The Relative Worst Order Ratio for On-Line Algorithms

Joan Boyar [*†] Lene M. Favrholdt [*]

Abstract

We consider a new measure for the quality of on-line algorithms, the *relative worst order ratio*, using ideas from the Max/Max ratio [2] and from the Random Order Ratio [7]. The new ratio is used to compare on-line algorithms directly by taking the ratio of their performances on their respective worst orderings of a worst-case sequence. Two variants of the bin packing problem are considered: the Classical Bin Packing Problem and the Dual Bin Packing Problem. Standard algorithms are compared using this new measure. For the Classical Bin Packing Problem, all algorithms considered can be linearly ordered by this measure. For the Dual Bin Packing problem some algorithms are found to be incomparable under this new measure, i.e., there are sequences where the one algorithm does better and sequences where the other does better. Many results obtained are consistent with those previously obtained with the competitive ratio or the competitive ratio on accommodating sequences, but new separations and easier results are also shown to be possible with the relative worst order ratio.

1 Introduction

The standard measure for the quality of on-line algorithms is the competitive ratio [4, 10, 6], which is, roughly speaking, the worst-case ratio, over all possible input sequences, of the on-line performance to the optimal off-line performance. There have been many attempts to provide alternative measures which either give more realistic results than the competitive ratio or do better at distinguishing between algorithms which have very different behaviors in practice. Two very interesting attempts at this are the Max/Max ratio [2] and the Random Order Ratio [7].

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. Consider first the cost A_ℓ defined as $\mathbb{A}_\ell = \max_{|I|=\ell}\{\mathbb{A}(I)/\ell\}$, where $\mathbb{A}(I)$ is the cost of the algorithm $\mathbb{A}$ on the input sequence I. The Max/Max ratio of a k-server algorithm $\mathbb{A}$ is $\lim_{\ell\to\infty}\{\mathbb{A}_\ell/\mathrm{OPT}_\ell\}$, where OPT is an optimal off-line algorithm. The amortized cost of algorithm $\mathbb{A}$ is $\mathbb{A}_\infty = \lim_{\ell\to\infty}\mathbb{A}_\ell$. For any algorithm $\mathbb{A}$ for the k-server problem on bounded metric spaces, the Max/Max ratio is the same as $\mathbb{A}_\infty/\mathrm{OPT}_\infty$. Hence,

[*] Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark, {joan,lenem}@imada.sdu.dk. 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 visiting the Computer Science Department, University of Toronto.

the ratio of the Max/Max ratio of algorithm $\mathbb{A}$ to the Max/Max ratio of algorithm $\mathbb{B}$ is the same as $\mathbb{A}_\infty/\mathbb{B}_\infty$.

Ben-David and Borodin demonstrate that results obtained for the Max/Max ratio may be more optimistic and provide more detailed information than results on the competitive ratio. As an example they give a memoryless k-server algorithm with a Max/Max ratio of $2k$ for any bounded metric space and prove that the competitive ratio of any memoryless k-server algorithm grows unboundedly as a function of the distances in the metric space. Moreover, they show that the best possible Max/Max ratio depends on the metric space; for the uniform metric space with $n \geq k+1$ points, the Max/Max ratio is $\frac{n-1}{n-k}$, i.e., depending on the number of points, the ratio varies from k to values close to 1. In contrast, [9] shows that, for any metric space with at least $k+1$ points, the competitive ratio of any k-server algorithm is at least k. The same paper conjectures that this bound is tight, i.e., an algorithm exists which is k-competitive [9], and it was later proven that the competitive ratio is at most $2k-1$ [8].

The Random Order Ratio. The Random Order Ratio for an on-line algorithm $\mathbb{A}$ for a minimization problem is the maximum ratio over all multi-sets of requests of the expected performance of A compared with OPT on a random permutation of the multi-set. The Random Order Ratio thus gives the possibility of considering some randomness of the request sequences without specifying a complete probability distribution. If, for all possible multi-sets of requests, any ordering of these requests is equally likely, the Random Order Ratio gives a meaningful worst-case measure of how well an algorithm can do. This assumption that all orderings are equally likely may be reasonable for many "real life" problems where further assumptions on the probability distributions of input sequences are difficult to make or vary greatly from one application to another. Many applications of bin packing and scheduling problems may have this property.

