COMBINATORICA 6 (2) (1986) 179—200

THE AVERAGE-CASE ANALYSIS OF SOME ON-LINE ALGORITHMS FOR BIN PACKING

P. W. SHOR

Received 1 March 1985
Revised 1 December 1985

In this paper we give tighter bounds than were previously known for the performance of the bin packing algorithms Best Fit and First Fit when the inputs are uniformly distributed on [0, 1]. We also give a general lower bound for the performance of any on-line bin packing algorithm on this distribution. We prove these results by analyzing optimal matchings on points randomly distributed in a unit square. We give a new lower bound for the up-right matching problem.

1. Introduction

The bin packing problem is: given n items of size between 0 and 1, fit them into the least number of bins such that the sum of the sizes of the items in any bin does not exceed 1. This is an NP-complete problem that has received much study [6].

One topic of study has been the behavior of simple algorithms for bin packing. Some of the first results, proved by Johnson et al. [12, 13] were that in the worst case, the First Fit algorithm does not use more than $(17/10)\mathrm{OPT}+1$ bins and First Fit Decreasing does not use more than $(11/9)\mathrm{OPT}+4$ bins, where OPT is the number of bins used in the optimal packing. Recently, asymptotically optimal algorithms have been found [9, 15] where the ratio of bins used to the optimal number approaches 1 as the number of items goes to infinity. The best asymptotic result so far is that of Karmarkar and Karp [15], who give an algorithm that never uses more than $\mathrm{OPT}+$ $+O((\log \mathrm{OPT})^2)$ bins.

Work has also been done on average-case analysis. For the average case, one must assume some distribution on the item sizes. Much of the work done on this problem has assumed that the item sizes are uniformly distributed on [0, 1]. For this distribution, Coffman, Hofri, So and Yao [7] showed that the expected ratio between the algorithm's performance and the optimal packing was 4/3 for the algorithm Next Fit. More recently, Frederickson [10] showed that for First Fit Decreasing, this ratio approaches 1 as the number of items goes to infinity. It was then shown [16, 21] that the expected wasted space for this algorithm is $\Theta(n^{1/2})$, where wasted space is the total amount of empty space in partially filled bins, and n is the number of items. The latest

A preliminary version of this paper appeared in *Proc. 25th IEEE Symposium on Foundations of Computer Science, 193—200.*

This research was done while the author was at the Massachusetts Institute of Technology Dept. of Mathematics and was supported by a NSF Graduate Fellowship and by Air Force grant OSR-82-0326.

AMS subject classification (1980): 68 C 05

results along these lines are in Bentley et al. [3]. They show that the expected wasted space for First Fit is $O(n^{4/5})$.

There are also results for other distributions. Karmarkar [14] analyzed Next Fit for a uniform distribution on $[0, \alpha]$. Karmarkar, Karp, Lueker and Murgolo [22] have generalized the upper bound of $O(n^{1/2})$ wasted space for First Fit Decreasing to any symmetric or decreasing distribution. Recently, Bentley et al. [3] proved the surprising result that First Fit Decreasing, when packing items uniformly distributed on $[0, \alpha]$, $\alpha \leqq 1/2$, produces with high probability constant wasted space.

In this paper we consider on-line bin packing algorithms. On-line algorithms are algorithms, for example Next Fit or First Fit, that assign items to bins as soon as the items are input. For some applications, on-line algorithms are necessary. Unfortunately, unlike off-line algorithms, on-line algorithms can never achieve asymptotically optimal worst-case performance. It has been shown [5, 20] that in the worst case, any on-line algorithm must use at least 1.536 times as many bins as the optimal packing.

We will investigate the average-case behavior of on-line algorithms. We first show, in Section 2, that any on-line algorithm that does not know in advance the number of items cannot produce expected wasted space $o(n^{1/2}(\log n)^{1/2})$, when packing items uniformly distributed on $[0, 1]$. This contrasts with off-line algorithms (or on-line algorithms that are given the number of items in advance) such as First Fit Decreasing, which can produce $\Theta(n^{1/2})$ wasted space. In Section 3, we show that the algorithm Best Fit, given items from a uniform distribution on $[0, 1]$, is equivalent to planar upright matching [17] and the discrepancy of a lower layer [8]. Their bounds of $O(n^{1/2} \log n)$ and $\Omega(n^{1/2}(\log n)^{1/2})$ thus apply to the wasted space produced by Best Fit. We then improve their lower bound to $\Omega(n^{1/2}(\log n)^{3/4})$. In [19], a matching upper bound of $O(n^{1/2}(\log n)^{3/4})$ for up-right matching is proved, showing that Best Fit wastes $\Theta(n^{1/2}(\log n)^{3/4})$ space. In the following section, we show that First Fit is equivalent to a different planar matching problem, and analyze this problem to obtain bounds of $\Omega(n^{2/3})$ and $O(n^{2/3}(\log n)^{1/2})$. Section 5 contains remarks, including directions for further research. Section 6 contains acknowledgements, and Section 7 contains references.

Throughout this paper we use the phrase "with high probability" to mean with probability at least $1-n^{-1}$. In bin packing, anything with probability lower than $1/n$ can be ignored in analyzing expected behavior. This is because n items can never take more than n bins to pack. Thus, if a bin packing algorithm packs items using $O(f(n))$ bins with probability $1-n^{-1}$, the expected number of bins used is

$$\left(1-\frac{1}{n}\right) O(f(n))+\frac{1}{n} O(n) = O(f(n)).$$

We will be measuring the performance of bin packing algorithms both by number of bins used and by wasted space. The wasted space is the total amount of empty space in bins, so the amount of wasted space is the number of bins minus the sum of the sizes of the items. The expected item size is $1/2$, so we have

$$E(\text{number of bins}) = \frac{n}{2}+E\,(\text{wasted space}).$$

For some proofs, the number of bins is a more convenient measure, while for other proofs, wasted space is more convenient.

2. A General lower bound

In this section, we prove a general lower bound for any on-line algorithm that does not know the number of items that will be input. In particular, we show that any such algorithm cannot have an $o(\sqrt{n \log n})$ bound on expected wasted space, when packing items uniformly distributed on [0, 1]. We do this by relating the performance of the algorithm to a problem concerning bipartite matching of points randomly distributed in the unit square. We then use a theorem of Ajtai, Komlós and Tusnády [1] to obtain a bound for this matching problem. This lower bound contrasts with off-line algorithms or on-line algorithms that are given the number of items in advance. These can achieve $\Theta(\sqrt{n})$ average wasted space, which is within a constant factor of optimal.

Definition. An *on-line algorithm* for bin packing is one that packs each item as soon as it is received.

The model we will use is that the algorithm has no information about the number of items until it receives the last one. More specifically, we set up a distribution as follows:

1. Choose k from 1 to n at random.
2. Choose k items uniformly on [0, 1].
3. Input these items to the algorithm, ending with a "stop" signal after the kth item.

We now show the expected wasted space is $\Omega(\sqrt{n \log n})$ if the list of items is chosen in this manner.

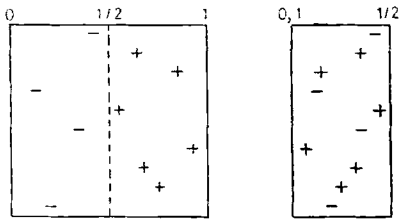

Fig. 1. Representing items as points in the plane

Theorem 1. *Let A be an on-line algorithm that receives k items uniformly distributed on* [0, 1] *and is not given any information about the value of k until it has received the last item. Let k be chosen with equal probability from the integers between* 1 *and n. Then A must waste an expected value of $\Omega(\sqrt{n \log n})$ space.*

Proof. For a random list L of n items, consider the wasted space that algorithm A produces when packing the first k items of the list. We will show the average of this wasted space over k must be with high probability $\Omega(\sqrt{n \log n})$. This proves the theorem, since choosing k first and then choosing a random sequence of length k is equivalent to first choosing a random sequence L of length n and then only looking at the first k items. We can choose k second because our conditions on the algorithm

A require it to pack the initial items of a sequence in a way that does not depend on the number of items in the sequence or on the items which will be received later.

