To appear in *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms,* Washington, DC © SIAM 2001

# Better Approximation Algorithms for Bin Covering

Janos Csirik [*] David S. Johnson [†] Claire Kenyon [‡]

**Abstract**

Bin covering takes as input a list of items with sizes in $(0, 1)$ and places them into bins of unit demand so as to maximize the number of bins whose demand is satisfied. This is in a sense a dual problem to the classical one-dimensional bin packing problem, but has for many years lagged behind the latter in terms of the quality of the best approximation algorithms. We design algorithms for this problem that close the gap, both in terms of worst- and average-case results. We present (1) the first asymptotic approximation scheme for the offline version, (2) algorithms that have bounded worst-case behavior for instances with discrete item sizes and expected behavior that is asymptotically optimal for all discrete "perfect-packing distributions" (ones for which optimal packings have sublinear expected waste), and (3) a learning algorithm that has asymptotically optimal expected behavior for *all* discrete distributions. The algorithms of (2) and (3) are based on the recently-developed online *Sum-of-Squares* algorithm for bin packing. We also present experimental analysis comparing the algorithms of (2) and suggesting that one of them, the *Sum-of-Squares-with-Threshold* algorithm, performs quite well even for discrete distributions that do not have the perfect-packing property.

## 1 Introduction

In the *bin covering* (or *dual bin packing*) problem, we are given a list $L = (a_1, a_2, \ldots, a_n)$ of items with sizes $s(a_i) \in (0, 1)$. One must pack the items into bins in such a way as to maximize the number of bins that receive items of total size 1 or more. A picturesque application would be selling packets of canceled stamps to novice philatelists, where each packet is guaranteed to contain stamps whose total face value is at least \$10. (Note the contrast to bin packing, where one wants to *minimize* the number of bins, subject to the constraint that each receives items of total size 1 or *less*.)

Bin covering is NP-hard, and so research has concentrated on polynomial-time approximation algorithms. Given a list $L$ and an algorithm $A$, let $A(L)$ be the number of bins filled to level at least 1 by $A$, $OPT(L)$ be the optimal number of bins that can be so filled, and $s(L)$ be the sum of the item sizes in $L$. Note that we have $s(L) \geq OPT(L) \geq A(L)$. In this paper we are interested both in worst-case and average-case behavior of approximation algorithms, but let us begin with our major new worst-case result. The following shorthand notations will be useful.

Worst-Case Ratio:

$$R^A = \min\left\{\frac{A(L)}{OPT(L)} : \text{all lists } L\right\}$$

Asymptotic Worst-Case Ratio:

$$R^A_\infty = \liminf_{n\to\infty}\left(\min\left\{\frac{A(L)}{OPT(L)} : OPT(L) = n\right\}\right)$$

It is easy to see that no polynomial-time approximation algorithm $A$ can have $R^A > 1/2$ (because of the difficulty of distinguishing between instances that can fill two bins as opposed to one). For typical applications where the number of filled bins is likely to be large, $R^A_\infty$ might thus be a better measure of performance, especially since low-order polynomial-time algorithms exist with $R^A_\infty = 2/3$ and $3/4$ [2, 3, 9].

Unfortunately, until now $3/4$ was the best asymptotic guarantee known. This is in contrast to the situation for bin packing, where there is a polynomial-time asymptotic approximation scheme (PTAAS), *i.e.*, a parameterized algorithm $A_\epsilon$ such that for any fixed $\epsilon > 0$ $A_\epsilon$ runs in polynomial time and satisfies $R^{A_\epsilon}_\infty < 1 + \epsilon$. The existence of such schemes for bin packing has been known for two decades [13], but it was feared by many experts that no analogous schemes (with "$< 1 + \epsilon$" replaced by "$> 1 - \epsilon$") might exist for bin covering. This was because the techniques used by [13] (and the subsequent improvements of [15]) did not seem adaptable to bin covering. They relied on the fact that tiny items (ones of size smaller than, say, $\epsilon/2$) could essentially be ignored in bin packing – adding them in at the end could only cause a new bin to be started if all other bins were already filled to levels exceeding $1 - \epsilon/2$, in which case the packing was already good enough. This allowed the bin packing approximation schemes to concentrate on a simplified situation with a controllable number of reasonably large items to worry about. In contrast, for bin covering small items may be crucial to the task of filling a bin, and so they cannot be ignored.

Theorem 1.1. *There is a PTAAS for bin covering.*

[*] `csirik@inf.u-szeged.hu`. Department of Computer Sciences, University of Szeged, Szeged, Hungary.

[†] `dsj@research.att.com`. AT&T Labs, Room C239, 180 Park Avenue, Florham Park, NJ 07932, USA.

[‡] `Claire.Kenyon@lri.fr`. Laboratoire de Recherche en Informatique, Bâtiment 490, Université Paris-Sud, 91405 Orsay Cedex, France.

The approximation scheme and a proof sketch are given in Section 2. The key idea is to recognize that although the small items cannot be ignored, they can be treated as a fluid in much of the computation, which is almost as good.

The approximation scheme of Theorem 1.1 is primarily of theoretical interest, since the running time is exponential in $1/\epsilon$. A second drawback for many applications is that the algorithm is an offline algorithm. The rest of our paper concerns what we can do in the online case. In terms of asymptotic worst-case performance alone, this is already known: In [12] it is shown that every online algorithm $A$ must have $R_\infty^A \le 1/2$, and it is easy to verify that the Next Fit ($NF$) heuristic (place each item in the current bin unless that bin is full, in which case we start a new "current" bin) has $R_\infty^{NF} = 1/2$.

We are interested in "robust" heuristics that not only have worst-case behavior close or equivalent to that of $NF$, but also have average-case behavior that is provably much better for many distributions. In the online algorithm world, it is all too common that the more one attempts to improve worst-case behavior, the more likely it is that average-case performance will approach that of the worst-case. Conversely, algorithms designed for good average-case performance often are unboundedly bad in the worst-case.

Average case behavior is a function of the algorithm $A$, the item-size distribution $F$, and the number of items $n$. Let $A(L_n(F))$ be the random variable denoting the number of bins filled by $A$ when applied to a list of $n$ items, where the size of each item is chosen independently according to $F$. The key average-case metrics for bin covering are the following.

