The Relative Worst Order Ratio Applied to Seat Reservation

Joan Boyar* and Paul Medvedev

Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark.
{joan,pashadag}@imada.sdu.dk

Abstract. The relative worst order ratio is a new measure for the quality of on-line algorithms, which has been giving new separations and even new algorithms for a variety of problems. Here, we apply the relative worst order ratio to the Seat Reservation Problem, the problem of assigning seats to passengers in a train. We consider the unit price problem, where all tickets have the same cost, and the proportional price problem, where the ticket price is proportional to the distance travelled.
Using the relative worst order ratio, we show that First-Fit and Best-Fit are better than Worst-Fit, for the unit price problem, even though they have not been separated using the competitive ratio. The same relative worst order ratio result holds for the proportional price problem, where no deterministic algorithm has a competitive ratio, or even a competitive ratio on accommodating sequences, which is bounded below by a constant.
Comparing algorithms to OPT with the relative worst order ratio gives the worst order ratio. The worst order ratio of any deterministic algorithm for either the unit price problem or the proportional price problem is always bounded above by the competitive ratio on accommodating sequences for the algorithm and bounded below by the competitive ratio on accommodating sequences for some algorithm. Thus, for the unit price problem, the worst order ratio of any algorithm is at least $\frac{1}{2}$, even though the competitive ratio is not bounded below by any constant. This gives a much more optimistic and realistic view of how well the algorithms perform, but it is still necessary to use the relative worst order ratio to separate their performances.

1 Introduction

The standard measure for the quality of on-line algorithms has been the competitive ratio [10,15,12], which is, roughly speaking, the worst-case ratio, over all possible input sequences, of the on-line performance to the optimal off-line performance. In many cases, the competitive ratio has been quite successful in predicting the performance of algorithms. However, in many others, it has given results that are either counter-intuitive or counter to the experimental data. There is therefore a need to develop better performance measures that, at the very least, would supplement the competitive ratio.

The competitive ratio resembles the approximation ratio, as on-line algorithms can be viewed as a special case of approximation algorithms. However, while it seems natural to compare an approximation algorithm to an optimal algorithm, which solves the same problem in unlimited time, it does not seem as natural to compare an on-line algorithm to an off-line optimal algorithm, which actually solves a different problem (an off-line version). Additionally, when there is need to compare two on-line algorithms against each other, it seems more appropriate to compare them directly, rather than involve an intermediate comparison to an optimal off-line algorithm.

For this reason, a new performance measure for the quality of on-line algorithms has been developed [3]. This measure, the relative worst order ratio, allows on-line algorithms to be compared directly to each other. It combines the desirable properties of some previously

* Supported in part by the Danish Natural Science Research Council (SNF).

considered performance measures, namely the Max/Max ratio [2] and the random order ratio [13]. To compare two algorithms, we consider a worst-case sequence and take the ratio of how the two algorithms do on their respective worst orderings of that sequence. Though intended for direct comparison of on-line algorithms, the relative worst order ratio may also be used to compare an on-line algorithm to the optimal off-line algorithm, in which case it more closely parallels the competitive ratio. We then refer to the ratio as simply the worst order ratio.

The relative worst order ratio has already been applied to some problems and has led to more intuitively and/or experimentally correct results than the competitive ratio, as well as to new algorithms. For paging, for example, it has shown that Least-Recently-Used(LRU) is strictly better than Flush-When-Full(FWF) [5], even though both algorithms have the same competitive ratio. This result is intuitively correct. Additionally, although LRU is an optimal deterministic algorithm according to the competitive ratio, a new algorithm RLRU has been discovered, which not only has a better relative worst order ratio than LRU, but is experimentally better as well according to initial testing [5]. It has also been shown that look-ahead is a significant advantage [5], which is again an intuitively correct result that is in contrast with the competitive ratio, which does not reflect that look-ahead can be helpful. Other problems where the relative worst order ratio has given more correct results are bin packing [3,4], scheduling [9], and bin coloring [14].

Given these encouraging results, this paper will use the relative worst order ratio to analyze algorithms for the seat reservation problem. This problem is defined in [7] as the problem of assigning as many passengers as possible, to seats on a train with n seats and k stations en-route, in an on-line manner. We focus on deterministic algorithms, although randomized algorithms for this problem have also been studied [7,1]. Three algorithms are studied: First-Fit, Best-Fit, and Worst-Fit. There are two variants of the seat reservation problem: the unit price problem and the proportional price problem.

Table 1. Bounds for the competitive ratio.

