Alcom-FT Technical Report Series
ALCOMFT-TR-03-21

The Relative Worst Order Ratio for On-Line Bin Packing Algorithms*

Joan Boyar† Lene M. Favrholdt‡

Abstract

The relative worst order ratio, a new measure for the quality of on-line algorithms, was defined in [3]. Classical bin packing algorithms such as First-Fit, Best-Fit, and Next-Fit were analyzed using this measure, giving results that are consistent with those previously obtained with the competitive ratio or the competitive ratio on accommodating sequences, but also giving new separations and easier results.

In this paper we consider newer algorithms; for the Classical Bin Packing Problem we analyze HARMONIC(k) [13] and some of its variants, and for the Dual Bin Packing Problem we introduce a new, simple, unfair variant of First-Fit.

1 Introduction

The standard measure for the quality of on-line algorithms is the competitive ratio [8, 15, 10], which is, roughly speaking, the worst-case ratio, over all

*Supported in part by the Danish Natural Science Research Council (SNF) and in part by the Future and Emerging Technologies program of the EU under contract number IST-1999-14186 (ALCOM-FT).

†Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark. Email: joan@imada.sdu.dk

‡Department of Computer Science, University of Copenhagen, Denmark. Email: lenem@imada.dk. Supported in part by the Villum Kann Rasmussen Fund

possible input sequences, of the on-line performance to the optimal off-line performance. The definition of the competitive ratio has been taken rather directly from that of the approximation ratio. This seems natural in that on-line algorithms can be viewed as a special class of approximation algorithms. However, for approximation algorithms, the comparison to an optimal off-line algorithm, OPT, is very natural, since one is comparing to another algorithm of the same general type, just with more computing power, while for on-line algorithms, the comparison to OPT is to a different type of algorithm.

Although the competitive ratio has been an extremely useful notion, in many cases it has appeared inadequate at differentiating between on-line algorithms. When this is the goal, doing a direct comparison between the algorithms, instead of involving an intermediate comparison to OPT, seems the obvious choice. A direct comparison on exactly the same sequences will produce the result that many algorithms are incomparable because one algorithm does well on one type of ordering, while the other does well on another type. With the relative worst order ratio, on-line algorithms are compared directly to each other on their respective worst orderings of multi-sets. In this way, the relative worst order ratio [3] combines some of the desirable properties of the Max/Max ratio [2] and the random order ratio [11].

The Max/Max Ratio

The Max/Max ratio allows direct comparison of two on-line algorithms for an optimization problem, without the intermediate comparison to OPT, as is necessary with the competitive ratio. Rather than comparing two algorithms on the same sequence, they are compared on their respective worst-case sequences of the same length. The Max/Max Ratio applies only when the length of an input sequence yields a bound on the profit/cost of an optimal solution.

The Random Order Ratio

The random order ratio gives the possibility of considering some randomness of the request sequences without specifying a complete probability distribution. For an on-line algorithm $\mathbb{A}$, the random order ratio is the worst-case ratio, over all multi-sets of requests, of the expected performance of $\mathbb{A}$ on a random permutation of the multi-set, compared with an optimal solution. If, for all possible multi-sets of requests, any ordering of these requests is equally likely, this ratio gives a meaningful worst-case measure of how well an algorithm can do.

The Relative Worst Order Ratio

With the relative worst order ratio, one considers the worst-case performance over all permutations instead of the average-case performance as with the random order ratio. Thus, when comparing two on-line algorithms, one considers a worst-case multi-set and takes the ratio of how the two algorithms do on their worst orderings of that multi-set. Note that the two algorithms may have different "worst orderings" for the same multi-set. Two algorithms $\mathbb{A}$ and $\mathbb{B}$ are said to be *incomparable* under the relative worst order ratio, if there exists a family of multi-sets where $\mathbb{A}$ is strictly better than $\mathbb{B}$ and another family of multi-sets where $\mathbb{B}$ is strictly better than $\mathbb{A}$.

The Bin Packing Problems

In the *Classical Bin Packing Problem* we are given an unlimited number of unit sized bins and a sequence of items each with a non-negative size of at most one, and the goal is to minimize the number of bins used to pack all the items. In contrast, in the *Dual Bin Packing Problem*, we are given a fixed number n of unit sized bins, and the goal is to maximize the number of items packed in the n bins.

Algorithms

For Dual Bin Packing, algorithms sometimes have to reject an item because it does not fit in any of the n bins. An algorithm that never rejects an item unless it has to is called a *fair* algorithm.

