Best-Fit Bin-Packing with Random Order

Claire Kenyon*

May 26, 1997

Abstract

Best-fit is the best known algorithm for on-line bin-packing, in the sense that no algorithm is known to behave better both in the worst case (when Best-fit has performance ratio 1.7) and in the average uniform case, with items drawn uniformly in the interval $[0, 1]$ (then Best-fit has expected wasted space $O(n^{1/2}(\log n)^{3/4})$). In practical applications, Best-fit appears to perform within a few percent of optimal. In this paper, in the spirit of previous work in computational geometry, we study the expected performance ratio, taking the *worst-case* multiset of items L, and assuming that the elements of L are inserted in *random order*, with all permutations equally likely. We show a lower bound of $1.08\ldots$ and an upper bound of 1.5 on the random order performance ratio of Best-fit. The upper bound contrasts with the result that in the worst case, any (deterministic or randomized) on-line bin-packing algorithm has performance ratio at least $1.54\ldots$.

1 Introduction

1.1 Background. Bin-packing is a basic problem of computer science: given a list of items between 0 and 1, $L = (x_1, \ldots, x_n)$, assign each item to a bin, so that the sum of the values of the items assigned to the same bin does not exceed 1, and the goal is to minimize the number of bins used. This problem is NP-hard [7] and heuristics have been developed to approximate the minimum number of bins. In the on-line version of the problem, the items arrive one by one, and x_i must be assigned to a bin without knowledge of the future items $(x_{i+1}, \ldots, x_n)$.

The simplest and most classical algorithms designed for this problem are Next-fit, First-fit and Best-fit. Best-fit maintains a list of current bins, ordered by sizes, and upon arrival of item x, puts it in the current fullest bin in which it fits, opening a new bin for x if this fails. First-fit maintains a list of current bins, ordered by the date at which they were opened, and upon arrival of item x, puts it in the first bin in which it fits, opening a new bin for x if this fails. Next-fit maintains the last opened bin, and upon arrival of item x, puts it in that bin if it fits and opens a new bin otherwise. More recently, the Harmonic algorithm was designed [12]; it is more complicated, but linear time, and tailored to behave well in worst-case situations.

The notion of performance ratio is used to evaluate bin-packing algorithms. Let OPT denote the (unknown) optimal off-line algorithm, and, for an algorithm A and a list L, let $A(L)$ denote the number of bins used by algorithm A run on L.

Definition 1. *The performance ratio $C(A)$ of algorithm A is:*

$$C(A) = \limsup_{OPT(L)\to\infty} \frac{A(L)}{OPT(L)}.$$

In the seminal paper [8], Johnson et al. proved that Best-fit and First-fit have performance ratio 1.7, while Next-fit has performance ratio 2. The Harmonic algorithm is proved in [12] to have performance ratio $1.69\ldots$, and improved versions have slightly lower performance ratios; the current best performance ratio is $1.58\ldots$ [16]. The quest for better algorithms was somewhat quelched by Yao's lower bound: no deterministic on-line algorithm can have performance ratio better than 1.5 [19]. This lower bound was later improved up to $1.54\ldots$ in [13, 6, 18], and proved to hold even for randomized algorithms [1].

The performance ratio has the drawback that for Best-fit, the worst-case sequences upon which it relies are very contrived and never occur in practice. These sequences are contrived in two ways: on the one hand, the values of the items inserted are very special, and on the other hand, they must be inserted in increasing order. In fact, it has been observed that Best-fit usually behaves within a few percent of optimal in practice, much better than predicted by the performance ratio. To explain this, researchers have studied the behavior of on-line algorithms when the items are drawn independently from particular distributions. Of particular interest is the uniform distribution in $[0, 1]$. In his thesis, Peter Shor analyzed Best-fit and First-fit under this distribution, and proved that they are

*Permanent address: LIP, URA CNRS 1398, ENS-Lyon, 46, Allée d'Italie, 69364 Lyon Cedex 07. This work was done at U.C. Berkeley, where the author is currently visiting, supported by CNRS and by a NATO fellowship.