We give a lower bound on average wasted space by analyzing a planar matching problem. To convert a list of items to a planar matching problem, we represent the items received by the algorithm by points in a unit square. The x-coordinate will be the size of the item. The y-coordinate will depend on the time the item was received. To fit the n items into the square, we put the jth item at distance j/n from the top. Next, we label the items larger than 1/2 with '+' and those smaller than 1/2 with '−'. We then fold the plane about the line $x=1/2$ (See Figure 1), so a + point with x-coordinate $s>1/2$ will be moved so it has x-coordinate $1-s$. For the time being consider only items with size between 1/3 and 2/3. We join every + point in this range to a — point in this range representing an item packed in the same bin, if there is such an item. This gives a bipartite matching M between + and − points.

We examine this bipartite matching M more closely. It is a matching on a bipartite graph G between the — points in (1/3, 1/2) and the + points in (1/2, 2/3). A − point can only be matched to a + point to the right of it, since otherwise the sizes of the corresponding items would sum to more than 1. Only one item larger than 1/2 can fit in a bin, only two items in the range (1/3, 1/2) can fit in a bin, and an item in (1/3, 1/2) cannot fit in a bin with an item in (2/3, 1). Thus, the total number of + points plus half the number of unmatched − points in G is a lower bound on the number of bins used. We prove the lower bound on the expected number of bins by giving a lower on the average number of unmatched − points. The following lemma proves this lower bound.

Lemma 1. *Let there be n points randomly distributed in a unit square, and let each point have an equal probability of being a + or − point. If we have a matching such that every matched − point is matched to a + point to its right, then the sum of the vertical lengths of the edges,* $\sum_e |y_{e_+}-y_{e_-}|$, *in a matching minimizing this sum has expected value* $\Omega(\sqrt{n}\log n)$. *An unmatched point is considered to be matched to the bottom of the square.*

The sum of the vertical lengths of the edges in the matching is equal to the integral over t of the number of edges crossing the horizontal line $y=t$. The y-axis is time, and the number of edges crossing the line $y=t$ is the number of unmatched points at time t. Thus, this integral is the average number of unmatched points over time. The lower bound given in Lemma 1 thus bounds the expected number of unmatched points. By applying this lemma to the points with x-coordinate greater than 1/3, we get a $\Omega(\sqrt{n\log n})$ lower bound on the expected number of unmatched points in G. This proves Theorem 1. ∎

Proof of Lemma 1. To prove the lemma, we use the following closely related result due to Ajtai, Komlós, and Tusnády.

Theorem. (Ajtai, Komlós and Tusnády) *If there are n + points and n − points randomly distributed in a unit square, then the sum of the lengths of the edges of a perfect matching with this sum minimized will average* $\Theta(\sqrt{n\log n})$.

This theorem does not directly prove the lemma because the models of the distributions of the points are different. In the model of Ajtai, Komlós and Tusnády, the points are distributed randomly throughout the unit square. In the model required to prove the theorem, the points are distributed randomly horizontally (in size) but come from a different distribution vertically (in time). In the AKT model, there are equal numbers of $+$ and $-$ points. In our model, each point has an equal chance of being a $+$ or a $-$. In the AKT model, we minimize the sum of the edge lengths instead of the vertical distances between the points, and we can match edges in any direction, and not just rightward.

The first two differences mentioned above are relatively minor. The first difference is in the vertical distribution of the points. In the AKT model, the points are distributed randomly vertically. In our model, the points are evenly spaced vertically before the points not in $[1/3, 2/3]$ are deleted. We thus get approximately one third of a set of evenly spaced points. To change from random points to evenly spaced points, we move all the points up or down without changing their vertical order until they are evenly spaced vertically. The expected distance moved by a point in this process is $\Theta(1/\sqrt{n})$. Similarly, our distribution can be changed to evenly spaced points by moving points an expected distance of $\Theta(1/\sqrt{n})$. Moving the points in this manner changes the expected average edge length by $O(1/\sqrt{n})$, and thus changes the expected sum of the edge lengths by $O(\sqrt{n})$.

The second difference between the models is that in the AKT model, there are an equal number of $-$ and $+$ points, while in our model, every point has an even chance of being $-$ or $+$. Assume without loss of generality that there are more $+$ points. To change from our model to the AKT model, place enough $-$ points randomly in the square to obtain an equal number of each kind of point. The expected number of points we need to add is $\Theta(\sqrt{n})$. Since the maximum length of an edge in the unit square is $\sqrt{2}$, adding or removing a point cannot change the sum of the edge lengths in the optimal matching by more than $\sqrt{2}$. Thus, adding $\Theta(\sqrt{n})$ points changes the sum of the edge lengths by $O(\sqrt{n})$.

The remaining difference between the models is that for our theorem, we require a rightward matching and only measure the vertical component of the edge lengths. To show that this difference does not matter, we go to the dual problem. Ajtai, Komlós and Tusnády prove their theorem by finding a dual weight function w that gives a lower bound for the cost of any matching. We will use their weight function w to find a new weight function w' which gives a lower bound on the vertical cost of any *rightward* matching.

The AKT weight function w is a function mapping the unit square to $[-1, 1]$ such that:

1. the slope of w is never more than 1,
2. $\sum w(P_+) - \sum w(P_-)$ has expected value $\Omega(\sqrt{n \log n})$,
3. at the boundary of the square, $w = 0$.

Here, $\sum w(P_+)$ and $\sum w(P_-)$ denote the sum of the weights of the $+$ and $-$ points, respectively. The slope of w between two points P_1 and P_2 is $|w(P_1) - w(P_2)|/d(P_1, P_2)$, where $d(P_1, P_2)$ is the distance between P_1 and P_2.

The lower bound on the sum of edge lengths in any matching M follows from the observation that if an edge e between points e_+ and e_- has length $d(e)$, then

$d(e) \geqq w(e_+) - w(e_-)$. Summing over the edges of the matching M, we get

$$\begin{aligned} \sum_{e \in M} d(e) &\geqq \sum w(e_+) - \sum w(e_-) \\ &= \sum w(P_+) - \sum w(P_-) \\ &= \Omega(\sqrt{n \log n}). \end{aligned}$$

We now use this result to obtain a dual function w' for rightward matching that maps the unit square to $[0, 1]$ such that:

1. the slope of w' is never more than 1,
2. $\sum w'(P_+) - \sum w'(P_-)$ has expected value $\Omega(\sqrt{n \log n})$,
3. w' is decreasing on any horizontal line (y constant),
4. $w' = 0$ at the bottom boundary of the square ($y = 0$).

We find this w' by using the AKT result to obtain a w on the middle third of the square ($1/3 \leqq y \leqq 2/3$) and then adding a certain function to it so that is satisfies the conditions above. By rescaling the middle third to a 1×1 square, we can apply the AKT theorem to it to obtain a function w that maps $[0, 1] \times [1/3, 2/3]$ to the interval $[-1/6, 1/6]$, such that:

1. the slope of w is never more than $1/6$,
2. $\sum w(P_+) - \sum w(P_-)$ has expected value $\Omega(\sqrt{n \log n})$ for the points in the middle third of the square,
3. $w = 0$ on the boundaries of this region.

