Online Bin Covering: Expectations vs. Guarantees[⋆]

Marie G. Christ, Lene M. Favrholdt, and Kim S. Larsen

University of Southern Denmark, Odense, Denmark
{christm,lenem,kslarsen}@imada.sdu.dk

Abstract. Bin covering is a dual version of classic bin packing. As usual, bins have size one and items with sizes between zero and one must be packed. However, in bin covering, the objective is to cover as many bins as possible, where a bin is covered if the sizes of items placed in the bin sum up to at least one. We are considering the online version of bin covering. Two classic algorithms for online bin packing that have natural dual versions are Harmonic$_k$ and Next-Fit. Though these two algorithms are quite different in nature, competitive analysis does not distinguish these bin covering algorithms.
In order to understand the combinatorial structure of the algorithms better, we turn to other performance measures, namely relative worst order, random order, and max/max analysis, as well as analyses under restricted input assumptions or uniformly distributed input. In this way, our study also supplements the ongoing systematic studies of the relative strengths of various performance measures.
We make the case that when guarantees are needed, even under restricted input sequences, the dual Harmonic$_k$ algorithm is preferable. In addition, we establish quite robust theoretical results showing that if items come from a uniform distribution or even if just the ordering of items is uniformly random, then dual Next-Fit is the right choice.

1 Introduction

Bin covering [1] is a dual version of classic bin packing. As usual, bins have size one and items with sizes between zero and one must be packed. However, in bin covering, the objective is to cover as many bins as possible, where a bin is covered if the sizes of items placed in the bin sum up to at least one. We are considering the online version of bin covering. A problem is online if the input sequence is presented to the algorithm one item at a time, and the algorithm must make an irrevocable decision regarding the current item without knowledge of future items.

Bin covering algorithms have numerous important applications. For instance when packing or canning food items guaranteeing a minimum weight or volume, reductions in the overpacking of even a few percent may have a large economic

⋆ Supported in part by the Danish Council for Independent Research.

impact. If items arrive on a conveyor belt, for instance, the problem becomes online.

Classic algorithms for online bin packing are Next-Fit and Harmonic$_k$ [21]. Next-Fit is a very simple and natural algorithm, and Harmonic$_k$ was designed to obtain a competitive ratio [24, 19] better than any Any-Fit algorithm (First-Fit and Best-Fit are examples of Any-Fit algorithms for bin packing, and the competitive ratio of Next-Fit is worse than both these algorithms). Harmonic$_k$ and variations of it have been analyzed extensively [22, 25, 23]. We consider the obvious dual version of these, DNF [1] and DH$_k$ [12]. These algorithms are quite different in nature and the bin packing versions are clearly separated, having competitive ratios of 2 and approximately 1.691, respectively. However, for bin covering, competitive analysis does not separate them! In fact, for bin covering, competitive analysis categorizes both algorithms as being worst possible (among reasonable algorithms). This is unlike the situation in bin packing, and in general, results from bin packing do not transfer directly to bin covering.

To understand the algorithmic differences better, it is therefore necessary to employ different techniques, and we turn to other generally applicable performance measures, namely relative worst order analysis, random order analysis, and max/max analysis. As for almost all performance measures, the idea is to abstract away some details of the problem to enable comparisons. Without some abstraction, it is hard to ever, analytically, claim that one algorithm is better than another, since almost any algorithm performs better than any other algorithm on at least one input sequence. For all the measures considered here, the abstraction can be viewed as being defined via first a partitioning of the set of input sequences of a given length and then an aggregation of the results from each partition. For each sequence length, competitive analysis, for instance, considers all the ratios of the online performance to the optimal offline performance obtained for each sequence of that length, and then takes the worst ratio of all of these. The measures above employ a less fine-grained partition of the input space. Worst order and random order analysis group permutations of the same sequence together instead of considering each sequence separately, considering worst-case or average-case performance, respectively, within each partition. With max/max analysis the partitioning of the input space is even coarser: for each sequence length n, the online worst-case behavior over all sequences of length n is compared to the worst-case optimal offline behavior over all sequences of length n. There is no one correct way to compare algorithms, but since these measures focus on different aspects of algorithmic behavior, considering all of the ones above lead to a very broad analysis of the problem. Extensive motivational sections can be found in the papers introducing these measures and in the survey [13]. As a further supplement, we analyze restricted input sequences, where items have similar size, which is likely to happen in practice if one is packing products with an origin in nature, for instance. Finally, we consider input sequences containing items having uniformly distributed sizes.

