Tight Bounds on the Competitive Ratio on Accommodating Sequences for the Seat Reservation Problem

Eric Bach [*] Joan Boyar [†] Leah Epstein [‡]
Lene M. Favrholdt [†] Tao Jiang [§] Kim S. Larsen [†]
Guo-Hui Lin [¶] Rob van Stee [‖]

Abstract

The unit price seat reservation problem is investigated. The seat reservation problem is the problem of assigning seat numbers on-line to requests for reservations in a train traveling through k stations. We are considering the version where all tickets have the same price and where requests are treated fairly, i.e., a request which can be fulfilled must be granted.

For fair deterministic algorithms, we provide an asymptotically matching upper bound to the existing lower bound which states that all fair algorithms for this problem are $\frac{1}{2}$-competitive on accommodating sequences, when there are at least three seats.

[*] Computer Sciences Department, University of Wisconsin – Madison, 1210 West Dayton Street, Madison, WI 53706-1685, U.S.A. Email: bach@cs.wisc.edu.

[†] Department of Mathematics and Computer Science, University of Southern Denmark, Main Campus: Odense University, Campusvej 55, DK-5230 Odense M, Denmark. Email: {joan,lenem,kslarsen}@imada.sdu.dk.

[‡] School of Computer and Media Sciences, The Interdisciplinary Center, P.O.B. 167, 46150 Herzliya, Israel. Email: epstein.leah@idc.ac.il.

[§] Department of Computer Science, University of California, Riverside. On leave from Department of Computing and Software, McMaster University, Hamilton, Ontario L8S 4L7, Canada. Email: jiang@cs.ucr.edu.

[¶] Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada and Department of Computing and Software, McMaster University, Canada. Email: ghlin@wh.math.uwaterloo.ca.

[‖] Centre for Mathematics and Computer Science (CWI), P.O.B. 94079, 1090 GB Amsterdam, The Netherlands. Email: Rob.van.Stee@cwi.nl.

Additionally, we give an asymptotic upper bound of $\frac{7}{9}$ for fair randomized algorithms against oblivious adversaries.

We also examine concrete on-line algorithms, First-Fit and Random, for the special case of two seats. Tight analyses of their performance are given.

1 Introduction

In many train transportation systems, passengers are required to buy seat reservations with their train tickets. The ticketing system must assign a passenger a single seat when that passenger purchases a ticket, without knowing what future requests there will be for seats. Therefore, the seat reservation problem is an on-line problem, and a competitive analysis is appropriate.

Assume that a train with n seats travels from a start station to an end station, stopping at $k \geq 2$ 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. Reservations can be made for any trip from a station s to a station t as long as $1 \leq s < t \leq k$. Each passenger is given a single seat number when the ticket is purchased, which can be any time before departure. The algorithms (ticket agents) may not refuse a passenger if it is possible to accommodate him when he attempts to make his reservation. That is, if there is any seat which is empty for the entire duration of that passenger's trip, the passenger must be assigned a seat. An algorithm of this kind is *fair*.

The algorithms attempt to maximize income, i.e., the sum of the prices of the tickets sold. Naturally, the performance of an on-line algorithm will depend on the pricing policies for the train tickets. In [6], two pricing policies are considered: one in which all tickets have the same price, the *unit price problem*; and one in which the price of a ticket is proportional to the distance traveled, the *proportional price problem*. This paper focuses on fair algorithms for the unit price problem.

The seat reservation problem is closely related to the problem of optical routing with a number of wavelengths [1, 5, 9, 14], call control [2], interval graph coloring [12] and interval scheduling [13]. The off-line version of the seat reservation problem can be used to solve the following problems [8]: minimizing spill in local register allocation, job scheduling with start and end times, and routing of two point nets in VLSI design. Another application of the on-line version of the problem could be to assign vacation bungalows (mentioned in [15]).

The performance of an on-line algorithm $\mathcal{A}$ is usually analyzed using the competitive ratio defined in the following way.

Let $\mathcal{A}(I)$ denote how much an on-line algorithm $\mathcal{A}$ earns with the request sequence I, and let $\mathrm{OPT}(I)$ denote how much is earned by an optimal off-line algorithm when given the sequence I. If $\mathcal{A}$ is randomized, $E[\mathcal{A}(I)]$ denotes the expected value of $\mathcal{A}(I)$.