Kenyon has shown 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. Furthermore, a whole class of algorithms to which Best-Fit and First-Fit belong have competitive ratio at least 1.7. This is the class of Any-Fit algorithms, which never open a new bin unless the item to be packed does not fit in any of the non-empty bins.

The Relative Worst Order Ratio. Attempting to combine the desirable properties of both 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 orderings of that sequence. Note that the two algorithms may have different "worst orderings" for the same sequence.

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 order in which requests arrive 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 [5] 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 non-negative 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. Dual Bin Packing can be seen as a scheduling problem where we have n machines and would like to schedule as many jobs as possible before a given deadline, whereas with classical bin packing we want to use as few machines as possible to schedule all jobs before a specific deadline. A variant of Dual Bin Packing is the *Fair Bin Packing Problem*, where the algorithms have to be *fair*, i.e., to reject items only when they do not fit in any bin.

Our Results. Many results obtained 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.

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 [5]. First-Fit is still the best Any-Fit algorithm and Next-Fit is no better than any Any-Fit algorithm.

The worst order ratio for any fair deterministic algorithm for the Dual Bin Packing Problem is not bounded below by any constant. In contrast, with the relative worst order ratio, one gets constant ratios. We find that First-Fit does at least as well as any Any-Fit algorithm and is better than Best-Fit, which is better than Worst-Fit. Worst-Fit is at least as bad as any fair on-line algorithm and strictly worse than First-Fit. This contrasts favorably with the competitive ratio, where Worst-Fit is better than First-Fit [3].

Unfair-First-Fit, an algorithm which is not fair and does better than First-Fit when using the competitive ratio on accommodating sequences, is incomparable to all Any-Fit algorithms.

2 The (Relative) Worst Order Ratio

This paper considers the worst order ratio as well as the relative Worst Order Ratio. The relative worst order ratio appears to be the more interesting measure for the two variants of bin packing studied here.

The definition of the relative worst order ratio uses $\mathbb{A}_{\mathrm{W}}(I)$, 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 2.1 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.

For a *maximization problem*, $\mathbb{A}(I)$ is the *value* of running the on-line algorithm $\mathbb{A}$ on I, and $\mathbb{A}_{\mathrm{W}}(I) = \min_{\sigma} \mathbb{A}(\sigma(I))$.

For a *minimization* problem, $\mathbb{A}(I)$ is a *cost*, and $\mathbb{A}_{\mathrm{W}}(I) = \max_{\sigma} \mathbb{A}(\sigma(I))$. □

Definition 2.2 Let $S_1(c)$ be the statement

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

and let $S_2(c)$ be the statement

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*.

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

□

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.

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

Table 1: Ratio values for minimization and maximization problems

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

Theorem 2.1 The ordering of algorithms for a specific problem is transitive.

Proof Suppose that three algorithms $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$ for some on-line problem, P, are such that $\mathbb{A}$ is at least as good as $\mathbb{B}$ and $\mathbb{B}$ is at least as good as $\mathbb{C}$, as measured by the relative Worst Order Ratio. If P is a minimization problem there exists a constant b such that for all I, $A_W(I) \leq B_W(I) + b$, and there exists a constant d such that for all I, $B_W(I) \leq C_W(I) + d$, so there exist constants b and d such that for all I, $A_W(I) \leq C_W(I) + b + d$, Thus $\mathbb{A}$ also performs at least as well as $\mathbb{C}$, according to the relative worst order ratio. The argument for a maximization problem is essentially the same. □

Finally we define the worst order ratio formally:

Definition 2.3 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., $\mathrm{WR}_{\mathbb{A}} = \mathrm{WR}_{\mathbb{A},\mathrm{OPT}}$. □

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, because of the restrictions on OPT's behavior. Clearly, the worst order ratio is never worse than the competitive ratio.