Relative worst order analysis [3, 4] has been applied to many problems; a recent list can be found in [15]. In [16], bin covering was analyzed, but using a version of the problem allowing items of size 1. We analyze the more commonly studied version for bin covering, where all items are strictly smaller than 1. Since worst-case sequences from [16] contain items of size 1, this leads to slightly different results. For completeness, we include these results. Random order analysis [20] was introduced for classic bin packing, but has also been used for other problems; a server problem, for instance [7]. Max/max analysis [2] was introduced as an early step towards refining the results from competitive analysis for paging and a server problem.

Relative worst order analysis emphasizes the fact that there exist multisets of input items where DNF can perform $\frac{3}{2}$ times as poorly as DH_k. On the other hand, DH_k's method of limiting the worst case also means that it has less of an opportunity to reach the best case, as opposed to DNF. This is reflected in the random order analysis, where DNF comes out at least as well as DH_k. Another way of approaching randomness is to analyze a uniform distribution. We establish new results on DH_k showing that its performance here is slightly worse than that of DNF, in line with the random order results. With the max/max analysis, a distinction between the two algorithms can only be achieved, when the item sizes are limited, and DH_k is the algorithm selected as best by this measure. With respect to competitive analysis, we also consider restricted input in the sense that item sizes may only vary across one or two consecutive DH_k partitioning points. This is a formal way of treating the case where items are of similar size, while allowing greater variation when this size is large. We show that with this restricted form of input, considering the worst case measures of competitive analysis, DH_k is deemed better than DNF, as DNF is vulnerable to worst-case sequences where DH_k can organize the packing differently.

This study also contributes to the ongoing systematic studies of the relative strengths of various performance measures, initiated in [7]. Up until that paper, most performance measures were introduced for a specific problem to overcome the limitations of competitive analysis. In [7], comparisons of performance measures different from competitive analysis were initiated, and this line of work has been continued in [5, 8, 6], among others. Our results supplement results in [11], showing that no deterministic algorithm for the bin covering problem can be better than $\frac{1}{2}$-competitive and giving an asymptotically optimal algorithm for the case of items being uniformly distributed on $(0, 1)$. For DNF, [10] established an expected competitive ratio of $\frac{2}{e}$ under the same conditions.

Due to space restrictions, several proofs have been omitted or shortened. Refer to [9] for all the details.

Bin Covering

In the one dimensional bin covering problem, the algorithm gets an input sequence $I = \langle i_1, i_2, \ldots \rangle$ of item sizes, where for all j, $0 < i_j < 1$. The goal is to pack the items in a maximum number of bins, each having size 1, such that the sum of the sizes of the items within each bin is at least one, i.e., the bin is *covered*. Requiring items to be strictly smaller than 1 corresponds to assuming that items of size 1 are treated separately. This makes sense, since there is no advantage in combining an item of size 1 with any other items in a bin. In other words, any algorithm not giving special treatment to items of size 1 could trivially be improved by doing so.

In algorithms for bin packing and covering, it is standard to use the terminology that a bin is *open* if it is one of the bins that an algorithm is currently considering for the next item, and *closed* if the bin has received items, but the algorithm will not consider that bin again for future items.

Thus, the objective for a bin covering algorithm A is to maximize the number of bins covered as a result of processing an input sequence I. We let $\mathrm{A}(I)$ denote this number of covered bins. We let $\textsc{Opt}$ denote an optimal offline algorithm. Thus, $\textsc{Opt}(I)$ is the largest number of bins that can be covered by any algorithm processing I.

Assmann, Johnson, Kleitman, and Leung [1] introduced the Dual $\textsc{Next-Fit}$ algorithm (DNF), an adaption of the $\textsc{Next-Fit}$ algorithm for bin packing. DNF always keeps a single bin open. The arriving items are packed into the open bin until the open bin has a content of at least one. Then the open bin is closed and a new empty bin becomes the open bin.

$\textsc{Harmonic}_k$ was introduced for bin packing by Lee and Lee [21]. This algorithm partitions the interval $(0,1]$ into k subintervals, with the partitioning points at $\frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{k}$, resulting in the intervals $(0, \frac{1}{k}], (\frac{1}{k}, \frac{1}{k-1}], \ldots, (\frac{1}{2}, 1)$. For each of these k subintervals, $\textsc{Harmonic}_k$ keeps one open bin into which the items belonging to this subinterval are packed at their arrival. This means that each closed bin for the interval $(\frac{1}{j}, \frac{1}{j-1}]$ contains exactly j items. The natural adaptation to the bin covering problem is to use $(0, \frac{1}{k}), [\frac{1}{k}, \frac{1}{k-1}), \ldots, [\frac{1}{2}, 1)$. The resulting algorithm, $\mathrm{D}\textsc{Harmonic}_k$ (DH_k), uses exactly j items from the interval $[\frac{1}{j}, \frac{1}{j-1})$ to cover a bin. All through the paper we assume that $k \geq 2$, since for $k = 1$, DH_k becomes DNF.