We now let

$$w' = \begin{cases} \frac{1}{2}(1-x)y & 0 \leqq y \leqq \frac{1}{3} \\ w(x, y) + \frac{1}{6}(1-x) & \frac{1}{3} \leqq y \leqq \frac{2}{3} \\ \frac{1}{2}(1-x)(1-y) & \frac{2}{3} \leqq y \leqq 1. \end{cases}$$

The function w' satisfies the conditions listed above for a dual function for rightward matching. Since we added $(1/6)(1-x)$ to the function w on the middle third of the square, it is now decreasing rightwards. It is easy to check the slope conditions, since slope is additive and the slope of the added function and of w are both small. We are adding a fixed function to w, so it does not affect the expectation of $\sum w(P_+) - \sum w(P_-)$. Thus w' satisfies the conditions above.

Now, suppose we have an edge (e_+, e_-) of our rightward matching. Let (x_+, y_+) be the coordinates of e_+ and (x_-, y_-) the coordinates of e_-. Since it is a rightward matching, $x_- \leqq x_+$. Let c be the point (x_+, y_-). Then since w' is decreasing rightward, $w'(e_+) \leqq w'(c)$. We thus have

$$\begin{aligned} w'(e_+) - w'(e_-) &\leqq w'(c) - w'(e_-) \\ &\leqq d(c, e_-) \\ &= |y_+ - y_-|. \end{aligned}$$

Summing the above equation over all edges (including "edges" from unmatched points to the boundary), we obtain Lemma 1. ∎

If we know in advance how many items the algorithm will receive, say n, then Theorem 1 no longer holds. The following algorithm wastes $\Theta(\sqrt{n})$ space: Pack the first $n/2$ items one per bin and sort them. Pack the next $n/2$ using First Fit. The proof in Lueker [21] that shows that First Fit Decreasing wastes $\Theta(\sqrt{n})$ space also holds for this algorithm.

3. Best Fit

We now discuss the Best Fit (BF) algorithm. In this algorithm, each item is placed into the fullest bin in which it fits at the time of arrival. In this section, we will give bounds on the expected space wasted by Best Fit.

Theorem 2. *The expected wasted space produced by the algorithm Best Fit, when packing n items uniformly distributed on* $[0, 1]$, *is* $\Theta(n^{1/2}(\log n)^{3/4})$.

We will first prove the upper bound and then the lower bound. These proofs use the techniques of matching in a plane that we introduced in the previous section. In Section 3a we show that the wasted space produced by Best Fit is bounded above by the up-right matching problem investigated by Karp, Luby and Marchetti [17]. The next section shows Best Fit is also bounded below by up-right matching, and is thus within a constant factor of it. By using previously known bounds for up-right matching, these results give that Best Fit wastes $\Omega(n^{1/2}(\log n)^{1/2})$ and $O(n^{1/2}\log n)$ space. Section 3c contains the proof of a new lower bound of $\Omega(n^{1/2}(\log n)^{3/4})$ for up-right matching. In [19], a matching bound of $O(n^{1/2}(\log n)^{3/4})$ for up-right matching is shown, thus giving the bound in the theorem.

3a. The upper bound

For the analysis of Best Fit, we will use a variation of the algorithm Best Fit, which we call Matching Best Fit (MBF). The algorithm MBF behaves exactly like BF, except that MBF may not put an item into a bin containing any item of size less than 1/2. That is, any bin containing an item of size less than 1/2 is considered full. Thus, MBF packs at most two items per bin, and in a two item bin, puts an item larger than 1/2 in the bin first.

We first show the upper bound on wasted space. This proof involves two parts. First we show that MBF always uses at least as many bins as BF. Second, we show that MBF is equivalent to the up-right matching problem discussed by Karp et al. [17]. The $O(n^{1/2}\log n)$ upper bound they obtain for this matching problem thus also applies to the wasted space in the BF algorithm.

To prove that MBF always uses at least as many bins as BF, we need the following lemma, to which we will be referring throughout the paper. Suppose that L is a list of items. We denote the packing of L using algorithm A by $A(L)$, and the number of bins used by this packing by $\# A(L)$. A similar lemma for the algorithm 2-First Fit was proved by Bentley et al. [3]; we use the same techniques for the proof that they did.

Lemma 2 does not hold for the algorithms Best Fit or First Fit; the restriction that the algorithm can never put more than two items in a bin seems necessary.

Lemma 2. *If L' is a list obtained by removing one item from L, then*

$$\#\mathrm{MBF}(L) \geqq \#\mathrm{MBF}(L') \geqq \#\mathrm{MBF}(L) - 1.$$

Proof. We show that, if we consider any bins with two items in them (and thus "full") to be identical, then L and L' differ in at most one bin. In fact, they are related in one of the following ways:

A) MBF (L) can be obtained from MBF (L') by replacing a 1-item bin by a full (2-item) bin.

B) MBF (L) can be obtained from MBF (L') by adding a 1-item bin.

We prove Lemma 2 by induction. We must show that if we add an item p to two packings related by A or B, they will still be related by A or B.

Suppose that MBF (L) and MBF (L') satisfy A. Then there is a 1-item bin in MBF (L') that is not in MBF (L). Call this bin b. If the next item to be added, say p, does not go into the bin b, then the two packings will still be related by A since the item p goes into the same bin in both of them. If the added item goes into bin b in MBF (L') then in MBF (L) it will either go into an empty bin or it will go into a different 1-item bin. In the first case, the two packings will be related by B. In the second case they will be related by A (See Figure 2). A similar analysis holds if MBF (L) and MBF (L') are related by B.

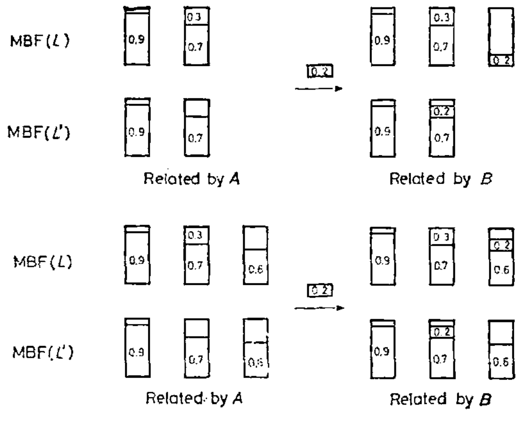

Fig. 2. Proof of Lemma 2

We must still do the base case of the induction. For this, we must show that when the item q in L and not in L' arrives, the two packings will be related by either A or B. Before this item arrives, MBF (L) = MBF (L'). The item q must either go into a bin by itself, or into a 1-item bin. In the first case, the packings are related by B; and in the second case, they are related by A. ∎

We can now prove the following lemma:

Lemma 3. *If L is a list of items, then*

$$\# \mathrm{MBF}(L) \geqq \# \mathrm{BF}(L).$$

Proof. Obtain a list L' by removing from L all items which were packed by BF into a bin already containing an item of size less than 1/2. We remove all items that are packed into bins considered full by MBF, so MBF (L') packs the items of L' in exactly the same way that BF (L) packs the items of L', and in BF(L) the items of $L-L'$ are packed into bins considered full by MBF. Thus, we get

$$\# \mathrm{MBF}(L') = \# \mathrm{BF}(L).$$

By Lemma 2, we have

$$\# \mathrm{MBF}(L) \geqq \# \mathrm{BF}(L),$$

since L' was derived from L by removing items. Combined with the previous equality, this proves Lemma 3. ▮

Proof of upper bound in Theorem 2. Now we are ready to convert the MBF problem to up-right planar matching. The up-right planar matching problem from [17] is: given +'s and −'s uniformly distributed in a unit square, match as many −'s to +'s as possible, with the constraint that a − can only be matched with a + above and to the right of it. Karp et al. used this problem to obtain bounds on 2-dimensional bin packing [17]. Here, by considering time as a second dimension, we use it to obtain bounds on 1-dimensional bin packing for the Best Fit algorithm.