For both problems, one can define the class of *Any-Fit* algorithms [9], which use an empty bin, only if there is not enough space in any partially full bin. For Dual Bin Packing, Any-Fit algorithms are defined to be fair.

Lee and Lee [13] defined a family of algorithms HARMONIC(k) which have a better competitive ratio than First-Fit. HARMONIC(k) is not an Any-Fit algorithm. It divides the items into classes based on their size, such that items with size in the range $(\frac{1}{j+1}, \frac{1}{j}]$ are in class C_j for $1 \leq j \leq k-1$, and all other items are in class C_k. Bins only contain items from a single class, and the items within a single class are packed using Next-Fit.

For the Dual Bin Packing Problem, we define a simple "unfair" variant of First-Fit, First-Fit^n. This algorithm behaves exactly as First-Fit would unless the item x is larger than $\frac{1}{2}$ and would be placed in the last bin, bin n. First-Fit^n rejects such an item and is thus not fair.

Other Measures

The results we obtain using the relative worst order ratio are compared to the standard competitive ratio and to the competitive ratio on accommodating sequences. Let $\mathbb{A}(I)$ be the *cost (profit)* of running the on-line algorithm $\mathbb{A}$ on I, and $\mathrm{OPT}(I)$ be the *optimal cost (profit)* that can be achieved on I. For a minimization (maximization) problem, an on-line algorithm $\mathbb{A}$ is said to be *c-competitive* if there exists a constant b such that for all input sequences I, $\mathbb{A}(I) \leq c \cdot \mathrm{OPT}(I) + b$ ($\mathbb{A}(I) \geq c \cdot \mathrm{OPT}(I) - b$). The competitive ratio of $\mathbb{A}$ is the infimum (supremum) over all c such that $\mathbb{A}$ is c-competitive. For the Dual Bin Packing Problem, the *competitive ratio on accommodating sequences* [7] is the same as the competitive ratio, except that the only request sequences considered are those for which all items could have been accepted in the n bins by an optimal off-line algorithm, so it is the worst case ratio over a restricted set of request sequences.

Previous Results

The Relative Worst Order Ratio. Previous study [3] of the relative worst order ratio for the two bin packing problems seems promising in that most results are consistent with those obtained with the competitive ratio, but new separations and easier results are possible.

More specifically, for the Classical Bin Packing Problem, First-Fit and Best-Fit are better than Worst-Fit, which is better than Next-Fit. This latter result is in contrast to the competitive ratio, where there appears to be no advantage to Worst-Fit being able to use empty space in earlier bins, since both have a competitive ratio of 2 [9]. First-Fit is still the best Any-Fit algorithm and Next-Fit is strictly worse than any Any-Fit algorithm.

For the Dual Bin Packing Problem, First-Fit is better than Best-Fit, which is better than Worst-Fit. Worst-Fit is at least as bad as any fair on-line algorithm. This contrasts favorably with the competitive ratio, where, for the restricted problem where all item sizes are multiples of some constant f, Worst-Fit is better than First-Fit [7].

The Competitive Ratio. Lee and Lee [13] have shown that the competitive ratio of HARMONIC(k) approaches about 1.691 as k approaches infinity. This is strictly better than First-Fit's competitive ratio of 1.7. Lee and Lee [13] also proposed the algorithm REFINED HARMONIC and showed that its competitive ratio is close to 1.636. Seiden defined the class of SUPER HARMONIC algorithms, which includes both REFINED HARMONIC and his own algorithm HARMONIC++, which has an asymptotic performance ratio of 1.58889.

New Results

Classical Bin Packing. We show that according to the relative worst order ratio, if the sequences can have arbitrarily small items, HARMONIC(k) is incomparable to First-Fit. However, when HARMONIC(k) is worse than First-Fit, it is only by a factor of at most $\frac{k}{k-1}$. On the other hand, if all items in the input sequences considered are restricted to having sizes greater than $\frac{1}{k+1}$, then First-Fit is strictly worse than HARMONIC(k), by a factor of $\frac{6}{5}$. The variants of HARMONIC(k) defined by Seiden, including REFINED HARMONIC, will place some of the items in certain size ranges in empty bins, where most of the space in those bins is reserved for items of other sizes. This can be done very cleverly, as Seiden did for HARMONIC++, guaranteeing that the algorithm never does more than a factor 1.58889 worse than OPT. These variants are not designed, however, to always do at least as well as First-Fit or HARMONIC(k). We show that these variants are incomparable to First-Fit and HARMONIC(k), using the relative worst order ratio. Thus, depending on the application and expected request sequence distribution, either HARMONIC(k) or HARMONIC++ could be the better algorithm.

