The Relative Worst Order Ratio for On-Line Algorithms

JOAN BOYAR and LENE M. FAVRHOLDT

We define a new measure for the quality of on-line algorithms, the *relative worst order ratio*, using ideas from the Max/Max ratio (Ben-David & Borodin 1994) and from the random order ratio (Kenyon 1996). The new ratio is used to compare on-line algorithms directly by taking the ratio of their performances on their respective worst permutations of a worst-case sequence.

Two variants of the bin packing problem are considered: the Classical Bin Packing problem, where the goal is to fit all items in as few bins as possible, and the Dual Bin Packing problem, which is the problem of maximizing the number of items packed in a fixed number of bins. Several known algorithms are compared using this new measure, and a new, simple variant of First-Fit is proposed for Dual Bin Packing.

Many of our results are consistent with those previously obtained with the competitive ratio or the competitive ratio on accommodating sequences, but new separations and easier proofs are found.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Sequencing and scheduling*

General Terms: Algorithms

Additional Key Words and Phrases: On-line, quality measure, relative worst order ratio, bin packing, dual bin packing

1. INTRODUCTION

The standard measure for the quality of on-line algorithms is the competitive ratio [Graham 1966; Sleator and Tarjan 1985; Karlin et al. 1988], which is, roughly speaking, the worst-case ratio, over all possible input sequences, of the on-line performance to the optimal off-line performance. The definition of the competitive ratio is essentially identical to 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 natural, since the approximation algorithm is compared 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. Intuitively, this could be because information is lost in the intermediate comparison to the much more powerful off-line algorithm. In a few cases (bin coloring [Krumke et al. 2001] and dual bin packing [Boyar et al. 2001]), one algorithm $\mathbb{A}$ even has a better competitive ratio than another algorithm $\mathbb{B}$, though intuitively, $\mathbb{B}$ is clearly better than $\mathbb{A}$.

Authors' address: J. Boyar and L.M. Favrholdt, Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark.

A preliminary version of some of these results appeared in the 5th Italian Conference on Algorithms and Complexity (CIAC 2003), *Lecture Notes in Computer Science* 2653: 58-69, Springer-Verlag, 2003.

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).

Part of this work was done while the first author was visiting the Computer Science Department, University of Toronto, and part was done while the second author was working at the Department of Computer Science, University of Copenhagen, Denmark.

The second author was supported in part by the Villum Kann Rasmussen Fund.

When differentiating between on-line algorithms is the goal, performing a direct comparison between the algorithms seems the obvious choice. A direct comparison on exactly the same sequences will produce the result that many algorithms are not comparable, because one algorithm does well on one type of sequence, 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 permutations of multi-sets. In this way, the relative worst order ratio combines some of the desirable properties of the Max/Max ratio [Ben-David and Borodin 1994] and the random order ratio [Kenyon 1996].

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. More specifically, comparing two on-line algorithms directly gives the same result as dividing their Max/Max ratios. 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.

In the paper [Ben-David and Borodin 1994] introducing the Max/Max ratio, Ben-David and Borodin demonstrate that, for the k-server problem on a bounded metric space, the Max/Max ratio can provide more optimistic and detailed results than the competitive ratio. However, for the paging problem, which is a special case of the k-server problem, the Max/Max ratio gives even less information than the competitive ratio; all deterministic algorithms have the same Max/Max ratio, and for an arbitrarily large slow memory, the Max/Max ratio of any algorithm (deterministic or randomized) is arbitrarily close to 1.

The Random Order Ratio

The random order ratio gives the possibility of considering some randomness of the input sequences without specifying a complete probability distribution. This ratio was introduced in connection with the Classical Bin Packing problem defined below. For an algorithm $\mathbb{A}$ for this problem, the random order ratio is the maximum ratio over all multi-sets of items of the expected performance of $\mathbb{A}$ compared with an optimal solution. If, for all possible multi-sets of items, any permutation of these items is equally likely, this ratio gives a meaningful worst-case measure of how well an algorithm can do. Unfortunately, the random order ratio seems to be difficult to compute.

In the paper [Kenyon 1996] introducing the random order ratio, Kenyon showed that for the Classical Bin Packing problem, the random order ratio of Best-Fit lies between 1.08 and 1.5. In contrast, the competitive ratio of Best-Fit is 1.7 [Johnson et al. 1974].

The Relative Worst Order Ratio

Combining the desirable properties of the Max/Max ratio and the random order ratio, we define the *relative worst order ratio*, where when comparing two on-line algorithms, we consider a worst-case sequence and take the ratio of how the two algorithms do on their worst permutations of that sequence. Note that the two algorithms may have different worst permutations for the same sequence.

Two algorithms $\mathbb{A}$ and $\mathbb{B}$ are said to be *incomparable* under the relative worst order ratio, if there exists a family of sequences where $\mathbb{A}$ is strictly better than $\mathbb{B}$ and another family of sequences where $\mathbb{B}$ is strictly better than $\mathbb{A}$. A formal definition is presented in Section 2. When this possibility occurs, it means to some extent that neither algorithm can be considered the better, as when the ratio is 1, but the incomparability result may include information as to when each algorithm is better.

The relative worst order ratio can be viewed as a worst case version of Kenyon's random order ratio, with the modification that on-line algorithms are compared directly, rather than indirectly through OPT. It can also be viewed as a modification of the Max/Max ratio, where a finer partition of the request sequences is used; instead of finding the worst sequence among those having the same length, one finds the worst sequence among those which are permutations of each other. This particular finer partition was inspired by the random order ratio. See Table 1 for

Measure	Value
Competitive Ratio	$\mathrm{CR}_{\mathbb{A}} = \max_I \frac{\mathbb{A}(I)}{\mathrm{OPT}(I)}$
Max/Max Ratio	$\mathrm{MR}_{\mathbb{A}} = \frac{\max_{\lvert I\rvert=n} \mathbb{A}(I)}{\max_{\lvert J\rvert=n} \mathrm{OPT}(J)}$
Random Order Ratio	$\mathrm{RR}_{\mathbb{A}} = \max_I \frac{E_\sigma\left[\mathbb{A}(\sigma(I))\right]}{\mathrm{OPT}(I)}$
Relative Worst Order Ratio	$\mathrm{WR}_{\mathbb{A}} = \max_I \frac{\max_\sigma\left\{\mathbb{A}(\sigma(I))\right\}}{\max_\sigma\left\{\mathbb{B}(\sigma(I))\right\}}$

Table I. Comparison of measures

a comparison of these different measures on minimization problems. Note that the ratios given in the table are not the exact definitions of the measures; they are all asymptotic measures but for simplicity, this is not reflected in the table. Thus, e.g., for the competitive ratio, the additive constant in the definition is ignored, so what the table shows is actually the strict competitive ratio.

The Worst Order Ratio

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*. Note that for many problems, the worst order ratio is the same as the competitive ratio, since the specific permutation of the input sequence does not matter for an optimal off-line algorithm. However, for the Fair Bin Packing problem mentioned below, the order does matter, even for OPT. The same is true for bounded space bin packing [Johnson 1974] where only a limited number of bins are allowed open at one time.

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 positive size, 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. A variant of Dual Bin Packing is the *Fair Bin Packing problem* [Boyar et al. 2001; Azar et al. 2002], where the algorithms have to be *fair*, i.e., to reject items only when they do not fit in any bin.

Other Measures

The results we obtain using the relative worst order ratio are compared to the standard competitive ratio [Graham 1966; Sleator and Tarjan 1985; Karlin et al. 1988] and to the competitive ratio on accommodating sequences [Boyar et al. 2001].

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 by any off-line algorithm. 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 a problem with some restricted resource, such as the bins in the Dual Bin Packing problem, accommodating sequences are defined. For the Dual Bin Packing problem, an accommodating sequence is a sequence of items that can be completely packed in the n bins by an optimal off-line algorithm. Thus, for the Dual Bin Packing problem, the *competitive ratio on accommodating sequences* is the same as the competitive ratio, except that the only input sequences considered are those for which all items could be packed in the n bins, so it is the worst case ratio over a restricted set of input sequences.

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 the Classical Bin Packing problem, Johnson [Johnson 1974] defined the class of *Any-Fit* algorithms, which use an empty bin, only if there is not enough space in any partially full bins. In [Boyar et al. 2001], this class of algorithms is defined for Dual Bin Packing: an algorithm is Any-Fit if it is fair and it uses an empty bin only if there is not enough space in any partially full bins (fairness does not apply to Classical Bin Packing where no items are ever rejected).

Classical Bin Packing. Three examples of Any-Fit algorithms for Classical Bin Packing are First-Fit, which places an item in the first bin in which it fits, Best-Fit, which places an item in one of the fullest bins in which it fits, and Worst-Fit, which will place an item in a least full open bin. First-Fit and Best-Fit both have competitive ratios of $\frac{17}{10}$ [Johnson et al. 1974], while Worst-Fit has a competitive ratio of 2 [Johnson 1974].

The algorithm, Next-Fit, which is not an Any-Fit algorithm also has a competitive ratio of 2 [Johnson 1974]. Next-Fit is the algorithm which first attempts to fit an item in the current bin, places it there if it fits, or opens a new bin if it does not.