	Unit Price	Proportional Price
Any det. alg.	$\frac{2}{k} \leq r \leq \frac{8}{k+5}$	$\frac{1}{k-1} \leq r \leq \frac{4+2\sqrt{13}}{3+2\sqrt{13}+k}$
Worst-Fit	$\frac{2}{k} \leq r \leq \frac{4}{k+1}$	$r = \frac{1}{k-1}$
First-Fit/Best-Fit	$\frac{2}{k} \leq r \leq \frac{2-\frac{1}{k-1}}{k-1}$	$\frac{1}{k-1} \leq r \leq \frac{4}{k+2}$

The competitive ratio has been applied to both variants in [7,1,6], and the known results are summarized in Table 1[1]. Note that the competitive ratio is $\Theta(\frac{1}{k})$ for all deterministic algorithms (randomized as well [7]), and thus not bounded below by a constant independent of k (recall that for a maximization problem, a low competitive ratio implies a bad algorithm). Given the results so far, the only definite conclusion that can be made about the relative performance of these algorithms is that Worst-Fit is no better than any other deterministic algorithm for the proportional price problem. Additionally for the unit price problem, given that First-Fit and Best-Fit have the same bounds on their competitive ratios, and that these

[1] All bounds come directly from [7], with the following exceptions: The upper bound on Worst-Fit for unit price follows from the proof of Theorem 8 in [7]. The upper bound on Worst-Fit for proportional price follows from the worst-case sequence used in Theorem 8 in [6].

bounds are asymptotically tight, it seems unlikely that First-Fit and Best-Fit can be separated. However, it seems hard to draw any other conclusions about the relative performance of these algorithms. Combined with the pessimistic upper bounds on the competitive ratios, these very inconclusive results make the seat reservation problem an ideal candidate to study with the relative worst order ratio.

1.1 Our Results

Using the relative worst order ratio, we are able to differentiate all three algorithms, for both the unit price and the proportional price problems, something that has not been done with the competitive ratio. We show that for a category of algorithms called Any-Fit, which includes both First-Fit and Best-Fit, First-Fit is at least as good as any other algorithm. Moreover, First-Fit is strictly better than Best-Fit with a relative worst order ratio of at least $\frac{4}{3}$ for the unit price problem and at least $\frac{k+2}{6}$ for the proportional price problem. We also show that Worst-Fit is at least as bad as any other deterministic algorithm, and is strictly worse than any Any-Fit algorithm by a ratio of at least $2 - \frac{1}{k-1}$ for the unit price problem and exactly $k - 1$ for the proportional price problem.

Additionally, we find that, for the seat reservation problem, an algorithm's worst order ratio is bounded from above by the competitive ratio on accommodating sequences for the algorithm (defined below) and bounded below by the competitive ratio on accommodating sequences for some algorithm. This gives bounds for the worst order ratio of $\frac{1}{2} \le r \le \frac{1}{2} + \frac{3n-3}{2k+6n-(8+2c)}$, where $c \equiv k - 1 \pmod 6$, for the unit price problem. This is a more useful estimate of how an algorithm performs than the competitive ratio, which is not bounded below by a constant.

2 The Seat Reservation Problem: Definitions and Algorithms

The *seat reservation problem* was originally studied in [7]. We consider a scenario where a train with n seats travels on a route passing through $k \ge 2$ stations, including the first and the last. The seats are numbered from 1 to n. The start station is station 1 and the end station is station k. A customer may, any time prior to departure, request a ticket for travel between stations s and f, where $1 \le s < f \le k$. At that time, the customer is assigned a single seat number, which cannot be changed. It is the role of the algorithm (ticket agent) to determine which seat number to assign. The customer may be refused a ticket only in the case when there is no single seat which is empty for the duration of the request. An algorithm which obeys this rule is called *fair*, and all algorithms for this problem must be fair.

The seat reservation problem is, by its very nature, an on-line problem. The algorithm attempts to maximize income, i.e., the total price of the tickets sold. The performance of an algorithm will thus clearly depend on the ticket pricing policy. In this paper, we consider two variants. In the *unit price problem*, the price of all tickets is the same. In the *proportional price problem*, the price of a ticket is directly proportional to the distance traveled. Some of the results we prove hold for any pricing policy where all tickets have positive cost; we refer to such results as holding "regardless of pricing policy."

Before continuing, we introduce some basic notation. We use the notation $x = [x_{\mathrm{s}}, x_{\mathrm{f}})$ to denote an interval x from station x_{s} to station x_{f}, where $1 \le x_{\mathrm{s}} < x_{\mathrm{f}} \le k$. We say an interval x is a subinterval of the interval y if $y_{\mathrm{s}} \le x_{\mathrm{s}}$ and $x_{\mathrm{f}} \le y_{\mathrm{f}}$. Since a *request* is just an interval, we will use the terms interchangibly, depending on what is more natural at the time. The *length* of an interval (request) x is simply $x_\mathrm{s} - x_\mathrm{f}$.

The following three algorithms which we will consider are all inspired by Bin Packing algorithms of the same name.

First-Fit This algorithm will place a request on the first seat which is unoccupied for the length of the journey.

Best-Fit Assume we have an interval x and a seat which is empty for that interval. The *empty space* containing x is the maximum length of a request which could be placed on that seat and which contains x as a subinterval. This algorithm will place a request on a seat such that the empty space containing that request is minimized. We note that to fully define the algorithm we must also specify a tie-breaker, that is, what happens when there is more than one such seat. However, since we would like to keep our results as widely applicable as possible, we will not assume any specific tie-breaker in any of our proofs. Our results will thus hold for any choice of a tie-breaker for Best-Fit. In some cases, bounds could be tightened with knowledge of the tie-breaker[2]. However, these improvements are minor, and do not change the meaning of the results.