Dual Bin Packing. For the Dual Bin Packing Problem, we define a new algorithm, First-Fit^n. In [3] First-Fit was shown to be the best Any-Fit algorithm and shown to be incomparable to one algorithm, Unfair-First-Fit, which has a better competitive ratio on accommodating sequences than First-Fit. The algorithm First-Fit^n is a simple "unfair" variant of First-Fit which can clearly do much better than First-Fit in some cases. According to the relative worst order ratio, First-Fit^n is unboundedly better than First-Fit, but it cannot be separated from First-Fit using either the competitive ratio or the competitive ratio on accommodating sequences.

2 The (Relative) Worst Order Ratio

The definition of the relative worst order ratio uses $\mathbb{A}_W(I)$, the performance of an on-line algorithm $\mathbb{A}$ on the "worst ordering" of the multi-set I of requests, formally defined in the following way.

Definition 1 Consider an on-line problem P and let I be any request sequence of length n. If σ is a permutation on n elements, then $\sigma(I)$ denotes I permuted by σ. Let $\mathbb{A}$ be any algorithm for P.

If P is a *maximization problem*, $\mathbb{A}(I)$ is the *profit* of $\mathbb{A}$'s solution on I, and

$$\mathbb{A}_{\mathrm{W}}(I) = \min_{\sigma} \mathbb{A}\{\sigma(I)\}.$$

If P is a *minimization* problem, $\mathbb{A}(I)$ is a *cost*, and

$$\mathbb{A}_{\mathrm{W}}(I) = \max_{\sigma} \mathbb{A}\{\sigma(I)\}.$$

□

Definition 2 Let S_1 and S_2 be statements defined in the following way.

$S_1(c)$: There exists a constant b such that $\mathbb{A}_{\mathrm{W}}(I) \leq c \cdot \mathbb{B}_{\mathrm{W}}(I) + b$ for all I
$S_2(c)$: There exists a constant b such that $\mathbb{A}_{\mathrm{W}}(I) \geq c \cdot \mathbb{B}_{\mathrm{W}}(I) - b$ for all I.

The *relative worst order ratio* $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ of on-line algorithm $\mathbb{A}$ to algorithm $\mathbb{B}$ is defined if $S_1(1)$ or $S_2(1)$ holds. Otherwise the ratio is undefined and the algorithms are said to be *incomparable*.

$$\text{If } S_1(1) \text{ holds, then } \mathrm{WR}_{\mathbb{A},\mathbb{B}} = \sup\{r \mid S_2(r)\}.$$
$$\text{If } S_2(1) \text{ holds, then } \mathrm{WR}_{\mathbb{A},\mathbb{B}} = \inf\{r \mid S_1(r)\}.$$

□

Note that if $S_1(1)$ holds, the supremum involves S_2 rather than S_1, and vice versa. A ratio of 1 means that the two algorithms perform identically with respect to this quality measure; the further away from 1 the greater the difference in performance. The ratio may be greater than or less than one, depending on whether the problem is a minimization or a maximization problem and on which of the two algorithms is better. These possibilities are illustrated in Table 1.

Although not all pairs of algorithms are comparable with the relative worst order ratio, for algorithms which are comparable, the measure is transitive, i.e., for any three algorithm $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$, $\mathrm{WR}_{\mathbb{A},\mathbb{B}} \leq 1$ and $\mathrm{WR}_{\mathbb{B},\mathbb{C}} \leq 1$ implies $\mathrm{WR}_{\mathbb{A},\mathbb{C}} \leq 1$, and similarly, $\mathrm{WR}_{\mathbb{A},\mathbb{B}} \geq 1$ and $\mathrm{WR}_{\mathbb{B},\mathbb{C}} \geq 1$ implies $\mathrm{WR}_{\mathbb{A},\mathbb{C}} \geq 1$ [3].

	minimization	maximization
$\mathbb{A}$ better than $\mathbb{B}$	< 1	> 1
$\mathbb{B}$ better than $\mathbb{A}$	> 1	< 1

Table 1: Values of $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ for minimization and maximization problems

Although one of the goals in defining the relative worst order ratio was to avoid the intermediate comparison of any on-line algorithm to the optimal off-line algorithm OPT, it is still possible to compare on-line algorithms to OPT. In this case, the measure is called the *worst order ratio*.