To convert a bin packing problem to an up-right matching problem, we use the same technique as before. We represent each item as a point in the unit square with x-coordinate the size of the item and y-coordinate the time of arrival. Then, we fold the square onto itself about the axis $x=1/2$, using + or − for items larger or smaller than 1/2. (See Figure 1.) For every bin containing two items in MBF, we will put an edge between these items to produce a matching M.

We claim that every edge in a MBF matching M matches a − to a + above and to the right of it, and furthermore that MBF finds a maximum such matching. Every + is above the — with which it is matched, since if two items are put in a bin by MBF, the first is larger than 1/2 (i.e., is a + item) and the second is smaller than 1/2 (i.e., a −item). Every + is to the right of the corresponding −, since if it were to the left, the two items would have sizes summing to more than 1 and so could not fit into a single bin. Furthermore, the algorithm MBF processes the —'s from top to bottom, matching each with the leftmost available +. (Here available means the + is above and to the right of the − and is still unmatched.) This is the algorithm shown in [17] to produce an optimal up-right matching.

Since there is a bound of $O(n^{1/2} \log^{3/4} n)$ on the expected number of unmatched points in an optimal up-right matching [19], we obtain a bound of $n/2+O(n^{1/2} \log^{3/4} n)$ on the number of bins used and hence the wasted space is $O(n^{1/2} \log^{3/4} n)$. ▮

3b. The lower bound

We now prove a lower bound on the wasted space for BF. We do this by showing that many bins are packed by putting a large item in the bin first and a small one second. This gives an up-right matching which has an expected value of $\Omega(\sqrt{n}(\log n)^{3/4})$ unmatched points. These unmatched points correspond to extra bins.

In this section, we will say that the items in a bin are stacked in the same order as they were put in the bin. Thus, the first item to be put in a bin is on the *bottom* and the last item on the *top*. The *height* of a bin is the sum of the sizes of the items it contains.

For the proof, we will consider what happens to the items with size between 1/3 and 2/3. We will call items with size between 1/3 and 1/2 *s-items* (s for small) an items with size between 1/2 and 2/3 *b-items* (b for big.) We now prove the following lemma.

Lemma 4. *At any time, the probability that the next item produces a bin with a b-item on top of an s-item is less than or equal to the probability that it produces a bin with two s-items.*

Proof. We can only get a bin with a b-item on top of an s-item if there is an s-item in a bin B which is less than half full. At any time, there can be only one such bin B. Suppose this bin is filled to height a. (The size of the s-item in the bin B might be smaller than a.) We claim that if the next item has size between a and $1-a$, it will go into B. Clearly, it will fit into the bin B. Furthermore, this bin is the only non-empty one with height less than $1-a$. This was certainly true when the first item was put into B, since otherwise it would not have been put there. It has remained true, since no items of size less than $1-a$ have started bins since. Thus, the probability of a b-item being put into the bin B is the probability of getting a b-item between 1/2 and $1-a$. This is equal to the probability of getting an s-item between a and 1/2, which is less than or equal to the probability of an s-item being put into B. ∎

Proof of lower bound in Theorem 2. We will call a bin with two s-items a $\begin{bmatrix} s \\ s \end{bmatrix}$ bin, and denote the number of these bins by $\begin{vmatrix} s \\ s \end{vmatrix}$. We will call a bin with a b-item on top of an s-item a $\begin{bmatrix} b \\ s \end{bmatrix}$ bin, and so on. If we restrict our attention to otems in the range (1/2, 2/3), there are five types of bins: $\begin{bmatrix} s \\ s \end{bmatrix}$, $\begin{bmatrix} s \\ b \end{bmatrix}$, $\begin{bmatrix} b \\ s \end{bmatrix}$, $\begin{bmatrix} \ \\ s \end{bmatrix}$ and $\begin{bmatrix} \ \\ b \end{bmatrix}$. There may also be non-empty bins with no items larger than 1/3 in them. As we are proving a lower bound, we may ignore these bins.

There is an equal probability of receiving an s-item and receiving a b-item. (The probability is 1/6.) By using bounds for the tail of the binomial distribution, we get that with probability at least $1-1/n$, the number of b-items and the number of s-items differ by $O(\sqrt{n \log n})$. It follows that

$$\begin{vmatrix} \ \\ b \end{vmatrix} = 2\begin{vmatrix} s \\ s \end{vmatrix} + \begin{vmatrix} \ \\ s \end{vmatrix} \pm O(\sqrt{n \log n}).$$

Lemma 4 shows that at any time, the probability of creating a $\begin{bmatrix} b \\ s \end{bmatrix}$ bin is less than the probability of an $\begin{bmatrix} s \\ s \end{bmatrix}$ bin. Thus, with probability at least $1-1/n$,

$$\begin{vmatrix} b \\ s \end{vmatrix} \leqq \begin{vmatrix} s \\ s \end{vmatrix} + O(\sqrt{n \log n}).$$

The probability of getting an s-item and the probability of getting a b-item are both 1/6. Thus, with probability $1-1/n$, there are $n/3 \pm O(\sqrt{n \log n})$ b-items and s-items. The $\begin{bmatrix} s \\ b \end{bmatrix}$ bins give an up-right matching between the s-items and the b-items. The s- and b-items are distributed uniformly on $[1/3, 2/3]$, so, by Theorem 3 proved in the next section, with high probability any up-right matching has $\Omega(\sqrt{n}\,(\log n)^{3/4})$ unmatched points. Thus, we have

$$\begin{vmatrix} s \\ b \end{vmatrix} = \frac{n}{6} - \Omega(\sqrt{n}\,(\log n)^{3/4}).$$

We have that the total number of b-items is $n/6 \pm O(\sqrt{n \log n})$. This gives

$$\begin{vmatrix} s \\ b \end{vmatrix} + \begin{vmatrix} \\ b \end{vmatrix} + \begin{vmatrix} b \\ s \end{vmatrix} = \frac{n}{6} \pm O(\sqrt{n \log n}).$$

Combining the two above equations, we obtain

$$\begin{vmatrix} \\ b \end{vmatrix} + \begin{vmatrix} b \\ s \end{vmatrix} = \Omega(\sqrt{n} \log^{3/4} n).$$

Using the inequalities on $\begin{vmatrix} \\ b \end{vmatrix}$ and $\begin{vmatrix} b \\ s \end{vmatrix}$ obtained above, we get

$$3\begin{vmatrix} s \\ s \end{vmatrix} + \begin{vmatrix} \\ s \end{vmatrix} = \Omega(\sqrt{n} \log^{3/4} n).$$

Thus, there are $\Omega(\sqrt{n} \log^{3/4} n)$ bins containing no items larger than 1/2. Since the number of items larger than 1/2 is with high probability $n/2 \pm O(\sqrt{n \log n})$, this shows that the number of bins used is with high probability $n/2 + \Omega(\sqrt{n} \log^{3/4} n)$. ▮

3c. Up-right matching

In this section, we obtain the $\Omega(n^{1/2}(\log n)^{3/4})$ lower bound for up-right matching.

Theorem 3. *Suppose there are n points uniformly distributed in a unit square, and each point has an equal probability of being a + or a − point. If these points are matched such that every − point is matched to a + point above and to the right of it, then the expected number of unmatched points in a maximum such matching is* $\Omega(n^{1/2}(\log n)^{3/4})$.

Proof. In order to make the proof easier, we rotate the square 45°. With this rotation a $-$ point can only be matched to a $+$ point above it, and the slope of the edge joining them must be larger than 1 or smaller than -1. (See Figure 3.) To obtain a lower bound of k for the number of unmatched points in an up-right matching in a rotated square, it suffices to split the square into two sections, such that in the lower section there are k more $+$ points than $-$ points, and such that the boundary dividing the sections is a curve that joins the left corner of the square to the right corner, and always has slope between -1 and 1. A lower section of a square divided in this way is also called a *lower layer* [8]. (See Figure 4.) With such a boundary, the excess $+$ points in the lower section can never be matched to $-$ points in the upper section, so they must remain unmatched. We will construct a boundary such that the expected number of extra $+$'s below it is $\Theta(n^{1/2}(\log n)^{3/4})$.