Worst-Fit This algorithm will place a request on a seat such that the empty space containing that request is maximized. Again, we assume that any tie-breaker may be chosen, and our results hold for all such choices. In this case, however, knowledge of the tie-breaker would not help tighten any of our bounds.

Additionally, we will look at the following class of algorithms, also inspired by a class of Bin Packing algorithms of the same name defined by Johnson in [11].

Any-Fit At any given time, we say that a seat is *active* if at least one request has been assigned to it, and *inactive* otherwise. An algorithm which belongs to this class will place a request on an inactive seat only if it does not fit into any of the active seats.

3 The Relative Worst Order Ratio

In this section, we define the relative worst order ratio and the notion of two algorithms being comparable (Definition 2) as in [3], though, for the sake of simplicity, only for maximization problems, such as the seat reservation problem.

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 permutation" of the sequence I of requests, formally defined as follows:

Definition 1. *Consider an on-line maximization 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. $\mathbb{A}(I)$ is the value of running $\mathbb{A}$ on I, and $\mathbb{A}_W(I) = \min_\sigma \mathbb{A}(\sigma(I))$.*

Definition 2. *Let $S_1(c)$ and $S_2(c)$ be statements about algorithms $\mathbb{A}$ and $\mathbb{B}$ defined in the following way.*

$S_1(c)$: *There exists a constant b such that $\mathbb{A}_W(I) \leq c \cdot \mathbb{B}_W(I) + b$ for all I.*

[2] Specifically, the relative worst order ratio of First-Fit to Best-Fit can be slightly improved in Theorem 3 and in Theorem 6.

$S_2(c)$: *There exists a constant b such that $\mathbb{A}_W(I) \geq c \cdot \mathbb{B}_W(I) - b$ for all I.*

The relative worst order ratio *$WR_{\mathbb{A},\mathbb{B}}$ of on-line algorithm $\mathbb{A}$ to algorithm $\mathbb{B}$ is defined if $S_1(1)$ or $S_2(1)$ holds. In this case, $\mathbb{A}$ and $\mathbb{B}$ are said to be* comparable. *If $S_1(1)$ holds, then $WR_{\mathbb{A},\mathbb{B}} = \sup\{r \mid S_2(r)\}$, and if $S_2(1)$ holds, then $WR_{\mathbb{A},\mathbb{B}} = \inf\{r \mid S_1(r)\}$.*

Note that if $S_1(1)$ holds, the supremum involves S_2 rather than S_1, and vice versa. A ratio of 1 means that the two algorithms perform identically with respect to this quality measure; the further away from 1, the greater the difference in performance. The ratio is greater than one if the first algorithm is better and less than one if the second algorithm is better. In most cases, it is convenient to place the better algorithm first.

It is easily shown [3] that the relative worst order ratio is a transitive measure, i.e., for any three algorithms $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$, $\mathrm{WR}_{\mathbb{A},\mathbb{B}} \leq 1$ and $\mathrm{WR}_{\mathbb{B},\mathbb{C}} \leq 1$ implies $\mathrm{WR}_{\mathbb{A},\mathbb{C}} \leq 1$.

4 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, $\mathbb{A}$, 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* [3] and is denoted $\mathrm{WR}_{\mathbb{A}}$. This ratio can be used to bound the relative worst order ratio between two algorithms and in some cases gives tight results. Thus, although it is generally most interesting to compare on-line algorithms directly to each other, the worst order ratio can also be interesting.

In this section, we show a connection between the worst order ratio and the competitive ratio on accommodating sequences[3], which is relevant to the seat reservation problem when the management has made a good guess as to how many seats are necessary for the expected number of passengers.

A sequence for which all requests can be accepted within n seats is called an *accommodating sequence.* For a maximization problem, an algorithm $\mathbb{A}$ is *c-competitive on accommodating sequences* if, for every accommodating sequence I, $\mathbb{A}(I) \geq c \cdot \mathrm{OPT}(I) - b$, where b is a fixed constant for the given problem, and, thus, independent of I. The *competitive ratio on accommodating sequences $\mathcal{A}$* is defined as

$$\sup\{c \mid \mathbb{A} \text{ is } c\text{-competitive on accommodating sequences}\}.$$

The major result of this section shows that the worst order ratio for any memoryless, deterministic algorithm for the seat reservation problem, regardless of the pricing policy, is equal to its competitive ratio on accommodating sequences. An algorithm is *memoryless* if it never uses any information about anything but the current request and the current configuration (which requests have been placed where) in making a decision about the current request. In particular, a memoryless algorithm never uses information about the order the requests came in or about any of the rejected requests. All algorithms considered in this paper are memoryless.

In the proof showing this connection, it is shown that there is a permutation of a particular subsequence which will force OPT to accept every item in that subsequence, using the following lemma:

[3] The competitive ratio on accommodating sequences was first studied in [7], but called the *accommodating ratio* there.