First-Fit has the best competitive ratio of any Any-Fit algorithm [Johnson 1974][1], but other algorithms have better competitive ratios. Lee and Lee [Lee and Lee 1985] defined a family of algorithms Harmonic$_k$. These algorithms divide 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. Lee and Lee [Lee and Lee 1985] have shown that the competitive ratio of Harmonic$_k$ approaches about 1.691 as k approaches infinity. Lee and Lee [Lee and Lee 1985] also proposed the algorithm Refined Harmonic and showed that its competitive ratio is close to 1.636. Seiden [Seiden 2002] defined the class of Super Harmonic algorithms, which includes both Refined Harmonic and his own algorithm Harmonic++, which has an asymptotic performance ratio of 1.589.

Dual Bin Packing. First-Fit and Best-Fit are defined in the same manner for the Dual Bin Packing problem as for the Classical Bin Packing problem, though clearly they reject items which do not fit in any of the n bins. We use a different, more natural, definition for Worst-Fit for the fixed number of bins: Worst-Fit always places an item in a least full bin. For the first n items, the least full bin will be empty, so the Worst-Fit we consider for Dual Bin Packing is not an Any-Fit algorithm. (Note that this definition is not at all natural for the Classical Bin Packing problem, since a new bin would be opened for every item.)

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.

Our Results

Many results obtained here with the relative worst order ratio are consistent with those previously obtained with the competitive ratio or the competitive ratio on accommodating sequences, but new separations and easier proofs are also shown to be possible with the relative worst order ratio. Furthermore, a new algorithm is proposed.

Classical Bin Packing. With the relative worst order ratio, 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. First-Fit is still the best Any-Fit algorithm and Next-Fit is strictly worse than any Any-Fit algorithm.

We show that according to the relative worst order ratio, if the sequences can have arbitrarily small items, Harmonic$_k$ is incomparable to any Any-Fit algorithm. However, when Harmonic$_k$

[1] Johnson attributes this result to private communication with A. Demers.

is worse than some Any-Fit algorithm, 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 any Any-Fit algorithm is strictly worse than HARMONIC$_k$, by a factor of at least $\frac{6}{5}$. For First-Fit, the ratio of $\frac{6}{5}$ is tight.

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 1.589 worse than OPT. However, these variants are designed to do better than HARMONIC$_k$ and Any-Fit algorithms on these algorithms' worst-case sequences. The price to pay is that they do worse than HARMONIC$_k$ and Any-Fit algorithms on some other sequences. We show that these variants of HARMONIC$_k$ are incomparable to HARMONIC$_k$ and to any Any-Fit algorithm, using the relative worst order ratio. Thus, depending on the application and expected input sequence distribution, either HARMONIC$_k$ or HARMONIC++ could be the better algorithm.

Dual Bin Packing. All deterministic algorithms for Dual Bin Packing and Fair Bin Packing have a worst order ratio of 0. These extremely pessimistic results are consistent with the results [Boyar et al. 2001; Boyar et al. 2001] showing that no such algorithm is competitive. Using the *relative* worst order ratio, however, we find that meaningful results can be obtained: 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 one obtains the counterintuitive result that, for the restricted problem where all item sizes are multiples of some constant f, Worst-Fit is better than First-Fit [Boyar et al. 2001] (without this restriction, neither of them is competitive). The competitive ratio on accommodating sequences also indicates that First-Fit is better than Worst-Fit [Boyar et al. 2001], but the proof using the relative worst order ratio is much easier.

Unfair-First-Fit [Azar et al. 2002] is a variant of First-Fit designed to do better than First-Fit on certain sequences containing many large items followed by many small items. It has a better competitive ratio on accommodating sequences than First-Fit, but on sequences containing only large items it actually does worse than First-Fit. Under the relative worst order ratio, Unfair-First-Fit is incomparable to all Any-Fit algorithms.

We propose a simple variant of First-Fit called First-Fit^n. This new algorithm partially avoids the 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 any Any-Fit algorithm, though it cannot be distinguished from First-Fit using either the competitive ratio or the competitive ratio on accommodating sequences.

More Recent Results on the Relative Worst Order Ratio

Since these first results applying the relative worst order ratio to bin packing problems, the measure has been applied to other problems.

For the paging problem, one might not expect very useful results, since for this problem, all permutations of a sequence are generally not equally likely. However, even for this problem, there are several meaningful results concerning the relative worst order ratio [Boyar et al. 2005]. A new deterministic paging algorithm, Retrospective-LRU, is proposed and shown to perform better than LRU (Least Recently Used). This is supported by experimental results, but contrasts with the competitive ratio. LRU is better than FWF (Flush-When-Full), though all deterministic marking algorithms have the same competitive ratio. In addition, look-ahead is shown to be a significant advantage, whereas the competitive ratio does not reflect that look-ahead can be helpful.

For the problem of minimizing makespan on two related machines with speed ratio s, the optimal competitive ratio of $\frac{s+1}{s}$ for $s \geq \Phi \approx 1.618$ is obtained both by the post-greedy algorithm, which schedules each job on the machine where it will finish earliest, and by the algorithm which simply schedules all jobs on the fast machine. In contrast, the relative worst order ratio shows that the post-greedy algorithm is better [Epstein et al. 2004]. A similar result is obtained for the problem of minimizing makespan on $m \geq 2$ identical machines with preemption [Epstein et al. 2004].

The relative worst order ratio was also found by [Epstein et al. 2004] to give the intuitively

correct result for the bin coloring problem, where the competitive ratio gives the opposite result [Krumke et al. 2001]: a trivial algorithm using only one open bin has a better competitive ratio than a natural greedy-type algorithm.

The proportional price version of the seat reservation problem has largely been ignored due to very negative impossibility results using competitive analysis: even on accommodating sequences, all deterministic algorithms have a competitive ratio of $\Theta(\frac{1}{k})$, where k is the number of stations [Boyar and Larsen 1999]. However, in [Boyar and Medvedev 2004], algorithms for the problem were compared and separated using the relative worst order ratio.

2. THE (RELATIVE) WORST ORDER RATIO

In this section we formally define the *worst order ratio* as well as the *relative worst order ratio*. We define the relative worst order ratio first. For this we need $\mathbb{A}_{\mathrm{W}}(I)$, the performance of an algorithm $\mathbb{A}$ on the "worst permutation" of the input sequence I, formally defined in the following way.

Definition 2.1. Consider an optimization 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 running the algorithm $\mathbb{A}$ 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 $\mathbb{A}_{\mathrm{W}}(I) = \max_\sigma \mathbb{A}(\sigma(I))$.

For many on-line problems, some algorithms perform well on particular types of inputs, while other algorithms perform well on other types of inputs. The purpose of comparing on the worst permutation of sequences, rather than on each sequence independently, is to be able to differentiate between such pairs of algorithms, rather than just concluding that they are incomparable. Sequences with the same "content" are considered together, but the measure is worst case, so the algorithms are compared on their respective worst permutations. This was originally motivated by problems, such as many bin packing applications, where all orderings are equally likely, but appears to be applicable to many other problems as well [Boyar et al. 2005; Boyar and Medvedev 2004; Epstein et al. 2004].

Definition 2.2. Let $S_1(c)$ and $S_2(c)$ be statements about algorithms $\mathbb{A}$ and $\mathbb{B}$ defined in the following way.

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

The *relative worst order ratio* $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ of 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*.

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

The statements $S_1(1)$ and $S_2(1)$ check that one algorithm is always at least as good as the other on every sequence (on their respective worst permutations). When one of them holds, the relative worst order ratio is a bound on how much better that algorithm can be. 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 II. Note that if $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ and $\mathrm{WR}_{\mathbb{B},\mathbb{A}}$ are bounded above by some constant, then $\mathrm{WR}_{\mathbb{A},\mathbb{B}} = 1/\mathrm{WR}_{\mathbb{B},\mathbb{A}}$. If $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ is unbounded, then $\mathrm{WR}_{\mathbb{B},\mathbb{A}}$ is not bounded away from zero, and vice-versa.

Although not all pairs of algorithms are comparable with the relative worst order ratio, the following theorem implies that, for algorithms which are comparable, the measure is transitive,

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

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

i.e., if $\mathbb{A}$ is better than $\mathbb{B}$ and $\mathbb{B}$ is better than $\mathbb{C}$, then $\mathbb{A}$ is better than $\mathbb{C}$.

