The Relative Worst Order Ratio Applied to Seat Reservation

JOAN BOYAR

University of Southern Denmark

AND

PAUL MEDVEDEV

University of Toronto

Abstract. The seat reservation problem is the problem of assigning passengers to seats on a train with n seats and k stations enroute in an online manner. The performance of algorithms for this problem is studied using the relative worst order ratio, a fairly new measure for the quality of online algorithms, which allows for direct comparisons between algorithms. This study has yielded new separations between algorithms. For example, for both variants of the problem considered, using the relative worst order ratio, First-Fit and Best-Fit are shown to be better than Worst-Fit.

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

General Terms: Algorithms

Additional Key Words and Phrases: Online, quality measure, relative worst order ratio, seat reservation

ACM Reference Format:

Boyar, J. and Medvedev, P. 2008. The relative worst order ratio applied to seat reservation. ACM Trans. Algor. 4, 4, Article 48 (August 2008), 22 pages. DOI = 10.1145/1383369.1383379 http://doi.acm.org/10.1145/1383369.1383379

1. *Introduction*

The standard measure for the quality of online algorithms is the competitive ratio [Graham 1966; Sleator and Tarjan 1985; Karlin et al. 1988], which is, roughly speaking, the worst-case ratio over all possible input sequences, of the online

A preliminary version of some of these results appeared in the *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory (SWAT)*, Lecture Notes in Computer Science. vol. 3111: 90–101.

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

Authors' address: J. Boyar, Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark. P. Medvedev, Departmental of Computer Science, University of Toronto, 10 King's College Road, Toronto, Ontario M5S 3G4, Canada.

performance to the optimal offline 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 counterintuitive or counter to the experimental data. There is therefore a need to develop performance measures that supplement the competitive ratio.

The competitive ratio resembles the approximation ratio, which is not surprising as efficient online 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 seems less natural to compare an online algorithm to an offline optimal algorithm, which actually solves a variant of the original problem (the offline version). Additionally, when there is need to compare two online algorithms against each other, it seems more appropriate to compare them directly rather than involve an intermediate comparison to an optimal offline algorithm.

For this reason, a new performance measure for the quality of online algorithms has been developed [Boyar and Favrholdt 2003]. This measure, the relative worst order ratio, allows online algorithms to be compared directly to each other. It combines the desirable properties of some previously considered performance measures, namely the Max/Max ratio [Ben-David and Borodin 1994] and the random order ratio [Kenyon 1996]. The Max/Max ratio allows direct comparison of two online 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 analgorithm 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 online algorithms, the relative worst order ratio may also be used to compare an online algorithm to the optimal offline 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[1] 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 [Boyar et al. 2005]; both results are 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 [Boyar et al. 2005]. Other problems where the relative worst order ratio has given more intuitively correct results are bin packing [Boyar and Favrholdt 2003], scheduling [Epstein et al. 2006], and bin coloring [Kohrt 2004].

Given these encouraging results, this article uses the relative worst order ratio to analyze algorithms for the seat reservation problem. This problem is defined

[1]There have been numerous modifications proposed to the competitive ratio for the analysis of online algorithms, some of which are also able to improve on the competitive ratio for various problems, especially paging. For a survey, see Dorrigiv and López-Ortiz [2005].

TABLE I. 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	$\frac{2}{k} \leq r \leq \frac{2-\frac{1}{k-1}}{k-1}$	$\frac{1}{k-1} \leq r \leq \frac{4}{k+2}$

in Boyar and Larsen [1999] as the problem of assigning passengers to seats on a train with n seats and k stations enroute in an online manner. Five algorithms are studied: First-Fit, Best-Fit, Worst-Fit, $\mathrm{KT}^{\mathbb{A}}$, and Random. There are two variants of the seat reservation problem, namely, the unit price problem and the proportional price problem.

The competitive ratio has been applied to both variants in Boyar and Larsen [1999], Bach et al. [2003], and Boyar et al. [2003], particularly for First-Fit, Best-Fit, and Worst-Fit. The known results are summarized in Table I[2]. For both variants, the competitive ratio is $\Theta(\frac{1}{k})$ for all deterministic algorithms [Boyar and Larsen 1999], 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). The results in Table I concerning the proportional price problem also hold for the competitive ratio on accommodating sequences [Boyar and Larsen 1999], which is defined in Section 5. For the unit price problem, the lower bound from Boyar and Larsen [1999] on the competitive ratio on accommodating sequences for any deterministic algorithm is $\frac{1}{2}$, and the upper bound from Bach et al. [2003] is $\frac{1}{2} + \frac{3n-3}{2k+6n-(8+2c)}$, where $c = (k-1) \mod 6$.

No pair of algorithms for the seat reservation problem has been conclusively separated using the competitive ratio (or the competitive ratio on accommodating sequences). Combined with the pessimistic upper bounds on the competitive ratios, these inconclusive results make the seat reservation problem an ideal candidate to study with the relative worst order ratio.

2. *Our Results*

Using the relative worst order ratio, we are able to differentiate First-Fit, Best-Fit, and Worst-Fit 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 that is at least $\frac{k+2}{6}$ for the proportional price problem and asymptotically at least $\frac{5}{3}$ for the unit price problem. We also show that Worst-Fit is at least as bad as any other 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. With regard to $\mathrm{KT}^{\mathbb{A}}$, we show that it is incomparable to First-Fit and Best-Fit and is strictly

[2]All bounds are directly stated in Boyar and Larsen [1999] with the following exceptions. The upper bound on Worst-Fit for unit price can be concluded from the proof of Theorem 8 in Boyar and Larsen [1999]. The upper bound on Worst-Fit for proportional price follows from the worst-case sequence used in Theorem 8 in Boyar et al. [2003].

better than Worst-Fit. For Random, we show that it is incomparable to any Any-Fit algorithm.

Additionally, we find that, for the seat reservation problem, a deterministic algorithm's worst order ratio is bounded from above by the competitive ratio on accommodating sequences[3] (defined in Section 5) for the algorithm and bounded below by the competitive ratio on accommodating sequences for some (possibly a different) algorithm. Thus, a general lower bound on the competitive ratio on accommodating sequences gives a general lower bound for the worst order ratio, so we get bounds for the worst order ratio of $\frac{1}{2} \leq r \leq \frac{1}{2} + \frac{3n-3}{2k+6n-(8+2c)}$, where $c = (k-1) \mod 6$, 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.

3. *The Seat Reservation Problem: Definitions and Algorithms*

The *seat reservation problem* was originally studied in Boyar and Larsen [1999]. We consider 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, anytime 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 online problem. The algorithm attempts to maximize income, that is, the total price of the tickets sold. The performance of an algorithm will thus clearly depend on the ticket-pricing policy. There are two main variants of this problem. 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 $x = [x_s, x_f)$ to denote an interval x from station x_s to station x_f, where $1 \leq x_s < x_f \leq k$. We use the notation x^k to denote k copies of x, where k is a nonnegative integer. We say an interval x is a subinterval of the interval y if $y_s \leq x_s$ and $x_f \leq y_f$. Since a *request* is just an interval, we use the terms interchangeably, depending on what is more natural at the time. The *length* of an interval (request) x is simply $x_f - x_s$. Given a seat, 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 define the *density* of an interval of length one as the

[3] The competitive ratio on accommodating sequences was first studied in Boyar and Larsen [1999] but was called the accommodating ratio there.

number of requests that overlap that interval. The density of a larger interval is the maximum density of any of its length one subintervals. In the case the interval is $[1, k)$, we say the *density of the input sequence*.

