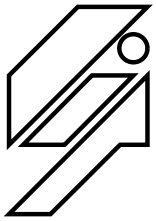
# Best-fit bin-packing with random order

Claire Kenyon



*Laboratoire de l'Informatique du Parallélisme*

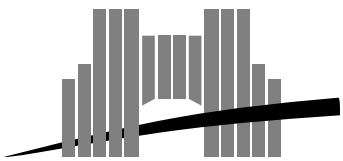
Ecole Normale Supérieure de Lyon
Unité de recherche associée au CNRS n°1398

# *Best-Fit Bin-Packing with Random Order*

Claire Kenyon

September 1995

Research Report N$^{o}$ 95-19



**Ecole Normale Supérieure de Lyon**
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : (+33) 72.72.80.00 Télécopieur : (+33) 72.72.80.80
Adresse électronique : lip@lip.ens−lyon.fr

# Best-Fit Bin-Packing with Random Order

Claire Kenyon

September 1995

**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 and in the average uniform case. In practice, Best-fit appears to perform within a few percent of optimal. In this paper, 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*. We show a lower bound of 1.07... and an upper bound of 1.5 on the random order performance ratio of Best-fit.

**Keywords:** Best Fit, Bin-Packing, On-line Algorithms, Approximation Ratio, Markov Chains

**Résumé**

L'algorithme de meilleur choix est le meilleur pour la mise en boîte en-ligne, en ce sens qu'on ne connaît pas d'algorithme qui lui est supérieur à la fois dans le pire cas et dans le cas moyen uniforme. En pratique, Meilleur choix semble être à quelques pour cent de l'optimum. Dans cet article, nous étudions la performance relative moyenne par rapport à l'optimum, considérant le pire cas de valeurs de données mais supposant que leur ordre d'arrivée est aléatoire. Nous montrons une borne inférieure de 107 % et une borne supérieure de 150% à la performance de Meilleur choix avec cette définition.

**Mots-clés:** Mise en Boîtes, Algorithmes En-Ligne, Rapport de Performance, Chaînes de Markov.

# Best-Fit Bin-Packing with Random Order

Claire Kenyon*

September 13, 1995

## 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 [10] 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 [11]; it is more complicated, but linear time, and tailored to behave well in worst-case situations.

*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 was supported by CNRS and by a NATO fellowship.

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:*

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

In the seminal paper [2], it is 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 [11] to have performance ratio $1.69\ldots$?, and improved versions have slightly lower performance ratios; to the author's knowledge, the current best performance ratio is $1.58\ldots$ [14]. 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 [4]. This lower bound was later improved up to $1.54\ldots$ in [5, 6, 7], and proved to hold even for randomized algorithms [9].

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. 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 asymptotically optimal on average, and that the expected amount of wasted space (number of bins minus the sum of the item sizes) is $O(n^{1/2}(\log n)^{3/4})$ for Best-fit and about $n^{2/3}$ for First-fit [3, 1]. 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 [15], it was shown, using a computer program to compute Lyapunov functions to analyze multi-dimensional bounded-jump 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 $\{1/12, 2/12, \ldots, 9/12\}$.

## 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 model 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 [8]).

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

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

*where $L_\sigma$ 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, 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 $1.07\ldots$.

**Theorem 1** *The random-order performance ratio of Best-fit satisfies:*

$$1.07 \leq RC(BF) \leq 1.5.$$

We expect the true answer to lie somewhere close to 1.07.

The proof of the lower bound analyzes the performance of Best-fit when the items are drawn uniformly and independently from $\{.29, .35, .36\}$, for which the optimum packing is perfect. The analysis can be reduced to studying a "one-dimensional" Markov chain (drawn on an infinite strip of width 3), 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; 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.

# 2 The upper bound

Let $L$ be a list of $n$ items, and let $L_\sigma$ denote the list ordered according to permutation $\sigma$. Let us first prove the upper bound for 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.

## 2.1 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. This implies $BF(L) \leq 2 + 3OPT(L)/2$.

## 2.2 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, only the large items can be alone in their bin (except possibly for the last 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 $BF(L) \leq 3OPT(L)/2 + 1$.

## 2.3 General case

In the general case, all sizes can occur. Let $t$ be the last time that a small item $z$ was inserted into a bin $B$ which either is new or was filled up to less than $1/2$ immediately prior to inserting $z$.