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.

We consider *Any-Fit algorithms*, a class of fair algorithms defined by Johnson [5], which use an empty bin, only if there is not enough space in any partially full bins. Three examples of Any-Fit algorithms are First-Fit (FF), which places an item in the first bin in which it fits, Best-Fit (BF), which places an item in one of the fullest bins in which it fits, and Worst-Fit (WF), which will place an item in a least full open bin. We also consider an algorithm, Next-Fit (NF), which is not an Any-Fit algorithm. 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.

Most of the results we obtain with the relative worst order ratio for the Classical Bin Packing Problem are very similar to those obtained for the competitive ratio and use similar techniques. However, the relative worst order ratio separates Worst-Fit and Next-Fit, which the competitive ratio cannot. We first present those results consistent with the competitive ratio.

3.1 First-Fit is the Best Any-Fit Algorithm

According to the competitive ratio for the Classical Bin Packing Problem, First-Fit is the best Any-Fit algorithm [5]. Using his proof, one can show that this also holds for the relative worst order ratio.

Theorem 3.1 The relative worst order ratio of First-Fit to $\mathbb{A}$, for any Any-Fit algorithm $\mathbb{A}$, is at most 1.

Proof Consider a request sequence I and its packing by First-Fit. Create a new sequence I' from I 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'. $\mathbb{A}$ will place the items in I' exactly as First-Fit places the items in I, since it 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 order I', $\mathbb{A}$ uses as many bins as First-Fit does with I. Hence, for all I, $\mathrm{FF}_{\mathrm{W}}(I) \leq \mathbb{A}_{\mathrm{W}}(I)$, giving a ratio of at most one. □

3.2 Worst-Fit is the Worst Any-Fit Algorithm

Not all Any-Fit algorithms perform as well. Worst-Fit is the worst among the Any-Fit algorithms, and it is significantly worse than First-Fit and Best-Fit.

Theorem 3.2 The relative worst order ratio of Worst-Fit to any Any-Fit algorithm $\mathbb{A}$ is at least 1.

Proof Consider a request sequence I and its packing by $\mathbb{A}$. Call the bins used by $\mathbb{A}$ $b_1, b_2, \ldots, b_n$, numbered according to the order in which they were opened by $\mathbb{A}$. 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. Let $\ell_{\mathbb{A}}^{\min}(j) = \min_{1 \le i \le j} \{\ell(b_i)\}$. Furthermore, let E_j be the set of items packed in b_j, $1 \le j \le n$.

Let I' be a permutation of I, where each item $e \in E_i$ appears before each item $e' \in E_j$, for $1 \le i < j \le n$. Consider the packing of I' produced by Worst-Fit. Let $\ell_{\mathrm{WF}}(b_j)$ be the level of b_j in this packing. We prove by induction on j that, for $1 \le j \le n$, $\ell_{\mathrm{WF}}(b_j) \ge \ell_{\mathbb{A}}^{\min}(j)$ and all items in $\cup_{i=1}^{j} E_i$ are packed in $b_1, \ldots, b_j$ by Worst-Fit.

The base case $j = 1$ is trivial: all items in E_1 are clearly packed in b_1, since Worst-Fit is an Any-Fit algorithm.

For the induction step with $j > 1$, the induction hypothesis says that $\ell_{\mathrm{WF}}(b_i) \ge \ell_{\mathbb{A}}^{\min}(i)$, $1 \le i \le j-1$, and that before giving the items of E_j, b_j is empty. If Worst-Fit packs all items of E_j in b_j, the result trivially follows. Assume now, that some item in E_j is packed in some bin $b_i \ne b_j$. Since b_j is empty before giving the items of E_j and Worst-Fit is an Any-Fit algorithm, we conclude that $i < j$. By the Worst-Fit packing rule, $\ell_{\mathrm{WF}}(b_j) \ge \ell_{\mathrm{WF}}(b_i) \ge \ell_{\mathbb{A}}^{\min}(i) \ge \ell_{\mathbb{A}}^{\min}(j)$.

