BBF_K	$H_K \geq$	$H_K \leq$	SH_K
2	2.0000	1.7000	2.0000	2.0000	2.0000
3	1.8500	1.7000	1.7500	1.7500	1.7500
4	1.8000	1.7000	1.7143	1.7143	1.7222
5	1.7750	1.7000	1.7000	1.7000	1.7000
6	1.7600	1.7000	1.7000	1.7000	1.6944
7	1.7500	1.7000	1.6944	1.6944	1.6939
8	1.7429	1.7000	1.6938	1.6939	1.6911
9	1.7375	1.7000	1.6933	1.6935	1.6910
10	1.7333	1.7000	1.6929	1.6931	1.6910

TABLE 1. Asymptotic worst case ratios.

An important observation is that *both* the packing and closing rules of the BBF_K algorithms are needed if we are to obtain a result like Theorem 1. Consider the following four rules:

P-FF: Place the current item b in the lowest indexed open bin that has room for it (if any do). Otherwise open a new bin and place b in it.

P-BF: Place the current item b is the fullest open bin that has room for it (if any do), ties broken in favor of the lowest index. Otherwise open a new bin and place b in it.

C-FF: Close the lowest indexed open bin.

C-BF: Close the fullest open bin, ties broken in favor of the lowest index.

Four algorithms with K-bounded space can be constructed using these rules. For any combination of a packing rule (P-FF or P-BF) with a closing rule (C-FF or C-BF), one applies the packing rule to each item in turn, subject to the following constraint: If no current bin has room for b and there are K open bins, then the closing rule must be invoked before starting a new bin. The combination of P-FF with C-FF yields NF_K and the combination of P-BF with C-BF yields BBF_K. Let ABF_K denote the (P-BF,C-FF) combination, and AFB_K denote the (P-FF,C-BF) combination. In the preliminary version of this paper [9] we showed that both AFB_K and ABF_K had asymptotic worst-case ratios of at least $1.7 + 3/(10K)$ for all $K \geq 2$. Subsequently, tight bounds have been obtained for the two algorithms by Mao [22] and Zhang [31] respectively:

Theorem [22]. *For all* $K \geq 2$, $\mathrm{ABF}_K = 1.7 + \dfrac{3}{10K}$.

Theorem [31]. *For all* $K \geq 2$, $\mathrm{AFB}_K = 1.7 + \dfrac{3}{10(K-1)}$.

Note that AFB_K has precisely the same worst-case behavior as NF_K, while ABF_K is slightly better for all $K \geq 2$.

This paper is organized as follows. We begin in Section 2 by presenting our extensive experimental results. These results serve to validate our claim that the algorithms BBF_K are substantially better than their counterparts on average, and raise many interesting theoretical questions. For instance, our results suggest that if items are uniformly distributed in the interval $(0,u]$, $u < 1$, then for sufficiently large fixed $K(u)$, $BBF_{K(u)}$ appears to outperform NF_K for *all* K, and indeed to do better than the limiting, unbounded-space First Fit algorithm. Our results

also suggest that if one allows K to grow slightly more quickly than $\sqrt{n}$, BBF_K will perform as well as Best Fit while using less space and time. We conclude the section by considering the one dimension under which algorithms like NF_K have an advantage over BBF_K, their *delay* characteristics. Note that under BBF_K, a bin may remain nonempty and open for an unbounded amount of time. This is a potential drawback in applications such as those mentioned above, where timely delivery may be important. We consequently examine various ways to bound the output delay of BBF_K, and experimentally confirm that there are ways to do this without seriously degrading overall performance.

In Section 3 we prove Theorem 1. The proof borrows a weighting function from the proof of [14] that $R^{\infty}[FF] = R^{\infty}[BF] = 1.7$, but beyond that takes an entirely different approach, using a complicated induction to arrive at a much more general conclusion. It implies $R^{\infty}[A] \le 1.7$ for a broad class of algorithms that includes the BBF_K, $K \ge 2$, as well as many variants. In Section 4 we provide proofs for some of the improved lower bounds mentioned in this paper, in particular the new lower bounds on $R^{\infty}[\mathrm{H}_K]$ cited in Table 1 and an improved lower bound on the average-case behavior of *any* on-line, bounded-space algorithm that is stated in Section 2. The paper concludes in Section 5 with a brief summary and a discussion of directions for future research.

2. The Average Case: Experimental Results

Our claims of the superiority of BBF_K ''on average'' are based on experiments with lists $L_{n,u}$ of n items with sizes drawn independently from uniform distributions on intervals $[0,u]$, $0 < u \le 1$. Such lists elicit an intriguing diversity of algorithmic behavior, and most of the interesting theoretical results about expected performance of bin packing algorithms that have been proved to date concern them. Our experiments will thus be placed in an informative theoretical context, and can suggest new questions to which probabilistic analysis might be applied.

Let us define $s(L)$ to be the sum of the sizes of the items in L, observing that $s(L) \le OPT(L)$ for all L. For all the distributions in question, we have $E[s(L_{n,u})] = \Theta(n)$ and $\lim_{n\to\infty} E[OPT(L_{n,u})/s(L_{n,u})] = 1$ [3]. Moreover,

$$E[OPT(L_{n,u}) - s(L_{n,u})] \;=\; \begin{cases} \mathrm{O}(1) & 0 < u < 1 \\ \Theta(n^{.5}) & u = 1 \end{cases}$$

where the result for $u = 1$ was first proved in [16,20] and the results for $u < 1$ are from [3].

Based on the above results, it is reasonable to perform one's comparisons between $A(L)$ and $s(L)$, and this is what we shall do, by means of the following definition.

Definition. *If A is a bin packing algorithm, $ER_u^{\infty}[A] = \lim_{n\to\infty} E[A(L_{n,u})/s(L_{n,u})]$.*

Precise values of $ER_u^{\infty}[A]$ have only been determined for a few choices of bounded-space on-line algorithms A and bounds u. Coffman, Hofri, and So [6] showed that $ER_1^{\infty}[\mathrm{NF}_1] = 4/3$, and Karmarkar [15] showed how the values of $ER_u^{\infty}[\mathrm{NF}_1]$ could be determined for $1/2 \le u < 1$. Lee and Lee [19] showed that $\lim_{K\to\infty} ER_1^{\infty}[\mathrm{H}_K] = 1.2898...$ For other algorithms and values of u, the best we can do at present is estimate the values of $ER_u^{\infty}[A]$ empirically by performing experiments on long random lists. Table 2 is the average-case analogue of Table 1, giving the average

K	H_K	SH_K	NF_K	BBF_K	LB
2	1.2986	1.2986	1.2387	1.1782	1.0833
3	1.2917	1.2917	1.1972	1.1378	1.0625
4	1.2904	1.2904	1.1729	1.1159	1.0500
5	1.2901	1.2912	1.1564	1.1015	1.0417
6	1.2900	1.2912	1.1443	1.0912	1.0357
7	1.2899	1.2912	1.1349	1.0832	1.0313
8	1.2899	1.2912	1.1272	1.0769	1.0278
9	1.2899	1.2912	1.1209	1.0717	1.0250
10	1.2899	1.2912	1.1155	1.0673	1.0227
20	1.2899	1.2912	1.0858	1.0439	1.0119
40	1.2899	1.2912	1.0636	1.0280	1.0061
80	1.2899	1.2912	1.0470	1.0175	1.0031
160	1.2900	1.2912	1.0345	1.0107	1.0016
320	1.2903	1.2912	1.0252	1.0065	1.0008
640	1.2909	1.2912	1.0183	1.0040	1.0004
∞	1.2932	1.2912	1.0069	1.0026	1.0000

TABLE 2. Average values of $A(L_{n,1})/s(L_{n,1})$ for $n = 1{,}000{,}000$.

values, over 100 trials, of $A(L_{n,1})/s(L_{n,1})$ when $n = 1{,}000{,}000$. For n this large, the variances of the performance ratios are so small that the 95% confidence range is less than ± 1 for the last decimal place displayed. The row labeled ''∞'' gives the results for the limiting, unbounded storage algorithms H_∞, FF, and BF. Algorithm H_∞ is the generalization of the H_K algorithms in which there is an infinite number of item classes, one for each interval $(1/j, 1/(j-1)]$, $j \geq 2$. For the results in Table 2, we simulated H_∞ by using $H_{1{,}000{,}000}$. The column labelled ''LB'' gives a lower bound on the best possible value of $ER_u^\infty[A]$ for *any* on-line algorithm using K-bounded space. The bounds are derived from the following theorem, which is a strengthening of an earlier result of Coffman and Shor [5] and will be proved in Section 4.