Definition 3 The *worst order ratio* $\mathrm{WR}_{\mathbb{A}}$ of an on-line algorithm $\mathbb{A}$ is the relative worst order ratio of $\mathbb{A}$ to an optimal off-line algorithm OPT, i.e., $\mathrm{WR}_{\mathbb{A}} = \mathrm{WR}_{\mathbb{A},\mathrm{OPT}}$. $\square$

Note that for many problems, the worst order ratio is the same as the competitive ratio, since the order in which requests arrive does not matter for an optimal off-line algorithm. However, for the Fair Bin Packing Problem [1] where the algorithms are required to be fair, the order does matter, even for OPT. The same is true for bounded space bin packing [9] where only a limited number of bins are allowed open at one time.

Clearly, the worst order ratio is never worse than the competitive ratio. However, for the Dual Bin Packing Problem, the result for the class of fair algorithms is as negative as with the competitive ratio; any fair algorithm has a worst order ratio of $O(s)$, where s is the smallest possible item size [3].

3 Classical Bin Packing

The Classical Bin Packing Problem is a minimization problem, so algorithms which do well compared to others have relative worst order ratios of less than 1 to the poorer algorithms.

Harmonic(k) can perform worse than First-Fit on sequences which have very small items. The statement of the following lemma gives a tight result using some very small items, but the proof also demonstrates what happens for slightly larger items.

Lemma 1 For any $k \geq 1$, there exists a family of multi-sets I_n such that

$$\textsc{Harmonic}(k)_W(I_n) \;\geq\; \frac{k}{k-1}\mathrm{FF}_W(I_n) - \frac{k+1}{k-1} \text{ for all } n$$

and $\lim_{n\to\infty} \mathrm{FF}_W(I_n) = \infty$.

Proof Let $\varepsilon < \frac{1}{kn}$. The sequence I_n consists of the following repeated n times: $k-1$ items of size $\frac{1}{k}$, followed by one item of size ε. All of these items will be in the same class for Harmonic(k), so they will be packed using Next-Fit, which uses n bins. First-Fit, on the other hand, combines the small items, so if ε is sufficiently small,

$$\mathrm{FF}_\mathrm{W}(I_n) \;=\; \left\lceil n\left(\frac{k-1}{k}+\varepsilon\right)\right\rceil \;<\; \frac{k-1}{k}n + \frac{k+1}{k}.$$

Thus,

$$\frac{k}{k-1}\mathrm{FF}_\mathrm{W}(I_n) - \frac{k+1}{k-1} \;<\; n \;=\; \textsc{Harmonic}(k)_\mathrm{W}(I_n).$$

Note that even if items as small as $\frac{1}{nk}$ cannot occur, we can obtain a ratio larger than 1. Let $\varepsilon \leq \frac{1}{2k}$, so that at least two of the small items can be combined with $k-1$ items of size $\frac{1}{k}$. In First-Fit's packing, the empty space in each bin, except for possibly one, will have size less than ε. Thus,

$$\mathrm{FF}^n_\mathrm{W}(I_n) \leq \left\lceil n\left(\frac{k-1}{k}+\varepsilon\right) + \varepsilon\mathrm{FF}^n_\mathrm{W}(I_n)\right\rceil.$$

We solve for n which is the number of bins used by Harmonic(k).

$$\begin{aligned}
\mathrm{FF}_\mathrm{W}(I_n) \;&\leq\; n\left(\frac{k-1}{k}+\varepsilon\right) + \varepsilon\mathrm{FF}_\mathrm{W}(I_n) + 1 \Leftrightarrow \\
(1-\varepsilon)\mathrm{FF}_\mathrm{W}(I_n) - 1 \;&\leq\; n\,\frac{k-1+k\varepsilon}{k} \qquad\qquad \Leftrightarrow \\
\frac{k-k\varepsilon}{k-1+k\varepsilon}\mathrm{FF}_\mathrm{W}(I_n) - \frac{k}{k-1+k\varepsilon} \;&\leq\; n \;=\; \textsc{Harmonic}(k)_\mathrm{W}(I_n)
\end{aligned}$$

□

The result of Lemma 1 is tight up to an additive constant.