asymptotically optimal on average, and that the expected amount of wasted space (number of bins minus the sum of the item sizes) is $\Theta(n^{1/2}(\log n)^{3/4})$ for Best-fit and about $n^{2/3}$ for First-fit [17, 4]. This is in fact even better than practice: real-life distributions are not always as nice as the uniform distribution! In recent years, people have also studied other distributions: a discretized version of the uniform distribution as well as some truncated versions, where the items are drawn uniformly in interval $[a, b]$. Analyzing these distributions precisely is a challenging problem. For example, in a recent paper [3], it was shown, using a computer program to compute Lyapunov functions to analyze multi-dimensional bounded-jump homogeneous Markov chains, that Best fit has linear expected waste when the items are drawn uniformly from the set $\{1/11, 2/11, \ldots 8/11\}$, and also when the items are drawn uniformly from the set $\{1/12, 2/12, \ldots, 9/12\}$. In [11], it is shown that Best fit has constant expected waste when the items are drawn uniformly from the set $\{1/k, 2/k, \ldots, (k-2)/k\}$.

1.2 The result. Best-fit emerges as the winner among the various on-line algorithms: it is simple, behaves well in practice, and no algorithm is known which beats it both in the worst-case and in the average uniform case. But the worst-case performance ratio and the uniform-distribution performance ratio are not quite satisfactory measures for evaluating on-line bin-packing algorithms. Moreover, it appears that studying given distributions accurately is an extremely challenging problem.

In this paper, we focus on Best-fit, and propose a new measure of performance evaluation, that of *worst-case list* of input items, but *random insertion order*, all permutations being equally likely. This model was used in computational geometry with extreme success (see for example [2]).

DEFINITION 2. *The random-order performance ratio $RC(A)$ of an on-line algorithm A is*

$$RC(A) = \limsup_{OPT(L)\to\infty} \frac{E_\sigma A(L_\sigma)}{OPT(L)},$$

where L_σ is the permuted list $(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$ and the expectation is taken over all permutations $\sigma \in \mathcal{S}_n$.

Note that the order is often crucial in the bad-case examples of bin-packing heuristics. A textbook example of why Best-fit is not optimal is the list

$$L = (\underbrace{1/2-\epsilon, \ldots, 1/2-\epsilon}_{n}, \underbrace{1/2+\epsilon, \ldots, 1/2+\epsilon}_{n}).$$

The optimal packing uses just n bins for L, while Best-fit uses $1.5n$ bins. However, if the list L is randomly permuted, the situation is completely different. It can be simulated by drawing each item independently and uniformly from $\{1/2-\epsilon, 1/2+\epsilon\}$. The sequence can be viewed as an unbiased random walk in the plane, where at each step we move by $(+1, \pm 1)$ depending on whether the arriving item is larger or smaller than $1/2$. the number of items left unpaired is bounded by the vertical span of the random walk, which is of order $o(n)$ with high probability. So, in dramatic contrast with its behavior when the items of L are inserted in increasing order, Best-fit behaves optimally for this list if the order is random!

We prove lower and upper bounds on the random-order performance ratio of Best-fit. First, we prove that for any list L, the random-order performance ratio is asymptotically less than 1.5. Second, we exhibit a list L such that the random-order performance ratio is $1.08\ldots$.

THEOREM 1.1. *The random-order performance ratio of Best-fit satisfies:*

$$1.08 \le RC(BF) \le 1.5.$$

We expect the true answer to lie somewhere close to 1.15.

The proof of the lower bound analyzes the performance of Best-fit when the items are drawn independently from the distribution which gives $1/2$ with probability p and $1/3$ with probability $q = 1 - p$, for which the optimum packing is perfect. The analysis can be reduced to studying a five-states Markov chain, which is then solved by linear algebra.

The proof of the upper bound, much more difficult, is a mixture of worst-case and average-case analysis, and its heart lies in proving that the number of items per bin in the optimum packing of the first t items converges quickly to its final value; this in turn can be reduced to upright-matching.

Another question of theoretical interest would be to design an algorithm tailored to behave well under the performance measure random-order performance ratio; it is likely that it is possible to design an optimal algorithm in this sense, since a recent paper by Rhee and Talagrand shows that if the input comes from an arbitrary fixed distribution then there is a distribution-dependent optimal algorithm [15]; however such an algorithm would be of theoretical interest only: in practice efficiency is a crucial issue and only the simplest algorithms, such as Next Fit, First Fit or Best Fit, are actually used.