Theorem 2. *For any integer K and any on-line bin packing algorithm A using K-bounded space,*

$$ER_1^\infty[A] \geq 1 + \frac{1}{4(K+1)}$$

Before considering the performance ratios in Table 2, let us first say something about the implementations. The implementations of all the algorithms in this study shared a common calling program that generated item sizes using a shift-register random number generator as described in [1,17]. (Limited experiments were performed with other random number generators to confirm that no major bias was introduced by this choice.) To reduce variance, a common seed was used for the random number generator for each set of experiments, meaning that all algorithms ran on the same set of lists. In order to be able to obtain results for long lists, we programmed in C and used relatively efficient implementations for all the algorithms, i.e., $O(n \log n)$ for Best Fit and

First Fit (the latter using the data structures of [12]), $O(nK)$ for the K-bounded space algorithms other than BBF_K, and $O(n\log K)$ for BBF_K. The slowest of these implementations was that for NF_{640}, but even this took only 19 seconds on a 194 Mhz MIPS™ R10000 processor to handle a million-item list. (MIPS is a trademark of Silicon Graphics, Inc.) The fastest, H_2, took about a half second.

Let us now look in more detail at the average performance ratios presented in Table 2. As predicted, for each value of $K > 1$, NF_K is better than H_K, and BBF_K is better than NF_K. In addition, however, Table 2 exhibits several perhaps unexpected trends. First, the average-case behavior of the Harmonic algorithms H_K actually gets *worse* once K becomes sufficiently large. This is an artifact of our having held n fixed; the expected ratio results for the H_K are asymptotic in n. The Harmonic algorithms create a separate bin class for items in each interval $(1/j, 1/(j-1)]$, $2 \le j \le K$, and if K is sufficiently large with respect to n, many of these classes will consist of a single, mostly empty bin. If one considers lists of length $n = 10^8$ instead of just 10^6, the average ratio of $\mathrm{H}_{640}(L_{n,1})/s(L_{n,1})$ drops to 1.2899.

A second surprise is that as soon as K is sufficiently large for the Simplified Harmonic algorithm SH_K to differ algorithmically from the Harmonic algorithm H_K, i.e., as soon as $K \ge 5$, the average-case behavior for the former becomes slightly *worse* than that for the latter, even though its worst-case behavior is better. This perhaps shouldn't be a surprise, given the many other situations in which actions taken to improve worst-case behavior result in degraded average-case behavior, but it does contradict the tentative claim of equivalent average-caes behavior made in [30]). The reason for the degradation is that SH_K, for $K \ge 5$, groups the items in $(1/6, 1/5]$ and $(1/5, 1/4]$ into one class that is packed by Next Fit, rather than two separate classes, and for items uniformly distributed between 1/4 and 1/6, using separate classes is marginally more effective.

As a final surprise, note that the average ratios for the bounded-space algorithms BBF_{320} and BBF_{640} are both better than the average for the unbounded storage FF algorithm! This is again an artifact of our having fixed n and does not hold asymptotically. For $n = 10^7$ instead of 10^6, the average for FF drops to 1.0034 versus 1.0038 for BBF_{640}. Indeed, it is impossible for *any* bounded-space on-line algorithm to asymptotically outperform FF on average, as a consequence of Theorem 2 and results from [5,26].

Theorem [3,4,26]. $ER_1^{\infty}[\mathrm{FF}] = 1$.

More precisely, in [3] it was shown that $E[\mathrm{FF}(L_{n,1}) - s(L_{n,1})] = O(n^{0.8})$, a result that was subsequently improved to $\Theta(n^{2/3})$ in [4,26]. Thus for $u = 1$ and all fixed K, FF asymptotically dominates BBF_K, which by Theorem 2 must have $ER_1^{\infty}[\mathrm{BBF}_K] > 1$. It may well be the case, however, that for smaller u the situation is reversed. So let us now consider random lists $L_{n,u}$ with $u < 1$.

Let us first examine the situation when K is small. Figure 1 charts the average ratios $A(L_{n,u})/s(L_{n,u})$ for the algorithms NF_K and BBF_K with assorted values of $K \le 5$ ($\mathrm{NF} = \mathrm{NF}_1$). The number n of items is 128,000, u ranges from 0.05 to 1.0 in steps of 0.05, and each data point represents the average of 50 runs. (The 95% confidence ranges are about 10 times as large as those for Table 2, but still smaller than the resolution of the figure.) The dashed lines represent the curves for the NF_K algorithms, and the solid lines represent the BBF_K. We also include a dotted line for ''Smart Next Fit'' (SNF) an intermediate algorithm between NF and BF_2 proposed

and studied in [23]. In SNF there is only one ‘‘current bin’’ allowed, but if the next item to be packed does not fit in that current bin and is bigger than the total contents of that bin, then the new item is packed into a new bin which is immediately closed, leaving the current bin open. This algorithm has the same worst-case behavior as NF, but its average case performance is significantly better (and even better than NF_2 for large u).

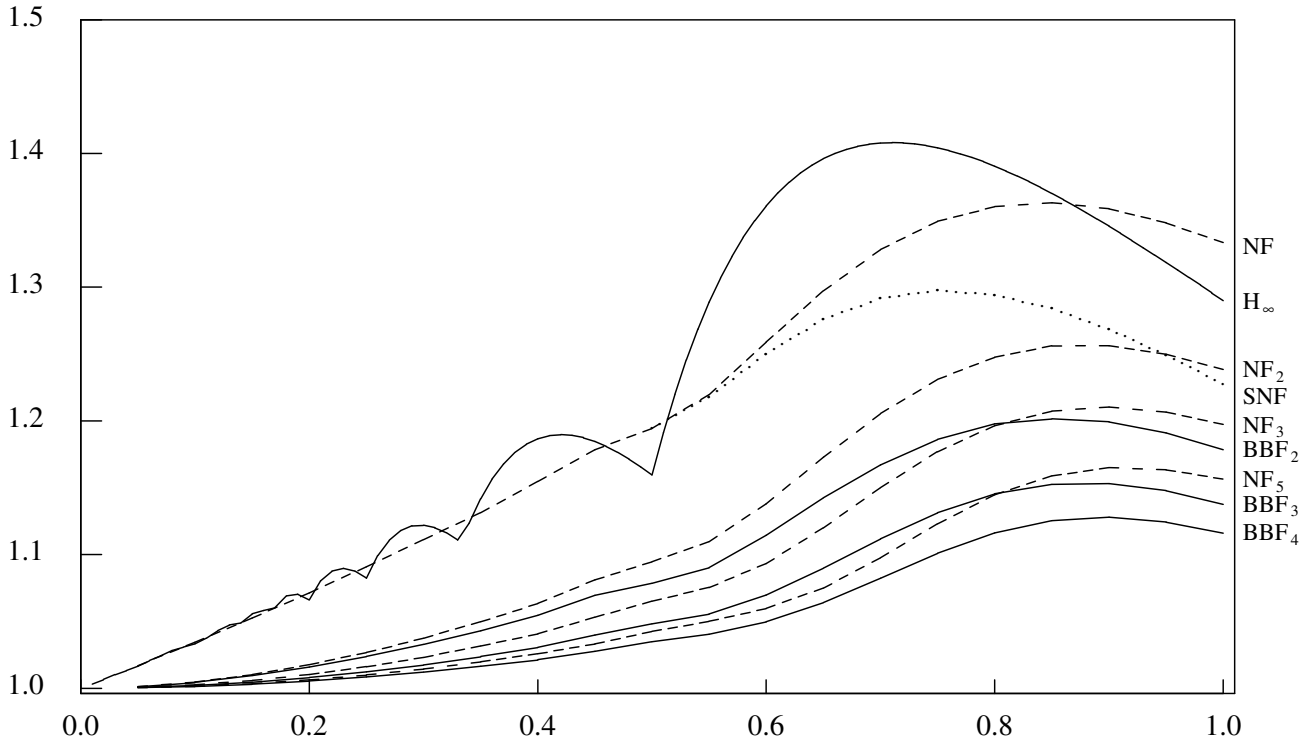

FIGURE 1. Average values of $A(L_{n,u})/s(L_{n,u})$ as a function of u for $n = 128{,}000$.

The figure also includes a solid line representing the analytically derived asymptotic expected ratio for the Harmonic algorithms H_K as K and n go to ∞. (The derivation is straightforward and left as an exercise for the reader.) Data points on this curve go from .01 to 1.00 in steps of .01, which explains its greater smoothness. As suggested by Table 2, choosing the correct K to attain the plotted ratios when n is fixed at 128,000 might have been a tricky task, but based on limited experiments, it appears that the ratios *can* be approached fairly closely, even for small u, if the correct K is chosen. (For $u \geq 0.5$, we could also have derived the points on the NF curve analytically, using results from [6,15], although these again would have been limiting values as $n \to \infty$, rather than values specifically for $n = 128{,}000$.)

Note that the Harmonic algorithm certainly lives up to its name, given the oscillatory behavior of its expected ratio as illustrated in Figure 1. More significantly, the average-case dominance of H_∞ over NF suggested by the theoretical results in [6,19] appears now to be an artifact of the standard bias toward $u = 1$ in the literature. The range of u's for which H_∞ is significantly worse than NF is substantially larger than that for which it is better. And for no u is it better than NF_2. More germane to our purpose here, note that for all K depicted in the figure, BBF_K significantly outperforms NF_K. The lead of the BBF_K over the NF_K seems to be growing with K, in the sense that NF_5 is dominated not only by BBF_5 but also by BBF_4. The lead also increases with u, so

that even BBF_3 dominates NF_5 at the high end of the scale. As K increases further, this phenomenon becomes more pronounced. See Figure 2.