Lemma 1. *Any algorithm $\mathbb{A}$ for the seat reservation problem will accept all requests in any accommodating sequence, I, if the requests in I are in nondecreasing order by left endpoint.*

Proof. Consider any request, $r = [r_s, r_f)$, in the sequence, I. Since the sequence is accommodating, there are at most n requests containing the subinterval $[r_s, r_{s+1})$. Thus, when r occurs in the sequence, there is some seat which $\mathbb{A}$ has left empty from r_s to r_{s+1}. Since the requests in I are in nondecreasing order according to the left endpoint, if the seat is empty from r_s to r_{s+1}, it has nothing placed to the right of r_s on that seat, yet. Since any algorithm for the seat reservation problem is fair, the request will be accepted. Thus, the entire sequence will be accepted. □

Theorem 1. *Let $\mathbb{A}$ be a deterministic algorithm for the seat reservation problem. If $\mathbb{A}$ is memoryless, then $\mathbb{A}$'s worst order ratio and its competitive ratio on accommodating sequences are equal, regardless of the pricing policy. Otherwise, $\mathbb{A}$'s worst order ratio is no larger than its competitive ratio on accommodating sequences and at least the competitive ratio on accommodating sequences of some algorithm.*

Proof. We first prove that if $\mathrm{WR}_{\mathbb{A}} \geq c$, then $\mathbb{A}$ is c-accommodating. Assuming that $\mathrm{WR}_{\mathbb{A}} \geq c$ implies that there exists a constant b such that $\mathbb{A}_W(I) \geq c \cdot \mathrm{OPT}_W(I) - b$ for all input sequences I. It follows from definitions that $\mathbb{A}(I) \geq \mathbb{A}_W(I)$ and $\mathrm{OPT}_W(I) = \mathrm{OPT}(I)$ for all accommodating sequences I. Combining these facts, we can say that there exists a constant b such that $\mathbb{A}(I) \geq c \cdot \mathrm{OPT}(I) - b$ for all accommodating sequences I, so $\mathbb{A}$ is c-accommodating. Therefore, if $\mathrm{WR}_{\mathbb{A}} \geq c$, then $\mathbb{A}$ is c-accommodating. Thus, the worst order ratio is at most as large as the competitive ratio on accommodating sequences.

To prove the other direction, we consider an arbitrary input sequence I and its worst-case permutation for $\mathbb{A}$, $I_{\mathbb{A}}$. Let I_{acc} be the subsequence of $I_{\mathbb{A}}$ containing all the requests in $I_{\mathbb{A}}$ which are accepted by $\mathbb{A}$. Order the requests in I_{acc} in nondecreasing order by their left endpoints. Then, place this ordered sequence at the beginning of a new sequence, I_{OPT}, followed by those requests remaining in I after removing those also in I_{acc}, This gives a permutation of I. Notice that by the above lemma, OPT will be forced to accept all requests in I_{acc} when given I_{OPT}. Let the subset of the requests it accepts from I_{OPT} be I'. In OPT's worst permutation of I, OPT accepts at most $|I'|$ requests.

Clearly, I' is an accommodating sequence. If $\mathbb{A}$ is memoryless, then we can without loss of generality assume that the items it rejects from a sequence are at the end of that sequence. Thus if, in a permutation of I', the items in I_{acc} are placed in the same relative order as in $I_{\mathbb{A}}$, followed by the remaining items from I', $\mathbb{A}$ will accept only those in I_{acc}.

If $\mathbb{A}$'s competitive ratio on accommodating sequences is c, then for some constant b,

$$\begin{array}{ll} & \mathbb{A}_W(I') \geq c \cdot |I'| - b \\ \Rightarrow & |I_{\mathrm{acc}}| \geq c \cdot |I'| - b \\ \Rightarrow & \mathbb{A}_W(I) \geq c \cdot |I'| - b \\ \Rightarrow & \mathbb{A}_W(I) \geq c \cdot \mathrm{OPT}_W(I) - b. \end{array}$$

Since this holds for any request sequence I, $\mathrm{WR}_{\mathbb{A}}$ is at least $\mathbb{A}$'s competitive ratio on accommodating sequences.

If $\mathbb{A}$ is not memoryless, it is not obvious that there is an permutation of I' which would cause $\mathbb{A}$ to accept only I_{acc}. However, there is clearly some on-line algorithm, $\mathbb{B}$ which would accept only I_{acc}. Following the reasoning above, assuming $\mathbb{B}$'s competitive ratio on accommodating sequences is c,

$$\begin{aligned} & \mathbb{B}_W(I') \geq c \cdot |I'| - b \\ \Rightarrow \qquad & \mathbb{A}_W(I) \geq c \cdot \text{OPT}_W(I) - b. \end{aligned}$$

Since this holds for any request sequence I, $\text{WR}_{\mathbb{A}}$ is at least $\mathbb{B}$'s competitive ratio on accommodating sequences. □

It would be interesting to know if the assumption that the algorithm is memoryless is necessary in the theorem above. As is, this result is interesting in that it gives a relationship between the relative worst order ratio and the competitive ratio on accommodating sequences. The direction showing that the worst order ratio for an algorithm $\mathbb{A}$ is no larger than its competitive ratio on accommodating sequences clearly applies to any maximization problem (and the opposite inequality for any minimization problem). However, the other direction does not hold for all problems. In the case of dual bin packing, a problem where most results have resembled those for the unit price seat reservation problem, $\text{WR}_{\mathbb{A}} = 0$ for any fair, deterministic algorithm $\mathbb{A}$ [3], although the competitive ratio on accommodating sequences is always at least $\frac{1}{2}$ [8].

