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. For the unit price problem, where all tickets have the same cost, we show that First-Fit and Best-Fit are better than Worst-Fit, 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 the ticket price is proportional to the distance travelled. In contrast, no deterministic algorithm has a competitive ratio, or even a competitive ratio on accommodating sequences, which is bounded below by a constant. It is also shown that the worst order ratio for seat reservation algorithms is very closely related to the competitive ratio on accommodating sequences.

1 Introduction

The standard measure for the quality of on-line algorithms is the competitive ratio [14, 23, 17], 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 is quite successful in predicting the performance of algorithms. However, in many others, it gives results that are either counter-intuitive or counter to the experimental data. There is therefore a need to develop performance measures that would supplement the competitive ratio.

The competitive ratio resembles the approximation ratio, which is not surprising 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 [6]. This measure, the relative worst order ratio, allows on-line algorithms to be compared directly to each other. It combines the desirable properties of

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

some previously considered performance measures, namely the Max/Max ratio [5] and the random order ratio [18]. The Max/Max ratio allows direct comparison of two on-line algorithms, without the intermediate comparison to OPT. The random order ratio, on the other hand, is the worst-case ratio of the expected performance of an algorithm on a random permutation of an input sequence, compared with an optimal solution. To compare two algorithms using the relative worst order ratio, 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, in contrast to the competitive ratio, it has shown that Least-Recently-Used(LRU) is strictly better than Flush-When-Full(FWF) and that look-ahead helps [8], both results being consistent with intuition and practice. 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 [8]. Other problems where the relative worst order ratio has given more correct results are bin packing [6, 7], scheduling [12], and bin coloring [20].

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 [10] as the problem of assigning passengers to seats on a train with n seats and k stations enroute, in an on-line manner. We focus on deterministic algorithms, although randomized algorithms for this problem have also been studied [10, 3]. 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. For both variants, the competitive ratio is $\Theta(\frac{1}{k})$ for all deterministic algorithms [10], 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). No pair of algorithms has been conclusively separated using the competitive ratio.

Using the relative worst order ratio, we are able to differentiate all three algorithms, for both the unit price and the proportional price problems. 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[1]

[1] The competitive ratio on accommodating sequences was first studied in [10], but called the *accommodating ratio* there.

(defined below) for the algorithm and bounded below by the competitive ratio on accommodating sequences for some algorithm. This gives bounds for the worst order ratio of $\frac{1}{2} \leq r \leq \frac{1}{2} + \frac{3n-3}{2k+6n-(8+2c)}$, 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

The *seat reservation problem* [10] concerns a scenario where a train with n seats travels on a route passing through $k \geq 2$ 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 $1 \leq s < f \leq k$. 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. An algorithm attempts to maximize income, i.e., the total price of the tickets sold, so the performance of an algorithm depends on the ticket pricing policy. 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 travelled. 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."

The seat reservation problem can be viewed as an interval graph coloring problem [15], with the assignment of seat numbers corresponding to the assignment of colors. An optimal on-line algorithm for the standard interval graph coloring problem, which tries to minimize the number of colors used, instead of maximizing the number of intervals given colors, is presented in [19]. The off-line seat reservation problem without the fairness restriction is equivalent to the maximum k-colorable subgraph problem for interval graphs, which is solvable in polynomial time [24]. Various other problems which can be viewed as variants of the seat reservation problem are optical routing with a limited number of wavelengths [1, 4, 13, 22], call control [2], and interval scheduling [21].

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 $1 \leq x_{\mathrm{s}} < x_{\mathrm{f}} \leq k$. We say an interval x is a subinterval of the interval y if $y_{\mathrm{s}} \leq x_{\mathrm{s}}$ and $x_{\mathrm{f}} \leq y_{\mathrm{f}}$. 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 *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. At any given time, we say that a seat is *active* if at least one request has been assigned to it, and *inactive* otherwise.

We consider the following three algorithms: *First-Fit* is the algorithm which places a request on the first seat which is unoccupied for the length of the journey. *Best-Fit* places

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 slightly with knowledge of the tie-breaker[2]. However, these improvements are minor, and do not change the meaning of the results. *Worst-Fit* places 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 consider the class of *Any-Fit* algorithms, inspired by a class of Bin Packing algorithms of the same name defined by Johnson in [16]. An Any-Fit algorithm places a request on an inactive seat 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 [6], though, for the sake of simplicity, only for maximization problems, such as the seat reservation problem.

Many algorithms are designed with certain kinds of permutations of the input in mind, making them very efficient for some permutations but very inefficient for others. Thus, given a set of requests, if we were to compare the performance of two algorithms directly to each other, we would get certain permutations where one algorithm strongly outperforms the other while the opposite would hold for other permutations, making the algorithms incomparable. Hence, we will consider sequences over the same set of requests together, and we will compare the performance of two algorithms on their respective worst-case permutations. To this end, we formally define $\mathbb{A}_W(I)$, the performance of an on-line algorithm $\mathbb{A}$ on the "worst permutation" of the sequence I of requests, 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.