Asymptotic Expected Ratio:

$$ER_\infty^A(F) = \liminf_{n\to\infty} E\left[\frac{A(L_n(F))}{OPT(L_n(F))}\right]$$

Expected Waste Rate:

$$EW_n^A(F) = E\left[s(L_n(F)) - A(L_n(F))\right]$$

Previous work on average-case analysis for bin covering either dealt only with the convergence properties of $A(L_n(F))/n$ (for arbitrary distributions) [14, 17, 18], or the value of $ER_\infty^A(F)$ for the particular distribution $U[0,1]$ in which item sizes are uniformly distributed in the real interval $[0,1]$, as in [8]. The distributions we consider here are the *discrete distributions* studied in [1, 5, 6, 10, 11, 16], in which item sizes are all integral multiples of $1/B$ for some integer $B$ and the probabilities are all rational numbers. When packing with such distributions, one may equivalently assume that the bin size is $B$ and the item sizes are all integers. Most real-world applications can be scaled so that they fit this model for some $B$, and for our study of online possibilities, we shall assume the model applies and allow ourselves to consider algorithms that exploit the discrete nature of the instances. This means that "polynomial time" can include polynomials in $n$ and $B$. For fixed distributions, this is equivalent to ordinary polynomial time.

Of special interest among discrete distributions are those distributions $F$ that satisfy the *perfect-packing* property: $EW_n^{OPT}(F) = o(n)$, in which case, by a result of Courcoubetis and Weber [7], $EW_n^{OPT}(F) = O(\sqrt{n})$. Note that if $F$ has the perfect-packing property, then the asymptotic expected ratio of $OPT(L_n(F))$ to the upper bound $s(L_n(F))$ is 1. There are many interesting distributions that have this property, which is easily seen to hold for bin covering if and only if it also holds for bin packing. For instance, if $U\{h,j,k\}$ is the distribution in which item sizes are $h$ through $j$, equally likely, and the bin size is $k$, then $U\{1,j,k\}$ has the perfect-packing property for all $j,k$, $1 \le j < k$ [5, 6], as do many such distributions with $h > 1$ [10, 11].

In [10] it was proved that for bin packing, the online Sum-of-Squares algorithm ($SS$) had $EW_n^{SS}(F) = O(\sqrt{n})$ for all perfect-packing distributions. The challenge is to find online algorithms for bin covering that provably have the same property and in addition have bounded worst-case behavior.

It is not immediately obvious how to do this. In $SS$, one keeps track of the numbers $n_i$, $1 \le i < B$, of partially-filled bins in the current packing $P$ whose contents total $i$. When a new item is packed, it is assigned to a bin so as to minimize $ss(P) = \sum_{i=1}^{B-1} n_i^2$ subject to the constraint that no bin is overfilled. One can use this algorithm directly as a bin covering heuristic and it will handle perfect-packing distributions as desired, but will fall down miserably in the worst case, since it never considers overfilling a bin, and for some lists the only way to reach a total $\ge B$ is to have the total *exceed* $B$. Thus $SS$ by itself has $R_\infty^{SS} = 0$. To do better, one needs an adaptation that allows overfilling bins, but doesn't allow this to be done too aggressively.

THEOREM 1.2. *For any $\epsilon > 0$ there is an $O(nB)$-time algorithm $SSNF_\epsilon$ such that*

1. $R_\infty^{SSNF_\epsilon} \ge 1/2 - \epsilon$
2. $EW_n^{SSNF_\epsilon}(F) = O(\sqrt{n})$ *for all perfect-packing distributions*

The $SSNF$ algorithms are straightforward hybrids of $NF$ and $SS$. To prove the results about their average-case performance, we rely on a key lemma from

[10], plus a new observation that the random process determining $ss(P)$ can be viewed as a submartingale when the distribution has the perfect-packing property.

These algorithms, although they work well for perfect-packing distributions, are less successful for other discrete distributions, where they can sometimes be outperformed by $NF$ acting alone. We thus also developed what appears to be a much more robust algorithm, "Sum-of-Squares with Threshold" ($SST$). In experiments we report for a variety of discrete distributions, the average-case performance of $SST$ was at least as good as and often significantly better than that of $NF$ and $SSNF$, both for large and small values of $n$. Moreover $SST$ is almost as good as the $SSNF$ algorithms from a theoretical point of view:

Theorem 1.3. *$R_{\infty}^{SST} \geq 1/3$ and $EW_n^{SST}(F) = O(n^{2/3})$ for all perfect-packing distributions.*

(It may be possible to improve these bounds. We know of no lists with $SST(L) < (1/2)OPT(L)$, and empirically the expected waste rates for $SST$ (and $SSNF$) appear to be within a constant factor of the optimal expected waste rates for all perfect-packing distributions we tested.)

If one is willing to forgo a bounded worst-case ratio, one can do even better in an average-case sense. In our final section we sketch a proof of the following:

Theorem 1.4. *There is a randomized $O(nB)$-time algorithm $SS^*$ such that $ER_{\infty}^{SS^*} = 1$ for all discrete distributions.*

This is an analogue of the learning algorithm of the same name for bin packing presented in [10]. As with the earlier algorithm, it repeatedly re-estimates the current distribution and solves an LP in order to construct a virtual perfect-packing distribution that it can simulate. A different construction is needed here, however. Instead of introducing "imaginary items" of size 1, we introduce imaginary items of a variety of sizes and in addition randomly truncate real items.

## 2 An Asymptotic Approximation Scheme

The basic idea for our approximation scheme is as follows. Partition the items into large items called "bricks," medium items that will not be used until the very end, and small items, called "pebbles." The definition of *large* is such that bins will on average contain many (but not too many) large items. Relax the problem by allowing pebbles to be melted into a fluid that can be arbitrarily divided between different bins. This new problem only depends on the brick set and on the total fluid volume. Using a rounding approach similar to that of de la Vega and Lueker [13], find a quasi-optimal way to fill the bins using bricks and fluid. From the solution to the relaxed problem, construct a feasible solution to the original problem by removing the fluid and placing pebbles and medium items in a greedy manner. A more formal description of the algorithm follows.

### 2.1 The algorithm

In order that we may use "$L$" to denote the set of large items, let us use $X$ to denote the set of all items in the instance. We may assume without loss of generality that $s(X) \geq 2$, $\epsilon < 1/2$ and $1/\epsilon$ is an integer. Note that this implies that $1 + 1/\epsilon \leq (4/3)/\epsilon$.

1. **Definition of large and small items.** If $n < \lfloor s(X) \rfloor (1 + 1/\epsilon)$, then let $L = X$ and $M = S = \phi$. If $n \geq \lfloor s(X) \rfloor (1+1/\epsilon)$, let $L$ be the largest $\lfloor s(X) \rfloor / \epsilon$ items of $X$, $M$ denote the next largest $\lfloor s(X) \rfloor$ items of $X$, and $S$ denote the remaining items.