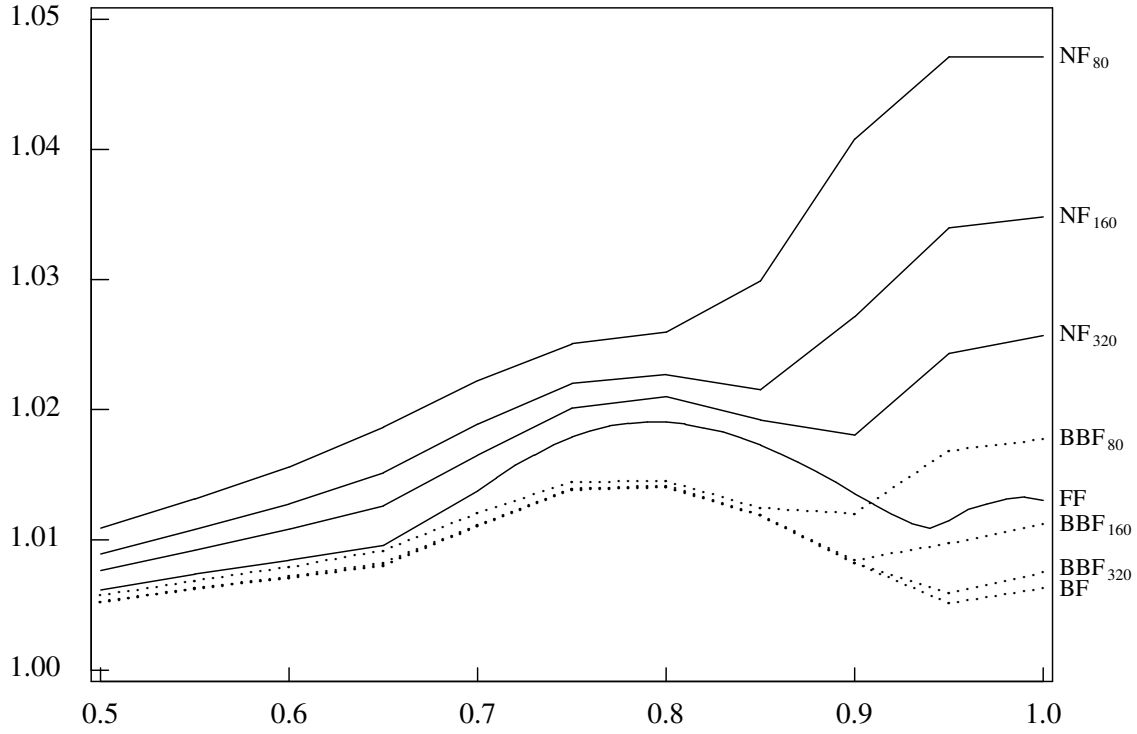

FIGURE 2. Average values of $A(L_{n,u})/s(L_{n,u})$ as a function of u for $n = 128{,}000$.

In Figure 2, we focus on the region where $u \geq .5$, comparing NF_K and BBF_K algorithms with each other and with FF and BF. (The curve for FF is smoother than the others for $u \geq .70$, since it includes data points going up by steps of .01 rather than .05, as a result of experiments performed in an earlier study.) Here we observe that BBF_{80} dominates NF_{320} across the board. Moreover, mirroring the behavior we saw for larger n in Table 2, BBF_{160} and BBF_{320} both dominate NF_K for *all* $K > 0$, since they dominate the limiting FF algorithm. Furthermore, in contrast to the situation when $u = 1$, here there is no strong reason why the dominance of BBF_K over FF might not hold for arbitrarily large n and at least some values of $u < 1$.

The large bumps at the right hand sides of the FF and BF curves are an artifact of the small value of n, but experimental evidence suggests that the gap between the curves at values $u < 1$ is real and persists for arbitrarily large n (see also [2]). Thus we join [27] in the following conjecture.

Conjecture. For all u, $0 < u < 1$,

$$ER_u^\infty[FF] > ER_u^\infty[BF] > 1.$$

We suspect that for $u \leq .90$ the true values are not much lower than those indicated by the curves in Figure 2, and are prepared to make the additional conjecture that, for $u \leq .90$, $ER_u^\infty[FF] > ER_u^\infty[\mathrm{BBF}_{160}] > ER_u^\infty[BF]$. We have studied this question in detail for the case of $u = .80$, where for $n = 128{,}000$ the gap between FF and BF seems to be close to a maximum. Table 3 compares the average ratios for FF, BBF_{80}, BBF_{160}, and BF for $u = .80$ as n increases by

powers of 2 from 128,000 to 8,192,000. For $n \geq 1{,}000{,}000$, the 95% confidence range is no more than ± 4 for the last decimal place displayed, and no more than ± 1 for the last decimal place in the $n = 8{,}192{,}000$ entries. Note that by $n = 8{,}192{,}000$, both the BBF_K algorithms appear to have converged, and both FF and BF seem to be very near their asymptotes, leaving the BBF_K algorithms with a sizable lead over FF.

$n/1000$	FF	BBF_{80}	BBF_{160}	BF
128	1.01905	1.01448	1.01409	1.01408
256	1.01873	1.01441	1.01394	1.01389
512	1.01855	1.01437	1.01387	1.01378
1024	1.01840	1.01435	1.01384	1.01371
2048	1.01832	1.01434	1.01382	1.01367
4096	1.01826	1.01434	1.01381	1.01364
8192	1.01822	1.01434	1.01381	1.01362

TABLE 3. Average values of $A(L_{n,0.80})/s(L_{n,0.80})$.

The question is thus raised: Given n and u, how large must K be for BBF_K to outperform FF? Of special interest is the case where $u = 1$, for which as we have seen above, no fixed value of K will work for all n. Assuming we allow K to grow with n, how slow a growth rate will suffice? For $u = 1$, a growth rate of $\Theta(n^{1/2}\log^{3/4} n)$ certainly suffices, as a consequence of the following result.

Theorem 3. *There exists a constant $c > 0$ such that if $g(n) = cn^{1/2}\log^{3/4} n$, then*

$$E[\mathrm{BBF}_{g(n)}(L_{n,1}) - s(L_{n,1})] = \Theta(n^{1/2}\log^{3/4} n).$$

Note that this expected difference is substantially smaller than the abovementioned $\Theta(n^{2/3})$ expected difference for FF, and is within a constant factor of the expected difference for BF, as determined by Shor in [26]. The proof of Theorem 3 is omitted, as it simply involves showing that the machinery of Shor's proof can be applied to BBF_K when K grows at the specified rate. The argument does not allow us to conclude, however, that the expected growth rates for $\mathrm{BF}(L) - s(L)$ and $\mathrm{BBF}_{g(n)}(L) - s(L)$ have the same constants of proportionality. To investigate this question, we compared BF to BBF_K for K equal to $(1/2)n^{1/2}\log^{3/4} n$ rounded to the nearest integer, logarithms taken to the base e. We used a testbed consisting of lists with lengths going up by factors of two from $n = 1{,}000$ to $8{,}192{,}000$, with 25 to 1600 random lists considered for each value of n, the higher numbers being for the smaller values of n. Surprisingly, the two algorithms used precisely the same number of bins for every list in the study! (We did not check to see if the packings within those bins were identical, although this seems less likely.)

That K must be allowed to grow roughly as fast as $n^{1/2}\log^{3/4} n$ for BBF_K to successfully mimic BF in this way is suggested by the data summarized in Figure 3. Here the average values of $(\mathrm{BF}(L) - s(L))/n^{1/2}$ for the above testbed are charted as a function of n and compared to the corresponding averages for BBF_K when K is the nearest integer to $n^{1/2}$. Observe that now BBF_K definitely produces more waste than BF, and the gap is growing with n.