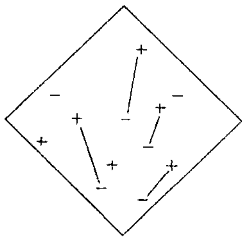

Fig. 3. Rotating the square

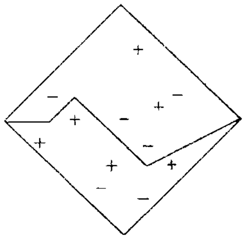

Fig. 4. A lower layer

We will produce the boundary in stages. At each stage, the boundary will consist of line segments and triangles. If a triangle is on the boundary, then in any subsequent stage the boundary will pass through two vertices of the triangle, and the portion of the boundary between these vertices will be contained within the triangle. For example, in Figure 5, the final boundary will lie in the shaded areas. To obtain the boundary at the next stage, we replace every triangle with either a line segment or with two triangles each having a quarter of the area of the old triangle. We repeat this step until the triangles are so small that they contain on the average only one point each. To keep the slope of the boundary from exceeding 1, we do not refine any triangle which has a side of slope 1. We shall later show that we only stop refining triangles on a small portion of the boundary, so this modification does not change the analysis.

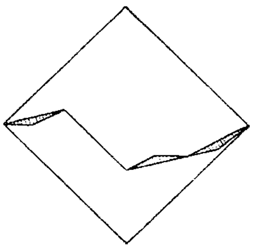
Fig. 5. The boundary at stage 2

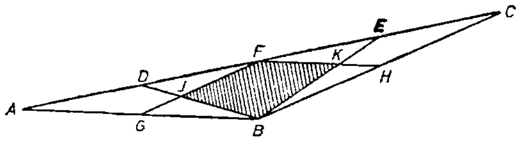

Fig. 6. A typical triangle on the boundary

The general step of replacing a triangle by two smaller triangles is illustrated in Figure 6. The points G and H are midpoints of AB and BC, and D, F and E divide AC into quarters. In the next stage, this triangle will be replaced either by triangles ADB and BEC or by triangles AGF and FHC. The *central quadrilateral* is defined by the four edges BJ, BK, FJ and FK; it is shaded in Figure 6. We put the central quadrilateral below the boundary when it contains more +'s than −'s. In the triangles the central vertex (B in Figure 6) can point either up or down. If the triangle's vertex points down, as in the picture, the refinement to AGF and FHC places the central quadrilateral below the boundary. Otherwise, the refinement to ADB and BEC will do this. We begin with one large triangle, as shown in Figure 7. This is an isosceles triangle which has two vertices at the left and right corners of the square, and which has two sides with slopes of $\pm s$, where $s=(\log n)^{-1/2}$.

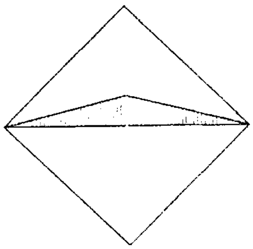

Fig. 7. The boundary at the start

We now list several properties of the triangles we will be using. The letters refer to the example in Figure 6. The first three properties can be proved using elementary Euclidean geometry; the fourth requires induction and the fact that slope $(AD)+$ $+$slope $(DB)=2\cdot$slope (AB).

1. The three vertices (A, B, C) have x-coordinates evenly spaced, so the x-coordinate of B is the average of the other two. Thus, the segment BF is vertical.
2. Each triangle has exactly 1/4 the area of the previous one.
3. The area of the central quadrilateral is 1/3 the area of the triangle.
4. If the slopes of the sides of a triangle are $(i-1)s$, is and $(i+1)s$, then the two triangles it is refined to will either both be similar to it, in which case they have sides with the same slopes, or one triangle will have sides with slopes $(i-2)s$, $(i-1)s$ and is, and the other will have sides with slopes is, $(i+1)s$ and $(i+2)s$.

If we let the first triangle be stage 0, at stage i we are testing 2^i regions each of area $(1/6)s/4^i$ to decide whether or not to include them. (Recall 0, $\pm s$ were the slopes of the sides of the original triangle.) The expected difference between the number of

$+$ and $-$ points in a region of area A is $\Theta(\sqrt{nA})$. This is true since a region of area A contains on the average nA points, and each of these has an equal probability of being a $+$ or a $-$ point. Thus, each stage adds on the average $\Theta(\sqrt{ns})$ extra $+$ points to the lower region. After $\Theta(\log n)$ stages, we have an expected number of $\Theta(\sqrt{ns}\log n)=\Theta(\sqrt{n}(\log n)^{3/4})$ extra $+$ points in the lower region.

We still must show that the slope of the boundary stays between -1 and 1. If we keep refining all the triangles, it will not. We modify the procedure so that any time we would produce an edge of a triangle with slope larger than 1 (or smaller than -1), we stop changing the boundary along this segment. We must show this will not affect the analysis.

We can consider the triangles to be organized in a binary tree, so the two children of any triangle are its two refinements on the next level. By property 4, the slopes of the sides of a triangle differ from the slopes of the sides of its parent by $-s$, 0, or $+s$. If one of the children of a triangle is obtained from its parent by adding $-s$, then by property 4, the other is obtained by adding $+s$. A stochastic process can be derived from such a tree by starting at the root and choosing each child with probability 1/2. At each node, the choice is between adding and subtracting the same value, either 0 or s. This process is therefore a martingale with variance at each step at most s^2. Thus, if the tree has depth $c\log n$, the variance of the values at the leaves is $s^2c\log n$. By making $s^2c\log n=1/4$, we ensure that the variance of the value of a randomly chosen leaf is at most 1/4. Since the mean is 0, this ensures that at most 1/4 of the leaves have slopes $\geqq 1$.

We have shown that after $O(\log n)$ steps, we stop refining the boundary on at most 1/4 of the triangles. The remaining 3/4 of the triangles will still give $O(n^{1/2}(\log n)^{-1/4})$ excess $+$ points at each stage with high probability because the excess number of $+$ points in a triangle containing nA points is distributed binomially, and even after removing the top quarter of this binomial distribution, the expected value of excess $+$ points is $\Theta(\sqrt{nA})$. ∎

4. First fit

In the algorithm First Fit (FF), the bins are kept in order. As each item arrives, it is placed into the first bin in which it fits. If it does not fit in any bin, it is placed in an empty bin at the end of list of bins. We also define Matching First Fit (MFF) as we did for Best Fit. In MFF, no item can be put into a bin that has an item smaller than 1/2 in it.

In this section, we prove the following:

Theorem 4. *The expected wasted space produced by the algorithm First Fit, when packing n items uniformly distributed on* $[0, 1]$, *is* $\Omega(n^{2/3})$ *and* $O(n^{2/3}(\log n)^{1/2})$.

The upper bound is obtained in a manner similar to that of Best Fit, and the lower bound uses some new techniques. We will discuss the upper bound first.

4a. The upper bound

The proof of the upper bound for First Fit is quite similar to the proof for Best Fit. As before, we convert the problem to a matching problem. We then prove an upper bound on the matching problem by producing a suitable matching.