2. **Handling the large items.** If $s(X) > 13/\epsilon^3$,

   (a) **Rounding the large items.** Partition $L$ into $1/\epsilon^2$ groups according to rank so that all groups have the same cardinality as far as possible : Let $q < 1/\epsilon^2$ and $p \geq 0$ such that $|L| = p/\epsilon^2 + q$. Then groups $G_1, G_2, \ldots, G_q$ each have cardinality $p+1$, and groups $G_{q+1}, \ldots, G_{1/\epsilon^2}$ each have cardinality $p$. The $i$th group $G_i$ contains items whose rank (by decreasing size) in $L$ is in $[1 + (i-1)(p+1), 1 + i(p+1))$ if $i \leq q$, and in $[1 + (i-1)p + q, 1 + ip + q)$ if $i > q$.

   For each $i$, round the values of the elements of $G_i$ down to the value of the smallest element of $G_i$. This creates a multiset $L_{\text{inf}}$. Let $H$ denote the set of element sizes in $L_{\text{inf}}$. (Note that $H$ has cardinality at most $1/\epsilon^2$.) For each item size $v$ in $H$, let $n(v)$ denote the number of items of size $v$ in $L_{\text{inf}}$.

   (b) **Solving the relaxed and rounded problem.** To deal with the relaxed bin covering problem with brick set $L_{\text{inf}}$ and fluid volume $s(S)$, solve the following linear program. A bin configuration is a multiset of items of $L_{\text{inf}}$ whose total size is less than 2. Let $\mathcal{C}$ denote the set of bin configurations. For each configuration $C \in \mathcal{C}$, let $s(C)$ denote the total size of items in $C$. In addition, for each configuration $C \in \mathcal{C}$ and item size $v \in H$, let $n(v, C)$ denote the number of occurrences of $v$ in $C$. The linear program has one variable $x_C$ for each bin configuration $C \in \mathcal{C}$. The constraints are as follows:

i. For every item size $v \in H$, we have $\sum_{C \in \mathcal{C}} n(v, C) x_C \leq n(v)$.
ii. $\sum_{C \text{ s.t. } s(C)<1} (1 - s(C)) \leq s(S)$.
iii. $\forall C \in \mathcal{C} \;\; x_C \geq 0$.

The goal of the linear program is to maximize $\sum_{C \in \mathcal{C}} x_C$. Let $x^* = (x^*_C)_{C \in \mathcal{C}}$ denote the solution obtained, and let $y_C = \lfloor x^*_C \rfloor$.

(c) **Converting to a packing of the large items.** Pack the items of $L_{\text{inf}}$ into bins according to the $y_C$'s (ignoring any excess items), and replace each item of $L_{\text{inf}}$ by the corresponding item of $L$.

Otherwise, $s(X) \leq 13/\epsilon^3$ and $L$ consists of at most $(4/3)\lfloor s(X) \rfloor/\epsilon < 18/\epsilon^4$ items. We can thus in constant time (albeit exponential in $1/\epsilon$) determine (via exhaustive search) the packing of $L$ into a maximum number of bins whose total shortfall is at most $s(S)$.

3. **Constructing a solution to the original problem.** If $S$ is empty, all the bins in the packing based on the $y_C$'s are full and this is our final packing (we ignore any unpacked items). Otherwise, put the items of $S \cup M$ in underfilled bins in a greedy manner, closing a bin as soon as its size is $\geq 1$.

We claim that the greedy process in Step 3 must fill all the underfilled bins. There are at most $\lfloor s(X) \rfloor$ bins in the packing based on the $y_C$'s, and the excess that any underfilled one receives is less than the largest element of $S \cup M$ that it receives. Thus the total excess volume is less than that of the largest $\lfloor s(X) \rfloor = |M|$ items in $S \cup M$, which is less than $s(S \cup M) - s(S)$, so there are enough items in $S \cup M$ to finish filling all the bins.

## 2.2 The analysis

THEOREM 2.1. *The algorithm described above is a polynomial-time asymptotic approximation scheme for bin covering.*

**Running time.** Everything except Step 2b takes $O(n)$ time for fixed $\epsilon$. To analyze the complexity of the linear program in Step 2b, recall that there are only $1/\epsilon^2 = O(1)$ element sizes, and at most $n$ elements of each size, so that there can be at most $n^{1/\epsilon^2}$ variables and $1 + 1/\epsilon^2$ constraints apart from the at most $n^{1/\epsilon^2}$ non-negativity constraints. Thus the LP is of polynomial size for fixed $\epsilon$ and can be solved in polynomial time by the ellipsoid method.

**Correctness.** It suffices to show that for some constants $c$ and $c'$, the number of bins filled to 1 or more is at least $OPT(X)(1 - c\epsilon) - c'$, in which case the algorithm $A_\epsilon$ would be based on the above scheme with $\epsilon$ replaced by $\epsilon/c$. We begin with three easy observations, the first two of which are sufficiently easy to be stated without proof.

OBSERVATION 2.2. $s(X) < 2OPT(X) + 1$.

OBSERVATION 2.3. *For a set $I$ of items and fluid volume $W$, let $OPT(I, W)$ denote the optimal bin covering using items from $I$ and completing bins with at most $W$ amount of fluid. Then $OPT(L, s(S)) \geq OPT(L \cup S)$.*

OBSERVATION 2.4.

$$OPT(L \cup S) \geq OPT(X) - \lfloor s(X) \rfloor \epsilon - (1 - \epsilon).$$

**Proof.** Take an optimal packing of $X = L \cup M \cup S$. We may assume $|L| \geq \lfloor s(X) \rfloor/\epsilon$ since otherwise $L = X$ and the Observation holds trivially. Thus since $OPT \leq \lfloor s(X) \rfloor$, the average number of large items per full bin is at least $1/\epsilon$. Remove the $\lceil \lfloor s(X) \rfloor \epsilon \rceil$ full bins that contain the most large items. There are at least this many bins by Observation 2.2 and our assumption that $\epsilon < 1/2$. Note also that by our assumption that $\epsilon$ is the reciprocal of an integer, we have $\lceil \lfloor s(X) \rfloor \epsilon \rceil \leq \lfloor s(X) \rfloor \epsilon + (1 - \epsilon)$. These bins collectively contain at least $\lfloor s(X) \rfloor$ large items, which can now be used in place of the medium items in the rest of the packing. We have thus constructed a packing of $L \cup S$ which fills at least $OPT(X) - \lfloor s(X) \rfloor \epsilon - (1 - \epsilon)$ bins. ■

Observations 2.2, 2.4, and 2.3 suffice to handle the case when $s(X) \leq 13/\epsilon^3$ and we fill at least $OPT(L, s(S))$ bins. In this case we have

$$\begin{aligned} A(X) &\geq OPT(X) - \lfloor s(X) \rfloor \epsilon - (1 - \epsilon) \\ &\geq OPT(X)(1 - 2\epsilon) - \epsilon - (1 - \epsilon) \\ &\geq OPT(X)(1 - 2\epsilon) - 1 \end{aligned}$$