THEOREM 2.3. *The ordering of algorithms for a specific problem is transitive. Moreover, if* $WR_{\mathbb{A},\mathbb{B}} \geq 1$ *and* $WR_{\mathbb{B},\mathbb{C}} \geq 1$*, then* $WR_{\mathbb{A},\mathbb{C}} \geq WR_{\mathbb{B},\mathbb{C}}$*. If furthermore* $WR_{\mathbb{A},\mathbb{B}}$ *is bounded above by some constant, then* $WR_{\mathbb{A},\mathbb{C}} \geq WR_{\mathbb{A},\mathbb{B}}$*. Similarly, if* $WR_{\mathbb{A},\mathbb{B}} \leq 1$ *and* $WR_{\mathbb{B},\mathbb{C}} \leq 1$*, then* $WR_{\mathbb{A},\mathbb{C}} \leq \min(WR_{\mathbb{A},\mathbb{B}}, WR_{\mathbb{B},\mathbb{C}})$*.*

PROOF. Suppose that three algorithms $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$ for some problem are such that $\mathrm{WR}_{\mathbb{A},\mathbb{B}} \geq 1$ and $\mathrm{WR}_{\mathbb{B},\mathbb{C}} \geq 1$. Then there exists a constant a such that for all I, $\mathbb{A}_W(I) \geq \mathbb{B}_W(I) - a$, and there exists a constant b such that for all I, $\mathbb{B}_W(I) \geq \mathbb{C}_W(I) - b$. Hence, there exist constants a and b such that for all I, $\mathbb{A}_W(I) \geq \mathbb{C}_W(I) - (a+b)$. Thus, $\mathrm{WR}_{\mathbb{A},\mathbb{C}} \geq 1$. This proves that the measure is transitive.

Moreover, for any I and any r such that $\mathbb{B}_{\mathrm{W}}(I) \geq r\,\mathbb{C}_{\mathrm{W}}(I) - d$,

$$\mathbb{A}_{\mathrm{W}}(I) \;\geq\; \mathbb{B}_{\mathrm{W}}(I) - a \;\geq\; r\,\mathbb{C}_{\mathrm{W}}(I) - (d+a)\,.$$

Therefore, since $\mathrm{WR}_{\mathbb{A},\mathbb{C}} \geq 1$, an adversary argument constructing sequences to prove a lower bound of r on $\mathrm{WR}_{\mathbb{B},\mathbb{C}}$ would also give a lower bound of r on $\mathrm{WR}_{\mathbb{A},\mathbb{C}}$. Thus, $\mathrm{WR}_{\mathbb{A},\mathbb{C}} \geq \mathrm{WR}_{\mathbb{B},\mathbb{C}}$.

Similarly, if $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ is bounded above by a constant, $\mathrm{WR}_{\mathbb{A},\mathbb{C}} \geq \mathrm{WR}_{\mathbb{A},\mathbb{B}}$, since $\mathbb{A}_{\mathrm{W}}(I) \geq r\,\mathbb{B}_{\mathrm{W}}(I) - d$ implies that there exist a constant b such that

$$\mathbb{A}_{\mathrm{W}}(I) \;\geq\; r\,\mathbb{B}_{\mathrm{W}}(I) - d \;\geq\; r\,\mathbb{C}_{W}(I) - (rb+d).$$

The arguments for the case where $\mathrm{WR}_{\mathbb{A},\mathbb{B}} \leq 1$ and $\mathrm{WR}_{\mathbb{B},\mathbb{C}} \leq 1$ are essentially the same, but there is no problem if $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ is not bounded away from zero; the value r simply makes the additive constant smaller. □

This transitivity implies that when a new algorithm is analyzed, it is unnecessary to compare it to all previously analyzed algorithms. Note that in the case where both ratios are constants greater than or equal to one, an upper bound of $\mathrm{WR}_{\mathbb{A},\mathbb{C}} \leq \mathrm{WR}_{\mathbb{A},\mathbb{B}} \cdot \mathrm{WR}_{\mathbb{B},\mathbb{C}}$ follows using essentially the same argument as above. If both ratios are at most one, a lower bound of $\mathrm{WR}_{\mathbb{A},\mathbb{C}} \geq \mathrm{WR}_{\mathbb{A},\mathbb{B}} \cdot \mathrm{WR}_{\mathbb{B},\mathbb{C}}$ holds. However, it is not necessarily the case that $\mathrm{WR}_{\mathbb{A},\mathbb{C}} = \mathrm{WR}_{\mathbb{A},\mathbb{B}} \cdot \mathrm{WR}_{\mathbb{B},\mathbb{C}}$, since the family of sequences giving the lower bound on $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ may not be the same as the family of sequences giving the lower bound on $\mathrm{WR}_{\mathbb{B},\mathbb{C}}$.

Finally we define the worst order ratio formally:

Definition 2.4. The *worst order ratio* $\mathrm{WR}_{\mathbb{A}}$ of an algorithm $\mathbb{A}$ is the relative worst order ratio of $\mathbb{A}$ to an optimal off-line algorithm OPT, i.e., $\mathrm{WR}_{\mathbb{A}} = \mathrm{WR}_{\mathbb{A},\mathrm{OPT}}$.

As mentioned in the introduction, if there is no restriction on the behavior of OPT, the worst order ratio is the same as the competitive ratio. This does not necessarily hold for the Fair Bin Packing problem or bounded space bin packing. Clearly, the worst order ratio is never worse than the competitive ratio.

2.1 The *Strict* Relative Worst Order Ratio

One reason for the constant in the definition of the relative worst order ratio is to ensure that some short sequence, I, where, e.g., $\mathbb{A}_{\mathrm{W}}(I) = \mathbb{B}_{\mathrm{W}}(I) + b$, for some small constant b, does not give a large separation for $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$. This constant in the definition forces one to find an infinite family of sequences to prove a separation. However, when analyzing the Dual Bin Packing problem, one would always find a family of sequences to show a separation, since the result must hold for an arbitrarily large number of bins; a single, short sequence cannot give separation results for the problem. Thus, motivated by its use for the Dual Bin Packing problem, we define the *strict relative worst order ratio*, which is identical to the relative worst order ratio except that

the additive constants in the definitions of $S_1(c)$ and $S_2(c)$ are removed. This corresponds to the strict competitive ratio's relation to the competitive ratio.

Definition 2.5. Let $\mathbb{A}$ and $\mathbb{B}$ be algorithms for some optimization problem. The *strict relative worst order ratio* of $\mathbb{A}$ to $\mathbb{B}$, $\mathrm{WR}'_{\mathbb{A},\mathbb{B}}$, is defined in the following way.

$$\text{If } \mathbb{A}_{\mathrm{W}}(I) \leq \mathbb{B}_{\mathrm{W}}(I) \text{ for all } I, \text{ then } \mathrm{WR}'_{\mathbb{A},\mathbb{B}} = \sup\{r \mid \mathbb{A}_{\mathrm{W}}(I) \geq r\, \mathbb{B}_{\mathrm{W}}(I) \text{ for all } I\}.$$
$$\text{If } \mathbb{A}_{\mathrm{W}}(I) \geq \mathbb{B}_{\mathrm{W}}(I) \text{ for all } I, \text{ then } \mathrm{WR}'_{\mathbb{A},\mathbb{B}} = \inf\{r \mid \mathbb{A}_{\mathrm{W}}(I) \leq r\, \mathbb{B}_{\mathrm{W}}(I) \text{ for all } I\}.$$

Note that results that bound the strict relative worst order ratio, $\mathrm{WR}'_{\mathbb{A},\mathbb{B}}$, *towards* 1 are also valid for the relative worst order ratio, whereas results that bound $\mathrm{WR}'_{\mathbb{A},\mathbb{B}}$ *away* from 1 (separation results for the algorithms $\mathbb{A}$ and $\mathbb{B}$) do not automatically carry over. On the other hand, results that bound the relative worst order ratio, $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$, *away* from 1 carry over to the strict relative worst order ratio, whereas results that bound $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ *towards* 1 hold automatically for the strict relative worst order ratio only if the additive constant is zero. This is analogous to the relationship between the competitive ratio and the strict competitive ratio, since results that bound $\mathrm{WR}'_{\mathbb{A},\mathbb{B}}$ away from 1 are negative results on the performance of algorithm $\mathbb{A}$, whereas results that bound $\mathrm{WR}'_{\mathbb{A},\mathbb{B}}$ towards 1 are positive results on the performance of $\mathbb{A}$.

For the strict relative worst order ratio, we can prove a result slightly stronger than that of Theorem 2.3.

THEOREM 2.6. *The ordering of algorithms for a specific problem is transitive.*
Moreover, if $WR'_{\mathbb{A},\mathbb{B}} \geq 1$ *and* $WR'_{\mathbb{B},\mathbb{C}} \geq 1$, *then* $WR'_{\mathbb{A},\mathbb{C}} \geq \max\{WR'_{\mathbb{A},\mathbb{B}}, WR'_{\mathbb{B},\mathbb{C}}\}$.
Similarly, if $WR'_{\mathbb{A},\mathbb{B}} \leq 1$ *and* $WR'_{\mathbb{B},\mathbb{C}} \leq 1$, *then* $WR'_{\mathbb{A},\mathbb{C}} \leq \min\{WR'_{\mathbb{A},\mathbb{B}}, WR'_{\mathbb{B},\mathbb{C}}\}$.

PROOF. The proof is a simplification of the proof of Theorem 2.3. □

3. CLASSICAL BIN PACKING

The Classical Bin Packing problem is a minimization problem, so the relative worst order ratio of $\mathbb{A}$ to $\mathbb{B}$ is greater than 1 when $\mathbb{B}$ is the better algorithm.