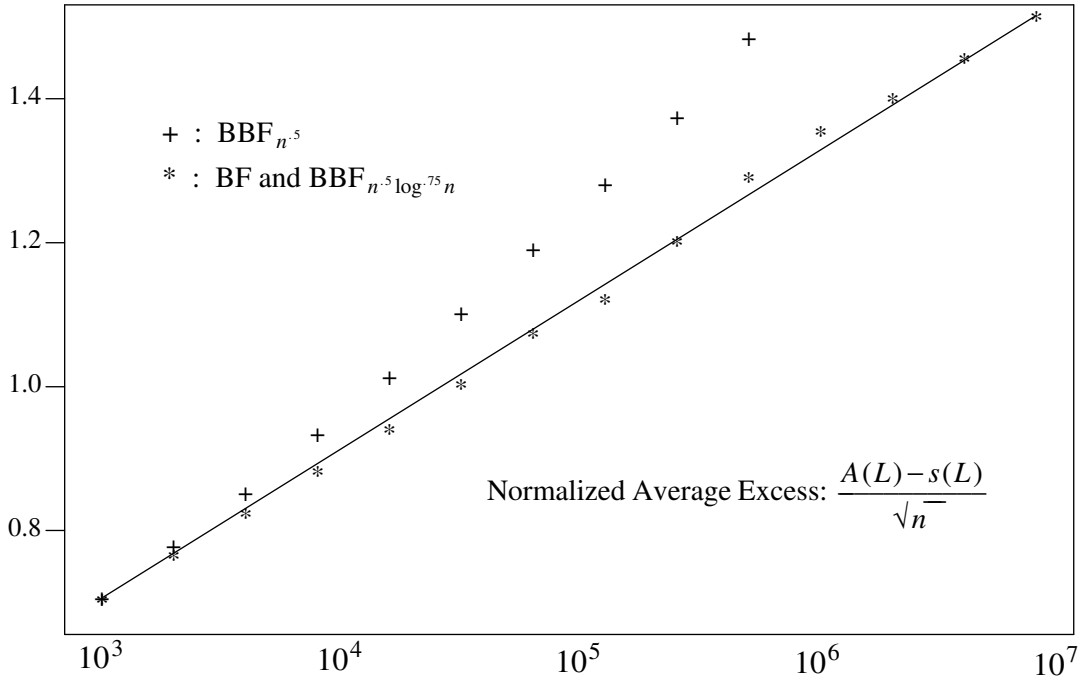

FIGURE 3. Normalized average excesses as a function of n, with n going from 1,000 to 8,192,000 by powers of 2.

Note also that the question of how fast K must grow for BBF_K to mimic BF is not entirely academic. At least for the straightforward $O(n\log n)$ and $O(n\log K)$ implementations used here, letting K grow as roughly $n^{1/2}$ yields BBF_K algorithms that should run roughly twice as fast as BF. For large n, BBF_K actually did substantially better than this, presumably because its lower space requirement led to better cache performance. Thus BBF_K may be of interest even when bounded space *per se* is not an issue.

When bounded space is important, however, there is one potential drawback to using algorithms like the BBF_K. Let us say an item is ''output'' when the bin into which it was packed is closed. In many applications, it may be important that the delay between the time an item is input and the time it is output be minimized. This could well be the case in both the loading dock and communication buffer applications described in the introduction. The NF_K algorithms provide a guarantee that an item will be output by the time subsequent items with total size K have arrived. No such bounded guarantee is provided by the BBF_K. Even on average, relatively large delays occur. For instance, over five runs of BBF_{10} on random lists $L_{n,1}$ with $n = 1{,}000{,}000$, the oldest item in an unclosed bin when the millionth item was packed had typically arrived over 67 items earlier, compared to the roughly $2K = 20$ maximum item delay one might expect for NF_{10}. For BBF_{320}, the oldest item had arrived an average of over 8,000 items earlier, compared to the roughly $2K = 640$ maximum item delay expected from NF_{320}. Note that the ratio of maximum delays for BBF_K and NF_K increased from roughly 3.3 to 12.5 as we went from $K = 10$ to $K = 320$.

Can we obtain the improved packings of the BBF_K algorithms without allowing such long delays. One option is the hybrid algorithm ABF_K described in Section 1, which combines the Best Fit packing rule with the First Fit closing rule. As we saw in Section 1, ABF_K does not

perform as well as BBF_K in the worst case, but it does do better than NF_K in the worst-case as soon as $K \geq 2$. One might hope that it would also produce better packings than NF_K, while having the same delay characteristics. Table 4 compares results for ABF_K with those for BBF_K and NF_K for lists $L_{n,1}$ with $n = 1{,}000{,}000$ and for various values of K from 2 to 640. The results are the averages over 100 trials each for the new algorithms, and once again the last digits are accurate to within ± 1 with a 95% confidence level.

K	NF_K	ABF_K	$\mathrm{BBF}_{K,2K}$	$\mathrm{BBF}_{K,4K}$	$\mathrm{BBF}_{K,8K}$	$\mathrm{BBF}_{K,16K}$	BBF_K
2	1.2387	1.2363	1.2131	1.1843	1.1785	1.1782	1.1782
5	1.1564	1.1520	1.1383	1.1116	1.1026	1.1016	1.1015
10	1.1155	1.1108	1.1026	1.0793	1.0692	1.0674	1.0673
20	1.0858	1.0810	1.0760	1.0566	1.0466	1.0441	1.0439
40	1.0636	1.0587	1.0556	1.0403	1.0313	1.0284	1.0280
80	1.0470	1.0420	1.0401	1.0285	1.0210	1.0181	1.0175
160	1.0345	1.0297	1.0285	1.0200	1.0141	1.0114	1.0107
320	1.0252	1.0208	1.0200	1.0139	1.0096	1.0073	1.0065
640	1.0183	1.0144	1.0139	1.0096	1.0066	1.0048	1.0040

TABLE 4. Average values of $A(L_{n,1})/s(L_{n,1})$ for bounded-delay algorithms when $n = 1{,}000{,}000$.

Table 4 also reports results for a sequence of algorithms that attempt to attack the delay problem head on, while maintaining more of the good packing behavior of the algorithms BBF_K. Algorithm $\mathrm{BBF}_{K,D}$ measures time by the number of items that have arrived so far, and assigns to each newly-opened bin a timestamp equal to the time when the first item was placed in it. When the jth item arrives to be packed, the algorithm first checks to see whether any bin with a timestamp of $j-D$ or less remains open. If so, it closes it (after first adding the jth item, should it fit). If no such *timed-out* open bin exists (or if the jth item didn't fit), the algorithm then proceeds as in BBF_K. Note that we may assume without loss of generality that $D \geq K$, as the time-outs imply that $\mathrm{BBF}_{K,D}$ can never have more than D bins open.

Now these modifications of BBF_K do not come without a significant worst-case penalty. As can be seen by considering lists consisting entirely of items of size ε/D for arbitrarily small ε, $R^{\infty}[\mathrm{BBF}_{K,D}] = \infty$ for all fixed D and K. However, one might hope on average to approach the quality of the BBF_K packings as D is allowed to increase, and the extra processing required by the timestamping should add little to the running times of the algorithms. Table 4 covers results for values of D equal to $2K$ (corresponding to NF_K and ABF_K), $4K$, $8K$, and $16K$.

From the results for the $\mathrm{BBF}_{K,D}$ algorithms, it is clear that the BBF_K need to allow an occasional large delay in order to attain their good results. Note that even the $\mathrm{BBF}_{K,16K}$ suffer a bit of degradation from the behavior of the BBF_K, at least for $K > 2$, although the degradation would be insignificant in most applications. In general the behavior of the $\mathrm{BBF}_{K,\alpha K}$ gets progressively worse as α decreases, although even for $\alpha = 2$ the algorithms are still better than the ABF_K. These in turn are better than the NF_K, as expected, but only by a small amount. The hybrid algorithms AFB_K (which don't have good delay characteristics and are therefore not included in the table) provide a much more substantial improvement over the NF_K, with performance ratios that fall somewhere between those for the $\mathrm{BBF}_{K,2K}$ and the $\mathrm{BBF}_{K,4K}$.

One concludes that it is possible to limit delays and still get packings almost as good as those provided by the BBF_K, although there is a trade-off between the quality of the packings and the amount of worst-case delay one is willing to tolerate.

3. Proof of Theorem 1: The 1.7 Upper Bound

Theorem 1 states that the asymptotic worst-case ratio for BBF_K is 1.7 for all $K \geq 2$. We shall show that this conclusion actually holds for a much larger class of algorithms than just the BBF_K, a class we shall call the (non-subscripted) BBF algorithms. These are like the algorithms BBF_K, but with the linkage between the packing and bin-closing operations relaxed: Whenever there are at least three non-empty open bins, one is free to close one of them, although one is under no stricture to do so. The application of a BBF algorithm to a list of items L thus consists of a series of *pack* and *close* operations, applied in sequence until a state is reached in which neither is applicable (at which time all items in L will have been packed and all but at most two of the bins will be closed). A BBF algorithm is defined by the choices it makes when both operations are applicable.

Packing must follow the standard Best Fit rule with respect to the open bins. *Close* operations must close the open bin whose gap is of minimum size among all ''mature'' open bins (ties broken in favor of the lowest indexed bin), where a mature bin is either the first bin or any non-empty bin that did not receive its first item in the most recent *pack* operation. (A bin other than the first bin that *did* receive its first item in the most recent pack operation is called *newborn*.)