It is also conceivable that this approach is of interest for offline bin packing. The best practical algorithm for offline bin packing is Best-fit-decreasing (BFD), in which the Best fit algorithm is applied to the items, previously sorted in decreasing order. Johson et al. [8] proved that the worst-case performance ratio of BFD is $11/9 \sim 1.22$. If, as seems likely, the performance ratio of random order Best fit is less than 1.22, this raises the possibility of a better alternate offline algorithm: first shuffle the items randomly, then apply Best fit.

2 The upper bound

Let L be a list of n items, and let L_σ denote the list ordered according to permutation σ. Let $L_\sigma(u, v)$ denote the list of items of L_σ inserted between time u and time v. The proof is in four steps.

2.1 Reduce the analysis of Best Fit to an analysis of OPT. An important fact to remember is that there is always at most one bin filled up to level $\leq 1/2$. Let us first look at easy restricted cases. We classify the items inserted into three types, according to their size: small ($x \leq 1/3$), medium ($1/3 < x \leq 1/2$), and large ($x > 1/2$). We first study Best Fit when not all types occur.

First, imagine that there are no large items. Then all items are less than or equal to 1/2. It is well-known that the worst-case performance ratio of Best Fit in this setting is 1.5. In fact, all bins except at most two are filled up to level 2/3 or more. To see that, first note that all bins except possibly the last one contain at least two items. Now, take the bins in the order in which they were opened, and consider the first bin B whose final size is less than 2/3. Any bin created later than B and with more than one item contains as first two items values between 1/3 and 1/2, whose sum is at least 2/3. Thus the only bins filled up to less than 2/3 are B and possibly the last bin (if it only contains one medium item). This implies

$$BF(L) \leq 2 + 3W(L)/2 \leq 2 + 3OPT(L)/2.$$

Second, imagine that there are no small items. Then all items are strictly greater than 1/3, and the worst-case performance ratio of Best-Fit in this setting is 1.5. To see this, observe that there are at most two items per bin, and that with the Best Fit algorithm, since only one bin can be less than half full, only the large items can be alone in their bin (except possibly for one bin). Let x be the number of large items and $y = n - x$ the number of medium items. The optimal algorithm uses at least $n/2$ bins; Best Fit uses at most $x + y/2 + 1$, which is maximized for $x = n/2$ and gives $BF(L) \leq 3OPT(L)/2 + 1$.

Third, in the general case, all sizes can occur. We will prove the following lemma.

LEMMA 2.1. *There exists a t_σ such that the total number of bins used by Best fit on L_σ is at most*

$$\frac{3}{2}[OPT(L_\sigma(1, t_\sigma)) + OPT(L_\sigma(t_\sigma + 1, n))] + 2.$$

Proof: Fix the permutation σ. Let t_σ be the last time that a small item z was inserted into a bin B which either was new or was filled up to less than 1/2 immediately prior to inserting z.

We first analyze what happens up to time t_σ. At time t_σ, B is the only bin filled up to less than 1/2. No bin B' is filled up to a level l, $1/2 < l \leq 2/3$, since otherwise z would have been inserted into B' rather than B. Thus all bins except B are filled up to level at least 2/3. Let $W(L_\sigma(1, t_\sigma))$ denote the total weight of items inserted up to time t_σ. We have: $BF(L_\sigma(1, t_\sigma)) \leq 3W(L_\sigma(1, t_\sigma))/2 + 1 \leq 3OPT(L_\sigma(1, t_\sigma))/2 + 1$.