Now, let e_j be the first item packed by $\mathbb{A}$ in b_j, $1 \le j \le n$. Since $\mathbb{A}$ is an Any-Fit algorithm, e_j does not fit in b_i, $1 \le i < j$. This means that e_j is larger than $1 - \ell_{\mathbb{A}}^{\min}(j-1)$. Since, in the Worst-Fit packing, b_i has a level of at least $\ell_{\mathbb{A}}^{\min}(i)$, this means that for each b_i and each e_j, $1 \le i < j \le n$, e_j does not fit in b_i. In words, for each bin b_i, the bottommost item in each bin b_j, $1 \le i < j \le n$, in the packing of $\mathbb{A}$ does not fit in b_i in the Worst-Fit packing. Hence, for each e_j, $1 \le j \le n$, 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 [5], one can show that the relative worst order ratio of Worst-Fit to either First-Fit or Best-Fit is 2.

Theorem 3.3 The relative worst order ratio of Worst-Fit to First-Fit or Best-Fit is 2.

Proof Since $\mathrm{FF_W}(I) \le \mathrm{BF_W}(I)$ for all I, by the proof of Theorem 3.1, it is only necessary to compare Worst-Fit and Best-Fit. The above theorem shows that $\mathrm{WR}_{\mathrm{WF,BF}} \ge 1$, so in order to prove a lower bound of 2, it is sufficient to find a family of sequences I_n, with $\lim_{n\to\infty} \mathrm{WF_W}(I_n) = \infty$, where there exists a constant b such that for all I_n, $\mathrm{WF_W}(I_n) \ge 2\mathrm{BF_W}(I_n) - b$. The family of sequences used in [5] to bound Worst-Fit's competitive ratio works here. Let $0 < \varepsilon \le \frac{1}{2n}$. Consider the sequence I_n with pairs $(\frac{1}{2}, \varepsilon)$, for $i = 1...n$. 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 ordering, using only $\lceil \frac{n+1}{2} \rceil$ bins. Thus, $\mathrm{WF_W}(I_n) = n \ge 2\lceil \frac{n+1}{2} \rceil - 2 = 2\mathrm{BF_W}(I_n) - 2$, so the relative worst order ratio is at least 2.

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 [5]. □

3.3 Next-Fit is Worst Than Worst-Fit

Now we consider Next-Fit, which is not an Any-Fit algorithm. First, we show that all Any-Fit algorithms perform at least as well as Next-Fit.

Theorem 3.4 The relative worst order ratio of Next-Fit to $\mathbb{A}$, for any Any-Fit algorithm $\mathbb{A}$, is at least 1.

Proof Consider a request sequence I and its packing by $\mathbb{A}$. Create a new sequence I' from I 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, $\mathrm{NF}_{\mathrm{W}}(I) \geq \mathbb{A}_{\mathrm{W}}(I)$, giving a ratio of at least one. $\square$

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 [5].

Theorem 3.5 The relative worst order ratio of Next-Fit to any Any-Fit algorithm $\mathbb{A}$ is 2.

Proof The above theorem shows that $\mathrm{WR}_{\mathrm{NF},\mathbb{A}} \geq 1$, so in order to prove a 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, $\mathrm{NF}_{\mathrm{W}}(I_n) \geq 2\mathbb{A}_{\mathrm{W}}(I_n) - b$. 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, $\mathrm{NF}_{\mathrm{W}}(I_n) = 2(n-1) = 2\mathbb{A}_{\mathrm{W}}(I_n) - 2$, so $\mathrm{WR}_{\mathrm{NF},\mathbb{A}} \geq 2$.

The relative worst order ratio of Next-Fit to any Any-Fit algorithm is at most 2, since Next-Fit's competitive ratio is 2 [5]. $\square$

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 of greater than 1, instead of less than 1, to the poorer algorithms.

First-Fit and Best-Fit are defined in the same manner for the 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. If one uses the same definition for Worst-Fit for dual bin packing as for classical bin packing, one can again show that it is the worst Any-Fit algorithm. However, we use a more natural definition for the fixed number of bins, where 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 here 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 opened for every item. This variant of Worst-Fit is clearly the worst possible algorithm for the classical problem.)