When $s(X) > 13/\epsilon^3$, we pay an extra price for using the rounding procedure of Steps 2a through 2c. We can analyze the rounding procedure as in [13]. Let $L_{\text{sup}}$ denote the set obtained by, for each $i$, rounding the values of the elements of $G_i$ *upwards,* to the value of the smallest element of $G_{i-1}$ (or to 1 if $i = 1$). We have $L_{\text{inf}} \prec L \prec L_{\text{sup}}$ in the sense that items can be put in one-to-one correspondence so that each item of $L_{\text{inf}}$ is no larger than the corresponding item of $L$ which is itself no larger than the corresponding item of $L_{\text{sup}}$. Thus

$$OPT(L_{\text{inf}}, s(S)) \leq OPT(L, s(S)) \leq OPT(L_{\text{sup}}, s(S)).$$

Notice now that all groups have cardinality $p$ or $p+1$, so that $L_{\text{sup}}$ can be obtained from $L_{\text{inf}}$ by removing the $p+1$ smallest elements and adding $p$ elements of size 1 plus possibly one more element (to account for when the group size switches from $p+1$ to $p$). Thus $OPT(L_{\text{sup}}, s(S)) \leq OPT(L_{\text{inf}}, s(S)) + p + 1$ and, recalling that $p \leq |L|\epsilon^2$, the solution of the linear program satisfies $\sum_C x_C^* = OPT(L_{\text{inf}}, s(S)) \geq OPT(L, s(S)) - |L|\epsilon^2 - 1$.

Since there are only $1 + 1/\epsilon^2$ constraints other than the non-negativity constraints, a basic optimal solution $x^*$ (which itself can be found in polynomial time by standard techniques) has at most $1 + 1/\epsilon^2$ fractional coordinates and so $\sum_C y_C > \sum_C x_C^* - (1 + 1/\epsilon^2)$.

The additional shortfall due to the rounding procedure is thus at most $(|L|\epsilon^2 + 1) + (1 + 1/\epsilon^2)$. Working through the math (and using the fact that $s(X) > 13/\epsilon^3$) we conclude that in this case, $A(x) \geq OPT(X)(1 - 5\epsilon) - 4$ which means that in all cases, $A(x) \geq OPT(X)(1 - 5\epsilon) - 4$, which suffices to prove that we have a PTAAS. ■

Note that although this asymptotic approximation scheme is a close analog in performance to the one of [13] for bin packing, it does not come close to the polynomial time bin packing algorithm of Karmarkar and Karp [15], which guaranteed a packing with $A(L) \leq OPT(L) + O(\log^2(OPT(L)))$. That algorithm improved on the one of [13] by taking $\epsilon$ to be a function of $n$ but not stating the resulting (superpolynomial-size) LP explicitly (or solving it exactly). Instead the algorithm used a column-generation approach to find new variables when needed (based on approximation schemes for the knapsack problem). A similar approach could probably be used in the context of our algorithm to solve the LP that arises when $s(X) > 13/\epsilon^3$. However, it won't work for the case of our algorithm where $s(X) \leq 13/\epsilon^3$ and we must resort to an exhaustive search over a space exponential in $1/\epsilon$. Thus for now it remains an open question as to whether a Karmarkar-Karp type algorithm is possible for bin covering.

## 3 Robust Online Algorithms

In this section, we consider two approaches to adapting the Sum-of-Squares algorithm for bin packing to produce a bin covering algorithm that both has sublinear expected waste for all perfect-packing distributions *and* has a relatively small constant bound on $R_\infty^A$.

**Algorithm $SSNF_r$.** We first consider a class of direct hybrids of $SS$ and $NF$. Algorithm $SSNF_r$, $r > 0$ maintains two separate packings, one governed by $SS$ and one by $NF$. If the current total number of full bins is less than $r$ times the number of partially filled bins in the $SS$ packing, the next item is put into the $NF$ packing, otherwise it is put into the $SS$ packing.

THEOREM 3.1. *For all $r > 0$, $R_\infty^{SSNF_r} \geq \frac{1}{2} - \frac{1}{4r+2}$.*

**Proof.** Let $f$ be the number of full bins in the final packing of a list $L$, and let $p$ be the number of partially full bins in the final $SS$ packing. By the definition of the algorithm, $SSNF_r(L) = f \geq r(p-1)$. Since every full bin contains items of total size less than 2, every partially filled bin contains items of total size less than 1, and the $NF$ packing has at most one partially filled bin, we have $OPT(L) \leq s(L) < 2f + p + 1 \leq 2f + f/r + 2 = SSNF_r(L)(2 + 1/r) + 2$. The theorem follows by standard mathematical manipulations. ■

THEOREM 3.2. *If $F$ is a discrete perfect-packing distribution and $r > 0$, $EW_n^{SSNF_r} = O(\sqrt{n})$.*

**Proof.** Assume the instance is scaled so that the bin size is $B$ and all item sizes are integers, as can be done for any discrete distribution. In the final packing, let $f_{NF}$ and $f_{SS}$ denote the numbers of full bins in the $NF$ and $SS$ packings, respectively, and similarly let $p_{NF}$ and $p_{SS}$ be the numbers of partially filled bins. The total waste consists of the total excess over $B$ in the full bins plus the contents of the partially filled bins. Since no bins are overfilled under $SS$, no bin can receive total contents exceeding $2B$ under $NF$, and there is at most one partially-filled in the $NF$ packing, we thus can bound the waste by $B(f_{NF} + p_{SS} + 1)$.

Now, as shown in [10], $E[p_{SS}] = O(\sqrt{nB})$, which is $O(\sqrt{n})$ for fixed $B$. And by the operation of $SSNF$, whenever an item is added to the $NF$ packing, the number of full bins in the $NF$ packing is at most $r$ times the number of partially-filled bins under $SS$. The technical difficulty here is that this only says that $f_{NF}$ is bounded by a constant times the *maximum* number of partially filled bins during the course of the $SS$ packing, not the final number. Here is where martingale arguments come in. The key lemma from [10] that yielded a bound on the expected number of partially filled bins under $SS$ when $F$ was a perfect-packing distribution can be stated as follows:

LEMMA 3.2.1. *Let $F$ be a discrete distribution satisfying $EW_n^{OPT}(F) = O(\sqrt{n})$. Then given any packing $P$ and an item $x$ randomly generated according to $F$, $SS$ will pack $x$ in such a way that the expected increase in $ss(P)$ is at most 4.*