We first present those results consistent with the competitive ratio.

Using essentially the same proof as is used in [Johnson 1974] to show that First-Fit is the best Any-Fit algorithm, one can show that this also holds for the relative worst order ratio. The idea is to consider the First-Fit packing of an arbitrary sequence. If the items are given bin by bin, any Any-Fit algorithm will produce exactly the same packing. As noted in Section 2.1, the following lower bound is valid for the relative worst order ratio as well as the strict relative worst order ratio.

THEOREM 3.1. *For any Any-Fit algorithm* $\mathbb{A}$, $WR'_{\mathbb{A},FF} \geq 1$ *(and* $WR_{\mathbb{A},FF} \geq 1$*).*

PROOF. Consider any input sequence I and its worst permutation for First-Fit, I_{W}. From First-Fit's packing of I_{W}, create a new permutation I' of I_{W} as follows: Take the items bin by bin from First-Fit's packing, starting with the first bin, and concatenate them together to form the sequence I'. Given I', $\mathbb{A}$ will place the items exactly as First-Fit places them when given I_{W}. To see this, note that $\mathbb{A}$ cannot open a new bin while items still fit in the last bin, and none of the items in First-Fit's later bins will fit in earlier bins. Thus, with this permutation, I', $\mathbb{A}$ uses as many bins as First-Fit does with I_{W}. Hence, for all I, $\mathbb{A}_{\mathrm{W}}(I) \geq \mathbb{A}(I') \geq \mathrm{FF}_{\mathrm{W}}(I)$, giving a ratio of at least one. □

Not all Any-Fit algorithms perform as well. Worst-Fit is the worst possible among the Any-Fit algorithms (Theorem 3.2), and it is significantly worse than First-Fit and Best-Fit (Theorem 3.3). Like Theorem 3.1, Theorem 3.2 is also valid for the relative worst order ratio $\mathrm{WR}_{\mathrm{WF},\mathbb{A}}$.

THEOREM 3.2. *For any Any-Fit algorithm* $\mathbb{A}$, $WR'_{WF,\mathbb{A}} \geq 1$ *(and* $WR_{WF,\mathbb{A}} \geq 1$*).*

PROOF. As in the proof of Theorem 3.1, we consider an arbitrary input sequence I, its worst permutation for $\mathbb{A}$, I_{W}, and the permutation I' of I_{W} where the items are given bin by bin, according to $\mathbb{A}$'s packing of I_{W}. We prove that Worst-Fit uses the same number of bins for I' as does $\mathbb{A}$ for I_{W}.

We need some notation: Call the bins used by $\mathbb{A}$ $b_1, b_2, \ldots, b_n$, numbered according to the order in which they were opened, and let S_j be the set of items packed in b_j, $1 \leq j \leq n$. Let $\ell_{\mathbb{A}}(b_j)$ be the level of b_j, i.e., the sum of the sizes of the items packed in b_j, in $\mathbb{A}$'s packing of I_{W}, and let $\ell_{\mathbb{A}}^{\min}(j) = \min_{1 \leq i \leq j} \{\ell_{\mathbb{A}}(b_i)\}$ be the minimum level among the first j bins. Let $\ell_{\mathrm{WF}}(b_i)$ be the level of b_i in Worst-Fit's packing of the prefix of I' considered so far.

We prove the following by induction on j (1. is used only in the proof of 2.).

(1) Worst-Fit uses at most j bins for the items in $S_1, \ldots, S_j$.
(2) For $1 \leq i \leq j$, after packing the items of $S_1, \ldots, S_i$, $\ell_{\mathrm{WF}}(b_i) \geq \ell_{\mathbb{A}}^{\min}(i)$.

The base case $j = 1$ is trivial: all items packed in b_1 by $\mathbb{A}$ are also packed in b_1 by Worst-Fit.

Consider now the induction step. If Worst-Fit packs all items of S_j in b_j, the result trivially follows, so assume that one of these items is packed in some bin $b_i \neq b_j$. Since, by 1. of the induction hypothesis, b_j is empty before giving the items of S_j, b_i must be an earlier bin, i.e., $i < j$. This proves 1. The inequalities below, combined with 2. of the induction hypothesis, prove 2.

$$\begin{aligned} \ell_{\mathrm{WF}}(b_j) &\geq \ell_{\mathrm{WF}}(b_i), \text{ by the Worst-Fit packing rule} \\ &\geq \ell_{\mathbb{A}}^{\min}(i), \text{ by 2. of the induction hypothesis} \\ &\geq \ell_{\mathbb{A}}^{\min}(j), \text{ since } i < j. \end{aligned}$$

Now, let e_j be the first item packed by $\mathbb{A}$ in b_j, $1 \leq j \leq n$. Since $\mathbb{A}$ is an Any-Fit algorithm, e_j does not fit in any earlier bin. In Worst-Fit's packing, all bins $b_1, \ldots, b_{j-1}$ have a level of at least $\ell_{\mathbb{A}}^{\min}(j-1)$, so e_j does not fit in any of these bins in the Worst-Fit packing either. Hence, for each e_j, $1 \leq j \leq n$, Worst-Fit must open a new bin, i.e., Worst-Fit uses the same number of bins as $\mathbb{A}$. □

Using Johnson's results and techniques [Johnson 1974], one can show that the relative worst order ratio of Worst-Fit to either First-Fit or Best-Fit is 2.

THEOREM 3.3. *$WR_{WF,FF} = WR_{WF,BF} = 2$.*

PROOF. The relative worst order ratio of Worst-Fit to either First-Fit or Best-Fit is at most 2, since Worst-Fit's competitive ratio is 2 [Johnson 1974].

By Theorems 2.6 and 3.1, we only need to compare Worst-Fit and Best-Fit to prove the lower bound. Since Theorem 3.2 shows that $\mathrm{WR}_{\mathrm{WF,BF}} \geq 1$, it is sufficient to find a family of sequences I_n, with $\lim_{n\to\infty} \mathrm{WF}_{\mathrm{W}}(I_n) = \infty$, where there exists a constant b such that for all I_n, $\mathrm{WF}_{\mathrm{W}}(I_n) \geq 2\mathrm{BF}_{\mathrm{W}}(I_n) - b$. The family of sequences used in [Johnson 1974] to bound Worst-Fit's competitive ratio from below works here. Let $0 < \varepsilon \leq \frac{1}{2n}$. Consider the sequence I_n with pairs $(\frac{1}{2}, \varepsilon)$, for $i = 1, \ldots, n$. In this order, Worst-Fit will pack all of the pairs, one per bin, using n bins. Best-Fit will pack the small items all in one bin, regardless of the permutation, using only $\lceil \frac{n+1}{2} \rceil$ bins. Thus, $\mathrm{WF}_{\mathrm{W}}(I_n) = n \geq 2\lceil \frac{n+1}{2} \rceil - 2 = 2 \cdot \mathrm{BF}_{\mathrm{W}}(I_n) - 2$, so the relative worst order ratio is at least 2. □

Now we consider Next-Fit, which is not an Any-Fit algorithm. Next-Fit is strictly worse than Worst-Fit and all other Any-Fit algorithms. This result is in contrast to the competitive ratio where Next-Fit and Worst-Fit both have ratios of 2 [Johnson 1974].

THEOREM 3.4. *For any Any-Fit algorithm $\mathbb{A}$, $WR_{NF,\mathbb{A}} = 2$.*

PROOF. To see that $\mathrm{WR}_{\mathrm{NF},\mathbb{A}} \geq 1$, consider any input sequence, I, and its worst permutation for First-Fit, I_{W}. From $\mathbb{A}$'s packing of I_{W}, create a new sequence I' from I_{W} by taking the items bin by bin from $\mathbb{A}$'s packing, starting with the first bin, in the order they were placed in the bins, and concatenate the contents together to form the sequence I'. Next-Fit also has to open a new bin for the first item put in each bin, so it ends up with the same configuration. Hence, for all I, $\mathrm{NF}_{\mathrm{W}}(I) \geq \mathrm{NF}(I') \geq \mathbb{A}_{\mathrm{W}}(I)$, giving a ratio of at least one.

Since $\mathrm{WR}_{\mathrm{NF},\mathbb{A}} \geq 1$, to prove the lower bound of 2 it is sufficient to find a family of sequences I_n, with $\lim_{n\to\infty} \mathrm{NF}_{\mathrm{W}}(I_n) = \infty$, where there exists a constant b such that for all I_n, $\mathrm{NF}_{\mathrm{W}}(I_n) \geq$

$2 \cdot \mathbb{A}_{\mathrm{W}}(I_n) - b$. Let $0 < \varepsilon \leq \frac{1}{n-1}$. Consider the sequence I_n with $n-1$ pairs $(\varepsilon, 1)$. In this order, Next-Fit will pack each item in a new bin, whereas $\mathbb{A}$ will pack all of the small items in the same bin. Thus, $\mathrm{NF}_{\mathrm{W}}(I_n) = 2(n-1) = 2 \cdot \mathbb{A}_{\mathrm{W}}(I_n) - 2$, so $\mathrm{WR}_{\mathrm{NF},\mathbb{A}} \geq 2$.