Note that for all $K \geq 2$, BBF_K can be viewed as an algorithm of this form. We apply the pack operation whenever there are unpacked items and K or fewer open bins, and we apply the close operation whenever there are $K+1$ open bins, or when all items have been packed and there are 3 or more open bins. (BBF_K need not use more than K ''physical'' bins, since a close operation must immediately follow the algorithmic creation of a $K+1$st open bin, and we can postpone the physical creation of that bin until after the close operation has taken place. Note that the $K+1$st bin is not a candidate for being closed in this situation, as it is newborn.) We shall show that for any BBF algorithm A and any list L,

$$A(L) \;\leq\; \lceil (1.7) \cdot \mathrm{OPT}(L) \rceil$$

We may assume without loss of generality that the $s(L) > 1$, as otherwise it is trivial that $A(L) = \mathrm{OPT}(L)$. Not surprisingly, we make use of several tools originally developed for the 17/10 proofs for FF and BF. In particular, a weighting function and several related lemmas from [11] serve as our starting point. The weighting function W: $[0,1] \rightarrow [0,8/5]$ is defined as follows.

$$W(\alpha) = \begin{cases} (6/5)\,\alpha & \text{for } 0 \leq \alpha \leq 1/6, \\ (9/5)\,\alpha - 1/10 & \text{for } 1/6 < \alpha \leq 1/3, \\ (6/5)\,\alpha + 1/10 & \text{for } 1/3 < \alpha \leq 1/2, \\ (6/5)\,\alpha + 4/10 & \text{for } 1/2 < \alpha \leq 1. \end{cases}$$

Lemma 1. *For all* $\alpha \in [0,1]$, $W(\alpha) \geq (6/5)\,\alpha$.

Proof. By inspection. □

If B is a set of items let us define $s(B)$ to be $\Sigma_{b\in B}\ s(b)$ and $W(B)$ to be $\Sigma_{b\in B}\ W(s(b))$. We also shall identify a bin B with the set of items it contains.

Lemma 2. *If B is a set of items such that $s(B) \le 1$, then $W(B) \le 1.7$.*

Proof. See proof of Lemma 1(a) in [11]. □

Lemma 3. *If $\alpha < 1/2$ and B is a set of items such that $|B| \ge 2$, $s(b) > \alpha$ for all $b \in B$, and $s(B) \ge 1 - \alpha$, then $W(B) \ge 1$.*

Proof. See proof of Lemma 3 in [11]. □

Lemma 4. *If $\alpha < 1/2$ and B is a set of items such that $|B| \ge 2$, $s(b) > \alpha$ for all $b \in B$, and $s(B) = 1-\alpha-\beta$ for some $\beta > 0$, then $W(B) \ge 1 - (6/5)\beta$.*

Proof. See proof of Lemma 4(b) in [11]. □

Lemma 5. *If $W(L) > A(L) - 1$, then $A(L) \le \lceil (1.7)\,\mathrm{OPT}(L) \rceil$.*

Proof. $W(L) \le (1.7)\,\mathrm{OPT}(L)$ by Lemma 2 applied separately to the bins of an optimal packing and then summed over all such bins. Thus if $W(L) > A(L) - 1$, we have $A(L) < (1.7)\,\mathrm{OPT}(L) + 1$, which yields the desired conclusion since $A(L)$ is an integer. □

In light of Lemma 5, the proof of Theorem 1 reduces to showing that $W(L) > A(L) - 1$. Our proof of this fact will be by induction on the number t of operations performed so far by algorithm A, starting with $t = 1$. We will have three induction hypotheses, (H1), (H2), and (H3). In order to specify them, we need a few more definitions and one more (easy) lemma. Let $P[t]$ denote the packing after the t^{th} operation has been performed, and let $t_{\max}$ denote the number of operations that algorithm A performs before it halts. Let $B[t]$ denote the contents of bin B in packing $P[t]$. (When the relevant value of t is clear from context, we will typically drop the ''$[t]$'' from $B[t]$, but sometimes it is useful to retain it.)

Lemma 6. *For all t, $1 \le t \le t_{\max}$, $P[t]$ contains at most one open bin B with $s(B) \le 1/2$.*

Proof. This follows immediately from the fact that no bin is started unless the item being packed is bigger than the gaps in all currently open bins. □

Let $N_C[t]$ be the number of closed bins in $P[t]$ and let $C[t]$ be the set of all items in closed bins in $P[t]$.

INDUCTION HYPOTHESIS (H1). *In packing $P[t]$, there exists a mature open bin $B_L[t]$, which we shall call the* leader*, such that*

$$W(C[t]) + W(B_L[t]) \ge N_C[t] + (6/5)\,s(B_L[t]).$$

In other words, the total weight of items in the closed bins plus the leader bin falls short of an average of 1 per bin by at most $1 - (6/5)\,s(B_L[t])$.

Hypothesis (H1) alone would be enough to imply that $W(L) > A(L) - 1$, as desired, if it could be shown to hold for $t = t_{\max}$. By our assumption that $s(L) > 1$, we know that $P[t_{\max}]$ contains at least two non-empty bins. Hence, by the definition of the close operation and its applicability, there must be precisely two open bins in $P[t_{\max}]$, a leader B_L and a non-leader B_N, and $N_C[t_{\max}] = A(L) - 2$. Moreover, by the same argument as used to prove Lemma 6, we

must have $s(B_N) > 1 - s(B_L)$. Thus by (H1) and Lemma 1 we would have

$$\begin{aligned} W(L) &= W(C[t_{\max}]) + W(B_L) + W(B_N) \\ &\geq N_C[t_{\max}] + (6/5)s(B_L) + (6/5)s(B_N) \\ &> A(L) - 2 + (6/5) > A(L) - 1, \end{aligned}$$

as desired.

Our remaining hypotheses are needed for technical reasons, in order to make the induction go through. To specify them, we will need some additional definitions.

Definition. *For any bin B, let* $gap(B) = 1 - s(B)$ *and* $min(B)$ *be the size of the smallest item in B.*

Definition. *If B and B′ are open bins, we say that* $gap(B') <^* gap(B)$ *if either* $gap(B') < gap(B)$ *or* $gap(B') = gap(B)$ *and B′ has lower index.*

Note that if $gap(B') <^* gap(B)$, an item that would fit in both bins B and B' will not be packed in B (by the definition of the pack rule). Also, with this relationship between gaps, bin B cannot be closed (by the definition of the close rule).

Definition. *An open bin in* $P[t]$ *is* senior *if* $W(B) \geq 1$ *and* junior *otherwise.*

Definition. *For every non-leading junior bin B,*

If $|B| = 1$, $generators(B) = \{B' : B'$ *is an open bin other than* $B\}$.

If $|B| \geq 2$, $generators(B) = \{B' : gap(B') <^* gap(B)$ *and B′ is either the leader or a junior bin}.*

Definition. *For every non-leading junior bin B,* $lb(B) = \max\{gap(B') : B' \in generators(B)\}$. *(By convention the maximum of an empty set is* 0*.)*

Definition. *A non-leading junior bin B is* well-bounded *if* $min(B) > lb(B)$.

INDUCTION HYPOTHESIS (H2). *For all non-leading junior bins B′*

$$gap(B_L[t]) <^* gap(B'[t]).$$

INDUCTION HYPOTHESIS (H3). *All non-leading junior bins are well-bounded.*

Note that if (H2) holds, then the leader bin is a generator for all non-leading junior bins.

To complete the proof, we show that (H1), (H2), and (H3) hold for all $P[t]$. It is easy to see that the hypotheses hold for $t = 1$: In $P[1]$ there is just one open bin and no closed bins. Hence $N_C[1] = 0$ and $C[1] = \phi$. We can take the one open bin to be the leader $B_L[1]$. Since there are no non-leader junior bins, (H2) and (H3) hold vacuously. (H1) holds since

$$\begin{aligned} W(C[1]) &+ W(B_L[1]) - N_C[1] \\ &= W(B_L[1]) \geq (6/5)s(B_L[1]) \end{aligned}$$

by Lemma 1.

Now suppose that the induction hypotheses hold for $t < t_{\max}$. We shall show that they continue to hold for $t + 1$. There are two main cases to consider, depending on whether the $t+1$st

operation is a pack or a close operation.

CASE 1. *Pack Operation.*

In this case the set of closed bins remains unchanged, so $C[t+1] = C[t]$ and $N_C[t+1] = N_C[t]$. We shall also leave the leader bin unchanged, i.e., $B_L[t+1] = B_L[t]$. Let b be the item packed. We divide into subcases depending on the type of bin into which b is packed.

CASE 1.1. *Item b goes into* $B_L[t]$.

Hypothesis (H1) continues to hold since

$$\begin{aligned} & W(C[t+1]) + W(B_L[t+1]) - N_C[t+1] \\ & = W(C[t]) + W(B_L[t]) + W(s(b)) - N_C[t] \\ & \geq (6/5)\,s(B_L[t]) + (6/5)\,s(b) \\ & = (6/5)\,s(B_L[t+1]), \end{aligned}$$

by Lemma 1 and the fact that (H1) holds for $P[t]$. Hypothesis (H2) continues to hold trivially, since $gap(B_L)$ can only have decreased. Finally, Hypothesis (H3) continues to hold since, for every non-leading junior bin B, the pack operation did not effect the contents of B, did not change the set $generators(B)$, and did not increase the gap in any of those bins.