Lemma 2 For any sequence, I,

$$\textsc{Harmonic}(k)_W(I) \leq \frac{k}{k-1}\mathrm{FF}_W(I) + k + 1.$$

Proof Consider $\textsc{Harmonic}(k)$'s packing of I. Let L_j, for $1 \leq j \leq k-1$, be the set of items from I in class C_j which $\textsc{Harmonic}(k)$ places in bins with fewer than j items in all, and let C'_j be the remaining items in C_j. Initially, let C'_k be the items in class C_k, except for those put in the last bin of that class, and let L_k contain the items from this last bin for items in class C_k. Consider First-Fit's packing of the sequence consisting only of those items in C'_k in the order in which they appear in I. Remove those items from C'_k which First-Fit places in its last bin and place them in L_k instead. Let $L = \cup_{i=1}^{k} L_i$ and $C' = \cup_{i=1}^{k-1} C'_i$. Thus, I consists of the items in L, C' and C'_k.

Create a sequence I' beginning with the items in C'_k in the order in which they appear in I, followed by the items from C' in nondecreasing order, and then those from L in any order. When First-Fit packs I', none of the items in C' will fit in any of the bins First-Fit uses for C'_k, so all of the items from C' in any class C_j will be packed j per bin. Thus, First-Fit and $\textsc{Harmonic}(k)$ will use the same number of bins for the items in C'. Suppose C'_k is nonempty, and the items in C'_k were placed in l bins by $\textsc{Harmonic}(k)$. Since the items in class C_k have size at most $\frac{1}{k}$ and since there are items in C_k which did not fit in the l bins used for C'_k, $\textsc{Harmonic}(k)$ fills each of those l bins to more than $1 - \frac{1}{k}$. Hence, the sum of the sizes in C'_k is at least $l(\frac{k-1}{k})$, so First-Fit must use at least $l(\frac{k-1}{k})$ bins for the items in C'_k. The result follows since $\textsc{Harmonic}(k)$ uses at most $k+1$ bins for the items in L. $\square$

Thus, for reasonably large k, $\textsc{Harmonic}(k)$ never does much worse than First-Fit. In fact, however, if all of the items in a sequence have sizes greater than $\frac{1}{k+1}$, then $\textsc{Harmonic}(k)$ does at least as well as First-Fit, except for a possible additive constant.

Lemma 3 For any sequence, I, where all items have size greater than $\frac{1}{k+1}$,

$$\textsc{Harmonic}(k)_W(I) \leq \mathrm{FF}_W(I) + k.$$

Proof The proof is a simplification of the proof of Lemma 2. The set C'_k only has items which are larger than $\frac{1}{k+1}$, so they can be put in C', since First-Fit and $\textsc{Harmonic}(k)$ will both put exactly k of them in each bin. In addition, the items moved from C'_k to L can be put in C'. Thus, $\textsc{Harmonic}(k)$ can use more bins than First-Fit only for the items in L, and there are at most k such bins. $\square$

Furthermore, on these sequences without very small items, $\textsc{Harmonic}(k)$ does strictly better than First-Fit with respect to the relative worst order ratio.

Theorem 1 For any fixed $k \geq 2$, for the Classical Bin Packing Problem restricted to sequences where all items have size greater than $\frac{1}{k+1}$,

$$\mathrm{WR}_{\mathrm{FF},\textsc{Harmonic}(k)} = \frac{6}{5}.$$

Proof By lemma 3, $\mathrm{WR}_{\mathrm{FF},\textsc{Harmonic}(k)} \geq 1$. For the lower bound, we consider the family of sequences, I_n, containing $6n$ items of size $\frac{1}{2}$ and $6n$ of size $\frac{1}{3}$, where the items of size $\frac{1}{2}$ and of size $\frac{1}{3}$ alternate. First-Fit uses $6n$ bins to pack I_n. $\textsc{Harmonic}(k)$ places items of sizes $\frac{1}{2}$ and $\frac{1}{3}$ in different bins since they are in different classes and thus uses only $5n$ bins.

To show the upper bound, we use a result from [6] showing that for the lazy bin packing problem, First-Fit-Increasing (FFI) has an approximation ratio of $\frac{6}{5}$. For that problem, the goal is to use as many bins as possible, subject to the restriction that no item in a later bin would fit in an earlier bin. This means that OPT packs a sequence I as First-Fit packs its worst ordering of I. Both $\textsc{Harmonic}(k)$ and FFI pack most of the items in any class C_j with j per bin, though FFI may use as many as $k-1$ fewer bins since it does not always start a new bin for a new class. $\square$

The idea behind the variants of Harmonic(k) defined in [13] and [14] is to sub-partition some of the classes of Harmonic(k) further, and then use some of the empty space which would necessarily occur in bins reserved for intervals with a right endpoint that cannot be expressed as $\frac{1}{j}$ for any integer j. A certain fraction of the items from some classes are thus designated to be placed in bins which are primarily reserved for other classes.