For the upper bound, note that the relative worst order ratio of Next-Fit to any other algorithm is at most 2, since Next-Fit's competitive ratio is 2 [Johnson 1974]. □

Now we turn to algorithms which have better competitive ratios than First-Fit, starting with Harmonic$_k$. This class of algorithms can be slightly worse than First-Fit and Best-Fit, but only on sequences which have very small items. For sequences with all items larger than $\frac{1}{k+1}$, Harmonic$_k$ is strictly better than any Any-Fit algorithm.

Recall that Harmonic$_k$ 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.

Lemma 3.5. *Let $\mathbb{A}$ denote First-Fit or Best-Fit. If the input sequences can contain arbitrarily small items, there exists a family of sequences I_n such that, for all n,*

$$(\text{Harmonic}_k)_W(I_n) \;>\; \frac{k}{k-1}\mathbb{A}_W(I_n) - \frac{k+1}{k-1}\,,$$

and $\lim_{n\to\infty} \mathbb{A}_W(I_n) = \infty$.

Furthermore, if the smallest items in the input sequence have size $\varepsilon < \frac{1}{2k}$, there exists a family of sequences I_n such that, for all n,

$$(\text{Harmonic}_k)_W(I_n) \;>\; \frac{k(1-\varepsilon)}{k(1+\varepsilon)-1}\mathbb{A}_W(I_n) - \frac{k}{k(1+\varepsilon)-1}\,,$$

and $\lim_{n\to\infty} \mathbb{A}_W(I_n) = \infty$.

Note that this ratio is slightly greater than 1 when $\varepsilon < \frac{1}{2k}$.

Proof. By Theorem 3.1, if we prove the lemma for the case where $\mathbb{A}$ denotes Best-Fit, we know that $\mathrm{BF}_{\mathrm{W}}(I_n) \geq \mathrm{FF}_{\mathrm{W}}(I_n)$, so the lemma also holds for First-Fit.

First, let $\varepsilon \leq \frac{1}{kn}$. 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. Best-Fit, on the other hand, combines the small items regardless of the permutation of the sequence, so

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

Thus,

$$\frac{k}{k-1}\,\mathrm{BF}_{\mathrm{W}}(I_n) - \frac{k+1}{k-1} \;<\; n \;=\; (\text{Harmonic}_k)_W(I_n).$$

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

$$\begin{aligned}\mathrm{BF}_{\mathrm{W}}(I_n) &\leq \left\lceil \left(\frac{k-1}{k} + \varepsilon\right) n + \mathrm{BF}_{\mathrm{W}}(I_n) \cdot \varepsilon \right\rceil \\ &< \left(\frac{k-1}{k} + \varepsilon\right) n + \mathrm{BF}_{\mathrm{W}}(I_n) \cdot \varepsilon + 1\end{aligned}$$

Solving for the number n of bins used by Harmonic$_k$ we get

$$\frac{k(1-\varepsilon)}{k(1+\varepsilon)-1}\mathrm{BF}_W(I_n) - \frac{k}{k(1+\varepsilon)-1} \;<\; n.$$

□

For any fixed k, the result of Lemma 3.5 is tight up to an additive constant.

LEMMA 3.6. *For any Any-Fit algorithm $\mathbb{A}$ and any sequence I,*

$$(\text{HARMONIC}_k)_W(I) \;\leq\; \frac{k}{k-1}\mathbb{A}_W(I) + k + 1.$$

PROOF. By Theorem 3.1, if we prove the lemma for the case where $\mathbb{A}$ denotes First-Fit, we know that $\text{FF}_\text{W}(I_n) \leq \mathbb{B}_\text{W}(I_n)$ for any Any-Fit algorithm $\mathbb{B}$, so the lemma also holds for $\mathbb{B}$. Consider HARMONIC$_k$'s packing of its worst permutation of I, I_W. The idea is to create a permutation of I on which First-Fit uses approximately the same number of bins as HARMONIC$_k$ for the items in $C_1, \ldots, C_{k-1}$ and at least $\frac{k-1}{k}$ as many bins as HARMONIC$_k$ for the C_k items.

Let L_j, for $1 \leq j \leq k-1$, be the set of items from I_W in class C_j which 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_W. Remove those items from C'_k which First-Fit places in its last bin and add them to 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_W, 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 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 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, 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 $\frac{k-1}{k}\ell$, so First-Fit must use at least $\frac{k-1}{k}\ell$ bins for the items in C'_k. The result follows since HARMONIC$_k$ uses at most $k+1$ bins for the items in L. □

Thus, for reasonably large k, HARMONIC$_k$ never does much worse than First-Fit. Furthermore, for all $k \geq 3$, HARMONIC$_k$ sometimes does strictly better than First-Fit.

LEMMA 3.7. *Let $\mathbb{A}$ denote any Any-Fit algorithm. There exists a family of sequences I_n such that, for all n and $k \geq 3$,*

$$\mathbb{A}_W(I_n) \;\geq\; \frac{6}{5} \cdot (\text{HARMONIC}_k)_W(I_n),$$

and $\lim_{n\to\infty} (\text{HARMONIC}_k)_W(I_n) = \infty$.

PROOF. 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. 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. This shows that $\text{FF}_\text{W}(I_n) \;\geq\; \frac{6}{5} \cdot (\text{HARMONIC}_k)_W(I_n)$. The result follows since there is no additive constant in the proof of Theorem 3.1. □

The following theorem is an immediate consequence of Lemmas 3.5 and 3.7.

THEOREM 3.8. *First-Fit and* HARMONIC$_k$ *are incomparable, as are Best-Fit and* HARMONIC$_k$.

However, if all of the items in a sequence have sizes greater than $\frac{1}{k+1}$, then HARMONIC$_k$ always does at least as well as any Any-Fit algorithm, except for a possible additive constant.

LEMMA 3.9. *Let $\mathbb{A}$ be any Any-Fit algorithm. For any sequence, I, where all items have size greater than $\frac{1}{k+1}$,*

$$(\text{HARMONIC}_k)_W(I) \;\leq\; \mathbb{A}_W(I) + k.$$

PROOF. The proof is a simplification of the proof of Lemma 3.6. The class C_k only has items which are larger than $\frac{1}{k+1}$, so First-Fit and HARMONIC$_k$ will both put exactly k of them in each

bin, except possibly the last. L can be put in C'. This means that the items in class C_k can be treated in the same manner as items in the other classes. Thus, HARMONIC$_k$ can use more bins than First-Fit only for the items in L, and there are at most k such bins. □

Thus, with item sizes greater than $\frac{1}{k+1}$, HARMONIC$_k$ is comparable to any Any-Fit algorithm. The following theorem is an immediate consequence of Lemmas 3.7 and 3.9.

THEOREM 3.10. *Let $\mathbb{A}$ be any Any-fit algorithm. For any fixed $k \geq 2$, for the Classical Bin Packing problem restricted to sequences where all items have size greater than $\frac{1}{k+1}$,*

$$WR_{\mathbb{A},\text{HARMONIC}_k} \geq \frac{6}{5}.$$

For First-Fit the lower bound of Theorem 3.10 is tight.

THEOREM 3.11. *For any fixed $k \geq 2$, for the Classical Bin Packing problem restricted to sequences where all items have size greater than $\frac{1}{k+1}$,*

$$WR_{FF,\text{HARMONIC}_k} = \frac{6}{5}.$$

PROOF. The lower bound follows from Theorem 3.10. The upper bound argument uses the off-line algorithm First-Fit-Increasing (FFI), which first sorts the items in nondecreasing order and then applies First-Fit to the resulting sequence. We use a result from [Boyar et al. 2005] 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 permutation of I. Both 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. □

Note that the proof of Lemma 3.5, showing that First-Fit can do better than HARMONIC$_k$ depends on HARMONIC$_k$ using Next-Fit for packing within each class. This is used for efficiency, keeping the number of open bins no more than k. However, one could consider using another algorithm, such as First-Fit or Best-Fit instead. The proof of Lemma 3.6 shows that the multiplicative factor, $\frac{k-1}{k}$, in the result comes from the items which are in the last class and have size at most $\frac{1}{k}$. If HARMONIC$_k$ packed these items using the algorithm $\mathbb{A}$ (First-Fit or Best-Fit), then $\mathbb{A}$ would use exactly as many bins for these items as HARMONIC$_k$. Thus, for this modified version of HARMONIC$_k$, $(\text{HARMONIC}_k)_W(I) \leq \mathbb{A}_W(I) + k + 1$, so this modified HARMONIC$_k$ is strictly better than First-Fit, and the ratio of $\frac{6}{5}$ is valid for this modification.