The following three algorithms which we consider are all inspired by bin packing algorithms of the same name.

—*First-Fit*. This algorithm places a request on the first seat which is unoccupied for the length of the journey.

—*Best-Fit*. This algorithm 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 tiebreaker, 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 do not assume any specific tiebreaker in any of our proofs. Thus, our results hold for any choice of a tiebreaker for Best-Fit. In some cases, bounds could be tightened with knowledge of the tiebreaker[4]. However, these improvements are minor, and do not change the meaning of the results.

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

Additionally, we consider the following class of algorithms, also inspired by a class of bin packing algorithms of the same name defined by Johnson [1974].

—*Any-Fit*. An algorithm which belongs to this class places a request on an inactive seat only if it does not fit into any of the active seats. This class includes both First-Fit and Best-Fit.

Finally, we consider a random algorithm, first studied in Bach et al. [2003].

—*Random*. Given a request, this algorithm uniformly chooses a random seat from all the seats that are free for the duration of the request.

We also study another algorithm, $\mathrm{KT}^{\mathbb{A}}$, but its definition is left for Section 7.

4. *The (Relative) Worst Order Ratio*

In this section, we define the relative worst order ratio and the notion of two algorithms being comparable (Definition 4.2) as in Boyar and Favrholdt [2003] and Boyar et al. [2005], though for the sake of simplicity only for maximization problems such as the seat reservation problem.

Many algorithms are designed with certain kinds of input in mind, giving very good results for some request sequences but very poor results for others. Thus, if we were to compare the performance of two algorithms directly to each other, we would get certain request sequences where one algorithm strongly outperforms the other while the opposite would hold for other sequences, making the algorithms incomparable. Hence, we consider sequences over the same multiset of requests

[4]Specifically, the relative worst order ratio of First-Fit to Best-Fit can be slightly improved in Theorem 6.2 and in Theorem 6.6.

together, and we 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 online algorithm $\mathbb{A}$ on the worst permutation of the sequence I of requests, as follows.

Definition 4.1. Consider an online 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. If $\mathbb{A}$ is deterministic, then $\mathbb{A}(I)$ is the value of running $\mathbb{A}$ on I, and

$$\mathbb{A}_W(I) = \min_{\sigma} \mathbb{A}(\sigma(I)).$$

If $\mathbb{A}$ is randomized, then $E[\mathbb{A}(I)]$ is the expected value of running $\mathbb{A}$ on I, and

$$\mathbb{A}_W(I) = \min_{\sigma} E[\mathbb{A}(\sigma(I))].$$

The following definition differs slightly from the definition given in the earliest papers on the relative worst order ratio but is identical to that in the journal version of Boyar et al. [2005].

Definition 4.2. For any pair of algorithms $\mathbb{A}$ and $\mathbb{B}$, we define

$$\begin{aligned} c_l(\mathbb{A},\mathbb{B}) &= \sup\{c \mid \exists b\colon \forall I : \mathbb{A}_W(I) \geq c\,\mathbb{B}_W(I) - b\} \text{ and} \\ c_u(\mathbb{A},\mathbb{B}) &= \inf\{c \mid \exists b\colon \forall I : \mathbb{A}_W(I) \leq c\,\mathbb{B}_W(I) + b\}\ . \end{aligned}$$

If $c_l(\mathbb{A},\mathbb{B}) \geq 1$ or $c_u(\mathbb{A},\mathbb{B}) \leq 1$, the algorithms are said to be *comparable* and the *relative worst order ratio* $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ of algorithm $\mathbb{A}$ to algorithm $\mathbb{B}$ is defined. Otherwise, $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ is undefined.

$$\begin{aligned} &\text{If } c_l(\mathbb{A},\mathbb{B}) \geq 1, \text{ then } \mathrm{WR}_{\mathbb{A},\mathbb{B}} = c_u(\mathbb{A},\mathbb{B}), \text{ and} \\ &\text{if } c_u(\mathbb{A},\mathbb{B}) \leq 1, \text{ then } \mathrm{WR}_{\mathbb{A},\mathbb{B}} = c_l(\mathbb{A},\mathbb{B})\,. \end{aligned}$$

If $\mathrm{WR}_{\mathbb{A},\mathbb{B}} > 1$, algorithms $\mathbb{A}$ and $\mathbb{B}$ are said to be comparable in $\mathbb{A}$'s favor. Similarly, if $\mathrm{WR}_{\mathbb{A},\mathbb{B}} < 1$, the algorithms are said to be comparable in $\mathbb{B}$'s favor.

The comparison of $c_l(\mathbb{A},\mathbb{B})$ and $c_u(\mathbb{A},\mathbb{B})$ to 1 checks that one algorithm is always at least as good as the other on every sequence (on their respective worst permutations). When one of the two conditions holds, the relative worst order ratio is a bound on how much better the one algorithm can be.

The constant b in the definitions of $c_l(\mathbb{A},\mathbb{B})$ and $c_u(\mathbb{A},\mathbb{B})$ 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 [Boyar and Favrholdt 2003] (the journal version of Boyar et al. [2005] for this exact definition) that the relative worst order ratio is a transitive measure, that is, for any three algorithms $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$, $\mathrm{WR}_{\mathbb{A},\mathbb{B}} \geq 1$ and $\mathrm{WR}_{\mathbb{B},\mathbb{C}} \geq 1$ implies $\mathrm{WR}_{\mathbb{A},\mathbb{C}} \geq 1$. It also follows from the definition that if $\mathbb{A}$ and $\mathbb{B}$ are comparable, then $\mathrm{WR}_{\mathbb{A},\mathbb{B}} = \frac{1}{\mathrm{WR}_{\mathbb{B},\mathbb{A}}}$. In this article, we use the convention that the ratios are greater than one, so we usually write the better algorithm first.

Although one of the goals in defining the relative worst order ratio was to avoid the intermediate comparison of an online algorithm $\mathbb{A}$ to the optimal offline algorithm, OPT, it is still possible to use it to compare online algorithms to OPT. In this case, the measure is called the *worst order ratio* [Boyar and Favrholdt 2003], denoted $\text{WR}_{\mathbb{A}} \triangleq \text{WR}_{\mathbb{A},\text{OPT}}$ (note that in this special case, we write the better algorithm second as we do with minimization problems, and we will get ratios less than 1). 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 online algorithms directly to each other, the worst order ratio can also be useful in its own right.

5. *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 [Boyar et al. 2001], 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, $\mathbb{A}(I) \geq c \cdot \text{OPT}(I) - b$, where b is a fixed constant for the given problem, and, thus, independent of I, k, and n. The *competitive ratio on accommodating sequences* for algorithm $\mathbb{A}$ is defined as

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

In this section, we compare the worst order ratio and the competitive ratio on accommodating sequences for the seat reservation problem. Note that even though OPT is offline, it must still be fair since this is part of the problem definition.