2 Competitive Analysis

In competitive analysis [24, 19], the performance of an online algorithm is compared to that of an optimal offline algorithm $\textsc{Opt}$. An algorithm A for a maximization problem is called *c-competitive* if there exists a fixed constant b such that for any input sequence I, it holds that $\mathrm{A}(I) \geq c\,\textsc{Opt}(I) + b$. The supremum over all such c is the *competitive ratio* CR(A) of A. Note that some authors reverse the order of the algorithm and $\textsc{Opt}$ to get ratios larger than one.

For bin covering, Csirik and Totik [11] showed that no deterministic online algorithm can be better than $\frac{1}{2}$-competitive. DNF was shown to be $\frac{1}{2}$-competitive in [1], and the same result for DH_k was noted in [16]. For completeness, to show that this result is tight for a large class of algorithms, we define a *reasonable* algorithm to be one that closes bins as soon as they are covered, does not close bins before they are covered, and does not have more than a constant number of open bins at any point.

Theorem 1. *Any deterministic reasonable algorithm has competitive ratio* $\frac{1}{2}$.

2.1 Limiting the item sizes

In some applications of the bin covering problem it is likely that the sizes of the items contained in an input sequence differ only slightly, e.g., packing similar food items into a container, guaranteeing the consumer a minimum weight. In the following, we investigate the performance of DNF and DH_k on sequences with similar-sized items. Since it seems reasonable to allow larger variance in size when the considered sizes are large, we consider sequences containing item sizes from consecutive DH_k intervals.

We first consider intervals $(a, b) \subseteq (0, 1)$ that contain exactly one DH_k partitioning point. Afterwards, we consider sequences with exactly two DH_k partitioning points. We emphasize that there are no restrictions on the endpoints a and b, which can be any real numbers, as long as the interval between them contains exactly one or two DH_k partitioning points. In both cases, DH_k turns out to have the better ratio.

For any $(a, b) \subseteq (0, 1)$, we let $\mathrm{CR}_{a,b}$ denote the competitive ratio on sequences where all item sizes are in (a, b).

If (a, b) does not contain at least one of the interval borders used by DH_k, then DH_k behaves exactly like DNF. If (a, b) contains a DH_k border, then we define $\frac{1}{p} = \max\left\{\frac{1}{l} \,\middle|\, l \in \mathbb{N}, \frac{1}{l} < b\right\}$, and refer to $\frac{1}{p}$ as the *maximal border in* (a, b).

Theorem 2. *If* $\frac{1}{p+1} \leq a < \frac{1}{p}$, *then* $\mathrm{CR}_{a,b}(\mathrm{DNF}) = \frac{p}{p+1}$.

Theorem 3. *If* $\frac{1}{p+1} \leq a < \frac{1}{p}$ *and* $k \geq p$, *then* $\mathrm{CR}_{a,b}(\mathrm{DH}_k) = \frac{p^2+1}{p(p+1)}$.

It follows that if (a, b) contains exactly one DH_k partitioning point, $\frac{1}{p}$, and $k \geq p$, then DH_k has a better competitive ratio than DNF:

Corollary 1. *If* $\frac{1}{p+1} \leq a < \frac{1}{p}$ *and* $k \geq p$, *then* $\mathrm{CR}_{a,b}(\mathrm{DH}_k) > \mathrm{CR}_{a,b}(\mathrm{DNF})$.

We now consider intervals $(a, b) \subseteq (0, 1)$ that contain exactly two DH_k partitioning points. For the following theorem, note that $\frac{1}{p} < \frac{p+2}{p(p+1)} < \frac{1}{p-1}$.

Theorem 4. *If* $a < \frac{1}{p+1}$, *then*

$$\mathrm{CR}_{a,b}(\mathrm{DNF}) \leq \begin{cases} \dfrac{p+1}{p+2}, & \text{if } b \leq \frac{p+2}{p(p+1)} \\ \dfrac{p(p+1)}{p^2+2p+2}, & \text{otherwise} \end{cases}$$

Proof. Replacing p by $p+1$ in Theorem 2, we get an upper bound of $\frac{p+1}{p+2}$, since the upper bound of Theorem 2 only assumes $a < \frac{1}{p} < b$. This proves the upper bound for $b \leq \frac{p+2}{p(p+1)}$.