The idea behind the variants of HARMONIC$_k$ defined in [Lee and Lee 1985] and [Seiden 2002] 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 [Lee and Lee 1985], 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. On the other hand, any Any-Fit algorithm (or HARMONIC$_k$) would pack them all two per bin using only $\frac{n}{2}$ bins. This gives a ratio of $\frac{8}{7}$ in the Any-Fit algorithm's (or HARMONIC$_k$'s) favor. The family of sequences showing that HARMONIC$_k$ is better than any Any-Fit algorithm 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 any Any-Fit algorithm by a factor $\frac{6}{5}$. This shows the following:

THEOREM 3.12. REFINED HARMONIC *is incomparable to any Any-Fit algorithm.*

There also exist sequences where REFINED HARMONIC does better on its worst permutation than HARMONIC$_k$ on its worst permutation. Consider the family of sequences 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 3.13. *REFINED HARMONIC and HARMONIC$_k$ are incomparable for $k \geq 2$.*

Thus, either HARMONIC$_k$ or REFINED HARMONIC may have the better performance, depending on the application and the expected input sequence distribution. For example, if 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. Similar arguments show that any instance of Seiden's SUPER HARMONIC algorithms [Seiden 2002], including HARMONIC++, are incomparable to HARMONIC$_k$ and any Any-Fit algorithm.

4. DUAL BIN PACKING

When switching to the Dual Bin Packing problem, which is a maximization problem, the relative worst order ratio of $\mathbb{A}$ to $\mathbb{B}$ is greater than 1 when $\mathbb{A}$ is the better algorithm, instead of when $\mathbb{B}$ is.

For Dual Bin Packing, the worst order ratio is the same as the competitive ratio, so the worst order ratio of any algorithm is 0 [Boyar et al. 2001]. Recall that for the variant of Dual Bin Packing called Fair Bin Packing, the worst order ratio is not the same as the competitive ratio, since the optimal off-line algorithm compared to must be fair, and hence the permutation of the input sequence may make a difference, even for the off-line algorithm. However, again we get the same pessimistic result as with the competitive ratio.

Throughout this section, n will refer to the fixed number of bins which exist. We prove the following result only for n even to avoid additionally complicating the proof.

THEOREM 4.1. *For any deterministic on-line algorithm $\mathbb{F}$ for the Fair Bin Packing problem with an even number n of bins, $WR_{\mathbb{F}} = 0$.*

PROOF. Consider any fair, deterministic on-line algorithm, $\mathbb{F}$. For the following sequence, I, defined on the basis of $\mathbb{F}$'s performance, $\mathbb{F}$ will accept all of the larger items and reject all of the small, while, for any permutation of I, OPT will be able to arrange to reject some of the larger items and accept many of the small ones.

Let $0 < \varepsilon < \frac{1}{24}$. The sequence I begins with n items of size $\frac{1}{3} + \varepsilon$, called items of type A. Since $\mathbb{F}$ is fair, it clearly accepts all of these items. Suppose $\mathbb{F}$ places them in the bins, leaving q bins empty. Then, exactly q bins have two items and $n - 2q$ have one. The sequence continues with

Type B items:	$n + q$	items of size	$\frac{1}{3}$
Type C items:	$n - 2q$	items of size	$\frac{1}{3} - \varepsilon$
Type D items:	q	items of size	$\frac{1}{3} - 2\varepsilon$
Type E items:	$\frac{n}{12\varepsilon} - \frac{n}{4}$	items of size	ε

Items of types A, B, C, and D are the "large" items, and items of type E are the "small" items.

Each of the $n - 2q$ bins with one type A item can hold one item of type B, and each of the q bins with no type A item can hold $3q$ items of type B. Hence, the algorithm accepts all items of type B. After this, $\mathbb{F}$ will pack one type C item in each of the bins holding exactly one type A item, thus accepting all items of type C. Finally, all of the items of type D will be packed, one per bin with two type A items. This fills all bins completely, so $\mathbb{F}$ packs none of the small items.

In the worst permutation of I for OPT, OPT will be able to reject the least number of large items and thus accept the least number of small items. Consider such a worst-case permutation $I' = \sigma(I)$. Without loss of generality, we may assume that all of the large items in I' come before all of the small.

We now describe a strategy which tries to do the "opposite" of $\mathbb{F}$, i.e., if $q \leq \frac{2n}{7}$, we combine the items of type A pairwise, and if $q \geq \frac{2n}{7}$, we spread them out, one per bin.

If $q \leq \frac{2n}{7}$, we consider the strategy where type A items are combined pairwise in the bins. Type B and C items are packed together, three per bin. Each type D item is packed with a pair of type

A items. Note that it is indeed possible to avoid packing D items with the B and C items, since the B and C items take up at most $\frac{2n-q}{3}$ bins, leaving at least $\frac{n+q}{3}$ bins for the A and D items. For $q \leq \frac{2n}{7}$, $\frac{n+q}{3}$ is strictly greater than the number q of type D items.

The bins with two type A items and one type D item are completely filled, and bins with three type B and/or C items have empty space of size at most 3ε, but the bins with only A items have empty space of size $\frac{1}{3} - 2\varepsilon$ each. Hence, in the worst case, all B and C items appear so early in the sequence that none of them are rejected. Since this leaves only $\frac{n+q}{3}$ bins for the type A items,

$$n - 2 \cdot \frac{n+q}{3} = \frac{n-2q}{3} \geq \frac{7n-4n}{21} = \frac{n}{7}$$

of these will be rejected, leaving room for more than $\frac{n}{7} \cdot \frac{1}{3} \cdot \frac{1}{\varepsilon} = \frac{n}{21\varepsilon}$ small items.

If $q \geq \frac{2n}{7}$, we consider the strategy where type A items are spread out. The idea is the following. We pack at most one A item in each bin. As long as less than n items of type B have been given, we pack only one B item in each bin. C and D items are also spread out such that each bin contains at most one C item or one D item.

Bins that have received one A item and one B item cannot hold any more items of type A or B. A bin with two type B items can hold one more type B item, and then it is completely filled. Note that we cannot be forced to put two items of type A in one bin, since as long as there are bins with an A item that still have room for another A item, there are also bins without A items that have room for an A item. This is because, if some bin containing an A item has room for one more A item, then this bin contains no B item. But as long as there are bins without B items, no bin has more than one B item. Therefore, bins without A items are filled to at most $\frac{2}{3} - \varepsilon$ (the total size of a B item and a C item).

Let n_B be the number of bins containing three B items. The total number of A and B items packed is no more than $2n + n_B$. On the other hand, since we pack at most one A item in each bin, one type A item must be rejected for each bin with three B items, i.e., this total number is also bounded by $2n + q - n_B$. The two upper bounds are equal when $n_B = \frac{q}{2}$, so the total number of items of type A or B packed is at most $2n + \frac{q}{2}$, meaning that at least $\frac{q}{2} \geq \frac{n}{7}$ of these will be rejected. Again this means that at least $\frac{n}{21\varepsilon}$ small items are accepted.

There are only $3n$ large items, and the number of small items accepted with the strategy sketched above, and hence by an optimal off-line algorithm can be arbitrarily large, depending only on ε. This shows that no deterministic on-line algorithm for the Fair Bin Packing problem can have a relative worst order ratio strictly greater than 0. □

We now turn to the *relative* worst order ratio. For Dual Bin Packing, one can again show that First-Fit is a best possible Any-Fit algorithm, also using the proof by Johnson [Johnson 1974].

THEOREM 4.2. *For any Any-Fit algorithm* $\mathbb{A}$, $WR'_{FF,\mathbb{A}} \geq 1$.

PROOF. The proof is essentially the same as that for the Classical Bin Packing problem, but now any items First-Fit rejects are concatenated to the end of the sequence created for $\mathbb{A}$ and will also be rejected by $\mathbb{A}$. □

For $n \geq 6$, First-Fit is strictly better than Best-Fit:

THEOREM 4.3. $WR'_{FF,BF} \geq \frac{10}{9} - \frac{5}{9n}$.

PROOF. The above theorem shows that $\mathrm{WR}'_{\mathrm{FF,BF}} \geq 1$. To show the separation, let $0 < \varepsilon < \frac{1}{8n^2}$. Consider the sequence I_n starting with pairs, $(\frac{1}{2} + 2in\varepsilon, \varepsilon)$, for $i = 0, \ldots, n-1$ and followed by $\frac{1}{2} - (2i+1)n\varepsilon$, for $i = 0, \ldots, n-1$ and $n-1$ of size $n\varepsilon$. This gives a total of $4n-1$ items. The items of size ε or $n\varepsilon$ are called *small* items, the remaining items are called *large*. The smallest large item has size greater than $\frac{1}{4}$ and the largest item has size less than $\frac{3}{4}$. The total size of the small items is less than $\frac{1}{8}$.

Best-Fit will pack the first n pairs, one pair per bin. After packing the large items smaller than $\frac{1}{2}$, the empty space in each bin will be exactly $(n-1)\varepsilon$, and the $n-1$ items of size $n\varepsilon$ are rejected, i.e., Best-Fit packs only $3n$ items.

The problem with Best-Fit is that it distributes the items of size ε, one per bin. With First-Fit, these items will all end up in the same bin, since the first bin with room for one of them will have room for all of them (if there is more than one large item in a bin, then the remaining space is a multiple of $n\varepsilon$). Similarly, bins that do not have room for any item of size $n\varepsilon$ must be completely full or contain at least one item of size ε. Since a bin can only be completely full, if it contains all items of size ε or at least one item of size $n\varepsilon$, all $n-1$ items of size $n\varepsilon$ will be packed too.