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 article are memoryless.

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

LEMMA 5.1. *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 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 5.2. *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 $\mathrm{WR}_{\mathbb{A}} \geq c$. Then for any $\epsilon > 0$, there exists a constant b such that $\mathbb{A}_W(I) \geq (1-\epsilon)c \cdot \mathrm{OPT}_W(I) - b$ 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. Hence, there exists a constant b such that $\mathbb{A}(I) \geq (1-\epsilon)c \cdot \mathrm{OPT}(I) - b$ 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 non-decreasing order by their left endpoints. Then, place this ordered sequence at the beginning of a new sequence I_{OPT}, followed by the remaining requests in I, giving a permutation of I. Notice that by Lemma, 5.1, 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'. Let $p(I')$ denote the profit obtained by accepting the requests in I'. In OPT's worst permutation of I, OPT earns at most $p(I')$ profit. 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 any constant $\epsilon > 0$, there exists a constant b such that

$$\begin{array}{rl} & \mathbb{A}_W(I') \geq (1-\epsilon)c \cdot p(I') - b \\ \Rightarrow & p(I_{\mathrm{acc}}) \geq (1-\epsilon)c \cdot p(I') - b \\ \Rightarrow & \mathbb{A}_W(I) \geq (1-\epsilon)c \cdot p(I') - b \\ \Rightarrow & \mathbb{A}_W(I) \geq (1-\epsilon)c \cdot \mathrm{OPT}_W(I) - b. \end{array}$$

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 a permutation of I' which would cause $\mathbb{A}$ to accept only I_{acc}. However, there is clearly some online algorithm $\mathbb{B}$ which would accept only I_{acc}. Following the preceding reasoning, assuming $\mathbb{B}$'s competitive ratio on accommodating sequences is c, $\mathbb{B}_W(I') \geq (1-\epsilon)c \cdot p(I') - b$ implies $\mathbb{A}_W(I) \geq (1-\epsilon)c \cdot \mathrm{OPT}_W(I) - b$. Thus, $\mathrm{WR}_{\mathbb{A}}$ is at least $\mathbb{B}$'s competitive ratio on accommodating sequences. □

Theorem 5.2, combined with results on the competitive ratio on accommodating sequences [Bach et al. 2003], 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 5.3. *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 at most $\frac{1}{2} + \frac{3n-3}{2k+6n-(8+2c)}$, where $k \geq 7$ and $c = (k-1) \mod 6$.*

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. Thus, if we similarly try to extend the Theorem 5.2 to the proportional price problem, we do not get any results that are different from the competitive ratio.

These results 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 5.4. *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 $\frac{1}{k-1} \leq WR_{\mathbb{A},\mathbb{B}} \leq k-1$.

PROOF. This follows from Theorem 5.2 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 [Boyar and Larsen 1999]. □

6. *First-Fit, Best-Fit, and Worst-Fit*

6.1. UNIT PRICE PROBLEM. In this subsection, we investigate the relative worst order ratios of First-Fit, Best-Fit, and Worst-Fit 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.

6.1.1. *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 6.1. *For any Any-Fit algorithm $\mathbb{A}$, regardless of pricing policy,*

$$WR_{FF,\mathbb{A}} \geq 1.$$

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

Without loss of generality, we 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 chooses the first available inactive seat.

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 nondecreasing 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 $i \dots n$ 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 nonoverlapping, $\mathbb{A}$ 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. $\square$

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 outperform Best-Fit.

THEOREM 6.2. *For the unit price problem with $k \geq 10$,*

$$\frac{5 - \frac{4}{\sqrt{4k+9}-5}}{3} \leq \mathrm{WR}_{FF,BF} \leq 2.$$

PROOF. The upper bound follows directly from Corollary 5.4. Since Theorem 6.1 shows that $\mathrm{WR}_{\mathrm{FF,BF}} \geq 1$, it is sufficient to find a family of sequences I_n with $\lim_{n\to\infty} \mathrm{FF}_{\mathrm{W}}(I_n) = \infty$ such that for all I_n, $\mathrm{FF}_{\mathrm{W}}(I_n) \geq \frac{5-\frac{1}{\sqrt{4k+9}-5}}{3}\mathrm{BF}_{\mathrm{W}}(I_n)$. As a convention, we assume an algorithm has n seats available when processing the sequence I_n. Note that as long as I_n is defined for arbitrarily large n, it is not necessary for it to be defined for every n.

Let $j = \lfloor \frac{\sqrt{4k+9}-3}{2} \rfloor$. We define the family of sequences $\{I_n \mid n \text{ is divisible by } j\}$. The sequence is made up of request tuples. For $1 \leq i \leq j$, let g_i be the request tuple $[1 + j + \sum_{x=i+1}^{j} x, j + \sum_{x=1}^{j} x + \sum_{x=1}^{i} x)$, $[i, i + 1)$, $[k - i, k - i + 1)$. Now, I_n consists of $\frac{n}{j}$ requests for g_1, followed by $\frac{n}{j}$ requests for g_2, and so on until finally there are $\frac{n}{j}$ requests for g_j. The sequence finally finishes with $n - \frac{n}{j}$ requests tuples for $[1, j + 1)$ and $[k - j, k)$.

Notice that every request falls into either the left interval $[1, j + 1)$, the middle interval $[j + 1, k - j - 1)$, or the right interval $[k - j, k)$. The middle interval contains only n requests. The left and right intervals contain requests of length either 1 or j (the length of the whole interval), and the density of the intervals is n. Based on these facts, First-Fit will accept all the requests regardless of the ordering,

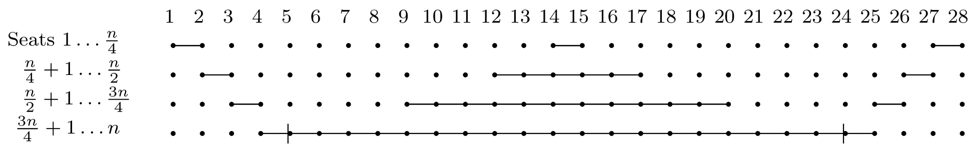
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
Seats $1 \ldots \frac{n}{4}$
$\frac{n}{4}+1 \ldots \frac{n}{2}$
$\frac{n}{2}+1 \ldots \frac{3n}{4}$
$\frac{3n}{4}+1 \ldots n$

FIG. 1. The behavior of Best-Fit for the sequence in Theorem 6.2, when $k = 28$.

so $\mathrm{FF_W}(I_n) = n(5 - \frac{2}{j})$. The seating arrangement of Best-Fit is shown in Figure 1. The requests in the middle interval extend closer and closer to the endpoints, so Best-Fit places the requests within a tuple on the same set of seats. Thus, Best-Fit places exactly one left interval, one middle interval, and one right interval on each seat. It does not accommodate any of the last $2 \cdot (n - \frac{n}{j})$ requests, so $\mathrm{BF_W}(I_n) = 3n$. Using the fact that $j = \lfloor \frac{\sqrt{4k+9}-3}{2} \rfloor \geq \frac{\sqrt{4k+9}-5}{2}$, we get that

$$\mathrm{FF_W}(I_n) \geq \frac{5 - \frac{2}{j}}{3}\mathrm{BF_W}(I_n) \geq \frac{5 - \frac{4}{\sqrt{4k+9}-5}}{3}\mathrm{BF_W}(I_n). \qquad \square$$

This result means that the lower bound on $\mathrm{WR_{FF,BF}}$ approaches $5/3$ asymptotically. It remains an open problem to close the gap between between the lower and upper bounds, though the relative performance of First-Fit to Best-Fit is established.