The result we obtain for the worst order ratio for the Dual Bin Packing Problem is similar to that obtained previously with the competitive ratio, while most of the results we obtain for the relative worst order ratio are similar to those with competitive ratio on accommodating sequences. The proof that First-Fit and Best-Fit are strictly better than Worst-Fit, which is true with the relative Worst Order Ratio and the competitive ratio on accommodating sequences, but not with the competitive ratio, is much easier using the relative worst order ratio.

4.1 The Worst Order Ratio

Computing the worst order ratio for deterministic algorithms for fair bin-packing, not surprisingly, gives similar results to the competitive ratio [3] — very pessimistic results.

Theorem 4.1 The worst order ratio for any fair deterministic algorithm for the Dual Bin Packing Problem is not bounded below by any constant.

Proof Consider any fair, deterministic on-line algorithm, $\mathbb{A}$. For the following sequence, I, defined on the basis of $\mathbb{A}$'s performance, $\mathbb{A}$ 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. Suppose $\mathbb{A}$ places these items 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. Since it is fair, $\mathbb{A}$ is forced to accept all of the large items, completely filling up the bins, so that it must reject all the small items. $\mathbb{A}_{\mathrm{W}}(I) = 3n$.

In the worst ordering 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 ordering $I' = \sigma(I)$. Without loss of generality, we may assume that all of the large items in I' come before all of the small. In such a sequence, all items of type D are accepted since there are only $3n$ large items in all, and an item of type D will fit in any bin which only contains two large items.

In addition, we may assume that all of the items of type D come after all of the larger items. To see this, suppose there exists an item x of type D, which occurs before some larger item y. Choose these items so that x occurs immediately before y. Now consider the sequence I'', which is identical to I', except that the order of x and y is inverted. OPT accepts x in both orderings, since it is of of type D. If it also accepts y in I', it will accept it in I'' too. If it rejects y in I', and also in I'', I'' is also a worst ordering. If it rejects y in I', but accepts it in I'', then its decision to reject y in I' was because that was better, so I'' is at least as bad as I'. One can continue in this manner moving all items of type D after the larger ones.

A similar argument shows that all of the items of type A can be assumed to come before all of the others.

If the items of type A are placed one per bin, then the items of type B will also have to be placed with at most one per bin, so at least q are rejected.

If the items of type A are placed so that r bins are empty, then those r bins will each get three large items, and r bins will get exactly two items of type A, plus possibly one of type D. The remaining $n-2r$ bins can each hold at most two items of type B or C. Thus, there is space for at most $3r+2(n-2r)=2n-r$ items of types B or C. This is smallest when $r=\frac{n}{2}$, its largest possible value. There are $2n-q$ items of types B and C, so $\frac{n}{2}-q$ are rejected.

OPT can choose whether to place the items of type A one per bin or two per bin, whichever leads to more rejected large items. If $q \geq \frac{n}{4}$, it will choose the first; otherwise it will choose the second. In any case, it rejects $s \geq \frac{n}{4}$ large items and accepts at least $s(\frac{1}{3}-\varepsilon)/\varepsilon$ small items.

Thus, the worst order ratio is at most

$$\begin{aligned}
\mathrm{WR}_{\mathbb{A}} &\leq \frac{3n}{(3n-s)+s(\frac{1}{3}-\varepsilon)/\varepsilon} = \frac{3n\varepsilon}{(3n-s)\varepsilon+s(\frac{1}{3}-\varepsilon)} \\
&= \frac{3n\varepsilon}{(3n-2s)\varepsilon+\frac{s}{3}} \leq \frac{3n\varepsilon}{(3n-\frac{n}{2})\varepsilon+\frac{n}{12}} = \frac{36\varepsilon}{30\varepsilon+1} < 36\varepsilon.
\end{aligned}$$

□

4.2 The Relative Worst Order Ratio

Note that for the Dual Bin Packing Problem, the competitive ratio on accommodating sequences [3] 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 orderings 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 of at most $r<p$, then there is an accommodating sequence I where $\mathbb{A}_{\mathrm{W}}(I) \geq p|I| > r|I| \geq \mathbb{B}_{\mathrm{W}}(I)$. 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.