The theorem above, combined with previous work on the competitive ratio on accommodating sequences, immediately gives the following corollaries. The first corollary shows that for k much larger than n, the worst order ratio for any deterministic algorithm for the unit price problem is close to $\frac{1}{2}$.

Corollary 1. *The worst order ratio for any deterministic algorithm for the unit price problem with $n \geq 3$ seats is at least $\frac{1}{2}$ and most $\frac{1}{2} + \frac{3n-3}{2k+6n-(8+2c)}$, where $k \geq 7$ and $c \equiv k-1 \pmod 6$.*

Proof. This statement holds for the competitive ratio on accommodating sequences [1], so the result follows from Theorem 1. □

This result is interesting in that it gives a much more optimistic prediction for the unit price problem than the competitive ratio, which is not bounded below by a constant. For the proportional price problem, the competitive ratio on accomodating sequences has not been shown to be different from the competitive ratio [7]. Thus, if we similarly try to extend the theorem above to the proportional price problem, we do not get any results that are different from the competitive ratio.

The results above are also useful when considering the relative worst order ratio. The next corollary gives bounds on the relative worst order ratios for the algorithms we consider.

Corollary 2. *For any two deterministic algorithms $\mathbb{A}$ and $\mathbb{B}$, $\frac{1}{2} \leq WR_{\mathbb{A},\mathbb{B}} \leq 2$ for the unit price problem, and $\frac{1}{k-1} \leq WR_{\mathbb{A},\mathbb{B}} \leq k-1$ for the proportional price problem.*

Proof. This follows from the above theorem, plus the fact that the competitive ratio on accommodating sequences for any algorithm is at least $\frac{1}{2}$ for the unit price problem and at least $\frac{1}{k-1}$ for the proportional price problem [7]. □

This result will be applied in the following sections.

5 The Relative Worst Order Ratio for the Unit Price Problem

In this section, we will investigate the relative worst order ratios of deterministic algorithms for the unit price problem. Without loss of generality, we will assume within the proofs that the price of all tickets is simply one unit of profit.

All the specific algorithms we are exploring make their decisions regardless of the pricing policy used, so in some cases we can make conclusions about their relative performance for the proportional price problem while analyzing their relative performance for the unit price problem. In the cases where we are interested in exactly how well one algorithms does against another, we will of course have to take into account the pricing policy.

5.1 First-Fit is at Least as Good as Any Any-Fit algorithm.

Our first result is based on the fact that given an input sequence and First-Fit's arrangement of it, an Any-Fit algorithm can be forced to make the exact same seat arrangements by permuting the sequence in an appropriate way.

Theorem 2. *For any Any-Fit algorithm $\mathbb{A}$, $WR_{FF,\mathbb{A}} \geq 1$, regardless of pricing policy.*

Proof. We will consider an arbitrary input sequence I and its worst-case permutation for First-Fit, I_{FF}. We will show that there exists a permutation of I, $I_{\mathbb{A}}$, such that $\mathrm{A}(I_{\mathbb{A}}) = \mathrm{FF}(I_{\mathrm{FF}})$. This will imply that $\mathrm{FF}_{\mathrm{W}}(I) = \mathrm{FF}(I_{\mathrm{FF}}) = \mathbb{A}(I_{\mathbb{A}}) \geq \mathbb{A}_{\mathrm{W}}(I)$. Since this will hold for all I, we will have proven the theorem.

Without loss of generality, we will assume that all requests which are rejected by First-Fit appear last in I_{FF}. Also, we will assume that when $\mathbb{A}$ must choose a new seat to activate, it will choose the seat with the smallest number. This places no effective restriction on $\mathbb{A}$ since we could always renumber the seats according to the order in which they would be activated by $\mathbb{A}$ when it would process $I_{\mathbb{A}}$.

Let the height of a request in I_{FF} be the seat it was assigned to by First-Fit, and ∞ if it was rejected by First-Fit. Let $I_{\mathbb{A}}$ be a permutation of I where all the requests appear in order of non-decreasing height. We prove that $\mathbb{A}(I_{\mathbb{A}}) = \mathrm{FF}(I_{\mathrm{FF}})$ by induction. The induction hypothesis is that after processing all requests with height up to and including i, $\mathbb{A}$ will make the same seat assignments as First-Fit. For the base case $i = 0$, no seats have been assigned, so the inductive hypothesis holds trivially.