If $b > \frac{p+2}{p(p+1)}$, we choose ε, $0 < \varepsilon < \min\left\{\frac{1}{2(p-1)(p+1)n}\left(\frac{1}{p+1} - a\right), b - \frac{p+2}{p(p+1)}\right\}$, the only purpose of this complicated expression being that we should ensure that all items below belong to (a, b). Now, we consider a sequence consisting of the following subsequences:

- $\langle\langle\frac{1}{p}\rangle^{p-1}, \frac{1}{p} - 2\varepsilon, \frac{p+2}{p(p+1)} + \varepsilon\rangle^{(p+1)(p-2)n}$
- $\langle\frac{1}{p+1}+i(p-1)\varepsilon, \frac{1}{p+1}-(i+1)(p-1)\varepsilon, \langle\frac{1}{p+1}+\varepsilon\rangle^{p-2}, \frac{1}{p+1}-\varepsilon, \frac{p+2}{p(p+1)}+\varepsilon\rangle$
  for $i = 1, 2, \ldots, (p+1)n$
- $\langle\frac{1}{p+1}+i(p-1)\varepsilon, \frac{1}{p+1}-(i+1)(p-1)\varepsilon, \langle\frac{1}{p+1}\rangle^{p-2}, \frac{1}{p+1}-\varepsilon, \frac{p+2}{p(p+1)}+\varepsilon\rangle$
  for $i = (p+1)n+1, (p+1)n+2, \ldots, 2(p+1)n-1$
- $\langle\frac{1}{p+1} + 2(p+1)n(p-1)\varepsilon, \langle\frac{1}{p+1}\rangle^{p-2}, \frac{1}{p+1} - \varepsilon, \frac{p+2}{p(p+1)} + \varepsilon\rangle$
- $\langle\frac{1}{p+1} - (p-1)\varepsilon\rangle$

Giving the items in this order, DNF covers $(p+1)(p-2)n + (p+1)n + (2(p+1)n - 1 - (p+1)n) + 1 = p(p+1)n$ bins. In the full version it is shown that Opt covers $(p^2 + 2p + 2)n$ bins. □

Theorem 5. *If* $\frac{1}{p+2} \leq a < \frac{1}{p+1}$ *and* $k \geq p+1$, *then*

$$\mathrm{CR}_{a,b}(\mathrm{DH}_k) = \begin{cases} \dfrac{p^3 + 2p^2 + p + 2}{p(p+1)(p+2)}, & \textit{if } b \leq \frac{p+2}{p(p+1)} \\ \dfrac{p^3 + 2p^2 + 2}{p(p+1)(p+2)}, & \textit{otherwise} \end{cases}$$

Proof. We only sketch the proof of the lower bound here.

Items of size less than $\frac{1}{p+1}$ are called *small*, items of size at least $\frac{1}{p}$ are called *large*, and the remaining items are called *medium*. Let s, m, and ℓ denote the number of small, medium, and large items, respectively.

Consider an optimal packing. For $i = 1, 2, 3$, let n_i denote the number of bins with exactly $p+i-1$ items. Then, $n = n_1+n_2+n_3$ is the number of bins covered by Opt. Since DH_k covers exactly $\lfloor\frac{s}{p+2}\rfloor + \lfloor\frac{m}{p+1}\rfloor + \lfloor\frac{\ell}{p}\rfloor$ bins, we can consider items from the three types of bins separately. The contribution to the number of bins covered by DH_k from the n_i items is at least $d_i - 3$, where

$$d_i \geq \begin{cases} \dfrac{p^3 + 2p^2 + p + 2}{p(p+1)(p+2)} n_i, & \text{if } b \leq \frac{p+2}{p(p+1)} \\ \dfrac{p^3 + 2p^2 + 2}{p(p+1)(p+2)} n_i, & \text{otherwise} \end{cases}$$

□

It follows that if (a, b) contains exactly two DH_k partitioning points, then DH_k has a better competitive ratio than DNF:

Corollary 2. *If* $\frac{1}{p+2} \leq a < \frac{1}{p+1}$, *then* $\mathrm{CR}_{a,b}(\mathrm{DH}_k) > \mathrm{CR}_{a,b}(\mathrm{DNF})$.

3 Relative Worst Order Analysis

Relative worst order analysis was introduced by Boyar and Favrholdt [3] and compares the performance of two algorithms A and B directly instead of via the comparison to Opt. Algorithms are compared on the same input sequence I, but on the worst possible permutation of I for each algorithm.

Formally, if n is the length of I, and σ is a permutation on n elements, then $\sigma(I)$ denotes I permuted by σ, and we define $\mathrm{A}_W(I) = \min_\sigma \mathrm{A}(\sigma(I))$. If there exists a fixed constant b such that, for any input sequence I, $\mathrm{A}_W(I) \geq \mathrm{B}_W(I) - b$, then A and B are *comparable* and the *relative worst order ratio* of A to B is defined as follows: $\mathrm{WR}(\mathrm{A},\mathrm{B}) = \sup\{c \mid \exists b \forall I\colon\ \mathrm{A}_W(I) \geq c\,\mathrm{B}_W(I) - b\}$.

Note that since the performance of DH_k does not depend on the order in which the items are given, relative worst order analysis of DNF versus DH_k gives the same result as simply comparing the two algorithms on each sequence separately, just as competitive analysis with Opt replaced by DH_k.