4.3 First-Fit is the Best Any-Fit Algorithm

For dual bin packing, one can again show that First-Fit is the best Any-Fit algorithm, also using the proof by Johnson [5].

Theorem 4.2 The relative worst order ratio of First-Fit to $\mathbb{A}$, for any Any-Fit algorithm $\mathbb{A}$, is at least 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 create for $\mathbb{A}$ and will also be rejected by $\mathbb{A}$. □

Theorem 4.3 The relative worst order ratio of First-Fit to Best-Fit is at least $\frac{7}{6}$.

Proof The above theorem shows that $\mathrm{WR}_{\mathrm{FF,BF}} \geq 1$, so it is sufficient to find a family of sequences I_n, with $\lim_{n\to\infty} \mathrm{FF}_{\mathrm{W}}(I_n) = \infty$, where there exists a constant b such that for all I_n, $\mathrm{FF}_{\mathrm{W}}(I_n) \geq \frac{7}{6}\mathrm{BF}_{\mathrm{W}}(I_n) - b$. Let $0 < \varepsilon < \frac{1}{8n^2}$. Consider the sequence I_n starting with pairs, $(\frac{1}{2} + 2ni\varepsilon, \varepsilon)$, for $i = 0...n-1$ and followed by $\frac{1}{2} - n(2i+1)\varepsilon$, for $i = 0,..n-1$ and $n-1$ of size $n\varepsilon$. Best-Fit will reject the last $n-1$ items when they are given in this order. The worst order for First-Fit would be such that First-Fit paired together the items of size just less than $\frac{1}{2}$, so that it could only accept $\lfloor \frac{n}{2} \rfloor$ of those larger than $\frac{1}{2}$. Thus, $\mathrm{FF}_{\mathrm{W}}(I_n) \geq 3n - 1 + \frac{n-1}{2} \geq \frac{7}{6}(3n) - \frac{3}{2} = \frac{7}{6}\mathrm{BF}_{\mathrm{W}}(I_n) - \frac{3}{2}$, as required. □

4.4 Worst-Fit is the Worst Fair Algorithm

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. Its relative worst order ratio to any other fair algorithm is less than or equal to one, so it is the worst such algorithm.

Theorem 4.4 The relative worst order ratio of Worst-Fit to $\mathbb{A}$, for any fair algorithm $\mathbb{A}$, is less than or equal to 1.

Proof Consider a request sequence I and its packing by $\mathbb{A}$. Create a new sequence I' from I as follows: As in the proof of Theorem 3.2, 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 $\mathbb{A}$ in the same bin as x. Let I' consist of the items in I sorted by $b(x)$, with the items $\mathbb{A}$ rejects placed at the end. Note that Worst-Fit will create essentially the same packing with I' as $\mathbb{A}$ did with I; 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. Thus, with this order, Worst-Fit will reject at least as many items of I' as $\mathbb{A}$ does with I. Hence, for all I, $\mathrm{WF}_{\mathrm{W}}(I) \leq \mathbb{A}_{\mathrm{W}}(I)$, giving a ratio of at most one. □

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 First-Fit and Best-Fit actually have worse ratios than Worst-Fit [3]. 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 [3]. The result concerning the relative worst order ratio is, however, much easier to prove.

Lemma 4.1 The relative worst order ratio of Worst-Fit to either First-Fit or Best-Fit is at most $\frac{n}{2n-1}$.