CASE 1.2. *Item b goes into a non-leading senior bin.*

Hypothesis (H1) holds trivially since it held for $P[t]$, Hypothesis (H2) is unaffected, and Property (H3) continues to hold for the same reasons as in the previous case.

CASE 1.3. *Item b goes into a non-leading junior bin.*

Hypothesis (H1) continues to hold trivially, but (H2) and (H3) now need more careful argument. Let B be the bin into which b goes.

First, let us argue that (H2) must be preserved. Suppose not. Then B must remain a junior bin in $P[t+1]$ and we must have $gap(B[t+1]) <^* gap(B_L[t+1]) = gap(B_L[t])$, despite the fact that $gap(B_L[t] <^* gap(B[t])$ by (H2) for $P[t]$. Now since (H3) holds for $P[t]$, we must have $min(B[t]) > lb(B[t]) \geq gap(B_L[t])$. Since b went into B despite the fact that $gap(B_L[t]) <^* gap(B[t])$, we must also have $s(b) > gap(B_L[t])$. Therefore $min(B[t+1]) > gap(B_L[t])$. Note also that the addition of b to B insures that $|B[t+1]| \geq 2$. Thus Lemma 3 must apply with $\alpha = gap(B_L[t])$. (We must have $\alpha < 1/2$, as required by the lemma, since otherwise $gap(B[t]) \geq gap(B_L[t]) \geq 1/2$, contradicting Lemma 6.) But Lemma 3 implies that $W(B[t+1]) \geq 1$, contradicting our assumption that $B[t+1]$ is a junior bin. Thus (H2) must be preserved.

In order to prove that (H3) is also preserved, we divide into two cases.

CASE 1.3.1. $W(B[t+1]) < 1$.

In this case, bin B remains a junior bin. We first show that Hypothesis (H3) continues to hold for bin B, i.e., that B remains well-bounded in $P[t+1]$. First note that by (H3) for $P[t]$, all items in $B[t+1]$ except b are of size at least $lb(B[t])$. Also, b must be larger than $lb(B[t])$: By the packing rule it must exceed $\max\{gap(B'[t]):\ B'[t]$ *is an open bin with*

$gap(B'[t]) <^* gap(B[t])\}$, and this quantity is itself an upper bound on $lb(B[t])$, since the set of bins being maximized over here is by definition a (not-necessarily proper) superset of $generators(B[t])$. Thus we conclude that $min(B[t+1]) \geq lb(B[t])$. But note that since $gap(B)$ decreases due to the addition of b and no other open bin has a change in gap, the set $generators(B[t+1])$ must be a (not-necessarily proper) subset of $generators(B[t])$. Thus $lb(B[t+1]) \leq lb(B[t])$, and so bin B remains well-bounded.

We now consider whether the addition of b to B can have caused any *other* non-leading junior bin B' to violate the well-boundedness property. For such a violation to be introduced, the packing of b must have caused an increase in $lb(B')$. If $generators(B')$ remains unchanged or shrinks after b is packed, then $lb(B')$ cannot increase. Thus if $lb(B')$ increased, it must be because $generators(B')$ gained a new member. Since only bin B has changed, this new member must be B, and so we must have $gap(B[t+1]) <^* gap(B'[t+1]) = gap(B'[t]) <^* gap(B[t])$.

But note that this means $B' \in generators(B[t])$ and hence that $lb(B[t]) \geq gap(B')$ by definition of $lb(B)$. Hence, by the well-boundedness of $B[t]$, $min(B[t]) > gap(B')$. Furthermore, since $gap(B') <^* gap(B[t])$ and yet b went into bin B, the pack rule implies that $s(b) > gap(B')$. Thus $min(B[t+1]) > gap(B')$, and Lemma 3 is applicable to $B[t+1]$ with $\alpha = gap(B')$. (We have $|B[t+1]| \geq 2$ since B was already open when b was added. We have $\alpha < 1/2$ since by Lemma 6 at most one bin in $P[t]$ has a gap of 1/2 or more, and $gap(B') <^* gap(B[t])$ implies that B' cannot be that bin.) By applying Lemma 3 we then have $W(B[t+1]) \geq 1$, a contradiction of the basic assumption for this subcase. Thus all non-leading junior bins remain well-bounded, and so (H3) continues to hold, as required.

CASE 1.3.2. $W(B[t+1]) \geq 1$.

In this case, bin B becomes a (non-leading) senior bin, and is no longer subject to Hypothesis (H3). The only question is whether the addition of b to B might cause a violation of well-boundedness for some remaining non-leading junior bin B', by increasing the value of $lb(B')$. As in the previous case, this can only happen if in going from $P[t]$ to $P[t+1]$, we add bin B to $generators(B')$. This cannot be the case if $|B'| \geq 2$, as in that case no non-leading senior bin can be in $generators(B')$. Nor can it be the case if $|B| = 1$, because then *all* open bins are in $generators(B')$ in the first place. Consequently, Hypothesis (H3) continues to hold, as required.

CASE 1.4. *Item b goes into a new(born) bin.*

We divide into two cases, depending on whether $s(b) > 1/2$ or not.

CASE 1.4.1. $s(b) > 1/2$.

In this case, $W(b) > 1$ and the new bin B into which b is placed will be a non-leading senior bin, so (H2) is unaffected. Since such bins are irrelevant to Hypothesis (H1), that property also must be preserved. Such bins also are not directly subject to Hypothesis (H3), so the only question is whether the creation of bin B causes a violation of (H3) for some already existing non-leading junior bin B'. If $|B'| \geq 2$, bin B is irrelevant, since $generators(B')$ does not contain any non-leading senior bins. On the other hand, if $|B'| = 1$, we have that $s(b) > gap(B') = 1 - min(B')$ since b would not have started a new bin if it had fit in the gap of bin B'. Thus $min(B') > 1 - s(b) = gap(B)$, and even if the creation of B increases $lb(B')$, B' will remain well-bounded. Consequently (H3) also continues to hold.

CASE 1.4.2. $s(b) \leq 1/2$.

In this case, $W(b) < 1$ and the new bin B is a non-leading junior bin. It is irrelevant to (H1), which hence continues to hold. Moreover, we must have $gap(B_L[t+1]) < gap(B[t+1])$ (else both $B_L[t+1]$ and $B[t+1]$ would have gaps of 1/2 or greater, contradicting Lemma 6). Thus (H2) is preserved. Finally, let us consider (H3). First note that since $B[t+1]$ is a one-item bin, $generators(B[t+1])$ is the set of all open bins other than B. But since b did not fit in any open bin of $P[t]$, we must have $min(B) = s(b) > gap(B')$ for all open bins B'. Consequently B is well-bounded. Could the creation of the new junior bin $B[t+1]$ have stopped any other non-leading junior bin B' from being well-bounded? Note that B' must contain at least two items, as otherwise B' and B would constitute a violation of Lemma 6. But then, since $gap(B[t+1])$ must be the strictly larger than the gap in any other non-leading junior bin of $P[t+1]$ (again by Lemma 6), B cannot be in $generators(B')$, and hence cannot affect $lb(B')$. Hence the well-boundedness property is preserved for all the non-leading junior bins of $P[t]$.

This completes the proof that the pack operation preserves (H1) through (H3).

CASE 2. *Close Operation.*

Let B be the bin that is closed.

CASE 2.1. *B was a non-leading junior bin.*

Since (H2) holds for $P[t]$, no non-leading junior bin can be closed in preference to $B_L[t]$, so this case cannot occur.

CASE 2.2. *B was a non-leading senior bin in $P[t]$.*

In this case, set $B_L[t+1] = B_L[t]$. Note that since the leader bin is unchanged, closing B cannot cause $lb(B')$ to increase for any non-leading junior bin B', and so can have no effect on Hypothesis (H3). Since neither the leader nor any non-leading junior bins are affected, (H2) also continues to hold. Finally, closing B must also preserve Hypothesis (H1), for observe that

$$\begin{aligned}
&W(C[t+1]) + W(B_L[t+1]) - N_C[t+1] \\
&\quad = (W(C[t]) + W(B)) + W(B_L[t]) - (N_C[t]+1) \\
&\quad = (W(C[t]) + W(B_L[t]) - N_C[t]) + W(B) - 1 \\
&\quad \geq (6/5)\, s(B_L[t]) + W(B) - 1 \\
&\quad \geq (6/5)\, s(B_L[t+1]),
\end{aligned}$$

as required, since $B_L[t+1] = B_L[t]$ and $w(B) \geq 1$ by the definition of *senior* bin.

CASE 2.3. *$B = B_L[t]$, i.e., B was the leader bin in $P[t]$.*

In this case we need a new leader.