For example, in Refined Harmonic [13], two of the intervals defining classes are $J_a = (\frac{1}{2}, \frac{59}{96}]$ and $J_b = (\frac{1}{3}, \frac{37}{96}]$. The third item with size in J_b and every seventh after that is placed in a bin reserved for class J_a items. Thus, for a sequence consisting of n items of size $s \in J_b$, Refined Harmonic will place approximately $\frac{6n}{7}$ items with two per bin and $\frac{n}{7}$ with one per bin, using $\frac{4n}{7}$ bins asymptotically. First-Fit or Harmonic(k), on the other hand, would pack them all two per bin using only $\frac{n}{2}$ bins. This gives a ratio of $\frac{8}{7}$ in First-Fit's (or Harmonic(k)'s) favor. The family of multi-sets showing that Harmonic(k) is better than First-Fit contained only items of sizes $\frac{1}{2}$ and $\frac{1}{3}$, so Refined Harmonic would do the same as Harmonic(k) on that sequence, thus out-performing First-Fit by a factor $\frac{6}{5}$. This shows the following:

Theorem 2 Refined Harmonic and First-Fit are incomparable.

There also exist multi-sets where Refined Harmonic does better than Harmonic(k). Consider the family of multi-sets I_n consisting of $2n$ items in class J_a and $14n$ items in class J_b. Regardless of the order, Harmonic(k) for $k \geq 2$ will place the items in class J_a one per bin, and those in class J_b two per bin, using $9n$ bins in all. The algorithm Refined Harmonic will place $12n$ of the items in class J_b two per bin, but place the others with an item from class J_a, using $8n$ bins. This gives a factor of $\frac{9}{8}$ in Refined Harmonic's favor, showing:

Theorem 3 Refined Harmonic and Harmonic(k) are incomparable for $k \geq 2$.

Similar arguments show that any instance of Seiden's Super Harmonic algorithms [14], including Harmonic++, are incomparable to both First-

Fit and HARMONIC(k). Thus, depending on the application and expected request sequence distribution, either HARMONIC(k) or HARMONIC++ could be the better algorithm.

4 Dual Bin Packing

When switching to the Dual Bin Packing Problem, which is a maximization problem, algorithms which do well compared to others have relative worst order ratios greater than 1, instead of less than 1, to the poorer algorithms.

It was shown in [3] that First-Fit is best possible among Any-Fit algorithms and incomparable to the only algorithm considered which was not fair. There is, however, an algorithm which does strictly better than First-Fit with respect to the relative worst order ratio. Recall that First-Fit^n (FF^n) is the algorithm which behaves exactly as First-Fit would unless the item x is larger than $\frac{1}{2}$ and would be placed in the last bin, bin n. First-Fit^n rejects such an item and is thus not fair. Intuitively, First-Fit^n partially avoids a major pit-fall with respect to First-Fit's performance: large items at the beginning of the sequence can cause the rejection of many small later items. Clearly, the constant $\frac{1}{2}$ in the definition of First-Fit^n could be changed to something smaller if this seems appropriate for the given application.

Theorem 4 For any constant $c > 1$, $\mathrm{WR}_{\mathrm{FF}^n,\mathrm{FF}} > c$.

Proof Clearly, on any sequence First-Fit accepts at most one more item than First-Fit^n, an item of size greater than $\frac{1}{2}$ which First-Fit puts in the last bin, so $\mathrm{WR}_{\mathrm{FF}^n,\mathrm{FF}} \geq 1$.

Define I_c to be the sequence containing n items of size 1 followed by cn items of size $\frac{1}{cn}$. First-Fit accepts only n items, while FF^n accepts $n-1$ items of size 1 and all cn items of size $\frac{1}{cn}$, regardless of the ordering. Thus, for every sequence I_c,

$$\mathrm{FF}^n(I_c) = cn + n - 1 = (c+1)\mathrm{FF}(I_c) - 1.$$

□

It was shown in [5] that no deterministic or randomized algorithm for Dual Bin Packing is competitive. For completeness, we give a simpler, direct proof to show that First-Fit^n is not competitive.