Now, we analyze what happens after time t_σ. Let x be the number of large items and y the number of medium items in $L_\sigma(t_\sigma + 1, n)$. Since all the bins created before t_σ are too full to accept even a medium item, the optimal algorithm uses at least $(x + y)/2$ additional bins when run on $L_\sigma(t_\sigma + 1, n)$ starting from the configuration of Best fit at t_σ. But since only one bin can be less than half full, every bin created after time t_σ by Best Fit contains either two medium items or one large item (except possibly for one bin). By definition of t_σ, the small items are only added either to bins created before t_σ, or to bins which already contain one large item or two medium items, so they are irrelevant for our analysis. At most $x + y/2 + 1$ bins are created by Best fit, and the ratio $(BF(L_\sigma) - BF(L_\sigma(t_\sigma + 1, n)))/OPT(L_\sigma(t_\sigma + 1, n))$ is maximized for $x = y$. Thus $BF(L_\sigma) - BF(L_\sigma(1, t_\sigma)) \leq 3OPT(L_\sigma(t_\sigma + 1, n))/2 + 1$.

Putting both inequalities together, we obtain the inequality of the lemma.

The rest of the proof consists in proving that the average number of items per bins at time u in the optimal packing, $OPT(L_\sigma(1, u)/u)$, converges quickly to its final value $OPT(L)/n$ for random σ. Since we cannot analyze the optimal algorithm, we design instead a related algorithm A^* which is about as good as the optimal algorithm, and which can be analyzed using upright-matching techniques, as for the analysis of Shor's Modified Best Fit algorithm [4].

2.2 Design of an efficient and analyzable algorithm A^*. We fix $\epsilon > 0$. In order to describe the algorithm, we first need a few definitions and notations.

A item is called *tiny* if it is less than $\epsilon/2$. Let R be the set of those pieces in L which are tiny, and $L'_\sigma(1, u)$ be the list, of size u', obtained from $L_\sigma(1, u)$ by removing the tiny items. We now define a "rank" for the items of L'. Let C be an optimal configuration of L'. The *rank* $r(x)$ of x is its rank among the items allocated to the same bin as x by configuration C. Let $l(x)$ denote the sum of the values of the items larger than x and packed by C in the same bin as x. Finally, let S_j denote the set of items of L' which have rank j, and let $k = 2/\epsilon$ denote the maximum possible rank of the items of L' (which are all greater than or equal to $\epsilon/2$). We first describe an algorithm A which only packs the items of L', and assigns them to bins as they arrive in the order of L_σ .

Algorithm A upon arrival of item x:
If $r(x) = 1$, then open a new bin for x.
If $r(x) > 1$, scan the items already packed to look for a y such that the following three conditions hold:

(a) $r(y) = r(x) - 1$
(b) $l(y) + y < l(x)$
(c) y's bin contains no item of rank $r(x)$.

If there exists such a y, put x in the bin such that $l(y) + y$ is maximum; otherwise open a new bin for x.

Algorithm A^* consists in first applying algorithm A to L', and then completing the packing with the items of R in a greedy fashion.

Note the similarity between algorithm A and Modified Best Fit. It can be seen as a version of Modified Best Fit adapted to the case when we want more than two items per bin.

It is easy to see that algorithm A is correct, i.e. never packs in the same bin items which sum to more than 1. Note that the optimal configuration C, if it packs items in decreasing order, puts x in a bin whose current level is $l(x)$ immediately prior insertion of x. One can see:

LEMMA 2.2. *If x is packed in a bin whose current level is l immediately prior to the insertion of x, then $l \leq l(x)$.*
This is proved by induction on the rank of x. It is obvious if $r(x) = 1$. If $r(x) = r > 1$, then the lemma is trivial if x is put in a new bin: $0 \leq l(x)$. Otherwise, x is put in a bin which contains y, some item of rank $r - 1$, as well as possibly other items of lower rank. By induction the level of that bin is at most $l(y) + y$. By condition (b), we have $l(y)+y \leq l(x)$, hence the lemma.

For an example of how algorithm A works, consider the sequence $a_1b_2a_3a_2b_1b_3$, where $a_1 = .5$, $a_2 = .4$, $a_3 = .1$, $b_1 = .6$, $b_2 = .2$, $b_3 = .2$, and the optimal packing packs $a_1a_2a_3$ together and $b_1b_2b_3$ together.
A new bin is opened for a_1.
Since $a_1 \leq b_1$, item b_2 is packed with a_1.
Since $b_1 + b_2 \leq a_1 + a_2$, item a_3 is packed with b_2.
Since the bin of a_1 already contains an item of rank 2, a new bin is opened for a_2.
A new bin is opened for b_1.
Since $a_1 + a_2 > b_1 + b_2$, a new bin is opened for b_3.
We thus obtain the packing with $a_1b_2a_3$ together and each other item in a bin of its own. The constraint $a_1+b_2+a_3 \leq 1$ is implied by $a_1 \leq b_1$, $b_1+b_2 \leq a_1+a_2$, and $a_1 + a_2 + a_3 \leq 1$.