In [16], a relative worst order analysis of DH_k and DNF is given for the model that allows items of size 1. It is shown that, for $i < j$, $\mathrm{WR}(H_j, H_i) = \frac{i+1}{i}$. Hence, in this model, $\mathrm{WR}(\mathrm{DH}_k, \mathrm{DNF}) = 2$, for $k \geq 2$, since DNF and DH_1 are equivalent. Note that, for $i \geq 2$, the result from [16] holds for our model too, since the lower bound sequences for these cases do not contain items of size 1.

We first show that DH_k and DNF are comparable. This is a special case of the corresponding result in [16].

Lemma 1. *For any $k \geq 1$ and any input sequence I,* $\mathrm{DH}_{kW}(I) \geq \mathrm{DNF}_W(I) - (k-1)$.

Thus, according to relative worst order analysis, DH_k is at least as good as DNF. The next lemma establishes a separation between the two algorithms.

Lemma 2. *For any $k \geq 2$,* $\mathrm{WR}(\mathrm{DH}_k, \mathrm{DNF}) \geq \frac{3}{2}$.

By providing a matching upper bound, we determine the exact relative worst order ratio of the two algorithms.

Theorem 6. $\mathrm{WR}(\mathrm{DH}_k, \mathrm{DNF}) = \frac{3}{2}$.

Thus, we conclude that according to relative worst order analysis, DH_k is a better algorithm than DNF.

4 The Random Order Ratio

The random order ratio was introduced by Kenyon [20] as the worst ratio obtained over all sequences I, comparing the expected value of an algorithm A, with respect to a uniform distribution of all permutations, σ, of I, to the value of Opt on I:

$$\mathrm{RR}(\mathrm{A}) = \liminf_{\textsc{Opt}(I)\to\infty} \frac{E_\sigma[\mathrm{A}(\sigma(I))]}{\textsc{Opt}(I)}$$

Note that $\textsc{Opt}$ is still assumed to know the entire sequence in advance, so there is no expectation involved in computing $\textsc{Opt}(I)$.

The following theorem gives a bound on how well DNF can perform with respect to the random order ratio.

Theorem 7. *The random order ratio of* DNF *is at most* $\frac{4}{5}$.

Proof. Let S^n denote all sequences of length n with item sizes from $\mathcal{I}$, where $\mathcal{I} = \{\varepsilon, 1-\varepsilon\}$ for an $0 < \varepsilon < \frac{1}{n}$. Define

$$S_i^n = \{I \in S^n \mid I \text{ contains } i \text{ items of size } \varepsilon \text{ and } n-i \text{ items of size } 1-\varepsilon\}$$

Then we can consider the following disjoint partitioning $S^n = \bigcup_{0 \le i \le n} S_i^n$. We let R^n denote the set of all sequences of length n.

The first inequality below follows from two facts:

– For any pair of sequences, $I, I' \in S_i^n$, $\textsc{Opt}(I) = \textsc{Opt}(I')$.
– For two sums $A = \sum_{i=1}^n a_i$ and $B = \sum_{i=1}^n b_i$, $\frac{A}{B} \ge \min_{1 \le i \le n} \frac{a_i}{b_i}$.

$$\begin{aligned}\frac{\mathrm{E}_{I \in S^n}[\mathrm{DNF}(I)]}{\mathrm{E}_{I \in S^n}[\textsc{Opt}(I)]} &\ge \min_{0 \le i \le n} \frac{\mathrm{E}_{I \in S_i^n}[\mathrm{DNF}(I)]}{\textsc{Opt}(I_i^n)}, \text{ where } I_i^n \in S_i^n \\ &= \min_{I \in S^n} \frac{\mathrm{E}_\sigma[\mathrm{DNF}(\sigma(I))]}{\textsc{Opt}(I)} \ge \min_{I \in R^n} \frac{\mathrm{E}_\sigma[\mathrm{DNF}(\sigma(I))]}{\textsc{Opt}(I)}\end{aligned}$$

Hence,

$$\lim_{n \to \infty} \frac{\mathrm{E}_{I \in S^n}[\mathrm{DNF}(I)]}{\mathrm{E}_{I \in S^n}[\textsc{Opt}(I)]} \ge \liminf_{\textsc{Opt}(I) \to \infty} \frac{\mathrm{E}_\sigma[\mathrm{DNF}(\sigma(I))]}{\textsc{Opt}(I)} = \mathrm{RR}(\mathrm{DNF}).$$

In the rest of the proof, we compute the leftmost expression from the above, which then gives us an upper bound on the random order ratio of DNF.