For the general case of $1 \leq i \leq n$, we consider when $\mathbb{A}$ encounters the first request with height i. At this point, $\mathbb{A}$ has filled the first $i-1$ seats exactly as First-Fit, and seats $i \ldots n$ remain inactive. Since this request could not be fit into any of the first $i-1$ seats by First-Fit, it cannot be fit into any of the first $i-1$ seats by $\mathbb{A}$. It will therefore be placed in the first available inactive seat, which is seat i.

Now consider when $\mathbb{A}$ encounters any other request r with height i. At this point, $\mathbb{A}$ has filled the first $i-1$ seats with at least the same requests as First-Fit, and now it has activated other seats as well. Seat i is now active. Again, r cannot fit into any of the first $i-1$ seats. Moreover, since the only possible requests to be placed on seat i at this point must have height i and all requests with the same height must be non-overlapping, $\mathbb{A}$ can fit r in seat i. Since $\mathbb{A}$ is an Any-Fit algorithm, it will necessarily assign r to seat i.

For the case of $i = \infty$, $\mathbb{A}$ is not able to accommodate these requests because if it would then First-Fit would have accommodated them as well. Therefore, $\mathbb{A}$ will reject these requests. □

This theorem alone does not separate First-Fit from Best-Fit, but the following theorem gives us a family of input sequences for which First-Fit will out-perform Best-Fit.

Theorem 3. *For the unit price problem with $k \geq 10$, $\frac{4}{3} \leq WR_{FF,BF} \leq 2$.*

Proof. The upper bound follows directly from Corollary 2. Since Theorem 2 implies that $\mathrm{WR}_{\mathrm{FF,BF}} \geq 1$, 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, $\mathrm{FF}_{\mathrm{W}}(I_n) \geq \frac{4}{3}\mathrm{BF}_{\mathrm{W}}(I_n) - b$.

Consider the sequence I_n beginning with $\lfloor \frac{n}{2} \rfloor$ request tuples $[1,2), [5,k-4), [k-1,k)$, followed by $\lfloor \frac{n}{2} \rfloor$ request tuples $[3,k-2), [2,3), [k-2,k-1)$. We then end the sequence with $\lfloor \frac{n}{2} \rfloor$ request tuples $[1,3), [k-2,k)$. Clearly, even in the worst-case ordering, First-Fit will accommodate all requests, so $\mathrm{FF}_{\mathrm{W}}(I_n) = 8 \cdot \lfloor \frac{n}{2} \rfloor$. Best-Fit, on the other hand, will accommodate at most two of the last $\lfloor \frac{n}{2} \rfloor$ tuples given the ordering above (when n is odd), so $\mathrm{BF}_{\mathrm{W}}(I_n) \leq 6 \cdot \lfloor \frac{n}{2} \rfloor + 2$. The needed ratio follows: $\mathrm{FF}_{\mathrm{W}}(I_n) \geq \frac{4}{3}\mathrm{BF}_{\mathrm{W}}(I_n) - \frac{8}{3}$. □

It remains an open problem to close the gap between $\frac{4}{3}$ and 2, though the relative performance of First-Fit to Best-Fit is established.

5.2 Worst-Fit is at Least as Bad as Any Deterministic Algorithm.

The idea behind Worst-Fit is that it will spread out all the requests as much as possible, preferring to have many shorter empty intervals over having fewer, but longer, empty intervals, as Best-Fit would prefer. It turns out that this idea does not work very well, as the following theorem shows.

Theorem 4. *For any deterministic algorithm $\mathbb{A}$, $WR_{\mathbb{A},WF} \geq 1$, regardless of pricing policy.*

Proof. We will consider an arbitrary input sequence I and its worst-case permutation for $\mathbb{A}$, $I_{\mathbb{A}}$. We will show that there exists a permutation of I, I_{WF}, for which Worst-Fit will reject at least all the elements that $\mathbb{A}$ rejected. This will imply $\mathbb{A}_{\mathrm{W}}(I) = \mathbb{A}(I_{\mathbb{A}}) \geq \mathrm{WF}(I_{\mathrm{WF}}) \geq \mathrm{WF}_{\mathrm{W}}(I)$. Since this will hold for all I, we will have proven the theorem.

We construct I_{WF} by ordering all the requests $\mathbb{A}$ accepted in nondecreasing order of their start station, followed by all the rejected requests in arbitrary order. Let r be any request rejected by $\mathbb{A}$. Consider the set of requests $S = \{s_1, s_2, \ldots, s_n\}$, which are the first n elements in I_{WF} which overlap r. Such a set must exist since r was rejected by $\mathbb{A}$. We claim that no two requests from S will be placed in the same seat by Worst-Fit. If the claim holds, then it will imply that r is rejected by Worst-Fit.