We wish to use this lemma to bound the *maximum* number of partially filled bins during the packing process. Let $P_i$ denote the $SS$ packing after $i$ items have been packed, $n_{i,j}$ denote the number of bins with level $j$ in $P_i$, and $ss(P_i) = \sum_{i=1}^{B-1} n_{i,j}$. The key observation is that Lemma 3.2.1 implies that the sequence of random variables $X_i = 4i - ss(P_i)$ is a submartingale. By Proposition 5.13 in [4], we thus have for all $x > 0$

$$p\left[\min_{1\le i\le n} X_i < -x\right] \le \frac{E|X_n| - E(X_1)}{x} \tag{3.1}$$

Note that $E[ss(P_i)] \ge 0$ for all $i \ge 0$, so that the right hand side of (3.1) is at most $4n/x$. We thus have

$$p\left[\max_{1\le i\le n} ss(P) > 4n + x\right] \le \frac{4n}{x} \tag{3.2}$$

Let $W_i$ denote the number of partially full bins in $P_i$. Now by the Cauchy-Schwartz inequality, we have that for any packing $P_i$,

$$W_i = \sum_{j=1}^{B-1} n_{i,j} \le \sqrt{B-1}\sqrt{\sum_{j=1}^{B-1} n_{i,j}^2} < \sqrt{B\cdot ss(P_i)} \tag{3.3}$$

Thus by (3.2) we have

$$p\left[\max_{1\le i\le n} W_i > \sqrt{B(4n+x)}\right] \le \frac{4n}{x} \tag{3.4}$$

This allows us to conclude that the expected maximum number of partially filled bins

$$\begin{aligned} &E\left[\max_{1\le i\le n} W_i\right] \\ \le\ & \sqrt{4Bn}\left(\sum_{j=0}^{\infty} p\left[\max_{1\le i\le n} W_i > j\sqrt{4Bn}\right]\right) \\ \le\ & \sqrt{4Bn}\left(\sum_{j=0}^{\infty} p\left[\max_{1\le i\le n} W_i > \sqrt{B(4n+(j^2-1)4n)}\right]\right) \\ \le\ & \sqrt{4Bn}\left(2 + \sum_{j=2}^{\infty} \frac{4n}{4n(j^2-1)}\right) = \frac{11}{2}\sqrt{Bn} \end{aligned}$$

From this we can conclude that the expected waste under $SSNF_r$ is less than $(r+1)(11/2)\sqrt{Bn}+1$ which for fixed $B$ is $O(\sqrt{n})$. ■

**Algorithm *SST*.** The performance of $SSNF_r$ can be dominated by that for $NF$ when the distribution is *not* a perfect-packing distribution, and $NF$'s average-case performance is typically unimpressive. Thus it is worth looking for more robust alternatives. To this end, we have devised the *Sum-of-Squares-with-Threshold* algorithm ($SST$). In this algorithm, we keep track not only of the counts $n_i$, $1 \le i \le B-1$, but also of $f$, the number of bins currently filled to level $\ge B$, and $s$, the sum of the sizes of the items seen so far. At any time, the current *threshold* is

$$T = \begin{cases} 2B, & \text{if } f = 0 \\ \max(B, (s/f) - 1), & \text{otherwise} \end{cases}$$

To pack an item $x$, the algorithm considers all the ways to insert $x$ into a partially-filled bin or into a new bin and, among all the possibilities that yield a bin level $\le T$, chooses the one that minimizes the value of $ss(P)$ for the resulting packing $P$. (Note that $ss(P)$ continues to be a sum over only the counts for partially filled bins.) Ties are broken first in favor of filling a bin, and then in favor of the bin whose new level will be closest to $B$.

THEOREM 3.3. $R_\infty^{SST} \ge 1/3$.

**Proof.** To prove this, it suffices to show that $s(L)/B \le 3SST(L)$. Consider the last time a new bin was started and, with a slight overloading of notation, let $x$ denote both the item that started the new bin and its size. We partition the partially filled bins of the current packing $P$ into $x$ classes $X_i$, $1 \le i \le x$, where $X_i$ consists of all those bins whose levels are congruent to $i$ (mod $x$).

CLAIM 3.3.1. *For each nonempty class $X_i$, the average level is more than $(T-x)/2$.*

**Proof of Claim.** We use the fact that since $x$ started a new bin, its placement caused an increase in the sum-of-squares potential function $ss(P)$ and hence, by the definition of $SST$, all legal ways of packing $x$ into a partially filled bin of $P$ must also increase $ss(P)$. Let

$$\begin{aligned} b &= \min\{h : n_{i+hx} > 0\} \\ t &= \max\{h : i + hx < B\} \end{aligned}$$

Observe that $n_{i+cx} \le n_{i+(c+1)x}$ for all $c$, $b \le c < t$. Otherwise, there would have to be a $c$ in this range such that $n_{i+cx} > 0$ but $n_{i+(c+1)x} < n_{i+cx}$. In this case, however, placing $x$ into a bin with level $i+cx$ would be a legal placement that does not increase $ss(P)$, a contradiction.

Note that as a consequence of the previous observation we must have $n_{i+tx} > 0$, which in turn implies that $i + (t+1)x > T$, as otherwise we could legally place $x$ in a bin with level $i+tx$ which would fill the bin and so reduce $ss(P)$. Thus the average level of bins in class $X_i$ is at least

$$\begin{aligned} \frac{\sum_{c=b}^{t}(i+cx)}{t-b+1} &= i + \frac{x(b+t)}{2} \\ &> \frac{T-x+i+b}{2} > \frac{T-x}{2}. \quad ■ \end{aligned}$$

Since the Claim holds for all the nonempty $X_i$, this implies that the average level of *all* partially filled bins in $P$ is at least $(T-x)/2$.

Let $F$ and $U$ denote the number of full and partially filled bins and at the time $x$ was packed. Let $F'$ be such that $B(F+F')$ is total volume of items in full bins. By the definition of the threshold $T$ in the description of the algorithm $SST$,

$$T > \frac{B(F+F')+U(T-x)/2}{F} - 1$$

This implies

$$F > \frac{BF'+U(T-x)/2}{T+1-B} > \frac{(F'+U)(T-x)}{2(T-(B-1))} \geq \frac{F'+U}{2}$$

After $x$ is packed, no new bins are started by assumption, and the only way that the waste can grow is if new items are added to the partially filled bins. Note that the worst situation is if all the partially-filled bins are filled to level $B-1$. If any partially-filled bin were to be completely filled, the total waste would increase by at most $2B-3$ while the number of full bins would increase by 1, which would drive the ratio of $(s(L)/B)/F$ down toward 2, and we are only trying to prove an upper bound of 3. Thus at the end of the packing of $L$, we will in the worst-case have

$$s(L)/B < F+F'+U \leq F+2F = 3SST(L),$$

as required. ■

THEOREM 3.4. *For any discrete perfect-packing distribution $F$, $EW_n^{SST} = O(n^{2/3})$.*

**Proof.** We first observe that by the operation of $SST$, the expected increase in $ss(P)$ at any step is no more than it would be under $SS$, so the martingale arguments used to prove Theorem 3.2 still apply. Thus we can conclude from (3.4) with $x = 4n^{4/3}$ that for some fixed $c$ the probability that the maximum number of partially filled bins ever encountered under $SS$ exceeds $cn^{2/3}$ is at most $1/n^{1/3}$. We break into cases.