We first analyze what happens up to time $t$ (if $t$ exists). At time $t$, all bins except $B$ are filled up to level at least $2/3$. Let $L_\sigma(1,t)$ denote the list of items inserted up to time $t$, and $W_\sigma(t)$ denote their total weight. We have: $BF(L_\sigma(1,t)) \leq 3W_\sigma(t)/2 + 1 \leq 3OPT(L_\sigma(1,t))/2 + 1$.

Now, we analyze what happens after time $t$. Let $L_\sigma(t+1,n)$ denote the list of items inserted after time $t$. Let $x$ be the number of large items and $y$ the number of medium items in $L_\sigma(t+1,n)$. The optimal algorithm uses at least $(x+y)/2$ bins when run on $L_\sigma(t+1,n)$. But every bin created after time $t$ by Best Fit contains either at least two medium items or one large item (except for the last bin). So at most $x + y/2 + 1$ are created, and the

ratio $(BF(L)-BF(L_\sigma(t+1,n)))/OPT(L_\sigma(t+1,n))$ is maximized for $x=y$. Thus $BF(L)-BF(L_t)\leq 3OPT(L_\sigma(t+1,n))/2+1$.

Putting both inequalities together, we obtain

$$BF(L)\leq \frac{3}{2}(OPT(L_\sigma(1,t))+OPT(L_\sigma(t+1,n)))+2.$$

Note that $t$ depends on the permutation $\sigma$. 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 $\sigma$. This relies heavily on up-right matching analysis [1].

## 2.4 Analysis of the optimal algorithm

We cannot easily analyze the optimal algorithm. Instead, we analyze another algorithm, less efficient but for which up-right matching results apply. The number of bins used by this algorithm is an upper bound on $OPT(L_\sigma(1,u))$.

We first present the algorithm and analysis in the simple case when $OPT$ packs exactly two items in each bin. From now on, we take the wording "with high probability" to mean with probability $1-o(1)$.

**Lemma 1** *Assume that $OPT(L)$ packs eaxctly two items per bin. Then, with high probability, we have:*

$$\sup_u \left[OPT(L_\sigma(1,u)-\frac{u}{n}OPT(L)\right]=O(n^{1/2}(\log n)^{3/4}).$$

**Proof :**
We start with some notation. In the packing of $OPT(L)$, each bin $i$ contains two items, $x_1^{(i)}\geq x_2^{(i)}$. In tis proof, we call $x_1^{(i)}$ "large" and $x_2^{(i)}$ "small". We can assume that $x_1^{(1)}\geq\ldots\geq x_{n/2}^{(1)}$. Then $OPT(L_\sigma[1\ldots u])$ is at least as good as the Modified Best Fit (MBF) algorithm of [3]. Briefly, a new large item is always put in a new bin; a new small item is matched to the largest possible previously inserted large item (which is not already matched), and put in a new bin otherwise; a bin is closed as soon as it receives a small item.

This is almost the setting of the up-right matching analysis of [3]; the difference is that in the present situation, there are exactly the same number $n/2$ of small and large items, while in the setting of [3], there are $n$ items, each of which has probability $1/2$ of being small and $1/2$ of being large;

this does not affect the analysis, since adding or removing $O(\sqrt{n})$ items only changes the number of bins which $MBF$ uses by $O(\sqrt{n})$.

The up-right matching analysis tells us that with high probability, the total number of unmatched items is $O(\sqrt{n}(\log n)^{3/4})$. This implies the lemma.

In the general case, let $b = OPT(L)$. We fix $\epsilon > 0$, and let $k = \lceil 1/\epsilon \rceil + 1$. In the optimum packing $OPT(L)$, we partition the bins into groups according to how many items they contain. Let $b_i$ be the number of bins with exactly $i$ items, for $1 \leq i < k$, and let $b_k$ denote the number pf bins with at least $k$ items. We have: $b = b_1 + \cdots + b_k$. Let $S_i$ denote the set of items in the $b_i$ bins with $i$ items (with $\geq k$ items in the special case $i = k$).