We prove the claim by contradiction. Suppose there exist two requests, $x, y \in S$ such that Worst-Fit places them in the same seat. Without loss of generality, we assume Worst-Fit processes x before y. Since requests appear in nondecreasing order of their start station in I_{WF}, we have that y lies to the right of x. Now consider the point in time when Worst-Fit processes y. Since S contains the first n requests in I_{WF} overlapping r, and Worst-Fit has not processed all of them yet, there must be a seat for which the interval r is still empty. Furthermore, since Worst-Fit hasn't yet processed any requests that lie completely to the right of r, there exists a free interval on this seat of length $s \geq k - r_{\mathrm{s}}$ into which Worst-Fit could place y. On the other hand, the free interval on the seat of x has length $s' \leq k - x_f$. Since $s > s'$, Worst-Fit would not place y on the same seat as x, and therefore we have reached a contradiction. □

Additionally, we can prove an asymptotically tight bound for the relative worst order ratio of Worst-Fit to both First-Fit and Best-Fit, which is as bad as Worst-Fit can be with respect to any algorithm. The following proof uses a family of sequences, first used in [6], which can be intuitively seen to cause Worst-Fit to perform very poorly. This idea is formalized with respect to the relative worst order ratio in the following theorem.

Theorem 5. *For any Any-Fit algorithm $\mathbb{A}$ for the unit price problem $2-\frac{1}{k-1} \leq WR_{\mathbb{A},WF} \leq 2$.*

Proof. The upper bound follows directly from Corollary 2. Since Theorem 4 implies that $\mathrm{WR}_{\mathbb{A},\mathrm{WF}} \geq 1$, to prove the lower bound, it is sufficient to find a family of sequences I_n with $\lim_{n\to\infty} \mathbb{A}_{\mathrm{W}}(I_n) = \infty$, where there exists a constant b such that for all I_n, $\mathbb{A}_{\mathrm{W}}(I_n) \geq (2-\frac{1}{k-1})\mathrm{WF}_{\mathrm{W}}(I_n) - b$.

We construct I_n as follows. We begin the request sequence with $\left\lfloor \frac{n}{k-1} \right\rfloor$ requests for each of the intervals $[1,2), [2,3), \ldots, [k-1,k)$. In the case when n is not divisible by $k-1$, we also give one additional request for each of the intervals $[1,2), \ldots, [(n \bmod k-1), (n \bmod k-1)+1)$. If n is divisible by $k-1$, then these requests are omitted. Then we finish the sequence with $n - \left\lceil \frac{n}{k-1} \right\rceil$ requests for the interval $[1,k)$. Regardless of the ordering, $\mathbb{A}$ will accommodate all requests, so that $\mathbb{A}_{\mathrm{W}}(I_n) = 2n - \left\lceil \frac{n}{k-1} \right\rceil$. For Worst-Fit, the given ordering is the worst case ordering, and it will fill all the available seats with the first n requests, while rejecting all the remaining requests. Therefore, $\mathrm{WF}_{\mathrm{W}}(I_n) = n$. This gives us the needed ratio: $\mathbb{A}_{\mathrm{W}}(I_n) \geq (2-\frac{1}{k-1})\mathrm{WF}_{\mathrm{W}}(I_n) - 1$. □

Corollary 3. $2-\frac{1}{k-1} \leq WR_{FF,WF} \leq 2$ *and* $2-\frac{1}{k-1} \leq WR_{BF,WF} \leq 2$.

Thus, we obtain a clear separation between Worst-Fit and First-Fit/Best-Fit, and the bounds on the ratio are asymptotically tight.

6 The Relative Worst Order Ratio for the Proportional Price Problem.

It turns out that many of the results for the unit price problem can be transfered to the proportional price problem. Specifically, we still have the result that First-Fit is at least as good as any Any-Fit algorithm, and Worst-Fit is at least as bad as any deterministic algorithm. One difference is that the value of the relative worst order ratio of First-Fit to Best-Fit is different, as we show in the following theorem.

Theorem 6. *For the proportional price problem with $k \geq 6$, $\frac{k+2}{6} \leq WR_{FF,BF} \leq k-1$.*

Proof. The upper bound follows directly from Corollary 2. Since Theorem 2 implies that $\mathrm{WR}_{\mathrm{FF},\mathrm{BF}} \geq 1$, it is sufficient to find a family of sequences I_n with $\lim_{n\to\infty} \mathrm{FF}_{\mathrm{W}}(I_n) = \infty$, such that for all I_n, $\mathrm{FF}_{\mathrm{W}}(I_n) \geq \frac{k+2}{6}\mathrm{BF}_{\mathrm{W}}(I_n)$.

We define the family of sequences I_n only for even n. Consider this sequence beginning with $\frac{n}{2}$ request tuples $[1,2), [k-1,k)$, followed by $\frac{n}{2}$ request tuples $[k-3,k)$ and $[2,3)$. Finally, the sequence concludes with $\frac{n}{2}$ requests tuples $[1,k-3)$. First-Fit will be able to place all the requests regardless of their ordering, so $\mathrm{FF}_{\mathrm{W}}(I_n) = (k+2)\cdot(\frac{n}{2})$. On the other hand, Best-Fit will not accommodate any of the last $\frac{n}{2}$ requests when given the ordering above, so $\mathrm{BF}_{\mathrm{W}}(I_n) = 6\cdot(\frac{n}{2})$. The needed ratio follows. □

Unlike for the unit price problem, the relative worst order ratio of First-Fit to Best-Fit is not bounded by a constant independent of k. However, the gap between the lower bound and the upper bound increases as k goes to infinity, meaning that the bounds are not asymptotically tight. It would be interesting to see if they can be tightened to be so.