In those instances for which the maximum number of partially filled bins exceeds $cn^{2/3}$, the waste can be no more than $nB$, so the contribution to the overall expected waste is at most $nB/n^{1/3} = Bn^{2/3}$. In those instances where the $cn^{2/3}$ bound applies, we can argue that the threshold $T$ must in bounded time decline to $B$ and stay there, since the contribution of the partially filled bins to the sum of item sizes $s$ becomes less and less significant, and when $T > B$, newly-filled bins must have contents at least 1 unit less than the average for previously filled bins. Once $T$ converges to $B$, the waste due to overfull bins can no longer grow. Since by assumption the waste due to partially filled bins is at most $cn^{2/3}$, the desired overall asymptotic bound follows. Details postponed to the full paper. ■

We implemented the bin covering algorithms $NF$, $SSNF_r$, $r \in \{1,9\}$, and $SST$, and tested them on the discrete distributions covered for bin packing in [11]. We also determined the asymptotic expected ratios of $OPT(L_n(F))$ to $s(L_n(F))$ for these distributions using the linear programs described in the next section. Experimental results, although limited, support our conjecture about the superiority of $SST$. Figures 1 through 3 cover our results for the interval distributions $U\{18,j,100\}$, in which the bin size is 100 and the item sizes are 18 through $j$, equally likely. This set of distributions was chosen for the variety of optimal behavior it exhibits.

Figure 1 compares the asymptotic expected ratios of $OPT(L_n(F))$ to $s(L_n(F))$ for bin packing and bin covering under these distributions, as a function of $j$, $18 \leq j \leq 99$. The $j$'s for which $U\{18,j,100\}$ is a perfect-packing distribution are those for which both curves coincide with the horizontal line at 1.00. The straight lines leaving the frame continue to be straight outside the frame. Note that for the non-perfect-packing distributions, the departure from ratio 1 is more pronounced for bin covering than bin packing.

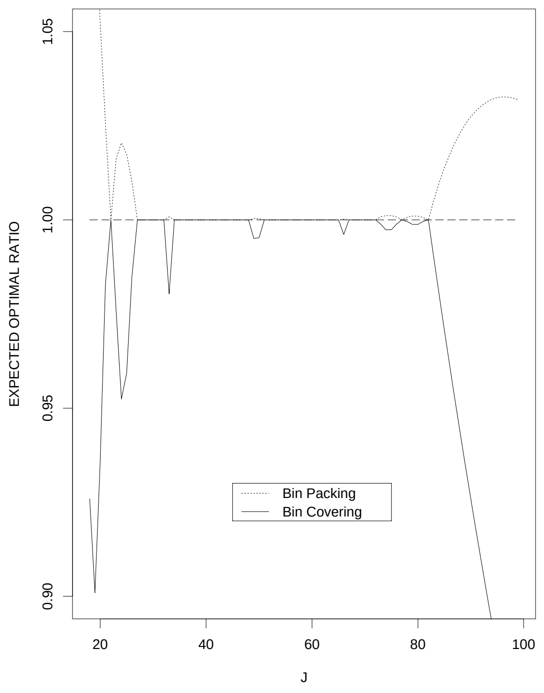

Figure 1: Comparison of asymptotic expected ratios of $OPT(L_n(F))$ to $s(L_n(F))$ for distributions $U\{18,j,100\}$, $18 \leq j \leq 99$.

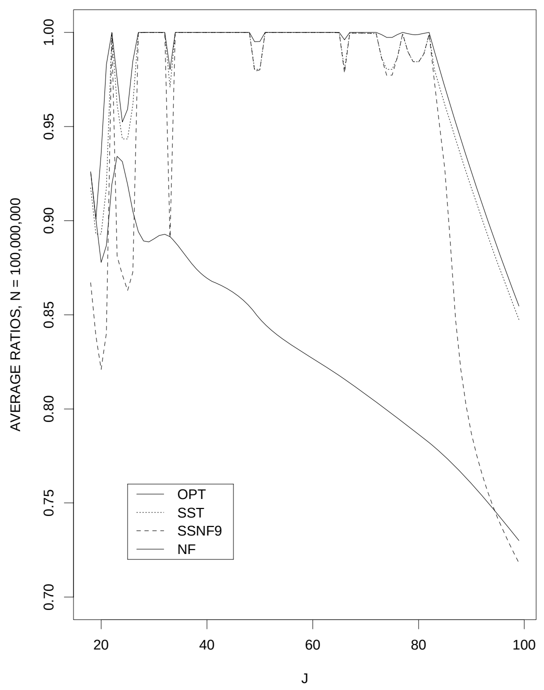

Figure 2: Measured average values of $A(L_n(F))/s(L_n(F))$ for various algorithms and distributions $U\{18, j, 100\}$, $18 \le j \le 99$, when $n = 100,000,000$.

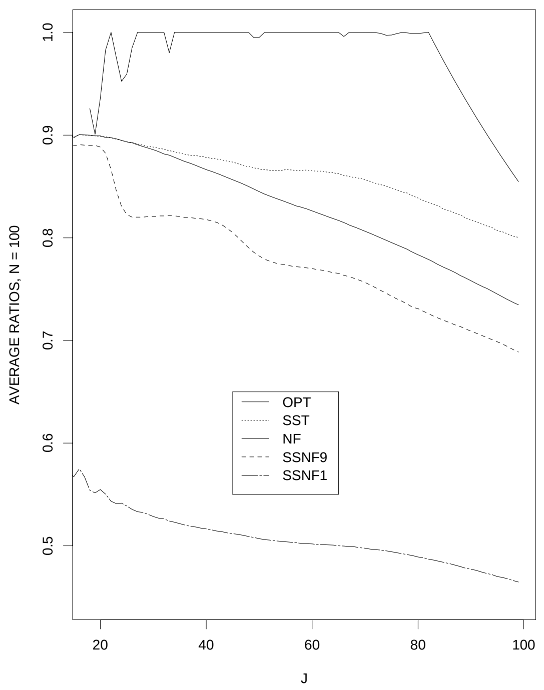

Figure 3: Measured average values of $A(L_n(F))/s(L_n(F))$ for various algorithms and distributions $U\{18, j, 100\}$, $18 \le j \le 99$, when $n = 100$.