Theorem 5 First-Fit^n is not competitive.

Proof Let $0 < \varepsilon \leq \frac{1}{2}$. An adversary gives the following request sequence, divided into three phases:

1. $n-1$ items of size 1;
2. 2 items of size $\frac{1}{2}$;
3. $n \cdot \lfloor \frac{1}{\varepsilon} \rfloor$ items of size ε.

First-Fit^n accepts the first two phases, since the second phase is designed for the last bin. An optimal off-line algorithm accepts only the requests in phase 3, giving a performance ratio of approximately ε. $\square$

To show that First-Fit^n has the same competitive ratio on accommodating sequences as First-Fit asymptotically, we use the sequences from [1] showing that First-Fit's competitive ratio on accommodating sequences is asymptotically $\frac{5}{8}$.

Theorem 6 If the given number of bins, n, is of the form $n = 9 \cdot 2^q - 5$ for some positive integer q, then First-Fit^n's competitive ratio on accommodating sequences is at most $\frac{5}{8} + O(\frac{1}{n})$.

Proof Let $\varepsilon > 0$ be small enough. An adversary can give the following sequence, I_q, of items, divided into $q+3$ phases:

Phase 1.	3 items of size $A = \frac{1}{3} - 2^{3q}\varepsilon$.
Phases $2 \dots (q+1)$.	In phase $j+1$ $(1 \le j \le q)$, $3 \cdot 2^j$ pairs, each with one item of size $B_j = \frac{1}{3} + 2^{3q-3j+2}\varepsilon$ followed by an item of size $C_j = \frac{1}{3} - 2^{3q-3j}\varepsilon$.
Phase $q+2$.	$3 \cdot 2^q$ items of size $D = \frac{2}{3} + \varepsilon$.
Phase $q+3$.	$9 \cdot 2^q - 6$ items of size $E = \frac{1}{3}$.

In each phase $j+1$, $1 \le j \le q$, First-Fit^n will pair each item of size B_j with one item of size C_j, thus using $3 \cdot (2^{q+1} - 2) + 1 = 6 \cdot 2^q - 5$ bins for these q phases and the first phase. After this, the first $3 \cdot 2^q - 1$ items of phase $q+2$ will be packed in separate bins, giving a total of $n-1$ used bins. Since the last item of phase $q+2$ is larger than $\frac{1}{2}$, it will be rejected and three of the items from phase $q+3$ will be packed in the last bin. The remaining $9 \cdot 2^q - 3$ from phase $q+3$ items do not fit in any bin and are rejected. This gives a total of $3 + 2 \cdot 3 \cdot (2^{q+1} - 2) + (3 \cdot 2^q - 1) + 3 = 15 \cdot 2^q - 7$ accepted items.

OPT, on the other hand, pairs each item from phase 1 with two items of size B_1 and each item of size C_j, $1 \le j \le q-1$, with two items of size B_{j+1}. Each item of size C_q is paired with an item from phase $q+2$. This uses $3 \cdot (2^{q+1} - 1) = 6 \cdot 2^q - 3$ bins, leaving room for all of the items of phase $q+3$. Thus, OPT accepts all $(15 \cdot 2^q - 9) + (9 \cdot 2^q - 6) = 24 \cdot 2^q - 15$ items.

This gives a ratio of

$$\begin{aligned}
\frac{\mathrm{FF}^n(I_q)}{\mathrm{OPT}(I_q)} &= \frac{15 \cdot 2^q - 7}{24 \cdot 2^q - 15} = \frac{5 \cdot (3 \cdot 2^q - \frac{7}{5})}{8 \cdot (3 \cdot 2^q - \frac{15}{8})} \\
&= \frac{5 \cdot (3 \cdot 2^q - \frac{15}{8}) + 5 \cdot (\frac{15}{8} - \frac{7}{5})}{8 \cdot (3 \cdot 2^q - \frac{15}{8})} \\
&\in \frac{5}{8} + O\left(\frac{1}{n}\right).
\end{aligned}$$

□

Theorem 6 considers special values of n. As in [1], it can be shown that, in general, First-Fit^n's competitive ratio on accommodating sequences is $\frac{5}{8} + O(\frac{1}{\sqrt{n}})$.