The algorithm which we use to bound $OPT(L_\sigma(1,u))$ is basically $MBF$ used independently on each $S_i$ and on each item rank. We order the items by decreasing size in each bin. The algorithm constructs a matching between the largest and the second largest items of $S_i$; and another matching between the second largest and the third largest items of $S_i$; and so on. It then takes matched items and allocates them to the same bin.

More precisely, we use the following notation for items of $S_i$. Order the bins by decreasing size of their largest items, $y_1^{(1)} \geq y_2^{(1)} \geq \ldots \geq y_{b_i}^{(1)}$. In each bin $j$, $1 \leq j \leq b_i$, order the items by decreasing size: $y_j^{(1)} \geq y_j^{(2)} \geq \ldots \geq y_j^{(j)}$.

For finding an upper bound to $OPT(L_\sigma[1 \ldots u])$, our algorithm works independently for each $S_i$, as follows: we apply $MBF$ to form a matching of the largest items (the "large" items) with the second largest items (the "small" items); we mark the unmatched items as having failed. We then apply MBF to find a matching of the second largest items which have not yet failed (the new "large" items) with the third largest items (the new "small" items). We mark the unmatched items as having failed. We continue in the same manner for $i-1$ steps. Any two items which are matched at any stage will be allocated to the same bin. Finally, in the special case of set $S_k$, we allocate the remaining items $R$ to bins using a greedy algorithm such as Best Fit.

**Lemma 2** *If $i < k$, then we have:*

$$\sup_u \left[ OPT(L_\sigma(1,u) \cap S_i) - \frac{u}{n} b_i \right] = O(\sqrt{b_i}(\log b_i)^{3/4}).$$

*If $i = k$, then we have:*

$$\sup_u \left[ OPT(L_\sigma(1,u) \cap S_k) - \frac{u}{n} b_k \right] \leq \epsilon \frac{u}{n} b_k + O(\sqrt{b_k}(\log b_k)^{3/4}.$$

The proof, ommitted here, is basically up-right matching applied $i$ times for $S_i$. For the special case of the $S_k$, note that the items in $R$ are all smaller than $1/\epsilon$, and thus won't create new bins unless all bins are filled up to level $1 - 1/\epsilon$ at least, which gives performance ratio $(1 + \epsilon)$ at most.

We finally obtain:

$$\begin{aligned} BF(L) &\leq \frac{3}{2}(OPT(L_\sigma(1,t)) + OPT(L_\sigma(t+1,n)) \\ &\leq \frac{3}{2} n \sup_u \frac{OPT(L_\sigma(1,u))}{u} \\ &\leq \frac{3}{2} OPT(L)(1+\epsilon) + o(OPT(L)) \end{aligned}$$

with high probability.

In the remaining, low-probability cases, we use the worst-case bound $BF(L) \leq 1.7 OPT(L)$. Altogether, we get:

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

This implies the theorem.

# 3 The lower bound

The calculations are only sketched here; they were done using Mathematica. Instead of taking items from a fixed list in random order, we will draw $n$ items independently and uniformly from a fixed set $S$. 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 $1.07\ldots$. It follows that there exists at least one multiset for which the random-order performance ratio is greater than or equal to $1.07\ldots$. We choose $S$ so that the optimal algorithm is perfect, and packs exactly three items per bin: $S$ has three elements $a \leq b \leq c$ which sum to 1. Now, we make sure that if Best-fit starts packing a bin $B$ the wrong way, by putting two copies of $c$ together in

$B$, the error is unretrievable: no more item can fit into $B$. In addition, by choosing all elements greater than $1/4$, we make sure that Best-fit can never pack four items in the same bin, and thus can never recoved from its errors. All these conditions are satisfied by $S = \{.29, .35, .36\}$, which is thus a good candidate to find a non-trivial lower bound.

Let $a = .29, b = .35, c = .36$. A bin is called *closed* if it can no longer receive any more items (i.e. its current size is greater than .71). 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 $a$, $b$ or $c$ with probability $1/3$ each.

An open bin has size either $.29, .35, .36, .58, .64, .65$ – with at most one bin of each of these sizes, and additional constraints on which sizes can occur simultaneously–, or size .70 or .71, or is closed. Note that bins of size .70 and bins of size .71 behave exactly in the same way: they can only receive one additional $a$. So they do not need to be distinguished, and the state of the system is determined by the number $i$ of bins of size .70 or .71, plus a constant amount of information on which other bin sizes are present: the Markov chain is infinite "one-dimensional". In addition, if $i$ is greater than 0, one can check that either there are no other bins, or there is just one other bin (which has size .35 or .36).