There is no difference between choosing some element from S^n uniformly at random and generating a length n sequence iteratively by choosing the next item from $\mathcal{I}$ with equal probability. Thus, we can analyze the behavior of DNF by considering a Markov chain, where the state of the system after i items have been processed is determined by the state of the open bin. The Markov chain is finite and has just three states: either there is no open bin (N – for "No"), one open bin containing one large item of size $1-\varepsilon$ (L – for "Large"), or one bin with a number of small items, each of size ε (S – for "Small"). Note that since $\varepsilon < \frac{1}{n}$, there is room for all the small items in one bin, if necessary.

This is an irreducible chain, where all states are positive recurrent, which implies that it has a stationary (equilibrium) distribution, and the probability of ending up in each of the states converges independently of the starting state [14]. The probability of being in one of the states N, L, or S can be calculated from the following equations:

$$\begin{aligned}1 &= \mathrm{Prob}[N] + \mathrm{Prob}[L] + \mathrm{Prob}[S] \\ \mathrm{Prob}[N] &= \mathrm{Prob}[L] + \mathrm{Prob}[S]/2 \\ \mathrm{Prob}[L] &= \mathrm{Prob}[N]/2 \\ \mathrm{Prob}[S] &= \mathrm{Prob}[N]/2 + \mathrm{Prob}[S]/2\end{aligned}$$

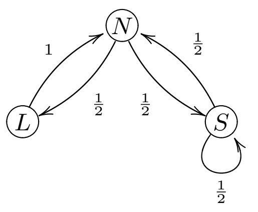

Fig. 1. A Markov chain describing DNF's behavior on the considered sequences.

This system has the solution $\text{Prob}[N] = \text{Prob}[S] = \frac{2}{5}$ and $\text{Prob}[L] = \frac{1}{5}$. From this it follows that $\text{E}_{I \in S^n}[\text{DNF}(I)]$ tends to $\text{Prob}[N]n = \frac{2}{5}n$.

For the optimal algorithm, note that its result only depends on the number of items of each size. In particular, after n items, it can cover $\lfloor \frac{n}{2} \rfloor$ bins, unless there are more small than large items. All the small items would be wasted.

Using random walks, it is easy to see that the expected difference between the number of large and small items is a low order term compared with n, and therefore does not affect the limit.

A sequence of independent stochastic variables $\{X_i\}_{i \geq 1}$, where $\text{Prob}[X_i = 1] = \text{Prob}[X_i = -1] = \frac{1}{2}$, is called a *simple random walk* [14]. It is well known that if we define $T_n = \sum_{i=1}^{n} X_i$, then $\lim_{n \to \infty} \frac{\text{E}[|T_n|]}{\sqrt{n}} = \sqrt{\frac{2}{\pi}}$ [17]. Hence, $\text{E}[|T_n|] \in O(\sqrt{n})$, and then $\text{E}_{I \in S^n}[\text{OPT}(I)] = \frac{n}{2} - O(\sqrt{n})$.

In conclusion, we get $\lim\limits_{n \to \infty} \frac{\text{E}_{I \in S^n}[\text{DNF}(I)]}{\text{E}_{I \in S^n}[\text{OPT}(I)]} = \lim\limits_{n \to \infty} \frac{\frac{2}{5}n}{\frac{n}{2} - O(\sqrt{n})} = \frac{4}{5}..$ □

Theorem 8. *The random order ratio of* DH_k *is* $\frac{1}{2}$.

Proof. The performance of DH_k does not depend on the order of the items in the sequence. Given a sequence containing n items of size $1 - \varepsilon$ and n items of size ε, where $\varepsilon < \frac{1}{n}$, DH_k will always cover $\frac{n}{2}$ bins, while OPT will cover n bins. The lower bound is given by Theorem 1, since the random order ratio of a bin covering algorithm is never worse than its competitive ratio. □

Thus, according to random order analysis, DNF is at least as good as DH_k. Though it seems hard to raise the lower bound on the random order ratio for DNF above $\frac{1}{2}$, and thereby separate the two algorithms, we conjecture that DNF is in fact strictly better than DH_k with respect to this measure. We discuss this further in the conclusion.

5 The Max/Max Ratio

The max/max ratio was introduced by Ben-David and Borodin [2] and compares an algorithm's worst-case behavior on any sequence of length n with OPT's worst-case behavior on any sequence of length n. The max/max ratio was introduced for the minimization problems paging and K-server. Since bin covering is a maximization problem, we actually need a min/min ratio. Additionally, since the input items can be arbitrarily small, letting the sequence length approach infinity does not give interesting results. Thus, we modify the measure to consider the volume, *vol*(I), of a sequence I, where *vol*(I) is the sum of the sizes of all the items in I:

$$\mathrm{MR}_{\mathrm{vol}}(\mathrm{A}) = \frac{\liminf_{v\to\infty} \min_{vol(I)=v} \mathrm{A}(I)/v}{\liminf_{v\to\infty} \min_{vol(I)=v} \textsc{Opt}(I)/v}$$

This measure cannot distinguish between DNF and DH_k in the general case:

Theorem 9. *Both* DNF *and* DH_k *have a min/min ratio of* 1.