Proof The proof of the above theorem shows that there is no sequence I where $\mathrm{WF}_{\mathrm{W}}(I) > \mathrm{FF}_{\mathrm{W}}(I)$ or $\mathrm{WF}_{\mathrm{W}}(I) > \mathrm{BF}_{\mathrm{W}}(I)$, so it is sufficient to find a family of sequences I_n, with $\lim_{n\to\infty} \mathrm{FF}_{\mathrm{W}}(I_n) = \infty$, where there exists a constant b such that for all I_n, $\mathrm{WF}_{\mathrm{W}}(I_n) \leq \frac{1}{2}\mathrm{BF}_{\mathrm{W}}(I_n) + b$. 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. First-Fit or Best-Fit will accept all of these items, regardless of their order. This gives a ratio of $\frac{n}{2n-1}$. □

The above result is tight:

Lemma 4.2 The relative worst order ratio of Worst-Fit to either First-Fit or Best-Fit is at least $\frac{n}{2n-1}$.

Proof Since Worst-Fit cannot have a lower ratio to Best-Fit than to First-Fit, it is sufficient to show that this holds for First-Fit. Consider any sequence I and the worst ordering of I for Worst-Fit. Without loss of generality, assume that all the items Worst-Fit accepts appear in I before those it rejects. Consider First-Fit's performance on this ordering of I, and suppose it accepts m items, but Worst-Fit only accepts $m' < m$.

Reorder the first m' items 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. 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 $m - m'$ 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.

Since Worst-Fit is fair, it must accept at least n items if it rejects any at all. Thus, the relative worst order ratio of Worst-Fit to either First-Fit or Best-Fit is at least $\frac{n}{2n-1}$. □

Theorem 4.5 The relative worst order ratio of Worst-Fit to either First-Fit or Best-Fit is $\frac{1}{2}$.

4.5 Unfair-First-Fit is Incomparable to Any Any-Fit Algorithm

An example of an algorithm for the Dual Bin Packing Problem which is not fair is Unfair-First-Fit [1]. It behaves as First-Fit, except that when given a request of size greater than 1/2, it automatically rejects that item if it has already accepted at least 2/3 of the items it has received so far. The intuition is that by rejecting some large items, it may have more room for more small items. The algorithm is defined in Figure 1.

Theorem 4.6 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 request sequence containing n items of size 1. Unfair-First-Fit will only accept 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 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 $\frac{2}{3} \pm \Theta(\frac{1}{n})$, for large enough n [1], there exists an accommodating sequence where Unfair-First-Fit outperforms First-Fit. Asymptotically, Unfair-First-Fit accepts $\frac{16}{15}$ times as many items as First-Fit. □

```
Input:  S = ⟨o_1, o_2, ..., o_n⟩
Output:  A, R, and a packing for those items in A
A:={}; R:={}
while S ≠ ⟨⟩
    o:=hd(S); S:=tail(S)
    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}
```

Figure 1: The algorithm Unfair-First-Fit

5 Conclusion and Open Problems

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

The definition presented here for the relative worst order ratio allows one to say that $\mathbb{A}$ is better than $\mathbb{B}$ even if there exist sequences where $\mathbb{B}$ does better than $\mathbb{A}$ by an additive constant. Another possible definition would require that $\mathbb{A}$ do at least as well as $\mathbb{B}$ on every sequence. In all the examples of comparable algorithms presented here, one algorithm does at least as well as the other on every sequence. It would be interesting to find an example showing that these definitions are different.

For the Classical Bin Packing Problem, there exist multi-sets where First-Fit's worst ordering uses one less bin than Best-Fits's worst ordering. 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 ordering, while First-Fit only uses two. However this seems to be hard to extend to an asymptotic result. Determining if $\mathrm{WR}_{\mathrm{FF,BF}} < 1$ is an open problem.

The most interesting open problem, though, is for which other problems does the relative worst order ratio give interesting and new results.

6 Acknowledgments

We would like to thank Kim Skak Larsen for helpful discussions.

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, 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. Also in *WADS 99*, pages 74–79.

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

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

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

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

[8] E. Koutsoupias and C. H. Papadimitriou. On the k-Server Conjecture. *Journal of the ACM*, 42(5):971–983, 1995.

[9] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive Algorithms for On-Line Problems. In *20th Annual ACM Symposium on the Theory of Computing*, pages 322–333, 1988.

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