6.1.2. *Worst-Fit Is at Least as Bad as Any 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 6.3. *For any algorithm $\mathbb{A}$, regardless of pricing policy,*

$$WR_{\mathbb{A},WF} \geq 1.$$

PROOF. We consider an arbitrary input sequence I and its worst-case permutation for $\mathbb{A}$, $I_{\mathbb{A}}$. First, suppose that $\mathbb{A}$ is deterministic. 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} that 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 has not 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.

Now, suppose $\mathbb{A}$ is a randomized algorithm. If B is some fixed setting of the random bits used by $\mathbb{A}$, then define $\mathbb{A}_B$ to behave exactly as $\mathbb{A}$ when the random bits are fixed to B. By the properties of expectation, there must be some B such that $\mathbb{A}_B(I_{\mathbb{A}}) \leq E[\mathbb{A}(I_{\mathbb{A}})]$. Since $\mathbb{A}_B$ is deterministic, we can use the preceding arguments for a deterministic algorithm to show that $\mathbb{A}_{\mathrm{W}}(I) = E[\mathbb{A}(I_{\mathbb{A}})] \geq \mathbb{A}_B(I_{\mathbb{A}}) \geq \mathrm{WF}(I_{\mathrm{WF}}) \geq \mathrm{WF}_{\mathrm{W}}(I)$. $\square$

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 Boyar et al. [2003], 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 6.4. *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 5.4. Since Theorem 6.3 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 a family of sequences $\{I_n \mid n \text{ is divisible by } k-1\}$ as follows. We begin the request sequence with $\frac{n}{k-1}$ requests for each of the intervals $[1, 2), [2, 3), \ldots, [k-1, k)$. We finish the sequence with $n - \frac{n}{k-1}$ requests for the interval $[1, k)$. Regardless of the ordering, $\mathbb{A}$ will accommodate all requests so that $\mathbb{A}_{\mathrm{W}}(I_n) = 2n - \frac{n}{k-1}$. 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; $\mathbb{A}_{\mathrm{W}}(I_n) \geq (2 - \frac{1}{k-1})\mathrm{WF}_{\mathrm{W}}(I_n) - 1$. $\square$

COROLLARY 6.5. $2 - \frac{1}{k-1} \leq WR_{FF,WF} \leq 2$ *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.

6.2. PROPORTIONAL PRICE PROBLEM.

For the proportional price problem, the ticket price is proportional to the distance traveled. Without loss of generality, we 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 transferred 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 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 6.6. *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 5.4. Since Theorem 6.1 shows that $\mathrm{WR}_{\mathrm{FF,BF}} \geq 1$, it is sufficient to find a family of sequences I_n with $\lim_{n\to\infty} \mathrm{FF}_{\mathrm{W}}(I_n) = \infty$, 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 $\mathrm{FF}_W(I_n) = (k+2)\cdot(\frac{n}{2})$. On the other hand, Best-Fit will not accommodate any of the last $\frac{n}{2}$ requests when given the specified ordering, so $\mathrm{BF}(I_n) = 6\cdot(\frac{n}{2})$. The needed ratio follows. □

Unlike 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 6.7. *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 5.4. Since Theorem 6.3 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 use the same sequence as was used in the proof of Theorem 6.4. 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 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. *The KT$^{\mathbb{A}}$ Algorithm*

The problem of coloring an interval graph online is equivalent to the seat reservation problem with a different goal, minimizing the number of seats. In this scenario, all requests are accepted and the goal is to do it using as few seats as possible. An optimal 3-competitive algorithm (KT) for this problem is given by Kierstead and Trotter [1981]. Specifically, for an input sequence of density d, KT uses no more than $3d-2$ seats. Since KT assigns every request to a seat, it can be used for the seat reservation problem as long as there are at least $3d-2$ seats (or, equivalently, the density is no more than $\frac{n+2}{3}$, where n is the number of seats). Of course, such a guarantee cannot be made in an online setting, but we can supplement the algorithm so that those requests assigned to seats that do not exist are placed on one of the existing seats, according to some other rule. This strategy gives rise to the KT$^{\mathbb{A}}$ algorithm. It is stated in the following as exactly the KT algorithm with the supplements in bold.

KT$^{\mathbb{A}}$ ALGORITHM. *Initialize the sets $B_i = \{\}$ for $1 \leq i \leq \frac{n+2}{3}$. To process a new interval x, find the smallest j, $1 \leq j \leq \frac{n+2}{3}$ such that for all h, $j \leq h \leq \frac{n+2}{3}$, the set of intervals $\{x\} \cup \bigcup_{i=1}^{h} B_i$ has density no more than h.* ***If such a j does not exist, then use algorithm $\mathbb{A}$ to process x.*** *Otherwise: Set $B_j = B_j \cup \{x\}$. If $j = 1$, then place x on the first seat, otherwise use First-Fit to place x on one of the seats in $\{3j-4, 3j-3, 3j-2\}$.* ***If First-Fit is not able to place x in the specified seat(s), then use $\mathbb{A}$ to process x.***

Note that KT$^{\mathbb{A}}$ is not fully specified unless $\mathbb{A}$ is specified. Thus KT$^{\mathbb{A}}$ actually refers to a class of algorithms defined over all possible seat reservation algorithms $\mathbb{A}$. However, since the following analysis applies to any algorithm $\mathbb{A}$, we will slightly abuse notation and treat KT$^{\mathbb{A}}$ as a single algorithm.

In Kierstead and Trotter [1981], it is shown that for densities no more than $\frac{n+2}{3}$ the cases in bold never occur. KT$^{\mathbb{A}}$, however, must be defined for all densities. Thus, it deals with the cases in bold by invoking $\mathbb{A}$. However, if the density does not exceed $\frac{n+2}{3}$, then those cases never occur, and KT$^{\mathbb{A}}$ reduces to KT. We thus have the following lemma.

LEMMA 7.1. *If the density of the input sequence is no more than $\frac{n+2}{3}$, then KT$^{\mathbb{A}}$ will accept all the requests.*

The major goal of this section is to prove Theorem 7.6, which states that KT$^{\mathbb{A}}$ is incomparable to First-Fit, incomparable to Best-Fit, and strictly better than Worst-Fit. We first show that there exist sequences where First-Fit and Best-Fit outperform KT$^{\mathbb{A}}$.