A large item in a bin with no other large item is said to be *alone* in that bin. Since there are $2n$ large items, the number of large items rejected is at most the number of large items alone in bins. Let S denote the set of large items of size less than $\frac{1}{2}$ and R denote the items First-Fit rejects. Suppose the smallest rejected item has size s.

If $s \geq \frac{1}{2}$, then all items in S are packed, and at most one is alone in a bin. Therefore, if more than $\frac{n}{2}$ large items are alone in bins, at least $\lfloor \frac{n}{2} \rfloor$ items larger than $\frac{1}{2}$ are packed. Since no items smaller than $\frac{1}{2}$ are rejected, this means that at most $\lceil \frac{n}{2} \rceil$ items are rejected. If at most $\frac{n}{2}$ large items are alone in bins, then by the argument in the previous paragraph, at most $\frac{n}{2}$ items are rejected.

If $s < \frac{1}{2}$, then $s = \frac{1}{2} - (2j+1)n\varepsilon$ for some $0 \leq j \leq n-1$. Thus, except for possibly one, all large items alone in bins must have size at least $s = \frac{1}{2} + 2(j+1)n\varepsilon$. There are only $n-j-1$ such items, so this gives an upper bound on the number of rejected items of $|R| \leq n-j$. To get another upper bound on $|R|$, note that the $n-j-1$ items from S which are smaller than s are all packed. So there are at most $n+j+1$ items which could potentially be rejected, and at least $\frac{n+j}{2}$ of these must be in bins alone to account for the rejections of the others. Thus, $|R| \leq \frac{n+j}{2}+1$. Multiplying this inequality by two and adding the previous inequality involving $|R|$ gives $3|R| \leq 2n+2$, so First-Fit rejects at most $\frac{2n+2}{3}$ items.

This shows that First-Fit packs at least $\frac{4n-2}{3}$ large items and all of the $2n-1$ small items, giving a total of $\frac{10n-5}{3}$ items, whereas Best-Fit packs only $\frac{9n}{3}$ items. This gives a ratio of at least $\frac{10}{9} - \frac{5}{9n}$. □

Recall that for the Dual Bin Packing problem, Worst-Fit is the algorithm which places an item in one of the bins which are least full; we assume it chooses the first such bin. Worst-Fit is a fair algorithm, and it is a worst possible such algorithm:

THEOREM 4.4. *For any fair algorithm $\mathbb{F}$, $WR'_{\mathbb{F},WF} \geq 1$.*

PROOF. Consider any input sequence, I, and its worst permutation for $\mathbb{F}$, I_{W}. From $\mathbb{F}$'s packing of I_{W}, create a new sequence I' from I as follows: For each item x in I, let $b(x)$ denote the sum of the sizes of the items which appeared before x in I and are placed by $\mathbb{F}$ in the same bin as x. Thus, $b(x)$ is the height at which x is placed. Let I' consist of the items in I sorted in non-decreasing order by $b(x)$, with the items $\mathbb{F}$ rejects placed at the end. Note that Worst-Fit will create essentially the same packing with I' as $\mathbb{F}$ did with I_{W}; if $b'(x)$ is defined as $b(x)$, except using the Worst-Fit's packing of I', then $b(x) = b'(x)$ for all x, except those where $b(x)$ is larger than the final height of some bin in $\mathbb{F}$'s packing. Thus, with this order, Worst-Fit will reject the same items of I' as $\mathbb{F}$ does with I_{W}. Hence, for all I, $\mathbb{F}_{\mathrm{W}}(I) \geq \mathbb{F}(I') \geq \mathrm{WF}_{\mathrm{W}}(I)$, giving a ratio of at least one. □

Theorem 4.5 below shows that, according to the relative worst order ratio, First-Fit and Best-Fit are strictly better than Worst-Fit. This is in contrast to the competitive ratio, where, for the restricted problem where all item sizes are multiples of some constant f, First-Fit and Best-Fit actually have worse ratios than Worst-Fit [Boyar et al. 2001], and without this restriction none of them are competitive. The relative worst order ratio corresponds more to the competitive ratio on accommodating sequences, where First-Fit and Best-Fit can be shown to perform better than Worst-Fit [Boyar et al. 2001]. The result concerning the relative worst order ratio is, however, much easier to prove.

THEOREM 4.5. *For any Any-Fit algorithm $\mathbb{A}$, $WR'_{\mathbb{A},WF} = 2 - \frac{1}{n}$.*

PROOF. Since Any-Fit algorithms are fair, Theorem 4.4 shows that $\mathrm{WR}'_{\mathbb{A},\mathrm{WF}} \geq 1$. To prove the separation and the lower bound, let $0 < \varepsilon \leq \frac{1}{n}$, and let I_n consist of n items of size ε, followed

by $n-1$ of size 1. Worst-Fit will accept only the n items of size ε when they are given in this order. $\mathbb{A}$ will accept all of these items, regardless of their order. Thus, $\frac{\mathbb{A}_W(I_n)}{WF_W(I_n)} = \frac{2n-1}{n}$.

Consider now the upper bound. By Theorems 2.6 and 4.2, it is sufficient to show that it holds for First-Fit. Consider any sequence I and the worst permutation of I for Worst-Fit. Without loss of generality, assume that the m items Worst-Fit accepts appear in I before those it rejects.

Reorder the first m items in this worst permutation for Worst-Fit, so that First-Fit gets them bin by bin, according to Worst-Fit's packing. Since no item will be packed in a later bin by First-Fit than by Worst-Fit, for each item Worst-Fit accepts, First-Fit will have room for it in the same bin, if not an earlier one. Thus, First-Fit will accept all the items Worst-Fit accepts. First-Fit accepts at most $n-1$ more items than Worst-Fit, since each of the items which Worst-Fit rejects must be larger than the empty space in any of Worst-Fit's bins. Thus, the total size of any n rejected items (if there are that many) would be more than the total empty space in the n bins after packing the items accepted by Worst-Fit. First-Fit will thus accept at most $m + (n-1)$ items in its worst ordering of the I.

Since Worst-Fit is fair, it must accept at least n items if it rejects any at all, and the ratio is worst when it is only n items. Thus, $\frac{FF_W(I)}{WF_W(I)} \leq \frac{m+(n-1)}{m} \leq \frac{2n-1}{n}$. □

An example of an algorithm for the Dual Bin Packing problem which is not fair is Unfair-First-Fit [Azar et al. 2002]. It behaves as First-Fit, except that when given an item of size greater than $\frac{1}{2}$, it automatically rejects that item if it has already accepted at least $\frac{2}{3}$ of the items seen so far. The intuition is that by rejecting some large items, it may have room for more small items. The algorithm is defined in Figure 1.

```
Input:  I = ⟨o_1, o_2, ..., o_n⟩
Output: A, R, and a packing for those items in A
A := {}; R := {}
while I ≠ ⟨⟩
    o := hd(I); I := tail(I)
    if size(o) > 1/2 and |A|/(|A|+|R|+1) ≥ 2/3
        R := R ∪ {o}
    else if there is space for o in some bin
            pack o according to the First-Fit rule
            A := A ∪ {o}
        else
            R := R ∪ {o}
```

Fig. 1. The algorithm Unfair-First-Fit

To prove that Unfair-First-Fit is incomparable to any Any-Fit algorithm we use the following observation.

Remark 4.6. Note that for the Dual Bin Packing problem, the competitive ratio on accommodating sequences can be used to get results concerning the relative worst order ratio, but the competitive ratio cannot necessarily. The problem with using the competitive ratio directly is that we are comparing to OPT which may be more able to take advantage of a fairness restriction with some permutations than with others. When the sequences are not accommodating sequences, then we may be looking at sequences where there is some order where OPT also does poorly. This cannot happen with accommodating sequences. For example, if algorithm $\mathbb{A}$ has a competitive ratio on accommodating sequences of at least p and $\mathbb{B}$ has a competitive ratio on accommodating sequences of at most $r < p$, then there is an accommodating sequence I where $\mathbb{A}_W(I) \geq p|I| > r|I| \geq \mathbb{B}_W(I)$. This can help give a result in the case where one has already shown that $\mathbb{A}$ is at least as good as $\mathbb{B}$ on all sequences.

THEOREM 4.7. *Under the relative worst order ratio, Unfair-First-Fit is incomparable to all Any-Fit algorithms.*

PROOF. It is easy to see that there exist sequences where Unfair-First-Fit does worse than any Any-Fit algorithm. Consider, for example, the sequence containing n items of size 1. Unfair-First-Fit will only accept $\frac{2}{3}$ of them, while any fair algorithm (and thus all Any-Fit algorithms) will accept all of them. Hence, on such a sequence, any Any-Fit algorithm accepts $\frac{3}{2}$ times as many items as Unfair-First-Fit.