Figure 2 compares the performance of $NF$, $SSNF_9$ and $SST$ on these distributions for $n = 10^8$, with averages taken over 3 instances. (We also tested $SSNF_1$, whose behavior was typically worse than that for $SSNF_9$ on the non-perfect-packing distributions.) Note that $SST$ outperforms $SSNF_9$ for all non-perfect packing distributions and is better than $NF$ (typically substantially so) for all $j > 19$, and within 1% for $j \in \{18, 19\}$. $SSNF_9$ on the other hand is substantially outperformed by $NF$ for several $j$.

Given that $SST$ intentionally leaves bins partially full, one might expect it not to do well until $n$ is fairly large. Figure 3 indicates that this is not the case, at least with reference to the competition. Here the algorithms (including $SSNF_1$) are compared for the above distributions when $n = 100$, with averages now taken over 10,000 instances so that equivalent precision can be obtained. Here both $SSNF_1$ and $SSNF_9$ are always outperformed by $NF$, while $SST$ holds its own for small $j$ and then pulls significantly ahead as $j$ increases.

## 4 Online Algorithms that are Asymptotically Optimal for all Discrete Distributions

We begin by showing that for any fixed discrete distribution $F$ there is an $O(nB)$ algorithm $SS^F$ such that $ER_\infty^{SS^F}(F) = 1$. We then indicate how to use these to build a learning algorithm $SS^*$ that has $ER_\infty^{SS^*}(F) = 1$ for all discrete distributions $F$.

The algorithms $SS^F$ are built using the linear programs alluded to in the previous section for determining the optimum expected waste rate for a given distribution $F$. These linear programs are similar to those presented for bin packing in [10]. Suppose our discrete distribution is as described above, with a bin capacity $B$, integer item sizes $s_1, s_2, \ldots, s_J$, and rational probabilities $p_1, p_2, \ldots, p_J$.

Let us say that a bin has *level* $h$ if the total size of the items it contains is $h$. Our linear program (LP) will have $JB$ variables $v(j,h)$, $1 \le j \le J$ and $0 \le h \le B-1$, where $v(j,h)$ represents the rate at which items of size $s_j$ go into bins whose current level is $h$. There are three sets of constraints. The first two essentially say that all items are packed into bins that were at the time incompletely filled:

$$v(j,h) \ge 0, \qquad 1 \le j \le J, \quad 0 \le h \le B-1 \quad (4.5)$$

$$\sum_{h=0}^{B-1} v(j,h) = p_j, \qquad 1 \le j \le J \qquad (4.6)$$

The third set of constraints says that bins with a given level are created at least as fast as they disappear. To formulate these constraints, we use the following shorthand to denote the net rate of creation of bins with level $h$, $1 < h \leq 2B-2$:

$$x(h) = \sum_{j=1}^{J} v(j, h - s_j) - \sum_{j=1}^{J} v(j,h)$$

where the values of $v(k, h - s_k)$ when $h - s_k < 0$ and of $v(j,h)$ when $h \geq B$ are taken to be 0 by definition for all $h$ and $k$. Our final set of constraints is then simply

$$x(h) \geq 0 \qquad 1 \leq h \leq B-1 \tag{4.7}$$

To specify the LP's optimization criterion, let

$$\begin{aligned} \underline{w} &= \sum_{h=1}^{B-1} h \cdot x(h) \\ \overline{w} &= \sum_{h=B+1}^{2B-2} (h-B)x(h) \end{aligned}$$

Note that these correspond to the waste due to under- and overfilling bins, respectively. We wish to minimize

$$c(F) \equiv \underline{w} + \overline{w}. \tag{4.8}$$

We then have the following analog of Theorem 4.1 from [10], which will be proved in the full paper:

THEOREM 4.1. *There is a constant $\alpha_B$, depending on $B$ but not otherwise on $F$, such that for all sufficiently large $n$*

$$\left|EW_n^{OPT}(F) - nc(F)\right| \leq \alpha_B \sqrt{n} \tag{4.9}$$

*and*

$$\lim_{n\to\infty} E\left[\frac{|P_n^{OPT}(F)|}{s(L_n(F))/B}\right] = 1 - \frac{c(F)}{\sum_{i=1}^{J} s_i p_i}. \tag{4.10}$$

Note that if $c(F) = 0$ then $F$ has the perfect packing property, and so we can simply use the Sum-of-Squares bin packing algorithm $SS$ for $SS^F$. By the results in [10], its expected waste (in bin packing terms) will be $O(\sqrt{n})$ and can be at most $B$ times that in bin covering terms, which is still $O(\sqrt{n})$ for fixed $F$.

Suppose, however, that $c(F) > 0$ and so $F$ does not have the perfect packing property. We now derive a new distribution $F'$ that does have the perfect packing property from $F$. We begin by modifying $p$ to a new function $q : \{1, \ldots, B-1\} \to \mathbf{Q}$ that will be our new probability distribution. This is accomplished in stages. We begin by setting $q_1(h) = \sum_{s(j)=h} p(j)$, $1 \leq h \leq B-1$. Then, from the values of the variables in an optimal solution to the LP, we derive the rates $r(i,j)$ at which items of size $i$ create overfilled bins with total contents $B+j$. For $1 \leq h \leq B-1$, define

$$q_2(h) = q_1(h) - \sum_{i=1}^{h-1} r(i,j) + \sum_{i=h+1}^{B-1} r(i, i-h).$$

For each $r(i,j)$ this is equivalent to replacing $r(i,j)$ items of size $i$ with items of size $i-j$, which in effect gets rid of all the overfilled bins in the optimal packing specified by the LP. To get rid of the underfilled bins, we set $q_3(h) = q_2(h) + x(B-h)$ and normalize to a probability distribution by letting $\bar{x} = \sum_{h=1}^{B-1} x(h)$ and setting

$$q(h) = \frac{q_3(h)}{1+\bar{x}}, \quad 1 \leq h \leq B-1.$$

It is not difficult to show that the distribution $F'$ determined by $q$ has the perfect packing property, and hence if packed by $SS$ will have $O(\sqrt{n})$ waste. Our algorithm $SS^F$ works by simulating the $SS$ packing of a list generated according $F'$ while packing a list generated according to $F$. We do this by a combination of introducing *imaginary* items and *truncating* real items. The basic packing loop goes as follows:

First, we flip a biased coin and with probability $\bar{x}/(1+\bar{x})$, we decide to generate an imaginary item. The size of the imaginary item is chosen to be $h$ with probability $x(B-h)/\bar{x}$. This item is then added to the packing according to the $SS$ rule. For the purposes of the packing, it takes up space just like a real item, but at the end of the packing a bin containing an imaginary item may not really be full, in which case all its "real" contents will be declared wasted.