We use MFF to obtain the upper bound. Lemma 2 holds for MFF as well as for MBF, and the proof is identical. We thus get

Lemma 5. *If L' is a list obtained by removing one item from L, then*

$$\# \mathrm{MFF}(L) \geqq \# \mathrm{MFF}(L') \geqq \# \mathrm{MFF}(L)-1.$$

The proof that $\# \mathrm{MFF}(L) \geqq \# \mathrm{FF}(L)$ proceeds exactly as did the proof for Best Fit; the similar inequality $\# 2\mathrm{FF}(L) \geqq \# \mathrm{FF}(L)$ is shown in [3]. As before, we represent items as points in the plane with x-coordinate size and y-coordinate time. We then mark items greater than 1/2 with $+$ and items less than 1/2 with $-$ and fold the plane around the line $x=1/2$, as in Figure 1. We derive a matching by connecting a $+$ point and a $-$ point corresponding to two items in the same bin in MFF. This will be an up-right matching for the same reasons that it was for MBF. However, MFF differs from MBF in that it used a different algorithm to find the matching. The MFF matching is derived by processing the $-$'s from top to bottom, and matching each $-$ with the *highest* available $+$. This algorithm no longer produces a maximum cardinality up-right matching. We will show that it produces a maximum cardinality up-right matching satisfying the following condition:

Condition 1. *If $e_1=(a, b)$ and $e_2=(c, d)$ are $(-, +)$ edges and their y-coordinates satisfy $y_a<y_c<y_d<y_b$, then their x-coordinates satisfy $x_b<x_c$. (See Figure 8.)*

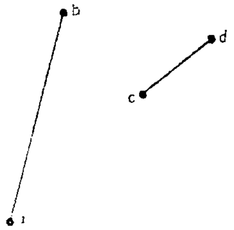

Fig. 8. Two edges satisfying Condition 1

The reason that any MFF matching satisfies this condition is that if c were to the left of b, then c would have been matched with b rather than with d. The proof that MFF gives a maximum cardinality matching satisfying this condition follows.

Lemma 6. *The algorithm MFF gives an up-right matching satisfying Condition 1 of maximum cardinality.*

We will first show that any *perfect* matching satisfying Condition 1 is a MFF matching. Then we will use this to prove Lemma 6.

Lemma 7. *If a perfect up-right matching M satisfying Condition 1 exists, it will be found by the algorithm MFF.*

Proof. Suppose we have a perfect matching M satisfying Condition 1. We show that if a $-$ point is not matched with the highest available $+$ point, then there is a pair of edges violating Condition 1. Consider the highest $-$ point which was not matched according to the rules of MFF. Call this $-$ point c. Since M is a perfect matching, this $-$ must be matched to some $+$ point by M, say point d. Let b be the $+$ point that point c would have been matched to using the MFF rule. Since b was the highest available $+$ point when we processed c, point b must be higher than point d. Since it was unmatched when we processed c, point b must be matched in M to a $-$ point below c, say a. Then (a, b) and (c, d) forn a pair of edges in M violating Condition 1. This contradicts our assumption and thus proves Lemma 7. ▮

Proof of Lemma 6. Let L be the list of items to be packed. Let M be a maximum cardinality up-right matching satisfying Condition 1. Let L' be those items of L matched by M. By Lemma 7, MFF must match every item in L'. By Lemma 5, MFF must match at least as many items of L as it does of L'. Thus, MFF produces a maximum cardinality matching. ▮

Next, we prove that if we have a up-right matching satisfying Condition 2 given below, we can obtain an equal cardinality matching satisfying Condition 1. We do this by successive switches of pairs of edges that satisfy Condition 2 but not Condition 1.

Condition 2. *If $e_1=(a, b)$ and $e_2=(c, d)$ are $(-, +)$ edges, and their y-coordinates satisfy $y_a<y_c<y_d<y_b$, then their x-coordinates satisfy $x_a<x_d$. (See Figure 9.)*

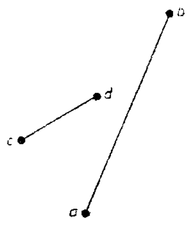

Fig. 9. Two edges satisfying Condition 2 but not Condition 1

Any two edges satisfying Condition 1 must also satisfy Condition 2.

Lemma 8. *Given a set of $+$ and $-$ points in a unit square, suppose that there is an up-right matching satisfying Condition 2. Then there is an up-right matching of equal cardinality satisfying Condition 1.*

Proof. Suppose that we have an up-right matching satisfying Condition 2 but not Condition 1. Consider all pairs of edges e_1 and e_2 which violate Condition 1. Of these, choose for e_1 an edge (a, b) such that b is the rightmost point of all such edges. Next, choose for e_2 an edge (c, d) such that (a, b) and (c, d) violate Condition 1, and c is the leftmost point in all such (c, d) edges. (We require a, b, c, d to be the points in the order stated in Condition 1.) Replace (a, b) and (c, d) with (a, d) and (c, b). We claim that this new matching satisfies Condition 2. Furthermore, by repeating this step, we will eventually obtain a matching that satisfies Condition 1.

We must first show that the new matching still satisfies Condition 2. This is done by case analysis. There are four cases, all of which are relatively easy.

Case A. Suppose (c, b) and (c', d') do not satisfy Condition 2, with (c', d') the edge e_2 as stated in the condition. Since this means d' is to the left of c, we must have c' to the left of c. But (a, b) and (c', d') now violate Condition 1. This contradicts our choice of c as the leftmost point violating Condition 1 with (a, b). (See Figure 10A.)
Case B. Suppose (a, d) and (c', d') do not satisfy Condition 2, with $e_1=(a, d)$ and $e_2=(c', d')$. Then (a, b) and (c', d') violate Condition 2, contradicting our assumption that we had a matching satisfying Condition 2. (See Figure 10B.)

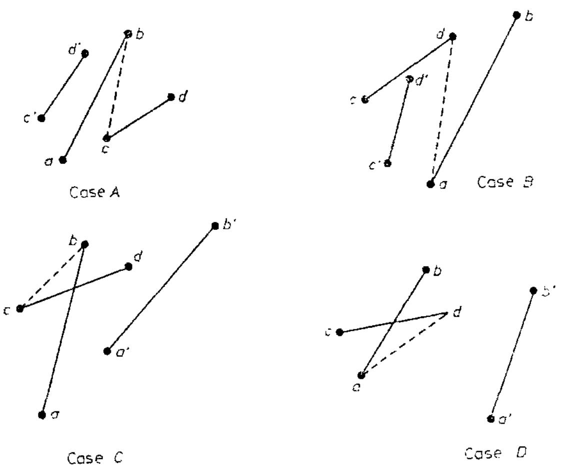

Fig. 10

Case C. Suppose (a', b') and (c, b) do not satisfy Condition 2, with $e_1=(a', b')$ and $e_2=(c, b)$. Then a' is to the right of b, implying that b' is to the right of b. Now (a', b') and (c, d) violate Condition 1, contradicting our choice of b as the rightmost point in an edge $e_1=(a, b)$ violating Condition 1. (See Figure 10C.)

Case D. Suppose $e_1=(a', b')$ and $e_2=(a, d)$ do not satisfy Condition 2. Then we have a' to the right of d. Thus, $(a'\ b')$ and (c, d) violate Condition 2, a contradiction. (See Figure 10D.)

We can continue the above process of switching pairs of edges that do not satisfy Condition 1 as long as our matching does not satisfy Condition 1. Thus, if the process terminates, the resulting matching must satisfy Condition 1. It is a simple calculation to see that the replacement always reduces the sum $\sum_e (y_{e_+} - y_{e_-})^2$ over all edges $e=(e_+, e_-)$ in the matching. Since there are a finite number of matchings, this sum cannot decrease forever. The process therefore terminates in a matching satisfying Condition 1. ▌

Proof of upper bound in Theorem 4. We will now produce a matching satisfying Condition 2 and using all but $O(n^{2/3}(\log n)^{1/2})$ points. We will need the following lemma on grid matching to prove this. This lemma is proved in Leighton and Shor [19]. A simpler but weaker lemma, with a bound of $\log n/\sqrt{n}$ instead of $(\log n)^{3/4}/\sqrt{n}$ can be shown using techniques in [1, 18].