We can now make the similarity with Modified Best fit (MBF) more explicit. Pick a rank $r > 1$. From the list L'_σ, construct a list U_σ by removing all items except those of ranks $r - 1$ or r, and replacing each y of rank $r - 1$ by $y' = l(y) + y$. Call these "large", and the items of rank r "small". Then algorithm A packs x, of rank r, in y's bin, if and only if MBF applied to U_σ packs x with y'. Thus the analysis of MBF can be used for A, with only slight modifications, due to the fact that the list U_σ is not quite the same as if the items were drawn from the uniform distribution $U[0, 1]$: one difference is that in our setting, the number of "small" items is exactly equal to the number of "large" items; for a uniform distribution, that would only be true on the average, with an expected discrepancy of $\sqrt{n}$. The other difference is that the set of "large" items which can potentially be matched to the ith smallest "small" item has size exactly i, for every i, while again this only holds on average for a uniform distribution. Nevertheless these differences are slight enough that the results derived for MBF on the distribution $U[0, 1]$ will hold in our case.

2.3 Elimination of tiny pieces.

The analysis of A^* can be reduced to an analysis of A, thanks to a lemma used in [5, 10].

LEMMA 2.3.

$$A^*(L) \leq \max(A(L'), (1 + \epsilon)W(L) + 1).$$

In fact, if adding tiny pieces does not require opening new bins, then $A^*(L) = A(L)$. If it does require opening new bins, then all bins are filled up to level at least $1 - \epsilon/2$ by algorithm A^* (with the possible exception of the last bin), and so

$$A^*(L) \leq \frac{1}{1 - \epsilon/2} W(L) + 1 \leq (1 + \epsilon)W(L) + 1.$$

2.4 Probabilistic analysis.

We have the following three lemmas. The first one relies heavily on upright-matching results. The proofs are omitted in this preliminary abstract.

LEMMA 2.4. *If $W(L') > \epsilon^2 W(L)/2$, then with high probability we have*

$$\sup_u \left[A(L'_\sigma(1, u)) - \frac{u'}{n'} OPT(L') \right] = o(OPT(L')).$$

LEMMA 2.5. *If $W(L') > \epsilon^2 W(L)/2$, then with high probability we have*

$$\sup_u \left[u' - \frac{u}{n} n' \right] = o(n').$$

LEMMA 2.6. *If $W(L') > \epsilon^2 W(L)/2$, then with high probability we have*

$$\sup_u \left[W(L_\sigma(1,u)) - \frac{u}{n} W(L) \right] = o(W(L)).$$

2.5 Putting everything together. Let us first look at the most important case, when $W(L') > \epsilon^2 W(L)/2$. From Lemma 2.1, and using the fact the OPT is always at least as good as A^*, we get:

$$BF(L_\sigma) \le \frac{3}{2} \left[A^*(L_\sigma(1,t_\sigma)) + A^*(L_\sigma(t_\sigma+1,n)) \right] + 2.$$

From Lemma 2.2, we can write:

$$BF(L_\sigma) \le 2 + \frac{3}{2} [$$
$$\max\left(A(L'_\sigma(1,t_\sigma)), (1+\epsilon)W(L_\sigma(1,t_\sigma)) + 1\right) +$$
$$\max\left(A(L'_\sigma(t_\sigma+1,n)), (1+\epsilon)W(L_\sigma(t_\sigma+1,n)) + 1\right)].$$

From lemmas 2.4 and 2.5, we know that no matter what t_σ is, with high probability we have (remembering that $t'_\sigma = |L_\sigma(1,t_\sigma)|$):

$$\begin{aligned} & A(L'_\sigma(1,t_\sigma)) \\ \le\ & \frac{t'_\sigma}{n'} OPT(L') + o(OPT(L')) \\ \le\ & \frac{t_\sigma}{n} OPT(L') + o(OPT(L')). \end{aligned}$$