To show the other direction, it suffices to compare Unfair-First-Fit to First-Fit, the best among the Any-Fit algorithms. Since the competitive ratio on accommodating sequences for First-Fit is bounded above by $\frac{5}{8} + O(\frac{1}{\sqrt{n}})$, and the competitive ratio on accommodating sequences for Unfair-First-Fit is $\frac{2}{3} - \Theta(\frac{1}{n})$, for large enough n [Azar et al. 2002], there exists an accommodating sequence where Unfair-First-Fit outperforms First-Fit. By Remark 4.6, Unfair-First-Fit accepts asymptotically $\frac{16}{15}$ times as many items as First-Fit. □

So far we have seen that, with respect to the relative worst order ratio, First-Fit is best possible among Any-Fit algorithms and incomparable to Unfair-First-Fit. There is, however, an algorithm which does strictly better than First-Fit. 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 if this seems appropriate for the given application.

We first show that with the relative worst order ratio, First-Fit^n is unboundedly better than First-Fit (Theorem 4.8) and then demonstrate that the two algorithms cannot be distinguished by either their competitive ratios or their competitive ratios on accommodating sequences (Theorems 4.9 and 4.10).

THEOREM 4.8. *For any Any-Fit algorithm $\mathbb{A}$, $WR_{FF^n,\mathbb{A}} > c$, for any constant $c > 1$.*

PROOF. By Theorems 2.3 and 3.1, it is sufficient to prove that $\mathrm{WR}_{\mathrm{FF}^n,\mathrm{FF}} > c$, for any constant $c > 1$. 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 First-Fit^n accepts $n-1$ items of size 1 and all cn items of size $\frac{1}{cn}$, regardless of the permutation. 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 [Boyar et al. 2001] 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 4.9. *First-Fit^n is not competitive.*

PROOF. Let $0 < \varepsilon \leq \frac{1}{2}$. An adversary gives the following item 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 items in phase 3, giving a performance ratio of approximately ε. □

To show that First-Fit^n has the same competitive ratio on accommodating sequences as First-Fit asymptotically, we use the sequences from [Azar et al. 2002] showing that First-Fit's competitive ratio on accommodating sequences is $\frac{5}{8} + O(\frac{1}{n})$ for some special values of n.

THEOREM 4.10. *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 0:	3 items of size $A = \frac{1}{3} - 2^{3q}\varepsilon$.
Phases $j = 1, \ldots, q$:	$3 \cdot 2^j$ 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 $q + 1$:	$3 \cdot 2^q$ items of size $D = \frac{2}{3} + \varepsilon$.
Phase $q + 2$:	$9 \cdot 2^q - 6$ items of size $E = \frac{1}{3}$.

In each phase j, $1 \leq j \leq q$, 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 phase 0. After this, the first $3 \cdot 2^q - 1$ items of phase $q+1$ will be packed in separate bins, giving a total of $n-1$ used bins. Since the last item of phase $q + 1$ is larger than $\frac{1}{2}$, it will be rejected and three of the items from phase $q + 2$ will be packed in the last bin. The remaining $9 \cdot 2^q - 3$ items from phase $q + 2$ 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 0 with two items of size B_1 and each item of size C_j, $1 \leq j \leq q - 1$, with two items of size B_{j+1}. Each item of size C_q is paired with an item from phase $q + 1$. This uses $3 \cdot (2^{q+1} - 1) = 6 \cdot 2^q - 3$ bins, leaving room for all of the items of phase $q + 2$. 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

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

□

Theorem 4.10 considers special values of n. As in [Azar et al. 2002], 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 AND OPEN PROBLEMS

This first study of the relative worst order ratio seems very promising in that most results obtained are consistent with those obtained with the competitive ratio, but new separations are found, along with a new algorithm. Even more promising results have recently been found when applying the relative worst order ratio to paging, bin coloring, some scheduling problems, and the seat reservation problem.

The new performance measure gives the advantage that one can compare two on-line algorithms directly and is more widely applicable than the Max/Max ratio. It is intuitively suitable for some natural problems where any permutation of the input is equally likely and is easier to compute than the random order ratio.

It is also better than the competitive ratio at distinguishing between algorithms for Classical and Dual Bin Packing. With the relative worst order ratio, Worst-Fit is better than Next-Fit for Classical Bin Packing. For Dual Bin Packing, First-Fit is better than Best-Fit which is better than Worst-Fit. Although the competitive ratio on accommodating sequences can also be used to show that First-Fit is better than Worst-Fit for Dual Bin Packing, the proof is much easier with the relative worst order ratio.

For Dual Bin Packing as well as Fair Bin Packing, all deterministic algorithms have a worst order ratio of 0, confirming the intuition that information is lost in an intermediate comparison to an algorithm much more powerful than the on-line algorithms.

The relative worst order ratio should be applied to other on-line problems. We hope this will lead to many new separations and algorithms being discovered. Furthermore, it would be interesting

to apply this new measure to NP-hard off-line problems where the approximation ratio would generally be used or to heuristics where no ratio has been proven.

For the Classical Bin Packing problem, there exist sequences where First-Fit's worst permutation uses one less bin than Best-Fit's worst permutation. One example of this is the following sequence: $\frac{1}{4}, \frac{1}{4}, \varepsilon, \frac{3}{4}, \varepsilon, \frac{1}{4}, \frac{1}{4}$, where Best-Fit uses three bins for its worst permutation, while First-Fit only uses two. Considering the strict relative worst order ratio, this gives a lower bound of $\frac{3}{2}$. However, this seems to be hard to extend to an asymptotic result; determining if $\mathrm{WR}_{\mathrm{BF,FF}} > 1$ is an open problem.

ACKNOWLEDGMENTS

We would like to thank Kim Skak Larsen and Yossi Azar for helpful discussions. We would also like to thank the anonymous referees for useful comments, especially the referee who noted that the proof of Lemma 3.5 depended on HARMONIC$_k$ using Next-Fit and asked about the possibility of it calling First-Fit instead.

REFERENCES

Azar, Y., Boyar, J., Epstein, L., Favrholdt, L. M., Larsen, K. S., and Nielsen, M. N. 2002. Fair versus unrestricted bin packing. *Algorithmica 34,* 2, 181–196.

Ben-David, S. and Borodin, A. 1994. A new measure for the study of on-line algorithms. *Algorithmica 11,* 1, 73–91.

Boyar, J., Epstein, L., Kohrt, J. S., Larsen, K. S., Pedersen, M. M., and Wøhlk, S. 2005. The maximum resource bin packing problem. In *Fundamentals of Computation Theory: 15th International Symposium.* Lecture Notes in Comput. Sci., vol. 3623. Springer-Verlag, 397–408.

Boyar, J., Favrholdt, L. M., and Larsen, K. S. 2005. The relative worst order ratio applied to paging. In *Proc. 16th Annual ACM-SIAM Symp. on Discrete Algorithms.* 718–727.

Boyar, J., Favrholdt, L. M., Larsen, K. S., and Nielsen, M. N. 2001. The competitive ratio for on-line dual bin packing with restricted input sequences. *Nordic J. Comput. 8,* 4, 463–472.

Boyar, J. and Larsen, K. S. 1999. The seat reservation problem. *Algorithmica 25*, 403–417.

Boyar, J., Larsen, K. S., and Nielsen, M. N. 2001. The accommodating function — a generalization of the competitive ratio. *SIAM J. Comput. 31,* 1, 233–258.

Boyar, J. and Medvedev, P. 2004. The relative worst order ratio applied to seat reservation. In *Proc. 9th Scandinavian Workshop on Algorithm Theory.* Lecture Notes in Comput. Sci., vol. 3111. Springer-Verlag, 90–101.

Epstein, L., Favrholdt, L. M., and Kohrt, J. S. 2004. The relative worst order ratio applied to scheduling problems. Manuscript, included in J. S. Kohrt's PhD disseration, *Online Algorithms under New Assumptions*, University of Southern Denmark.

Graham, R. L. 1966. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal 45*, 1563–1581.

Johnson, D. S. 1974. Fast algorithms for bin packing. *J. Comput. Syst. Sci. 8*, 272–314.

Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., and Graham, R. L. 1974. Worst-case performance bound for simple one-dimensional packing algorithms. *SIAM J. Comput. 3*, 299–325.

Karlin, A. R., Manasse, M. S., Rudolph, L., and Sleator, D. D. 1988. Competitive snoopy caching. *Algorithmica 3,* 1, 79–119.

Kenyon, C. 1996. Best-Fit bin-packing with random order. In *Proc. 7th Annual ACM-SIAM Symp. on Discrete Algorithms.* 359–364.

Krumke, S. O., de Paepe, W. E., Rambau, J., and Stougie, L. 2001. Online bin coloring. In *Proc. 9th Annual European Symp. on Algorithms.* Lecture Notes in Comput. Sci., vol. 2161. Springer-Verlag, 74–85.

Lee, C. and Lee, D. 1985. A simple on-line bin-packing algorithm. *J. ACM 32,* 3, 562–572.

Seiden, S. S. 2002. On the online bin packing problem. *J. ACM 49,* 5, 640–671.

Sleator, D. D. and Tarjan, R. E. 1985. Amortized efficiency of list update and paging rules. *Commun. ACM 28,* 2, 202–208.

Received March 2005; revised September 2005; accepted Month Year