CASE 2.3.1. *There exists a non-leading junior bin B' in $P[t]$ with $|B'| \geq 2$.*

Our choice for the new leader will be the bin of this type with smallest value for $gap(B')$ (ties broken in favor of the lowest indexed bin). Note that since B' contains at least two items it cannot be a newborn bin and hence is eligible for promotion to leader. We also note that it has the smallest value for $gap(B')$ over *all* junior bins, with value strictly smaller than for those bins that contain only one item. (Such a 1-item bin B'' would have to have $gap(B'') \geq 1/2$ if it is to be junior, and by Lemma 6 there can be at most one bin in $P[t]$ with gap that large, so we must

have $gap(B') < gap(B'')$.) Thus (H2) is satisfied.

Hypothesis (H3) will be preserved by this promotion, as bin B' was already in $generators(B'')$ for any other non-leading junior bin B'', given that $gap(B') <^* gap(B'')$ by our choice of B'.

To see that Hypothesis (H1) is preserved, we first note that by the close rule we have $gap(B_L[t]) <^* gap(B')$. Thus $lb(B') \geq gap(B_L[t])$, and by the well-boundedness property $min(B') > gap(B_L[t])$. Moreover, $gap(B_L[t]) < 1/2$ (otherwise both B' and $B_L[t]$ would have gaps 1/2 or greater, again violating Lemma 6). Thus either Lemma 3 or Lemma 4 applies to bin B' with $\alpha = gap(B_L[t])$, depending on whether $gap(B')$ is equal to $gap(B_L[t])$ or is strictly less than it. If $gap(B') = gap(B_L[t])$, Lemma 3 would apply, but this impossible, since the conclusion would be that $W(B') \geq 1$, contrary to our assumption that B' is a junior bin. Thus $gap(B') > gap(B_L[t])$ and Lemma 4 applies. Consequently we have $W(B') >= 1 - (6/5)\Delta$, where $\Delta = gap(B') - gap(B_L[t])$. Closing $B_L[t]$ and making B' the new leader thus yields

$$
\begin{aligned}
&W(C[t+1]) + W(B_L[t+1]) - N_C[t+1] \\
&\quad = (W(C[t]) + W(B_L[t])) + W(B') - (N_C[t]+1) \\
&\quad \geq (6/5)s(B_L[t]) + (1-(6/5)\Delta) - 1 \\
&\quad = (6/5)(s(B_L[t]) - \Delta) = (6/5)s(B') \\
&\quad = (6/5)s(B_L[t+1])
\end{aligned}
$$

and (H1) holds as required.

CASE 2.3.2. *There exists no non-leading junior bin B' with $|B'| \geq 2$.*

In this case, there can be at most one non-leading junior bin, since as seen above, one-item non-leading junior bins must have gaps of at least 1/2, and there can be only one bin with a gap that large. Thus, since the close operation applies only when there are *three* open bins, there must be a non-leading senior bin B', and our choice for the new leader will be the one with the smallest gap (ties broken in favor of lowest index). Hypothesis (H2) holds for the by now standard reason that $gap(B') < 1/2$ by Lemma 6, implying that $gap(B')$ is smaller than the gap of any non-leading junior bin (should one exist). Hypothesis (H3) will be preserved since if there is a non-leading junior bin B'' it has $|B''| = 1$ and so bin B' is already in $generators(B'')$.

Thus the only question is whether (H1) is preserved. By the close rule, we know that $s(B') \leq s(B_L[t])$. By the definition of senior, we have $W(B') \geq 1$. Consequently,

$$
\begin{aligned}
&W(C[t+1]) + W(B_L[t+1]) - N_C[t+1] \\
&\quad = (W(C[t]) + W(B_L[t])) + W(B') - (N_C[t]+1) \\
&\quad \geq (6/5)s(B_L[t]) + (1 - 1) \\
&\quad \geq (6/5)s(B')
\end{aligned}
$$

This exhausts the possibilities for the close operation, and so we have shown that Hypotheses (H1) through (H3) are preserved by both the pack and the close operations. By induction, this completes the proof that $R^\infty[A] \leq 1.7$ for all BBF algorithms A, and hence of Theorem 1. □

We should remark that our proof actually applies to even more algorithms than those in the general class of BBF algorithms defined at the beginning of this section. A careful consideration of Case 2.2 allows one to conclude that the 1.7 upper bound holds even if one allows a generalized closing rule in which one can close either the fullest bin or any bin B with $W(B) \geq 1$.

4. Lower Bound Proofs

In this section, we present bin packing instances to prove the lower bounds claimed in Section 1 for the H_K and the general lower bound claimed in Section 2 for the average-case behavior of any bounded-space on-line algorithm when lists have item sizes chosen uniformly from $(0,1]$.

4.1 New lower bounds for H_K algorithms

In this subsection, we provide instances that justify the new lower bound claims for H_K, $K \geq 7$, listed in Table 1. (These bounds have been independently obtained by A. van Vliet [29].) For these values of K, we will only be using items in the size ranges $(0,1/K]$, $(1/7,1/6]$, $(1/3,1/2]$, and $(1/2,1]$. Our lists will be parameterized by constants x_K and y_K, the values of x_K and y_K being chosen so that $x_k = 42(1 - 1/K - 1/6)y_K$ and both are integers. The lists will consist of $6N(x_K+y_K)$ items of size $1/2 + \delta$, $6N(x_K+y_K)$ items of size $1/3 + \delta$, $6Nx_K$ items of size $1/7 + \delta$, and $6Ny_K$ copies of the sequence $1/K, <S(1 - 2/K + \delta, \delta)>$.

We first observe that the H_K packing will use $10Nx_K + 15Ny_K$ bins. Next, assuming δ is sufficiently small and divides $1/42$ and $1/K$ evenly, note that the optimal packing will consist of $6Nx_K$ bins containing $1/2 + \delta, 1/3 + \delta, 1/7 + \delta, <S(1/42 - 3\delta, \delta)>$, $6Ny_K$ bins containing $1/2 + \delta, 1/3 + \delta, 1/K, <S(1/6 - 1/K - 2\delta, \delta)>$, and one bin containing the remaining $18N(x_K + y_K)$ items of size δ. The limiting ratio $\mathrm{H}_K(L)/\mathrm{OPT}(L)$ is thus at least

$$\frac{10Nx_K + 15Ny_K}{6Nx_K + 6Ny_K} = \frac{Ny_K\left[420(1 - \frac{1}{K} - \frac{1}{6}) + 15\right]}{Ny_K\left[252(1 - \frac{1}{K} - \frac{1}{6}) + 6\right]} = \frac{365K - 420}{216K - 252}$$

Plugging in the values $K = 7,8,9,10$ yields the bounds cited in Table 1.

4.2 New Lower Bound on Best Possible Average-Case Behavior

In this subsection we present a proof of Theorem 2, which stated that for *any* on-line bin packing algorithm A that uses K-bounded space, $ER_1^{\infty}[A] \geq 1 + 1/(4K+4)$. As in Section 2, let $L_{n,1}$ be a random n-item list with item sizes drawn independently and uniformly from $[0,1]$. We will show that

$$E[A(L_{n,1}) - s(L_{n,1})] \geq \frac{n}{8(K+1)}.$$

The claimed result will follow since $E[s(L_{n,1})] = n/2$.

Our proof of the above inequality follows the basic framework introduced by Coffman and Shor [5] in proving the first lower bound of this sort. Suppose we are about to pack x_j, the jth item in $L_{n,1}$. Let $l_1, l_2, \cdots, l_K$ denote the levels of (sum of the item sizes in) the bins inspected

by algorithm A in deciding where to pack x_j, and let $l_{K+1}=0$ denote the level of the next available empty bin. For any c, $0<c<1$, define $p_c(l_t)$, $1\le t\le K+1$ to be the probability that x_j fits in the bin with level l_t and would fill this bin to a level at least $1-c/(K+1)$. Then

$$p_c(l_t)=\begin{cases} \dfrac{c}{K+1} & \text{if } 1-l_t > \dfrac{c}{K+1} \\ 1-l_t & \text{if } 1-l_t \le \dfrac{c}{K+1} \end{cases}$$

and hence $p_c(l_{t)}\le c/(K+1)$ for all t.

if p_c^* denotes the probability that x_j is packed so as to fill a bin to within $c/(K+1)$ of its capacity, we thus have

$$p_c^* \le \sum_{t=1}^{K+1} p_c(l_t) \le c$$

for all c, $0<c<1$. This also holds true for x_j, $1\le j\le n$, and so the expected number of times a bin gets filled to within $c/(K+1)$ of its capacity over the entire packing is at most cn. Thus clearly the expected number of bins filled to within $c/(K+1)$ of capacity in the final packing can also be at most cn, $0<c<1$. Since the expected number of bins required in the final packing is at least $n/2$, we thus have an average of at least $n/2-cn$ bins with gaps of $c/(K+1)$ or more in the final packing.