If the item sizes are restricted to be from an interval $(a, b) \subseteq (0, 1)$, the min/min ratio can distinguish between DNF and DH_k. If (a, b) does not contain at least one of the interval borders used by DH_k, then DH_k behaves exactly like DNF. If (a, b) contains a DH_k border, then we define, as in Section 2, $\frac{1}{p}$ as the *maximal border in* (a, b).

Theorem 10. *With item sizes in* $(a, b) \subseteq (0, 1)$*, where* $\frac{1}{p} \in (a, b)$*,* DH_k *has a min/min ratio of* 1 *and* DNF *has a min/min ratio of* $\max\left\{\frac{1+\frac{1}{p}}{1+b}, \frac{pb}{1+b}\right\}$.

Note that $\frac{1+\frac{1}{p}}{1+b} < 1$ is equivalent to $\frac{1}{p} < b$, which follows from the definition and maximality of $\frac{1}{p}$. Furthermore, $\frac{pb}{1+b} < 1$ is equivalent to $b < \frac{1}{p-1}$, which is satisfied as long as b is not equal to $\frac{1}{p-1}$. Thus, according to min/min analysis, DH_k is better than DNF when item sizes are restricted to an interval $(a, b) \in (0, 1)$ containing a DH_k border, and $b \neq \frac{1}{p-1}$ where $\frac{1}{p}$ is the maximal border.

6 Uniform Distribution

In this section, we study the expected performance ratio of DNF and DH_k on sequences containing items drawn uniformly at random from the interval $(0, 1)$.

The expected performance ratio $\mathrm{ER}_{\mathrm{U}}(\mathrm{A})$ is the ratio between the expected performance of the algorithms A and $\textsc{Opt}$ on sequences of length n, containing items drawn uniformly at random from the interval $(0, 1)$:

$$\mathrm{ER}_{\mathrm{U}}(\mathrm{A}) = \lim_{n\to\infty} \frac{\mathrm{E}_{I\in U_n(0,1)}[\mathrm{A}(I)]}{\mathrm{E}_{I\in U_n(0,1)}[\textsc{Opt}(I)]}.$$

Theorem 11. *On a sequence containing items drawn uniformly at random from the interval* $(0, 1)$*,*

$$\mathrm{ER}_{\mathrm{U}}(\mathrm{DH}_2) = \frac{1}{2} + \frac{1}{e^2 - e} \approx 0.7141 \text{ and}$$

$$\lim_{k\to\infty} \mathrm{ER}_{\mathrm{U}}(\mathrm{DH}_k) = \frac{12 - \pi^2}{3} \approx 0.7101\,.$$

This should be compared with a result from [10], showing that on a uniform distribution, DNF has an expected performance ratio of $\frac{2}{e} \approx 0.7358$. Thus, under this assumption, DNF is a little better than DH_k.

7 Concluding Remarks

Our starting point was the fact that the very different bin covering algorithms, DNF and DH_k, are not separated by competitive analysis. Thus, the question is which algorithm to use in different scenarios. DH_k was designed to guard against worst-case sequences, and since these are often made up using pathological input, mixing very large and very small items, we have carried out analyses using the worst-case performance, but on restricted input of items of similar size. The comparison is still in DH_k's favor, though less so. Under similar conditions, Max/max analysis and relative worst order analysis also point to DH_k.

In contrast, DNF is a little better than DH_k when considering expected performance under a uniform distribution. This seems fairly robust; even if we add an element of worst-case requirements in the form of random order analysis, DNF does not appear worse than DH_k. Thus, even if an adversary gets to choose the worst sequence for the algorithm, just the fact that the items are received in the order of a random permutation removes DH_k's advantage over DNF.

Thus, unless guarantees are desired or it is known that items do not arrive in a random order, it is worth considering DNF as the algorithm of choice.

DH_k has a random order ratio of $\frac{1}{2}$, which is worst possible, whereas the upper bound we have on DNF is $\frac{4}{5}$. We conjecture that these two algorithms can be separated, and discuss this issue in rest of the section. It seems intuitively almost obvious that DNF would always get a ratio larger than $\frac{1}{2}$. The difficulty in establishing this formally stems from problems handling the size aspects using probability theory. In the hardest case, there are a linear number of very large items such that if they end up on top of each other pairwise, we get the ratio of $\frac{1}{2}$. Thus, we need to prove that some fraction of these large items do not end up pairwise on top of each other. The small items that would be packed with the large items in an optimal packing can be cut into very small pieces so there are orders of magnitude more small items than large items—but still of possibly dramatically varying size, relatively. Whereas we have strong theoretical tools for bounding the deviation from the expected number of items in certain locations in the form of Chebyshev's inequality, for instance, it is much harder to reason regarding deviations from the expected size, and it is exactly the sum of sizes of small items surrounding a large item that decides whether or not two large items end up on top of each other.