$S_2(c)$: 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 $WR_{\mathbb{A},\mathbb{B}} = \inf\{r \mid S_1(r)\}$.*

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

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 the one algorithm can be. Note that if $S_1(1)$ holds, the supremum involves S_2 rather than S_1, and vice versa.

The constant b in the definitions of $S_1(c)$ and $S_2(c)$ must be independent of the sequence I, and for the seat reservation problem, it must also be independent of k and n. 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. It is easily shown [6] 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 $\mathrm{WR}_{\mathbb{A},\mathbb{C}} \leq 1$.

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* [6], denoted $\mathrm{WR}_{\mathbb{A}} \triangleq \mathrm{WR}_{\mathbb{A},\mathrm{OPT}}$. 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 useful in its own right.

4 The Relation Between the Worst Order Ratio and the Competitive Ratio on Accommodating Sequences

In this section, we show a connection between the worst order ratio and the competitive ratio on accommodating sequences [10], 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* for algorithm $\mathbb{A}$ 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. 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:

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}. Because of the the ordering of the requests, if the seat is empty from r_s to r_{s+1}, it is also empty to the right of r_s. 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. First assume that $\text{WR}_{\mathbb{A}} \geq c$. Then, there exists a constant b such that $\mathbb{A}_W(I) \geq c \cdot \text{OPT}_W(I) - b$ for all input sequences I. It follows from definitions that $\mathbb{A}(I) \geq \mathbb{A}_W(I)$ and $\text{OPT}_W(I) = \text{OPT}(I)$ for all accommodating sequences I. Hence, there exists a constant b such that $\mathbb{A}(I) \geq c \cdot \text{OPT}(I) - b$ for all accommodating sequences I, so $\mathbb{A}$ is c-competitive on accommodating sequences. 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 a worst-case permutation of I 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 the remaining requests remaining in I, giving 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, $\mathbb{A}_W(I') \geq c \cdot |I'| - b$, so $\mathbb{A}_W(I) = |I_{\text{acc}}| \geq c \cdot |I'| - b$, and $\mathbb{A}_W(I) \geq c \cdot \text{OPT}_W(I) - b$. Since this holds for any request sequence I, $\text{WR}_{\mathbb{A}}$ 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, $\mathbb{B}_W(I') \geq c \cdot |I'| - b$ implies $\mathbb{A}_W(I) \geq$

$c \cdot \mathrm{OPT}_W(I) - b$. Thus, $\mathrm{WR}_{\mathbb{A}}$ is at least $\mathbb{B}$'s competitive ratio on accommodating sequences. □

The theorem above, combined with results on the competitive ratio on accommodating sequences [3], immediately gives 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 $c \equiv k-1 \pmod 6$.*

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 accommodating sequences has not been shown to be different from the competitive ratio [10]. 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, which follows from Theorem 1 and the results from [10], gives bounds on the relative worst order ratios for the algorithms we consider.

Corollary 2. *For any two comparable deterministic algorithms $\mathbb{A}$ and $\mathbb{B}$,*

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

5 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 assume within the proofs that the price of all tickets is one unit of profit. The algorithms we consider make the same decisions regardless of the pricing policy used. Thus, we can make some conclusions about their relative performance for the proportional price problem while analyzing their relative performance for the unit price problem.

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} and that when $\mathbb{A}$ must choose a new seat to activate, it will choose the seat with the smallest number.

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 $\mathbb{A}(I_{\mathbb{A}}) = \text{FF}(I_{\text{FF}})$ 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 shows that $\text{WR}_{\text{FF,BF}} \geq 1$, it is sufficient to find a family of sequences I_n with $\lim_{n\to\infty} \text{FF}_{\text{W}}(I_n) = \infty$, where there exists a constant b such that for all I_n, $\text{FF}_{\text{W}}(I_n) \geq \frac{4}{3}\text{BF}_{\text{W}}(I_n) - b$.

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 $\text{FF}_{\text{W}}(I_n) = 8 \cdot \lfloor \frac{n}{2} \rfloor$. Best-Fit, on the other hand, will accommodate at most two of the last $\lfloor \frac{n}{2} \rfloor$ tuples given this ordering (when n is odd), so $\text{BF}_{\text{W}}(I_n) \leq 6 \cdot \lfloor \frac{n}{2} \rfloor + 2$. The result follows: $\text{FF}_{\text{W}}(I_n) \geq \frac{4}{3}\text{BF}_{\text{W}}(I_n) - \frac{8}{3}$. □

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

Worst-Fit spreads out the requests, creating many short empty intervals, instead of fewer, but longer, empty intervals, as with Best-Fit. The following theorem shows that this strategy is not very successful,

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 $\mathbb{A}_{\text{W}}(I) = \mathbb{A}(I_{\mathbb{A}}) \geq \text{WF}(I_{\text{WF}}) \geq \text{WF}_{\text{W}}(I)$. 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 $s \geq k - r_{\text{s}}$ 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 [9], 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 $\text{WR}_{\mathbb{A},\text{WF}} \geq 1$, to prove the lower bound, it is sufficient to find a family of sequences I_n with $\lim_{n\to\infty} \mathbb{A}_{\text{W}}(I_n) = \infty$, where there exists a constant b such that for all I_n, $\mathbb{A}_{\text{W}}(I_n) \geq (2 - \frac{1}{k-1})\text{WF}_{\text{W}}(I_n) - b$.