The states of the Markov chain are the following. Let $a_i$ be the state with $i$ bins of size .70 or .71. Let $b_i$ be the state which has $i$ bins of size .70 or .71, plus one bin of size .35. Let $c_i$ be the state which has $i$ bins of size .70 or .71, plus one bin of size .36. Let $A$ be the state with one bin of size .29, $AA$ be the state with one bin of size .58, $AB$ be the state with one bin of size .64, $AC$ be the state with one bin of size .65, and $X$ be the state with one bin of size .65 and one bin of size .36.

The transitions are drawn in the figure 1. The chain is aperiodic and irreducible. The stationary probabilities exist iff the following system of equations has a positive normalized solution (where the name of a state is identified with its stationary probability for notational ease):

$$\begin{cases} a_{i+1} &= \frac{1}{3}c_i + \frac{2}{3}b_i + \frac{1}{3}a_{i+2} + \frac{1}{3}c_{i+1} \\ b_{i+1} &= \frac{1}{3}a_{i+1} + \frac{1}{3}b_{i+2} \\ c_{i+1} &= \frac{1}{3}a_{i+1} + \frac{1}{3}c_{i+2} \end{cases}$$

plus some additional equations for the initial part of the chain. From the second and third equations, we infer that $b_i = c_i$ for all $i \geq 1$. We obtain the

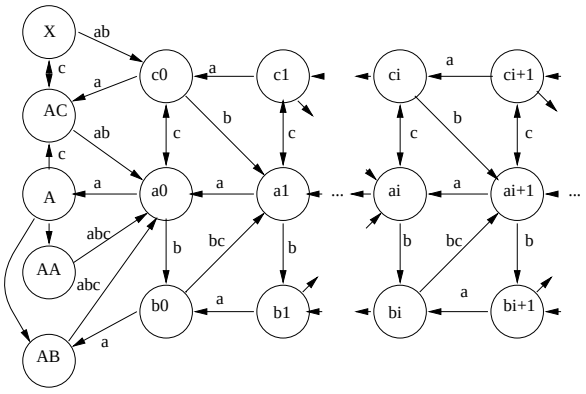



Figure 1: Markov chain describing Best-fit under random insertions from $\{.29, .35, .36\}$.

following linear recurrence relation:

$$\begin{pmatrix} a_{j+1} \\ b_{j+1} \\ b_j \end{pmatrix} = \begin{pmatrix} 3 & -1 & -3 \\ -1 & 3 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_j \\ b_j \\ b_{j-1} \end{pmatrix}$$

for all $j \geq 2$. The characteristic polynomial has three roots, $(5-\sqrt{13})/2 < 1 < (5+\sqrt{13})/2$. Since $a_j, b_j$ go to 0 as $n$ goes to infinity (the probabilities sum to 1), the vector $(a_{n+1}, b_{n+1}, b_n)^T$ is in the eigenspace of $(5-\sqrt{13})/2$, thus a multiple of the eigenvector, $(-2+\sqrt{13}, (5-\sqrt{13})/2, 1)^T$. We use the initial part of the chain, and the fact that the probabilities sum to 1, to determine the mutiplicative factor and thus the stationary distribution. The probability that an arrival causes a new bin to open is then $p_{BF} = a_0 + AC/3 + 2\sum_{i\geq 1} a_i/3$.

On the other hand, the optimal algorithm packs three items per bin, so the probability that a new arrival causes OPT to use one more bin is $p_{OPT} = 1/3$. The average performance ratio of Best-fit is thus $RC(BF) = p_{BF}/p_{OPT} = 3p_{BF}$. Calculations show $RC(BF) = 1.07322\ldots$.

## Acknowledgments

The author wishes to thank Richard Karp for numerous discussions, Mor Harchol for help with finding the relevant references, and Richard Kenyon for helping to solve the Markov chain used in the lower bound.

# References


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

[2] 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).

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

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

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

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

[8] K. Clarkson, Disc. and Comput. Geometry.

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

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

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

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

[13] 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.

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

[15] 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.