Results on the random order ratio are often difficult to establish. An exceptionally tight result appears in [18], where it is shown that the random order ratio of Next-Fit for bin packing is exactly 2. Note, however, that this result does not give indication that the random order ratio of DNF for bin covering should be $\frac{1}{2}$. The sequence establishing the lower bound of 2 consists of n items of size $\frac{1}{2}$ and kn items of size $\epsilon < \frac{1}{kn}$, for some large k. For a random ordering of these items, each item of size $\frac{1}{2}$ has a high probability of being combined with at least one of the small items. For bin covering, the problem is reversed; we must prove that each large item has a significant probability of being surrounded by a sufficient volume of small items so that it will not go into the same bin as a neighboring large item.

References

1. S.F. Assmann, 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.
2. S. Ben-David and A. Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11(1):73–91, 1994.
3. J. Boyar and L.M. Favrholdt. The relative worst order ratio for on-line algorithms. *ACM Trans. Algorithms*, 3(2), 2007.
4. J. Boyar, L.M. Favrholdt, and K.S. Larsen. The relative worst order ratio applied to paging. *J. Comput. Sys. Sci.*, 73(5):818–843, 2007.
5. J. Boyar, S. Gupta, and K.S. Larsen. Access graphs results for LRU versus FIFO under relative worst order analysis. In *SWAT*, volume 7357 of *LNCS*, pages 328–339. Springer, 2012.
6. J. Boyar, S. Gupta, and K.S. Larsen. Relative interval analysis of paging algorithms on access graphs. In *WADS*, LNCS. Springer, 2013. Accepted for publication.
7. J. Boyar, S. Irani, and K.S. Larsen. A comparison of performance measures for online algorithms. In *WADS*, volume 5664 of *LNCS*, pages 119–130. Springer, 2009.
8. J. Boyar, K.S. Larsen, and A. Maiti. A comparison of performance measures via online search. In *FAW-AAIM*, LNCS, pages 303–314. Springer, 2012.
9. M. Christ, L.M. Favrholdt, and K.S. Larsen. Online bin covering: Expectations vs. guarantees. arXiv:1309.6477 [cs.DS], 2013.
10. J. Csirik, J.B.G. Frenk, G. Galambos, and A.H.G.R. Kan. Probabilistic analysis of algorithms for dual bin packing problems. *J. Algorithms*, 12(2):189–203, 1991.
11. J. Csirik and V. Totik. Online algorithms for a dual version of bin packing. *Discrete Appl. Math.*, 21(2):163–167, 1988.
12. J. Csirik and G. Woeginger. On-line packing and covering problems. In *Online Algorithms*, volume 1442 of *LNCS*, pages 147–177. Springer, 1998.
13. R. Dorrigiv and A. López-Ortiz. A survey of performance measures for on-line algorithms. *SIGACT News*, 36(3):67–81, 2005.
14. R. Durrett. *Probability: Theory and Examples*. Dixbury Press, 1991.
15. M.R. Ehmsen, J.S. Kohrt, and K.S. Larsen. List factoring and relative worst order analysis. *Algorithmica*, 66(2):287–309, 2013.
16. L. Epstein, L.M. Favrholdt, and J.S. Kohrt. Comparing online algorithms for bin packing problems. *J. Scheduling*, 15(1):13–21, 2012.
17. J. Hoffmann-Jørgensen. *Probability with a View towards Statistics*, volume I. Chapman & Hall, 1994.
18. E.G. Coffman Jr., J. Csirik, L. Rónyai, and A. Zsbán. Random-order bin packing. *Discrete Appl. Math.*, 156:2810–2816, 2008.
19. A.R. Karlin, M.S. Manasse, L. Rudolph, and D.D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.
20. C. Kenyon. Best-fit bin-packing with random order. In *SODA*, pages 359–364, 1996.
21. C.C. Lee and D.T. Lee. A simple on-line bin-packing algorithm. *J. ACM*, 32(3):562–572, 1985.
22. P.V. Ramanan, D.J. Brown, C.C. Lee, and D.T. Lee. On-line bin packing in linear time. *J. Algorithms*, 10(3):305–326, 1989.
23. S.S. Seiden. On the online bin packing problem. *J. ACM*, 49(5):640–671, 2002.
24. D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. ACM*, 28(2):202–208, 1985.
25. G. Woeginger. Improved space for bounded space, on-line bin-packing. *SIAM J. Disc. Math.*, 6(4):575–581, 1993.