LEMMA 7.2. *For any Any-Fit algorithm $\mathbb{B}$, there does not exist a constant b such that $KT^{\mathbb{A}}_{W}(I) \geq \mathbb{B}_W(I) - b$ for all I, for both the unit and proportional price problem.*

PROOF. We define a family of sequences I_n such that $\lim_{n\to\infty} \mathbb{B}_W(I_n) - \text{KT}^{\mathbb{A}}_{W}(I_n) = \infty$. Thus, given a constant b to counter the claim of the lemma, we can always pick n large enough so that the sequence I_n disproves the counter claim.

For the sequence I_n, we require that $n = 3j-2$ for some positive integer j and that $k \geq 2j+1$. There are four groups of requests, appearing in the order given.

—$[4, 2j+1), [6, 2j+1), \ldots, [2j, 2j+1)$.
—$[1, 3), [3, 5), \ldots, [2j-1, 2j+1)$.
—$[2, 4), [4, 6), \ldots, [2j-4, 2j-2)$.
—$2j-5$ requests for $[1, 2j+1)$.

First, consider KT$^{\mathbb{A}}_{W}(I_n)$, and, more specifically, KT$^{\mathbb{A}}(I_n)$. The ith request from the first group is put in set B_i and placed on seat $3i-4$ except that the first request is placed on seat 1. The same is true for the ith requests from the second group. The ith request from the third group is put in set B_{i+1} and placed on seat $3i$. At this point, because of the fairness restriction, it is not important how KT$^{\mathbb{A}}$ is defined, as exactly j of the last requests are accommodated. So KT$^{\mathbb{A}}$ accepts all requests except the last $j-5$. See Figure 2 for an illustration.

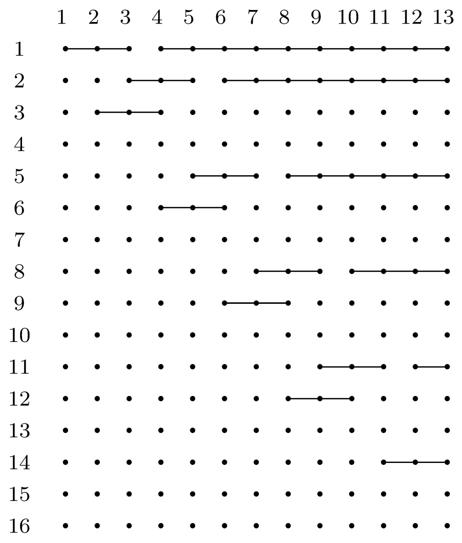

FIG. 2. An example of how $\mathrm{KT}^{\mathbb{A}}$ seats the requests from the first three groups with $n = 16$ and $k = 13$.

Now, consider $\mathbb{B}_{\mathrm{W}}(I_n)$. We claim that $\mathbb{B}$ accommodates all the requests regardless of their order. Let $\sigma(I_n)$ be the worst-case permutation of I_n for $\mathbb{B}$. Call the requests from groups two and three small and the requests from groups one and four big. We partition $\sigma(I_n)$ into phases. We define the 0th phase to be empty and to precede all others. It is followed by $3j - 6$ phases, each consisting of some (possibly zero) consecutive small requests, followed by one big request. Finally, the last phase consists of all the requests (possibly zero) after the last big request. Note that the phase partitioning is unique and that the number of the last phase is $3j - 5$.

We prove that, after the i^{th} phase, all requests have been accepted and there are at most $i + 3$ active seats. Since the number of the last phase is three less than the number of seats, it suffices to show that all requests from the ith phase are assigned to a seat number no more than $i + 3$. The base case of $i = 0$ is trivial since there are no requests. For the inductive step, consider the beginning of the ith phase. By the induction hypothesis, there are at most $i - 1 + 3$ active seats. Of these seats, $i - 1$ must have a big request assigned to them (since no two big requests can share a seat). Assume without loss of generality that those seats are labeled $1 \ldots i - 1$ if $i > 1$. Then, seats $i, i + 1$, and $i + 2$ are either inactive or have only small requests assigned to them. Then, any small request appearing in the ith phase can be assigned to one of these 3 seats since a small request can overlap at most two other small requests (this is just a property of the request sequence). Then, if there is a big request at the end of the phase, it will be assigned to at most the $i + 3$rd seat, completing the inductive step.

Specifically, the induction hypothesis applied to the last phase implies that $\mathbb{B}$ accepts all requests from $\sigma(I_n)$. Thus, $\mathbb{B}$ is able to accommodate $j-5$ requests more than $\mathrm{KT}^{\mathbb{A}}$. Given that both the unit and proportional pricing policies assign at least one unit of profit to an accepted request, we have that $\mathbb{B}_{\mathrm{W}}(I_n) \geq \mathrm{KT}^{\mathbb{A}}(I_n) + j - 5 \geq \mathrm{KT}^{\mathbb{A}}_{\mathrm{W}}(I_n) + \frac{n-13}{3}$. The statement of the limit follows directly. $\square$

To achieve the incomparability result for First-Fit, we now show that there also exist sequences where $\mathrm{KT}^{\mathbb{A}}$ outperforms First-Fit. In fact, First-Fit has also been studied in the context of online interval graph coloring. In this variant, First-Fit never rejects any requests, but otherwise behaves exactly as it would for the seat reservation problem. A lower bound exists [Chrobak and Ślusarek 1988] that gives a family of sequences where First-Fit will use at least $4d - 9$ seats, where d is the density. An adaptation of that argument for this problem, with very minor modifications [Boyar and Medvedev 2007], gives that First-Fit uses $4d - 21$ seats for a sequence with density $d = (n + 2)/3$. By Lemma 7.1, $\mathrm{KT}^{\mathbb{A}}$ accepts all of these requests regardless of order, but First-Fit must reject at least one request for each seat it needs over the $n = 3d - 2$ seats available. So

$$\begin{aligned}\mathrm{KT}^{\mathbb{A}}_{\mathrm{W}}(I_n) - \mathrm{FF}_{\mathrm{W}}(I_n) &\geq \mathrm{KT}^{\mathbb{A}}_{\mathrm{W}}(I_n) - \mathrm{FF}(I_n) \geq 4d - 21 - (3d - 2)\\ &= \frac{n+2}{3} - 19 \to \infty \quad \text{as } n \to \infty.\end{aligned}$$

We thus have the following lemma.

LEMMA 7.3. *There does not exist a constant b such that $KT^{\mathbb{A}}_{W}(I) \leq \mathrm{FF}_{\mathrm{W}}(I) + b$ for all I for both the unit and proportional price problem.*

The family of sequences used in the previous proof requires k to be exponential in the number of seats [Boyar and Medvedev 2007]. In the proof of Lemma 7.2, we require k to be of the order of n. These requirements may seem too strong, but they nevertheless disprove any possibility of a statement that the relative worst order ratio is defined for some $k \geq c$, for any constant c. However, there remains the possibility that, for some special case where k is restricted from above, the relative worst order ratio can be defined.

Now, we turn our attention to Best-Fit. Given Lemma 7.2, it only remains to show that there exist sequences where $\mathrm{KT}^{\mathbb{A}}$ outperforms Best-Fit. This follows immediately from the previous lemma because Best-Fit packs the family of sequences from the proof exactly the same as First-Fit. However, we give a different family of sequences in the next lemma, because the value of k required is much smaller.