If the coin flips the other way, then we take the next item from our online list of real items to be packed. Let $h$ be its size, and let $r(h,0) = q_1(h) - \sum_{i=1}^{h-1} r(i,j)$. Then declare the "truncation" $t$ for the item to be $i$ with probability $r(h,i)/q_1(h)$, $0 \leq i < h$, and pack it using $SS$, treating it from now until the end of the packing as if its size is really $h-t$. If in the end this item is in a full bin, that bin will actually be overfilled by at least $t$, and this item can thus contribute $t$ to the overall waste for the final packing.

It is easy to verify that, counting the imaginary items and truncations as real, this process produces the $SS$ packing of a list $L$ of items generated according to $F'$, and that the expected length of $L$ is $n(1+\bar{x}) < 2n$, including $n\bar{x}$ imaginary items. Thus by the results of [10] for $SS$, this packing is expected to contain at most $O(\sqrt{n})$ wasted space.

Let us now consider separately the additional waste caused by imaginary items and by truncated items. An imaginary item of size $i$ in a full bin of the $SS$ packing can cause real items of total (truncated) size at most

$B - i$ to be wasted. Since imaginary items of size $i$ are generated with probability $x(B-i)/\bar{x}$ whenever an imaginary item is generated, the expected total waste of this sort is thus at most

$$n\bar{x} \sum_{i=1}^{B-1} (B-i) \frac{x(B-i)}{\bar{x}} = n \sum_{i=1}^{B-1} ix(i) = n\underline{w}.$$

Item truncation can cause additional waste in two ways. First, it can cause a bin to be overfilled. Second, it can cause a bin that contained an imaginary item of size $i$ to yield more than $B - i$ real waste. In either case, the additional waste attributable to the item is at most the amount by which it was truncated. Thus the expected waste due to item truncation is at most

$$\begin{aligned} n \sum_{h=1}^{B-1} \sum_{i=1}^{h-1} r(h,i)(i) &= n \sum_{i=1}^{B-2} \sum_{h=i+1}^{B-1} r(h,i)(i) \\ &= n \sum_{i=1}^{B-2} x(B+i)(i) = n\overline{w} \end{aligned}$$

From this we can conclude that the expected waste is within $O(\sqrt{n})$ of the expected waste in an optimal solution, and so $ER_{\infty}^{SS^F}(F) = 1$.

Our algorithm $SS^*$ that has $ER_{\infty}^{SS^*}(F) = 1$ for all discrete distributions $F$ works by refining its estimate $F_i$ of $F$ at ever increasing intervals, solving the LP, and then applying $SS^{F_i}$ until the next refinement. Details and proofs will appear in the full paper.

**Acknowledgment**. The authors thank Jim Reeds for pointing out the relevance of submartingales to our average-case analysis.

## References

[1] S. Albers and M. Mitzenmacher. Average-case analyses of first fit and random fit bin packing. In *Proc. Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 290–299, Philadelphia, 1998. Society for Industrial and Applied Mathematics.

[2] S. B. Assman, D. S. Johnson, D. J. Kleitman, and J. Y-T. Leung. On a dual version of the one-dimensional bin packing problem. *J. Algorithms*, 5(4):502–525, 1984.

[3] S. B. Assmann. *Problems in Discrete Applied Mathematics.* PhD thesis, Department of Mathematics, MIT, Cambridge, MA, 1983.

[4] L. Breiman. *Probability.* Addison-Wesley, Reading, Mass., 1968.

[5] E. G. Coffman, Jr., C. Courcoubetis, M. R. Garey, D. S. Johnson, L. A. McGeoch, P. W. Shor, R. R. Weber, and M. Yannakakis. Fundamental discrepancies between average-case analyses under discrete and continuous distributions. In *Proceedings 23rd Annual ACM Symposium on Theory of Computing*, pages 230–240, New York, 1991. ACM Press.

[6] E. G. Coffman, Jr., C. Courcoubetis, M. R. Garey, D. S. Johnson, P. W. Shor, R. R. Weber, and M. Yannakakis. Bin packing with discrete item sizes, Part I: Perfect packing theorems and the average case behavior of optimal packings. *SIAM J. Disc. Math.*, 13:384–402, 2000.

[7] C. Courcoubetis and R. R. Weber. Stability of on-line bin packing with random arrivals and long-run average constraints. *Prob. Eng. Inf. Sci.*, 4:447–460, 1990.

[8] J. Csirik, J. B. G. Frenk, G. Galambos, and A. H. G. Rinnooy Kan. Probabilistic analysis of algorithms for dual bin packing problems. *J. Algorithms*, 12:189–203, 1991.

[9] J. Csirik, J. B. G. Frenk, M. Labbé, and S. Zhang. Two simple algorithms for bincovering. *Acta Cybernetica*, 14:13–25, 1999.

[10] J. Csirik, D. S. Johnson, C. Kenyon, J. B. Orlin, P. W. Shor, and R. R. Weber. On the sum-of-squares algorithm for bin packing. In *Proceedings of the 32nd Annual ACM Symposium on the Theory of Computing*, pages 208–217, New York, 2000. ACM.

[11] J. Csirik, D. S. Johnson, C. Kenyon, P. W. Shor, and R. R. Weber. A self organizing bin packing heuristic. In M. Goodrich and C. C. McGeoch, editors, *Proceedings 1999 Workshop on Algorithm Engineering and Experimentation*, pages 246–265, Berlin, 1999. Lecture Notes in Computer Science 1619, Springer-Verlag.

[12] J. Csirik and V. Totik. On-line algorithms for a dual version of bin packing. *Disc. Appl. Math.*, 21:163–167, 1988.

[13] W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

[14] S. Han, D. Hong, and J. Y.-T. Leung. Probabilistic analysis of a bin covering algorithm. *Oper. Res. Lett.*, 18:193–199, 1996.

[15] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin packing problem. In *Proc. 23rd Ann. Symp. on Foundations of Computer Science*, pages 312–320, 1982. IEEE Computer Soc.

[16] C. Kenyon, Y. Rabani, and A. Sinclair. Biased random walks, Lyapunov functions, and stochastic analysis of best fit bin packing. *J. Algorithms*, 27:218–235, 1998. Preliminary version under the same title appeared in *Proc. Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 351–358, 1996.

[17] W. T. Rhee. A note on optimal bin packing and optimal bin covering with items of random size. *SIAM J. Comput.*, 19:705–710, 1990.

[18] W. T. Rhee and M. Talagrand. Optimal bin covering with items of random size. *SIAM J. Comput.*, 18:487–498, 1989.