Similarly we have:

$$\begin{aligned} A(L'_\sigma(t_\sigma+1,n)) &\le \frac{n-t_\sigma}{n} OPT(L') + o(OPT(L')) \\ W(L_\sigma(1,t_\sigma)) &\le \frac{t_\sigma}{n} W(L) + o(W(L)) \\ W(L_\sigma(t_\sigma+1,n)) &\le \frac{n-t_\sigma}{n} W(L) + o(W(L)). \end{aligned}$$

Since both $OPT(L')$ and $W(L)$ are bounded by $OPT(L)$, we obtain:

$$BF(L_\sigma) \le \frac{3}{2}(1+\epsilon)OPT(L) + o(OPT(L))$$

with high probability. In addition, when the low probability event occurs, we can always use the upper bound $BF(L) \le 1.7 OPT(L) + 2$, valid for any L and proved in [8]. Thus we obtain for the expectation:

$$E_\sigma(BF(L_\sigma)) \le \frac{3}{2} OPT(L) + o(OPT(L)).$$

Now, let us look at the case when $W(L') \le \epsilon^2 W(L)/2$. Since all items of L' are $\ge \epsilon/2$, we have:

$$n' \le W(L')2/\epsilon \le \epsilon W(L).$$

Thus the bins used by Best fit on L are of two types: either they contain some item of L'; there are at most $\epsilon W(L)$ such bins. Or they contain items less than $\epsilon/2$ only, and are thus all filled up to level $1-\epsilon/2$ at least except for possibly one: there are at most $(1+\epsilon)W(L)+1$ such bins. The total number of bins is at most

$$E_\sigma(BF(L_\sigma)) \le (1+2\epsilon)W(L) + 1 \le (1+2\epsilon)OPT(L) + 1.$$

Hence overall we get: $RC(BF) \le (1+\epsilon)3/2$. Since this holds for any ϵ, it implies $RC(BF) \le 3/2$, which concludes the proof of the upper bound.

3 The lower bound

The lower bound presented here was suggested by David Johnson and worked out in collaboration with Johnson, Shor and Young. It is both an improvement and a significant simplification of the original lower bound.

The calculations are only sketched here. Instead of taking items from a fixed list in random order, we will draw n items independently from a fixed distribution. This will generate a random multiset L_n of n items inserted in random order. We will show that as n goes to infinity, the average performance ratio of Best-fit is $1.08\ldots$. It follows that there exists at least one multiset for which the random-order performance ratio is greater than or equal to $1.08\ldots$. We assume that each insertion is $1/2$ with probability p and $1/3$ with probability $q = 1-p$. The optimal algorithm is perfect, while Best-Fit will occasionally create bins with one $1/2$ and one $1/3$.

A bin is called *closed* if it can no longer receive any more items (i.e. its current size is $5/6$ or 1). We have a Markov chain, where the state of the system after i insertions is determined by the collection of open bins, and the transitions correspond to inserting $1/2$ or $1/3$ with probability p or q.

The Markov chain is finite and has just five states: either there is no open bin (a), or one open bin whose current content is $1/2$ (b), or one bin containing $1/3$ (c), or one bin containing a total of $2/3$ (d), or two bins, one containing $1/2$ and one containing $2/3$ (e).

The transitions are drawn in the figure 1. The chain is aperiodic and irreducible. The stationary probabilities are given by the following system (where the name of a state is identified with its stationary

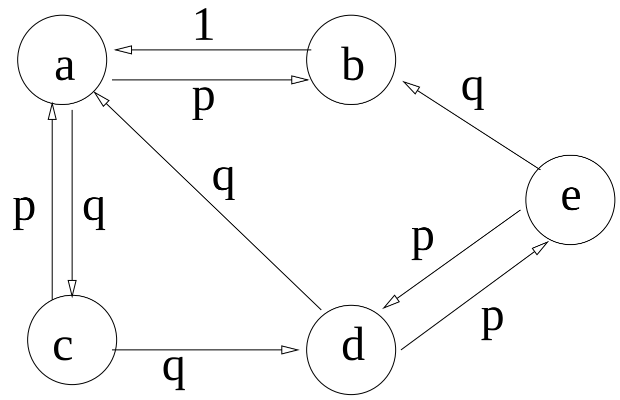

Figure 1: Markov chain describing Best-fit under insertions from $\{1/2, 1/3\}$.