Now note that $A(L)-s(L)$ is simply the sum of the gaps in all the bins of the packing produced for L by algorithm A. Coffman and Shor [5] obtained a lower bound on this quantity by simply determining that value of c that maximizes $(n/2-cn)(c/(K+1))$, yielding $E[A(L_{n,1})-s(L_{n,1})]\ge n/(16(K+1))$. We obtain a better lower bound by in effect integrating an appropriate function of c from $c=0$ to $c=1/2$.

More precisely, let us consider values of c of the form

$$\frac{1}{2} - \frac{i}{m}, \quad 1 \le i \le m-1$$

for a large fixed integer m.

For $i=1$ we will have at least $n/(2m)$ bins with gaps at least $\dfrac{m-1}{2m(K+1)}$, for a total gap of at least $\dfrac{n(m-1)}{(2m)^2(K+1)}$. Inductively, for $1<i\le m-1$ we will have at least $ni/(2m)$ bins with gaps at least $\dfrac{m-i}{2m(K+1)}$, of which at most $n(i-1)/(2m)$ will have been previously been accounted for. The remaining $n/(2m)$ bins have total gap at least $\dfrac{n(m-i)}{(2m)^2(K+1)}$.

Summing for all i, we get that the total overall gap is at least

$$\sum_{i=1}^{m-1} \frac{n(m-i)}{(2m)^2(K+1)} = \frac{n}{4m^2(K+1)} \cdot \frac{m(m-1)}{2} = \frac{n(m-1)}{8m(K+1).}$$

For arbitrarily large m, this is arbitrarily close to $\dfrac{n}{8(K+1)}$, as claimed. □

5. Directions for Further Research

In this paper we have considered the behavior of on-line bin packing algorithms subject to a space bound, and introduced a new sequence of algorithms which are currently the champions in this class when it comes to combined worst-case/average-case performance. An interesting theoretical question is whether any other sequence (bounded-space or not) can do substantially better. More specifically, can there be a sequence of on-line algorithms A_K whose expected ratios $ER_1^\infty[A_K]$ approach 1 (as do the ratios for our BBF_K algorithms), and whose asymptotic worst-case ratios $R^\infty[A_K]$ approach some constant smaller than 1.7 (the value of $R^\infty[\mathrm{BBF}_K]$ for all $K \geq 2$)? We conjecture that the answer is no, but at present do not know how to go about proving such a result.

An even more fundamental issue that we don't know how to attack analytically is the question of just what the asymptotic ratios $ER_u^\infty[A]$ are. Experiments appear to yield reliable estimates, but to date probabilistic analysis has only succeeded with the simplest of algorithms (Next Fit [6,15], Smart Next Fit [23], and variants on the Harmonic algorithms [7,18,24]). Barring exact analysis, can we prove the following conjecture, suggested by our experiments: For any u, $0 < u < 1$, there exists a constant K such that BBF_K performs better on average than FF when item sizes are drawn independently and uniformly from the interval $(0,u]$. Can we even prove simply that there exists a $u < 1$ such that $ER_u^\infty[FF] > 1$? (Recall that we conjectured in Section 2 that this held for *all* such u.) We consider this latter question to be *the* major open problem in the probabilistic analysis of bin packing algorithms, and offer it as a challenge to the reader.

References

1. J. L. BENTLEY, ''Software Exploratorium: Some random thoughts,'' *UNIX Review* **10:6** (June 1992), 71-77.
2. J. L. BENTLEY, D. S. JOHNSON, F. T. LEIGHTON, AND C. C. MCGEOCH, ''An experimental study of bin packing,'' *in Proc. 21st Ann. Allerton Conf. on Communication, Control, and Computing*, University of Illinois, Urbana, IL, 1983, 51-60.
3. J. L. BENTLEY, D. S. JOHNSON, F. T. LEIGHTON, C. C. MCGEOCH, AND L. A. MCGEOCH, ''Some unexpected expected behavior results for bin packing,'' *in* ''Proceedings 16th Ann. ACM Symp. on Theory of Computing,'' Association for Computing Machinery, New York, 1984, 279-288.
4. E. G. COFFMAN, JR, D. S. JOHNSON, P. W. SHOR, R. R. WEBER, ''Tight bounds on first fit,'' *Random Structures and Algorithms* **10** (1997), 69-101.
5. E. G. COFFMAN, JR AND P. W. SHOR, ''Packing in two dimensions: Asymptotic average-case analysis of algorithms,'' *Algorithmica* **9** (1993), 253-277.
6. E. G. COFFMAN, JR, M. HOFRI, K. SO, AND A. C. YAO, ''A stochastic model of bin packing,'' *Information and Control* **44** (1980), 105-115.
7. J. CSIRIK, J. B. G. FRENK, A. FRIEZE, G. GALAMBOS, AND A. H. G. RINNOOY KAN, ''A probabilistic analysis of the next fit decreasing bin packing heuristic,'' *Operations Res. Lett.* **5** (1986), 233-236.
8. J. CSIRIK AND B. IMREH, ''On the worst-case performance of the *NkF* bin-packing heuristic,'' *Acta Cybernetica* **9** (1989), 89-105.
9. J. CSIRIK AND D. S. JOHNSON, ''Bounded space on-line bin packing: Best is better than first,'' *in* ''Proceedings 2nd Ann. ACM-SIAM Symp. on Discrete Algorithms,'' Society for Industrial Mathematics, Philadelphia, PA, 1991, 309-319.

10. G. N. FREDERICKSON, ''Probabilistic analysis for simple one- and two-dimensional bin packing algorithms,'' *Inform. Process. Lett.* **11** (1980), 156-161.
11. M. R. GAREY, R. L. GRAHAM, D. S. JOHNSON, AND A. C.-C. YAO, ''Resource constrained scheduling as generalized bin packing,'' *J. Comb. Theory* **21** (1976), 257-298.
12. D. S. JOHNSON, *Near-Optimal Bin Packing Algorithms*, Doctoral Dissertation, Department of Mathematics, Massachussetts Institute of Technology, Cambridge, MA, 1973.
13. D. S. JOHNSON, ''Fast algorithms for bin packing,'' *J. Comput. System Sci.* **8** (1974), 272-314.
14. 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. Comput.* **3** (1974), 299-325.
15. N. KARMARKAR, ''Probabilistic analysis of some bin-packing alogorithms,'' *in* ''Proceedings 23rd Ann. Symp. on Foundations of Computer Science,'' IEEE Computer Society, Los Angeles, Calif., 1982, 107-111.
16. W. KNÖDEL, ''A bin packing algorithm with complexity $O(n\log n)$ and performance 1 in the stochastic limit,'' in ''Proc. 10th Symp. on Mathematical Foundations of Computer Science,'' J. Gruska and M. Chytil (eds.), *Lecture Notes in Computer Science 118*, Springer-Verlag, 1981, 369-378.
17. D. E. KNUTH, *The Art of Computer Programming: Seminumerical Algorithms*, Addison-Wesley, Reading, MA, 1981, Section 3.6.
18. C. C. LEE AND D. T. LEE, ''A new algorithm for on-line bin packing,'' Report No. 83-03-FC-02, Dept. of EECS, Northwestern University, Evanston, IL, 1983.
19. C. C. LEE AND D. T. LEE, ''A simple on-line packing algorithm,'' *J. Assoc. Comput. Mach.* **32** (1985), 562-572.
20. G. S. LUEKER, ''An average-case analysis of bin packing with uniformly distributed item sizes,'' Report No. 181, Department of Information and Computer Science, University of California, Irvine, CA, 1982.
21. W. MAO, ''Tight worst-case performance bounds for next-*k*-fit bin packing,'' *SIAM J. Comput.* **22** (1993), 46-56.
22. W. MAO, ''Best-*k*-Fit Bin Packing,'' *Computing* **50** (1993), 265-270.
23. P. RAMANAN, ''Average-case analysis of the smart next fit algorithm,'' *Inform. Process. Lett.* **31** (1989), 221-225.
24. P. RAMANAN AND K. TSUGA, ''Average-case analysis of the modified harmonic algorithm,'' *Algorithmica* **4** (1989), 519-534.
25. M. B. RICHEY, ''Improved bounds for harmonic-based bin packing algorithms,'' *Disc. Applied Math.* **24** (1991), 203-227.
26. P. W. SHOR, ''The average case analysis of some on-line algorithms for bin packing,'' *Combinatorica* **6** (1986), 179-200.
27. P. W. SHOR, private communication (1990).
28. A. VAN VLIET, ''An improved lower bound for on-line bin packing algorithms,'' *Inform. Process. Lett.* **43** (1992), 277-284.
29. A. VAN VLIET, ''On the asymptotic worst case behavior of harmonic fit,'' *J. Algorithms* **20** (1996), 113-136.
30. G. WOEGINGER, ''Improved space for bounded-space, on-line bin-packing,'' *SIAM J. Disc. Math.* **6** (1993), 575-581.
31. G. ZHANG, ''Tight Worst-case Performance Bound for AFB_k,'' Report No. 015, Institute of Applied Mathematics, Academia Sinica, Beijing, China, May, 1994.