5 Conclusion

The investigation of the relative worst order ratio, as applied to bin packing, has been continued. Using the relative worst order ratio, Harmonic(k) has been compared to First-Fit, which is theoretically the best Any-Fit algorithm for the Classical Bin Packing Problem, according to both the competitive ratio and the relative worst order ratio. Harmonic(k) was found to be incomparable to First-Fit, but this was only the case if the sequences had very small items. When First-Fit was better than Harmonic(k), it was only by a factor of at most $\frac{k}{k-1}$. For sequences where all items are larger than $\frac{1}{k+1}$, the relative worst order ratio shows that Harmonic(k) is better than First-Fit by a factor of $\frac{6}{5}$. This is consistent with the results obtained using the competitive ratio, showing that Harmonic(k) is the better algorithm asymptotically.

The variants of Harmonic(k), Refined Harmonic and Harmonic++, were found to be incomparable to First-Fit and Harmonic(k) using the relative worst order ratio. This differs from the results using the competitive ratio, where these variants are strictly better. This means that, depending on the expected distribution of the request sequence, either Harmonic(k) or Harmonic++ may have the better performance. In the case, for example, where one expects no items of size greater than $\frac{1}{2}$, there is no reason to use Refined Harmonic, since Harmonic(k) will perform at least as well and possibly better.

With respect to the Dual Bin Packing Problem, the relative worst order ratio has inspired the discovery of a new algorithm, First-Fit^n. This new algorithm partially avoids a problem with fair algorithms, that large earlier items could cause the rejection of many later, small items. According to the relative worst order ratio, First-Fit^n is unboundedly better than First-Fit, though it cannot be distinguished from First-Fit using either the competitive ratio or the competitive ratio on accommodating sequences.

Thus, this second investigation of the relative worst order ratio supports the promising results of the first [3]. The new performance measure gives the advantages that one can compare two on-line algorithms directly, that it is intuitively suitable for some natural problems where any ordering of the input is equally likely, and that it is easier to compute than the random order ratio.

Work in progress [4] shows that for the paging problem, the relative worst order ratio of LRU (FIFO) to LRU (FIFO) with lookahead l is $\min(k, l+1)$ when there are k pages in fast memory. This compares well with the competitive ratio, where these algorithms have the same competitive ratio. This result is similar to that which Koutsoupias and Papadimitriou obtained using comparative analysis [12], and stronger than that obtained with the Max/Max ratio [2]. The relative worst order ratio should be applied to other on-line problems to see if it is also useful for those problems.

References

[1] Y. Azar, J. Boyar, L. Epstein, L. M. Favrholdt, K. S. Larsen, and M. N. Nielsen. Fair versus Unrestricted Bin Packing. *Algorithmica*, 34(2):181–196, 2002.

[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. In *5th Italian Conference on Algorithms and Complexity*, volume 2653 of *LNCS*, pages 58–69, 2003.

[4] J. Boyar, L. M. Favrholdt, and K. S. Larsen. Work in progress.

[5] J. Boyar, L. M. Favrholdt, K. S. Larsen, and Morten N. Nielsen. The Competitive Ratio for On-Line Dual Bin Packing with Restricted Input Sequences. *Nordic Journal of Computing*, 8(4):463–472, 2001.

[6] J. Boyar, J. S. Frederiksen, K. S. Larsen, M. M. Pedersen, and S. Wøhlk. Work in progress.

[7] J. Boyar, K. S. Larsen, and M. N. Nielsen. The Accommodating Function — a Generalization of the Competitive Ratio. *SIAM Journal of Computation*, 31(1):233–258, 2001.

[8] R. L. Graham. Bounds for Certain Multiprocessing Anomalies. *Bell Systems Technical Journal*, 45:1563–1581, 1966.

[9] D. S. Johnson. Fast Algorithms for Bin Packing. *Journal of Computer and System Sciences*, 8:272–314, 1974.

[10] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive Snoopy Caching. *Algorithmica*, 3(1):79–119, 1988.

[11] C. Kenyon. Best-Fit Bin-Packing with Random Order. In *7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 359–364, 1996.

[12] E. Koutsoupias and C. H. Papadimitriou. Beyond Competitive Analysis. In *35th Annual Symposium on Foundations of Computer Science*, pages 394–400, 1994.

[13] C. Lee and D. Lee. A simple on-line bin-packing algorithm. *Journal of the ACM*, 32(3):562–572, 1985.

[14] S. S. Seiden. On the Online Bin Packing Problem. *Journal of the ACM*, 5:640–671, 2002.

[15] D. D. Sleator and R. E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communications of the ACM*, 28(2):202–208, 1985.