probability):

$$\begin{cases} a &= b + pc + qd \\ b &= pa + qe \\ c &= qa \\ d &= qc + pe \\ e &= pd \\ 1 &= a + b + c + d + e. \end{cases}$$

The number of bins open after n insertions tends to $n(p/2 + q/3)$ for OPT and to $n(a + e)$ for Best-Fit. Solving, we obtain:

$$\frac{E(BF(L_n))}{OPT(L_n)} \sim \frac{1 + 2p - p^2}{1 + 3p/2 + 3p^2/2 - p^3/6}.$$

Maximizing over p yields $1.08\ldots$

Simulations of various distributions seem to indicate that 1.15 should be a lower bound.

Acknowledgments

The author wishes to thank Richard Karp for numerous discussions and Mor Harchol for help with finding the relevant references. Moreover, the author particularly wishes to thank David Johnson, Peter Shor and Neal Young for working on the lower bound (both for getting the new proof and for doing simulations).

References

[1] B. Chandra, *Does randomization help in on-line bin-packing?* IPL 43, 1, 15-19, 1992.

[2] K.L. Clarkson and P.W. Shor. *Algorithms for diametral pairs and convex hulls that are optimal, randomized, and incremental.* In Proceedings of the 4th Ann. ACM Symp. on Comput. Geometry, 12-17, 1988.

[3] E.G. Coffman, D.S. Johnson, P.W. Shor, R.R. Weber. *Markov chains, computer proofs, and average-case analysis of best-fit bin packing.* STOC 1993, 412-421.

[4] E.G. Coffman, Jr. and George S. Lueker. *Probabilistic Analysis of Packing and Partitioning Algorithms.* Wiley & Sons, 1991.

[5] W. Fernandez de la Vega and G.S. Lueker. *Bin Packing can be solved within* $1+\epsilon$ *in linear time.* Combinatorica 1 (4) (1981) 349-355.

[6] Galambos and Frenk, *A simple proof of Liang's lower bound for on-line packing and the extension to the parametric case*, Discrete Applied Math, 41, 1993, 173-178.

[7] Michael R. Garey and David S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness.* Freeman & Co., 1979.

[8] D.S. Johnson, A.Demers, J.D. Ullman, M.R. Garey, and R.L. Graham. *Worst-case performance bounds for simple one-dimensional packing algorithms*, SIAM J. on Computing 3, 229-325 (1974).

[9] D.S. Johnson, *Fast algorithms for bin-packing*, JCSS8, 272-314, 1974.

[10] N. Karmakar and R.M. Karp. *An Efficient Approximation Scheme for the One-Dimensional Bin-Packing Problem.* FOCS 23 (1982), 312-320.

[11] C. Kenyon, Y. Rabani and A. Sinclair. *Biased Random Walks, Lyapunov Functions, and Stochastic Analysis of Best Fit Bin Packing.* These proceedings.

[12] C.C. Lee and D.T. Lee, *A simple on-line packing algorithm*, JACM 32, 562-572, 1985.

[13] Frank M. Liang, *A lower bound for on-line bin-packing*, IPL 10,2, 1980.

[14] P.Ramanan, D.J. Brown, C.C. Lee and D.T. Lee. *On-line bin-packing in linear time*, J. Alg. 10, 305-326, 1989.

[15] W.T. Rhee and M. Talagrand. *On Line Bin Packing with Items of Random Size*, Mathematics of Operations Research, 18 (2), 1993, 438-445.

[16] M.B. Richey, *Improved bounds for refined harmonic bin packing*, unpublished, 1990.

[17] P.W. Shor. *The average-case analysis of some on-line algorithms for bin packing.* Combinatorica 6 (2) (1986) 179-200.

[18] A. Van Vliet, *An improved lower bound for on-line bin packing algorithms*, IPL 43, 5, 277-284, 1992.

[19] A.C. Yao, *New algorithms in bin packing*, JACM 27, 207-227, 1980.