The second difference between the proportional and unit price problem is the relative worst order ratio of Worst-Fit to any Any-Fit algorithm. Specifically, we have the following theorem.

Theorem 7. *For any Any-Fit algorithm $\mathbb{A}$ for the proportional price problem, $WR_{\mathbb{A},WF} = k-1$.*

Proof. The upper bound follows directly from Corollary 2. Since Theorem 4 shows that $\mathrm{WR}_{\mathbb{A},\mathrm{WF}} \geq 1$, it is sufficient to find a family of sequences I_n with $\lim_{n\to\infty} \mathbb{A}_{\mathrm{W}}(I_n) = \infty$, such that for all I_n, $\mathbb{A}_{\mathrm{W}}(I_n) \geq (k-1)\mathrm{WF}_{\mathrm{W}}(I_n)$.

We will use the same sequence as was used in the proof of Theorem 5, except that we will define it only for n divisible by $k-1$. The algorithms will still accept and reject the same requests, but the profit must be calculated differently. $\mathrm{WF}_{\mathrm{W}}(I_n) = n$ still holds, but now $\mathbb{A}_{\mathrm{W}}(I_n) = n \cdot (k-1)$. The resulting ratio for the lower bound follows. □

Thus, the ratio of Worst-Fit to any Any-Fit algorithm is exact, and is as bad as can be. We note that in the above proof we consider the same ordering of the sequence for both Worst-Fit and $\mathbb{A}$, and $\mathbb{A}$ behaves exactly as OPT. This means we can also use the same sequence to prove that the competitive ratio for Worst-Fit is the worst possible among deterministic algorithms.

7 Concluding Remarks and Open Problems

The relative worst order ratio has already been applied to some problems, and has led to intuitively and/or experimentally correct results which could not be obtained with the competitive ratio. For the seat reservation problem, applying the relative worst order ratio has proven very helpful in differentiating between various deterministic algorithms that could not be differentiated with the competitive ratio. Moreover, previous work studying the seat reservation problem with respect to the competitive ratio and the competitive ratio on accommodating sequences has essentially ignored the proportional price problem, since all the results have been so negative. In contrast, the relative worst order ratio allows us to easily compare algorithms for the proportional price problem.

With respect to the algorithms described in this paper, the most interesting open problem is to close the gap between $\frac{4}{3}$ and 2 for the ratio of Fist-Fit to Best-Fit. It also remains interesting to see if the requirement that $\mathbb{A}$ is memoryless is necessary in Theorem 1.

The relative worst order ratio has very recently been extended to apply to randomized algorithms [5]. It is thus interesting to study the quality of randomized algorithms for the seat reservation problem, which were introduced in [7,1]. Ultimately, the goal is to find an algorithm that is better than the existing ones, as has been done for the paging problem [5]. In this sense, the most interesting open problem remains to find an algorithm that does better than First-Fit, or show that one does not exist.

References

1. E. Bach, J. Boyar, L. Epstein, L. M. Favrholdt, T. Jiang, K. S. Larsen, G.-H. Lin, and R. van Stee. Tight bounds on the competitive ratio on accommodating sequences for the seat reservation problem. *J. Sched.*, 6:131–147, 2003.
2. S. Ben-David and A. Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11(1):73–91, 1994.
3. J. Boyar and L. M. Favrholdt. The relative worst order ratio for on-line algorithms. In *5th Italian Conference on Algorithms and Complexity*, volume 2653 of *LNCS*, pages 58–69, 2003.
4. J. Boyar and L. M. Favrholdt. The relative worst order ratio for on-line bin packing algorithms. Tech. report PP–2003–13, Department of Mathematics and Computer Science, University of Southern Denmark, Main Campus: Odense University, 2003.
5. J. Boyar, L. M. Favrholdt, and K. S. Larsen. The relative worst order ratio applied to paging. Tech. report ALCOMFT-TR-03-32, Future and Emerging Technologies program under the EU, contract number IST-1999-14186, 2003.
6. J. Boyar, L. M. Favrholdt, K. S. Larsen, and M. N. Nielsen. Extending the accommodating function. *Acta Informatica*, 40:3–35, 2003.
7. J. Boyar and K. S. Larsen. The seat reservation problem. *Algorithmica*, 25:403–417, 1999.
8. J. Boyar, K. S. Larsen, and M. N. Nielsen. The accommodating function: A generalization of the competitive ratio. *SIAM J. Comput.*, 31(1):233–258, 2001.
9. L. Epstein, L. M. Favrholdt, and J. S. Kohrt. The relative worst order ratio applied to scheduling problems. Work in progress, 2003.
10. R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal*, 45:1563–1581, 1966.
11. D. S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8:272–314, 1974.
12. A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.
13. C. Kenyon. Best-fit bin-packing with random order. In *7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 359–364, 1996.
14. J. S. Kohrt. The relative worst order ratio applied to bin coloring. Work in progress, 2003.
15. D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. of the ACM*, 28(2):202–208, 1985.