LEMMA 7.4. *There does not exist a constant b such that $KT^{\mathbb{A}}_{W}(I) \leq \mathrm{BF}_{\mathrm{W}}(I) + b$ for all I for both the unit and proportional price problem.*

PROOF. We define a family of sequences I_n such that $\lim_{n\to\infty} \mathrm{KT^{\mathbb{A}}}_{\mathrm{W}}(I_n) - \mathrm{BF}_{\mathrm{W}}(I_n) = \infty$. For the sequence I_n, we require that $n = 6j$ for some positive integer j and that $k \geq 28$. I_n consists of $[27, 28)^j$, $[1, 2)^j$, $[25, 28)^j$, $[2, 3)^j$, $[22, 28)^j$, $[3, 4)^j$, $[18, 28)^j$, $[4, 5)^j$, $[13, 28)^j$, $[5, 6)^j$, $[7, 28)^j$, $[6, 7)^j$, $[1, 7)^j$, where $[s, f)^j$ denotes j requests for interval $[s, f)$. Note that all these requests are subintervals of either [1,7) or [7,28).

First, we claim that $\mathrm{KT}^{\mathbb{A}}$ accepts all the requests in this sequence regardless of the ordering. Note that the density within the interval [1,7) is $2j = \frac{n}{3} \leq \frac{n+2}{3}$. For each request which is a subinterval of [1,7), there exists a $1 \leq i \leq 2j$ such that it is placed in bin i and put in seat $3i - 4$ if $i > 1$ or in seat 1 if $i = 1$. Furthermore, there are only n requests which are subintervals of [7,28), so by the principle of fairness, $\mathrm{KT}^{\mathbb{A}}$ must accept all of them. Thus, $\mathrm{KT}^{\mathbb{A}}$ accepts all the requests.

Best-Fit, on the other hand, given the specified ordering, will never place any of the length 1 requests on the same seat, thereby rejecting the last j requests. See

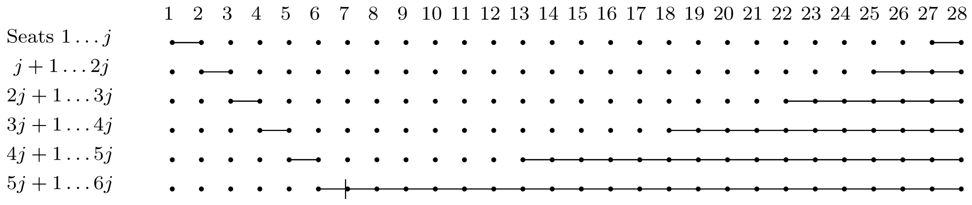

FIG. 3. The behavior of Best-Fit for the sequence in Lemma 7.4.

Figure 3 for an illustration. Since each accepted request contributes at least one unit of profit, we have:

$$\mathrm{KT}^{\mathbb{A}}_{\mathrm{W}}(I_n) - \mathrm{BF}_{\mathrm{W}}(I_n) \geq \mathrm{KT}^{\mathbb{A}}_{\mathrm{W}}(I_n) - \mathrm{BF}(I_n) \geq j = \frac{n}{6} \to \infty \quad \text{as } n \to \infty. \qquad \square$$

Again, the combination of Lemmas 7.2 and 7.4 disproves any possibility for the relative worst order ratio to be defined in the general case but allows for the possibility of the relative worst order ratio to be defined for some cases of k restricted from above. In this scenario, however, $\mathrm{KT}^{\mathbb{A}}$ would have to be better than BF since the proof of Lemma 7.4 only requires $k \geq 28$.

This concludes the analysis of Best-Fit. Turning our attention to Worst-Fit, Theorem 6.3 tells us that $\mathrm{KT}^{\mathbb{A}}$ is no worse than Worst-Fit. The exact ratio will depend on the algorithm $\mathbb{A}$. However, we can show using the following lemma that, regardless of $\mathbb{A}$, $\mathrm{KT}^{\mathbb{A}}$ is strictly better than WF.

LEMMA 7.5. *For $k \geq 5$, there does not exist a constant b such that $KT^{\mathbb{A}}_{W}(I) \leq WF_W(I) + b$ for all I for both the unit and proportional price problem.*

PROOF. We define a family of sequences I_n such that $\lim_{n\to\infty} \mathrm{KT}^{\mathbb{A}}_{\mathrm{W}}(I_n) - \mathrm{WF}_{\mathrm{W}}(I_n) = \infty$. We assume that n is divisible by $k-1$. I_n consists of $\frac{n}{k-1}$ requests for each of the intervals $[1, 2), [2, 3), \dots, [k-1, k)$, followed by $\lfloor \frac{n(k-4)}{3(k-1)} \rfloor$ requests for $[1, k)$. Since the density of I_n is $\frac{n}{k-1} + \lfloor \frac{n(k-4)}{3(k-1)} \rfloor \leq \frac{n}{3}$, Lemma 7.1 gives that $\mathrm{KT}^{\mathbb{A}}$ will accept all the requests regardless of the ordering. Worst-Fit, however, will reject the last $\lfloor \frac{n(k-4)}{3(k-1)} \rfloor$ requests given the specified ordering. Since each accepted request contributes at least one unit of profit, we have that $\mathrm{KT}^{\mathbb{A}}{}_{\mathrm{W}}(I_n) - \mathrm{WF}_{\mathrm{W}}(I_n) \geq \lfloor \frac{n(k-4)}{3(k-1)} \rfloor \to \infty$ as $n \to \infty$. $\square$

In the following theorem, all the results obtained in this section are summarized.

THEOREM 7.6. *For both the unit and proportional price problem,*

—*$KT^{\mathbb{A}}$ is incomparable to First-Fit,*

—*$KT^{\mathbb{A}}$ is incomparable to Best-Fit,*

—*for $k \geq 5$, $WR_{KT^{\mathbb{A}},WF} > 1$.*

PROOF. The incomparability of $\mathrm{KT}^{\mathbb{A}}$ to First-Fit is established by Lemmas 7.2 and 7.3. The incomparability of $\mathrm{KT}^{\mathbb{A}}$ to Best-Fit is established by Lemmas 7.2 and 7.4. The fact that $\mathrm{KT}^{\mathbb{A}}$ is strictly better than Worst-Fit is established by Theorem 6.3 and Lemma 7.5. $\square$

8. *Randomized Algorithms*

In this section, we consider the most intuitive randomized algorithm, Random (RND), defined in Section 3, and show that it is incomparable to any Any-Fit algorithm. The first step is to prove that Random's performance cannot be better than or equal to that of any Any-Fit algorithm for all sequences, and this is achieved by the following lemma.