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 $\mathbb{A}_{\text{W}}(I_n) = 2n - \left\lceil \frac{n}{k-1} \right\rceil$. 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, $\mathrm{WF_W}(I_n) = n$. This gives us the needed ratio: $\mathbb{A}_\mathrm{W}(I_n) \geq (2 - \frac{1}{k-1})\mathrm{WF_W}(I_n) - 1$. □

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 Proportional Price Problem

For the proportional price problem, the ticket price is proportional to the distance travelled. Without loss of generality, we will assume in the proofs that the price of a ticket from station i to station j is $j - i$. 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 shows that $\mathrm{WR_{FF,BF}} \geq 1$, it is sufficient to find a family of sequences I_n with $\lim_{n\to\infty} \mathrm{FF_W}(I_n) = \infty$, such that for all I_n, $\mathrm{FF_W}(I_n) \geq \frac{k+2}{6}\mathrm{BF_W}(I_n)$.

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 $\mathrm{FF_W}(I_n) = (k+2) \cdot (\frac{n}{2})$. On the other hand, Best-Fit will not accommodate any of the last $\frac{n}{2}$ requests when given the ordering above, so $\mathrm{BF_W}(I_n) = 6 \cdot (\frac{n}{2})$. 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. Moreover, 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 $\mathrm{WR_{\mathbb{A},WF}} \geq 1$, 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, $\mathbb{A}_\mathrm{W}(I_n) \geq (k-1)\mathrm{WF_W}(I_n)$.

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 [6, 8, 12, 20]. 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.

It remains interesting to see if the assumption that $\mathbb{A}$ is memoryless is necessary in Theorem 1. As is, Theorem 1 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. For dual bin packing, a problem where most results have resembled those for the unit price seat reservation problem, $\mathrm{WR}_{\mathbb{A}} = 0$ for any fair, deterministic algorithm $\mathbb{A}$ [6], although the competitive ratio on accommodating sequences is always at least $\frac{1}{2}$ [11].

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. Ultimately, the goal is to find an algorithm that is better than the existing ones, as has been done for the paging problem [8]. 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. B. Awerbuch, Y. Azar, A. Fiat, S. Leonardi and A. Rosén. On-line competitive algorithms for call admission in optical networks. In *4th Annual European Symposium on Algorithms*, volume 1136 of *LNCS*, pages 431–444, 1996.

2. B. Awerbuch, Y. Bartal, A. Fiat and A. Rosén. Competitive non-preemptive call control. In *5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 312–320, 1994.
3. 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.
4. A. Bar-Noy, R. Canetti, S. Kutten, Y. Mansour and B. Schieber. Bandwidth allocation with preemption. In *27th Annual ACM Symposium on the Theory of Computing*, pages 616–625, 1995.
5. S. Ben-David and A. Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11(1):73–91, 1994.
6. 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.
7. 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.
8. 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.
9. J. Boyar, L. M. Favrholdt, K. S. Larsen, and M. N. Nielsen. Extending the accommodating function. *Acta Informatica*, 40:3–35, 2003.
10. J. Boyar and K. S. Larsen. The seat reservation problem. *Algorithmica*, 25:403–417, 1999.
11. 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.
12. L. Epstein, L. M. Favrholdt, and J. S. Kohrt. The relative worst order ratio applied to scheduling problems. Work in progress, 2004.
13. J. A. Garay, I. S. Gopal, S. Kutten, Y. Mansour and M. Yung. Efficient on-line call control algorithms. *J. Algorithms*, 23:180–194, 1997.
14. R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal*, 45:1563–1581, 1966.
15. T. R. Jensen and B. Toft. Graph Coloring Problems. John Wiley & Sons, 1995.
16. D. S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8:272–314, 1974.
17. A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.
18. C. Kenyon. Best-Fit bin-packing with random order. In *7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 359–364, 1996.
19. H. A. Kierstead and W. T. Trotter. An extremal problem in recursive combinatorics. *Congr. Numer.*, 33:143–153, 1981.
20. J. S. Kohrt. The relative worst order ratio applied to bin coloring. Work in progress, 2004.
21. R. J. Lipton and A. Tomkins. Online interval scheduling. In *5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 302–311, 1994.
22. P. Raghavan and E. Upfal. Efficient routing in all-optical networks. In *26th Annual ACM Symposium on the Theory of Computing*, pages 134–143, 1994.
23. D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. of the ACM*, 28(2):202–208, 1985.
24. M. Yannakakis and F. Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24(2):133–137, 1987.