Lemma. *Suppose there are n points randomly distributed in a unit square, and there is a $\sqrt{n}\times\sqrt{n}$ grid imposed on this square. Then with probability at least $1-1/n$ we can match each point to a grid point with edges of length $O((\log n)^{3/4}/\sqrt{n})$.* ∎

We now produce the matching satisfying Condition 2. Assume that we have $2n$ points, approximately half $+$'s and half $-$'s, in the unit square. We will first lay out a grid of $n^{1/3}\log^{1/4} n\times n^{2/3}\log^{-1/4} n$ points, with $u=n^{1/3}\log^{1/4} n$ points in each row and $v=n^{2/3}\log^{-1/4} n$ points in each column. (See Figure 11.) From the lemma above, we show that by using edges that pass at most $k\log^{3/4} n$ grid points, we can match all but $O(n^{2/3}\log^{1/4} n)$ of the $+$ points and of the $-$ points to grid points. We do this by breaking the grid into $v/u=n^{1/3}\log^{-1/2} n$ square grids of size $u\times u$ and matching points within each of these using the above lemma. (See Figure 11.) With probability $1-n^2$, in each of these square grids the number of grid points, $+$ points, and $-$ points differs by $O(u\sqrt{\log u})=O(n^{1/3}\log^{3/4} n)$. Using the lemma above, we can match $+$ points to grid points in each of the square grids so that each edge passes $O(\log^{3/4} n)$ grid points and so that only the $O(n^{1/3}\log^{3/4} n)$ extra points in each grid are unmatched. We do the same for the $-$ points. There are $n^{1/3}\log^{-1/2} n$ square grids, so the total number of unmatched points is $O(n^{2/3}\log^{1/4} n)$.

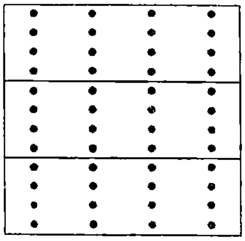
Fig. 11. Breaking the grid into square grids

We now have nearly all the $+$ points and nearly all the $-$ points matched to grid points with edges that move only $k\log^{3/4} n$ grid points horizontally and vertically, where k is a constant. Let $l=k\log^{3/4} n$. We match the $+$ points to the $-$ points by matching the $+$ point matched to one grid point to the $-$ point matched to a nearby grid point. In order to ensure an upward right matching, we must match a $-$ point to a $+$ point $2l$ grid points up and to the right. We can ensure a matching satisfying Condition 2 by making the edges on the left longer vertically than the ones on the right. If we do this properly, we match all but $O(n^{2/3}\log^{1/2} n)$ of the points.

We match the $-$ point matched to the (i,j) grid point to the $+$ point matched to the $(i+7l, j+u-i)$ grid point, where $u=n^{1/3}\log^{1/4} n$ is the number of grid points in a row. This matches all but $O(n^{2/3}\log^{1/2} n)$ of the $+$ points to $-$ points. (See Figure 12.) This matching is up-right since we matched each grid point $7l$ grid

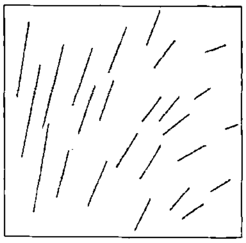
Fig. 12. A matching satisfying Condition 2

points to the right, and the edges used in matching the + points and − points to the grid points have length of at most l grid points. We have thus matched each point at least $5l$ grid points to the right. Similarly, we have matched all the − points up (except those within $7l$ grid points of the right border, which are unmatched anyway). This matching also satisfies Condition 2. If there are two edges (a, b) and (c, d) with b to the left of c, then the grid points matched to a and c are separated by at least $5l$ grid points, since the grid point matched to c is at worst $2l$ grid points left of that matched to b. There is a possible difference of $2l$ between the vertical length of an edge and the vertical length of the corresponding grid point edge. Since a is at least $5l$ grid points to the left of c, we have that the vertical length of the grid point edge corresponding to (a, b) is at least $5l$ grid points more than the grid point edge for (c, d), so that edge (a, b) is vertically at least l grid points longer than edge (c, d). This matching therefore satisfies Condition 2. By Lemma 8, there is an equal cardinality matching satisfying Condition 1. Thus MFF, and also FF, has wasted space at most $O(n^{2/3}\log^{1/2} n)$. ∎

4b. The lower bound

We prove that the algorithm First Fit gives an expected value of $\Omega(n^{2/3})$ wasted space when packing items which are uniformly distributed on [0, 1].

Proof. In this proof we will be considering two different sequences; the sequence of items which are to be packed (this does not change) and the sequence of bins which are partially full (this changes as items are packed). To avoid confusion, we will call the sequence of items the *list* of items and the sequence of bins the *queue* of bins. A *subsequence* of one of these sequences will be a subset of its elements, in the same order as they were in the original sequence. If all these elements are adjacent, we call it a *contiguous subsequence.*

Consider an item that is being packed by First Fit. Into what bins can this item go? It can only go into a bin if this bin is less full than all previous bins in the queue (otherwise it would have been packed in one of the previous bins). We will call this subsequence of bins that have more empty space than all preceding bins the *greedy decreasing subsequence* of bins. This subsequence is the decreasing subsequence formed greedily, i.e., by starting with the first bin and always adding the next bin in the queue which will make the subsequence decreasing. This subsequence will end with the first empty bin. First Fit can only add items to bins in the greedy decreasing subsequence.

For this proof, we will look at the tail of the greedy decreasing subsequence starting with the first bin less than 2/3 full. We call this tail Q. No item less than 1/3 can be in any bin later in the queue than the first bin of Q, since any items less than 1/3 would be put in the first bin of Q or an earlier bin. Thus, any 2-item bin after the first bin of Q is more than 2/3 full, and thus not in Q. This shows that all the bins in Q, except possibly for the first one, contain only one item.

We consider the items which fill bins. Since we almost certainly use at least $n/2$ bins (the optimal packing almost certainly contains more bins than this [21]), at least half of all items will be the top item of a bin in the final packing. Thus, if we can show that $1/2+\delta$ of the time an item leaves at least ε empty space in a bin, we can conclude that $\delta n/2$ bins in the final packing have ε empty space.

Assume there are k bins in the greedy decreasing subsequence Q. Any item larger than 1/3 must be placed in a bin in Q. An item which will fill a bin to higher than $1-(12k)^{-1}$ must have size within $(12k)^{-1}$ of the amount of empty space in a bin in Q, so the probability of an item filling a specific bin to within $(12k)^{-1}$ of full is at most $(12k)^{-1}$. Since there are only k bins in Q, the probability of an item filling one of these to within $(12k)^{-1}$ of full is less than $k\cdot(12k)^{-1}=1/12$. Since an item goes into a bin in Q with probability 2/3, there is at least a $2/3-1/12=7/12$ probability than an item will leave a space of $(12k)^{-1}$ or greater in a bin. Thus, if there are never more than k bins in Q, with high probability the final packing will have $(1/13)n$ bins with at least $(12k)^{-1}$ empty space.

We now use Hammersley's result [11] that with probability at least $1-e^{-\alpha n}$, a random sequence of length n has no decreasing subsequence longer than $cn^{1/2}$, for some $c>2$ and some $\alpha>0$. Except possibly for the first bin of Q, the bins in Q each have only one item in them. These items were put in the bins in the order of the bins in the queue. Thus, these items form a decreasing subsequence of the list of items. Let P be the contiguous subsequence of the list of items starting with the item packed in the first bin of Q and ending with the item packed in the last non-empty bin of Q. All the items in P having size greater than 2/3 are packed in bins by themselves. Assume there are $n^{1/3}$ bins in Q. Applying Hammersley's result to each of the $n-n^{2/3}/\sqrt{c}$ contiguous subsequences of the list of items of length $n^{2/3}/\sqrt{c}$, we get that with probability at least $1-ne^{-\alpha n^{2/3}/\sqrt{c}}$ none of these contains a decreasing subsequence of length $n^{1/3}$. Thus, with high probability, P has length at least $n^{2/3}/\sqrt{c}$.