LEMMA 8.1. *Let $\mathbb{A}$ be any Any-Fit algorithm. Then there does not exist a constant b such that, for all I, $\text{RND}_{\text{W}}(I) \geq \mathbb{A}_{\text{W}}(I) - b$, for both the unit price and proportional price problem.*

PROOF. Consider the following family of sequences I_n, defined for n divisible by 2. First there are $\frac{n}{2}$ requests for [1,2), then there are $\frac{n}{2}$ requests for [k-1,k), and finally, there are $\frac{n}{2}$ requests for [1,k). Clearly, $\mathbb{A}$ will accept all requests regardless of the ordering. We will show that $\lim_{n\to\infty} \mathbb{A}_{\text{W}}(I_n) - \text{RND}_{\text{W}}(I_n) = \infty$, which will imply the statement of the lemma.

In general, for any random algorithm $\mathbb{R}$, we have

$$\begin{aligned}\mathbb{A}_{\text{W}}(I_n) - \mathbb{R}_{\text{W}}(I_n) &= \text{OPT}(I_n) - \min_{\sigma} E[\mathbb{R}(\sigma(I_n))]\\ &\geq \text{OPT}(I_n) - E(\mathbb{R}(I_n))\\ &= \sum_{x=0}^{\text{OPT}(I_n)} (\text{OPT}(I_n) - x) \cdot P[\mathbb{R}(I_n) = x].\end{aligned}$$

This result is intuitively simple. When the deterministic algorithm is optimal regardless of the ordering, then the goal of minimizing the expected difference of accepted requests is the same as the goal of maximizing the expected number of rejected requests of the random algorithm. Moreover, for a lower bound, we can consider any permutation of the sequence we wish, which in this case will be the specified order. For I_n, the probability of rejecting x requests is the probability of having exactly x requests from the second group assigned to inactive seats. Given an assignment for the first group of requests, there are $\binom{\frac{n}{2}}{x}\binom{\frac{n}{2}}{\frac{n}{2}-x}$ possibilities for this out of a total of $\binom{n}{\frac{n}{2}}$. Since all assignments are equally likely, we can conclude the preceding calculation as follows:

$$\begin{aligned}\mathbb{A}_{\text{W}}(I_n) - \text{RND}_{\text{W}}(I_n) &\geq \sum_{x=0}^{\frac{n}{2}} x \cdot \frac{\binom{\frac{n}{2}}{x}^2}{\binom{n}{\frac{n}{2}}} = \frac{\frac{n}{2}}{\binom{n}{\frac{n}{2}}} \sum_{x=0}^{\frac{n}{2}} \binom{\frac{n}{2}}{x}\binom{\frac{n}{2}-1}{x-1}\\ &= \frac{\frac{n}{2}}{\binom{n}{\frac{n}{2}}}\binom{n-1}{\frac{n}{2}-1} = \frac{n}{4}.\end{aligned}$$

Here, we assume that $\binom{n}{k} = 0$ when $k < 0$, and the summation is removed by applying Vandermonde's convolution [Graham et al. 1988]. The inequality comes from the fact that both the unit price and proportional price algorithms must assign at least one unit of profit to each accepted request. The statement of the limit follows directly. □

To prove the incomparability result, we next show that the performance of any Any-Fit algorithm $\mathbb{A}$ can be worse than that of Random for some sequences. We consider the following family of sequences I_n, defined for n divisible by 2. It consists of $\frac{n}{2}$ requests for [1,2), $\frac{n}{2}$ requests for $[k-1, k)$, $\frac{n}{2}$ requests for $[1, k-1)$, and $\frac{n}{2}$ requests for $[2, k)$. Call the requests of length 1 short, and the requests of length $k-2$ long. In an ordering where all the short requests come before all the long requests, $\mathbb{A}$ will seat only $\frac{3n}{2}$ requests.

It is necessary to find the worst permutation of I_n for Random. Regardless of the ordering, all the short requests are accepted. The more short requests that have been placed already, the greater the chances of a long request having to be rejected. This intuition explains why having all the short requests before all the long requests gives a worst permutation for Random. We now proceed to prove that this is in fact the case.

Call the requests $[1, k-1)$ long-left requests, the requests $[2, k)$ long-right requests, the requests $[1, 2)$ short-left requests, and the requests $[k-1, k)$ short-right requests.

LEMMA 8.2. *None of the short requests can be rejected by any (deterministic or randomized) algorithm.*

PROOF. The density is at most n everywhere, so no length 1 requests can be rejected. □

LEMMA 8.3. *The number of requests rejected by any (deterministic or randomized) algorithm is equal to the number of seats which receive two short requests.*

PROOF. First, note that if there are j seats with two short requests, then there are j seats which do not contain a long request. At most one long request can be placed on any seat, and there are n long requests in all, so at least j long requests must be rejected.

Suppose there are j rejected requests. By Lemma 8.3, they must all be long, so they contain the subinterval [2, 3). Since I is an accommodating sequence, there must be at least j seats which are empty from station 2 to station 3. Each of these j seats must contain some request which overlaps each rejected request. If both long-left and long-right requests are rejected, then each of these j seats must contain both types of short requests. Suppose only long-left (long-right) requests are rejected. Then, every seat must contain either a long-right (long-left) or a short-right (short-left) request since there are n such requests and none are rejected. The j seats which are empty in the subinterval [2, 3) must thus have short-right (short-left) requests. They must also have short-left (short-right) requests since long-left (long-right) requests were rejected. In either case, there must be at least j seats with two short requests.

This shows that the number of rejected requests must be exactly the number of seats with two short requests. □

LEMMA 8.4. *There is no ordering of the requests for which the expected number of seats to which Random assigns two short requests is greater than when all the short requests come first.*

PROOF. Suppose for the sake of contradiction that on the ordering I_1, the expected number of seats where Random assigns two short requests is greater than when all the short requests come first. We will modify I_1, step-by-step, eventually getting to an ordering I_{final} with all short requests first but without decreasing the expected number of seats where Random assigns two short requests. This will give the contradiction.

Denote by $f(I)$ the expected number of seats where Random assigns two short requests, where I is a permutation of I_n. In order to show that $f(I_{\mathit{final}}) \geq f(I_1)$, we consider Random's source of randomness. We assume, without loss of generality, that Random has a source of random numbers which can be represented as a set of sequences of random bits where a sequence corresponding to one execution of Random is called a random sequence. In the following, this set of random sequences is partitioned based on the output of Random on I_1 given these random sequences, and it is shown for each part of the partition that the reordering does not decrease the expected number of seats where Random assigns two short requests.

Let x be the last long request in I_1 which is immediately followed by a short request. Assume without loss of generality that x is long-left, and move all short-right requests which occur after it immediately before it. Call this new ordering I_2. The short requests which have been moved before x are placed independently of x in I_1's ordering. Therefore, $f(I_1) = f(I_2)$. If x is no longer followed by any short requests, we are finished with this step and proceed to find a new request x if such a request exists. Otherwise, assume x is followed by $i > 0$ short-left requests. Move x just after these i Type C requests and call the new ordering I_3. Observe that we can continually apply these two reorderings until we get to a permutation I_{final} where all the small requests come first. Thus, all that remains to show is that $f(I_3) \geq f(I_2)$.

For those sequences of random bits which cause Random to reject x in I_2, x is also rejected in I_3, in which case $f(I_3) = f(I_2)$. For the other case, consider the set $R(r, s, t)$ of random sequences which lead Random to accept x in I_2 and cause Random to place the prefix of I_2 preceding x such that r seats have only long-right requests, s seats have only short-right requests, and t seats are empty. The request x can only be placed on empty seats or those containing only short-right requests, whereas the i short-left requests can also be placed on any seat containing only a long-right request. When Random is run using a random sequence from $R(r, s, t)$ on the sequence I_2, the request x has probability $\frac{s}{s+t}$ of being placed on the same seat as a short-right request and probability $\frac{t}{s+t}$ of being placed on a seat which is empty. Thus, the expected number of the last i short-left requests which are placed with short-right requests (all of which have already been placed) is $\frac{s}{s+t}(\frac{(s-1)i}{r+s+t-1}) + \frac{t}{s+t}(\frac{si}{r+s+t-1}) = \frac{si}{s+t}(\frac{s+t-1}{r+s+t-1}) \leq \frac{si}{r+s+t}$. This last value is also the expected number of the last i short-left requests which are placed with short-right requests when Random is run using a random sequence from $R(r, s, t)$ on the sequence I_3. Since this holds for all possible sets $R(r, s, t)$, this implies that $f(I_3) \geq f(I_2)$. $\square$

By Lemma 8.3, the probability of rejecting x requests is the probability of having exactly x seats which have two short requests placed on them. But the expected number of such seats was already calculated in the proof of Lemma 8.1 and is $\frac{n}{4}$. Thus, for the unit price problem and any Any-Fit algorithm $\mathbb{A}$, $\min_\sigma E[\mathbb{A}(\sigma(I_n))] = \frac{3n}{2}$, while $\min_\sigma E[\text{RND}(\sigma(I_n))] = 2n - \frac{n}{4} = \frac{7n}{4}$, and $\lim_{n\to\infty} \text{RND}_{\text{W}}(I_n) - \mathbb{A}_{\text{W}}(I_n) = \infty$. Similarly, for the proportional price problem, $\min_\sigma E[\mathbb{A}(\sigma(I_n))] = \frac{nk}{2}$, while $\min_\sigma E[\text{RND}(\sigma(I_n))] = n(k-1) - \frac{n}{4}(k-2) = \frac{3nk}{4} - \frac{3n}{2}$.

This gives us the following:

THEOREM 8.5. *Random is incomparable to any Any-Fit algorithm for both the unit price problem and the proportional price problem.*

With respect to Worst-Fit, the results can be obtained more easily. However, finding the exact value of $\mathrm{WR}_{\mathrm{RND,WF}}$ does not seem to be an interesting direction of research since the fact that Worst-Fit is not an effective algorithm seems to be established.

9. *Concluding Remarks and Open Problems*

The relative worst order ratio has previously been applied to some problems and has led to intuitively and/or experimentally correct results which could not be obtained with the competitive ratio [Boyar and Favrholdt 2003; Boyar et al. 2005; Epstein et al. 2006; Kohrt 2004]. Now, for the seat reservation problem, applying the relative worst order ratio has also proven very helpful in differentiating between various 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 5.2. As is, Theorem 5.2 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 $\mathbb{A}$ [Boyar and Favrholdt 2003], although the competitive ratio on accommodating sequences is always at least $\frac{1}{2}$ [Boyar et al. 2001].

With respect to the algorithms described in this article, the most interesting open problem is to close the gap for the ratio of First-Fit to Best-Fit for both the unit price and proportional price problems. It is also of interest to see if First-Fit is strictly better than $\mathrm{KT}^{\mathbb{A}}$ for some limited values of k. Ultimately, the goal is to find an algorithm that is better than the existing ones as has been done for the paging problem [Boyar et al. 2005]. 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.

ACKNOWLEDGMENT. We would like to thank the anonymous referee who helped in improving the presentation of these results.

REFERENCES

BACH, E., BOYAR, J., EPSTEIN, L., FAVRHOLDT, L. M., JIANG, T., LARSEN, K. S., LIN, G.-H., AND VAN STEE, R. 2003. Tight bounds on the competitive ratio on accommodating sequences for the seat reservation problem. *J. Sched. 6*, 131–147.

BEN-DAVID, S., AND BORODIN, A. 1994. A new measure for the study of online algorithms. *Algorithmica 11,* 1, 73–91.

BOYAR, J., AND FAVRHOLDT, L. M. 2003. The relative worst order ratio for online algorithms. In *Proceedings of the 5th Italian Conference on Algorithms and Complexity*. Lecture Notes in Computer Science, vol. 2653. 58–69. Extended version in *ACM Trans. Algor. 3,* 2, 2007.

BOYAR, J., FAVRHOLDT, L. M., AND LARSEN, K. S. 2007. The relative worst order ratio applied to paging. *J. Comput. Syst. Sci. 73*, 818–843.

BOYAR, J., FAVRHOLDT, L. M., LARSEN, K. S., AND NIELSEN, M. N. 2003. Extending the accommodating function. *Acta Informatica 40*, 3–35.

BOYAR, J., AND LARSEN, K. S. 1999. The seat reservation problem. *Algorithmica 25*, 403–417.

BOYAR, J., LARSEN, K. S., AND NIELSEN, M. N. 2001. The accommodating function—a generalization of the competitive ratio. *SIAM J. Comput. 31,* 1, 233–258.

BOYAR, J., AND MEDVEDEV, P. 2007. The relative worst order ratio applied to seat reservation. Tech. rep. PP–2007–03, Department of Mathematics and Computer Science, University of Southern Denmark.

CHROBAK, M., AND ŚLUSAREK, M. 1988. On some packing problems relating to dynamical storage allocation. *RAIRO Theoret. Inform. Appl. 22*, 487–499.

DORRIGIV, R., AND LÓPEZ-ORTIZ, A. 2005. A survey of performance measures for online algorithms. *SIGACT News 36,* 3, 1–17.

EPSTEIN, L., FAVRHOLDT, L. M., AND KOHRT, J. S. 2006. Separating online scheduling algorithms with the relative worst order ratio. *J. Combin. Optim. 12,* 4, 362–385.

GRAHAM, R. L. 1966. Bounds for certain multiprocessing anomalies. *Bell Systems Tech. J. 45*, 1563–1581.

GRAHAM, R. L., KNUTH, D. E., AND PATASHNIK, O. 1988. *Concrete Mathematics*. Equation 5.23. Addison-Wesley.

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

KARLIN, A. R., MANASSE, M. S., RUDOLPH, L., AND SLEATOR, D. D. 1988. Competitive snoopy caching. *Algorithmica 3,* 1, 79–119.

KENYON, C. 1996. Best-fit bin-packing with random order. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*. 359–364.

KIERSTEAD, H. A., AND TROTTER, W. T. 1981. An extremal problem in recursive combinatorics. *Congressus Numerantium 33*, 143–153.

KOHRT, J. S. 2004. Online algorithms under new assumptions. Ph.D. thesis, Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark.

SLEATOR, D. D., AND TARJAN, R. E. 1985. Amortized efficiency of list update and paging rules. *Comm. ACM 28,* 2, 202–208.

RECEIVED JUNE 2006; REVISED JULY 2007; ACCEPTED APRIL 2008