Suppose the greedy decreasing subsequence Q is never longer than $n^{1/3}$. Then, by the preceding arguments, every item had a 7/12 probability of leaving $(1/12)n^{-1/3}$ or more empty space in the bin it entered, so we have $\Omega(n^{2/3})$ wasted space in the final packing. On the other hand, if the subsequence Q were longer than $n^{1/3}$ at some time t, we have that with high probability, at time t there were $\Omega(n^{2/3})$ bins in P. With high probability 1/7 of the items in P have size between 2/3 and 5/6. At time t these are all the only items in their bins. Thus, at time t there were $\Omega(n^{2/3})$ bins with at least 1/6 empty space. If there ever were $\Omega(n^{2/3})$ of these bins, we will show that with high probability there still are, so we still have $\Omega(n^{2/3})$ wasted space.

Suppose that at some point we have $\Omega(n^{2/3})$ bins filled to 5/6 or less. After this point, say we will be receiving $m<n$ more items. Any items larger than 1/2 will require new bins, so we need enough small items to fill all these new bins and the $\Omega(n^{2/3})$ bins that were filled to less than 5/6. With probability at least $1-1/n$, the empty space

we need to fill in these new bins (needed by large items) totals to $(1/4)m \pm O(\sqrt{n \log n})$ and the total size of the small items we can use to fill these totals to $(1/4)m \pm$ $\pm O(\sqrt{n \log n})$. Thus, we will with probability at least $1-1/n$ still have $\Omega(n^{2/3})$ wasted space. ∎

5. Remarks

In this paper we have mentioned several problems concerning the matching of points randomly distributed in the unit square. We have discussed weighted matching, rightward matching, up-right matching and grid matching. Ajtai, Komlós and Tusnády show that an optimal weighted matching has an expected weight of $\Theta(\sqrt{n \log n})$. Leighton and Shor [19] show that an optimal up-right matching leaves $\Theta(\sqrt{n}(\log n)^{3/4})$ points unmatched and that the longest distance in an optimal grid matching is $\Theta((\log n)^{3/4}/\sqrt{n})$. This still leaves a gap for rightward matching. It is bounded below by weighted matching and above by grid matching, so the weight for an optimal rightward matching is $\Omega(\sqrt{n \log n})$ and $O(\sqrt{n}(\log n)^{3/4})$.

An interesting question is how well an on-line algorithm for bin packing can do, given a uniform distribution of items. If it could be proved that weighted rightward matching required $\Omega(n^{1/2}(\log n)^{3/4})$ weight, this would mean that Best Fit is an optimal on-line algorithm up to a constant factor in wasted space. If weighted rightward matching can be done in $o(n^{1/2}(\log n)^{3/4})$, Best Fit might still be optimal, since finding an optimal rightward matching could require information not available to on-line algorithms.

Another area of research is to analyze the behavior of these bin packing algorithms for distributions other than uniform on [0, 1]. Experiments by Bentley et al. [2, 4] seem to show that First Fit and Best Fit waste linear space when packing a uniform distribution on $[0, \alpha]$, for some values of α near 0.8. These experimental results contrast with the theoretical proof that Best Fit is a good algorithm for $\alpha = 1$. Theoretical results for these distributions would be quite interesting.

Acknowledgements. I would like to thank Noga Alon, Jon Bentley, Mark Haiman, David Johnson, Jeff Kahn, Richard Karp, Tom Leighton, Michael Luby, George Lueker, Cathy McGeoch and Lyle McGeoch for their helpful comments and discussions. I am especially indebted to Jeff Kahn for suggesting a counterexample that led to the construction in Theorem 3, to Jon Bentley and Cathy McGeoch for doing computer experiments related to this research, and to Tom Leighton for his substantial assistance with this work and his insightful observations concerning the relationships between bin packing and planar matching.

References

[1] M. Ajtai, J. Komlós and G. Tusnády, On optimal matchings, *Combinatorica* 4 (1983), 259—264.
[2] J. L. Bentley, D. S. Johnson, F. T. Leighton and C. C. McGeoch, An experimental study of bin packing, *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. McGeoch, Some unexpected expected behavior results for bin packing, *Proc. 16th ACM Symp. on Theory of Computing*, (1984), 279—288.
[4] J. L. Bentley and C. C. McGeoch, *personal communications.*
[5] D. J. Brown, A lower bound for on-line one-dimensional bin packing algorithms, *Technical Report R86, Coordinated Science Lab., U. of Illinois, Urbana, IL*, 1979.
[6] E. G. Coffman, Jr., M. R. Garey and D. S. Johnson, Approximation algorithms for bin packing — an updated survey, *Algorithm Design for Computer System Design*, (G Ausiello, M. Lucertini and P. Serafini, eds.), Springer—Verlag, New York, (1984), 49—106.
[7] 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.
[8] R. M. Dudley, Empirical and Poisson processes on classes of sets or functions too large for central limit theorems, *Z. Wahrsch. verw. Gebiete* **61** (1982), 355—368,
[9] W. Fernandez de la Vega and G. S. Lueker, Bin packing can be solved within $1+\varepsilon$ in linear time, *Combinatorica* **1** (1981), 349—355.
[10] G. N. Frederickson, Probabilistic analysis for simple one- and two-dimensional bin packing algorithms, *Inf. Proc. Letters* **11** (1980), 156—161.
[11] J. M. Hammersley, A few seedlings of research, *Proc. 6th Berkeley Symp. on Mathematical Statistics and Probability*, 1 (1970), 345—394.
[12] D. S. Johnson, *Near-optimal bin packing algorithms*, Ph. D. Thesis, Massachusetts Institute of Technology, June 1973, Project MAC TR—109.
[13] 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. of Computing* 3 (1974), 299—325.
[14] N. Karmarkar, Probabilistic analysis of some bin-packing algorithms, *Proc. 23rd IEEE Symp. on Foundations of Computer Science*, (1982), 107—111,
[15] N. Karmarkar and R. M. Karp, An efficient approximation scheme for the one-dimensional bin packing problem, *Proc. 23rd IEEE Symp. on Foundations of Computer Science*, (1982), 312—320.
[16] W. Knödel, A bin packing algorithm with complexity $0(n \log n)$ and performance 1 in the stochastic limit, *Mathematical Foundations of Computer Science 1981, Lecture Notes in Computer Science 118*, (J. Gruska and M. Chytil, eds.), Springer, Berlin, (1981), 369—378.
[17] R. M. Karp, M. Luby and A. Marchetti-Spaccamela, Probabilistic analysis of multi-dimensional bin packing problems, *Proc. 16th ACM Symp. on Theory of Computing*, (1984), 289—298.
[18] F. T. Leighton and C. E. Leiserson, Wafer-scale integration of systolic arrays, *Proc. 23rd IEEE Symp. on Foundations of Computer Science*, (1982), 297—311.
[19] F. T. Leighton and P. W. Shor, Tight bounds for minimax grid matching, with applications to the average case analysis of algorithms. *Proc. 18th ACM Symp. on Theory of Computing*, (1986), 91—103.
[20] F. M. Liang, A lower bound for on-line bin packing, *Inf. Proc. Letters* **10** (1980), 76—79.
[21] G. S. Lueker, An average-case analysis of bin packing with uniformly distributed item sizes, *Report No.* **181**, Department of Information and Computer Science, U. of California, Irvine, CA, 1982.
[22] G. S. Lueker, personal communication.

Peter W. Shor

Mathematical Science Research Institute
1000 Centennial Drive
Berkeley, CA 94720