Definition 1.1 A deterministic on-line algorithm $\mathcal{A}$ is *c-competitive* if, for any sequence I of requests, $\mathcal{A}(I) \geq c\cdot\ \mathrm{OPT}(I) - b$, where b is a constant which does not depend on the input sequence I. The *competitive ratio* for $\mathcal{A}$ is the supremum over all such c. □

When measuring the performance of a randomized on-line algorithm there is a choice of adversaries. Throughout the paper we use an oblivious adversary. An *oblivious adversary* knows the exact definition of the on-line algorithm, but it constructs the whole input sequence before giving it. That is, the adversary does not know how the algorithm chooses to serve each request, but it knows the probabilities of the possible actions of the on-line algorithm.

Definition 1.2 A randomized on-line algorithm $\mathcal{A}$ is *c-competitive* against an oblivious adversary if, for any sequence I of requests, $E[\mathcal{A}(I)] \geq c\cdot$ $\mathrm{OPT}(I) - b$, where b is a constant which does not depend on the input sequence I. The *competitive ratio* for $\mathcal{A}$ is the supremum over all such c. □

Since we are trying to maximize income rather than minimize cost, a lower bound is obtained by proving a bound on the worst case behavior of an algorithm, and an upper bound is obtained by giving an adversary argument. Notice that the fairness criterion defined above is a part of the problem specification. Thus, even though the optimal off-line algorithm knows the whole sequence of requests in advance, it must process the requests in the same order as the on-line algorithm, and do so fairly.

In this paper, we investigate the competitive ratio in the special case where there are enough seats to accommodate all requests, i.e. an optimal off-line algorithm will not reject any of the requests. This restriction on the input sequences is used to reflect the assumption that the decision as to how many cars the train should have is based on expected ticket demand.

Definition 1.3 A sequence of requests that can be fully accommodated by an optimal off-line algorithm is called an *accommodating* sequence. □

In earlier papers [6, 7], the competitive ratio on accommodating sequences was called the accommodating ratio. The change is made here for consistency with common practice in the field.

1.1 Previous Results

We have the following known results:

Theorem 1.1 [6] On accommodating sequences, any fair (deterministic or randomized) on-line algorithm for the unit price problem is at least $\frac{1}{2}$-competitive. □

Theorem 1.2 [6] Even on accommodating sequences, any fair (deterministic or randomized) on-line algorithm for the unit price problem ($k \geq 6$) is at most $\frac{8k-8(k \bmod 3)-9}{10k-10(k \bmod 3)-15}$-competitive. □

Thus, even on accommodating sequences, no fair randomized on-line algorithm has a competitive ratio much better than $\frac{4}{5}$.

The results in [6] for the proportional price problem show that its competitive ratio, even on accommodating sequences, is $\Theta(\frac{1}{k})$. For the unit price problem, the competitive ratio is also $\Theta(\frac{1}{k})$. These very discouraging results explain the focus on the competitive ratio on accommodating sequences for the unit price problem in this paper.

1.2 Our Contributions

In Section 3, we lower the asymptotic upper bound on the competitive ratio on accommodating sequences for fair deterministic algorithms from $\frac{4}{5}$ to $\frac{1}{2}$ when k is large compared to n, and $n \geq 3$. This matches the lower bound from Theorem 1.1. For fair randomized algorithms against oblivious adversaries, we show an upper bound of $\frac{7}{9}$ for large k. A concrete on-line algorithm, First-Fit, is examined with regards to the unit price problem for the special case $n = 2$. Here, we show that First-Fit is $\frac{3}{5}$-competitive on accommodating sequences, and we show that this is asymptotically optimal. In Section 4, we examine a concrete randomized on-line algorithm, Random.

We prove an asymptotic upper bound of $\frac{17}{24}$ for $n \geq 3$, and for the special case of $n = 2$, we find asymptotically matching upper and lower bounds of $\frac{3}{4}$.

Our results are summarized in Table 1. For the sake of clarity, some of the values given there are not quite as tight as those proven in the paper. The lower bounds of $\frac{1}{2}$ and the upper bound on First-Fit for $n \geq 3$ are from [6].

	$n = 2$	$n \geq 3$
Any det. alg.	$\frac{1}{2} \leq c \leq \frac{3k-6}{5k-18}$	$\frac{1}{2} \leq c \leq \frac{1}{2} + \frac{3n-3}{2k+6n-18}$
First-Fit	$\frac{3}{5} \leq c \leq \frac{3k-6}{5k-18}$	$\frac{1}{2} \leq c \leq \frac{1}{2} + \frac{3}{2k-6}$
Any rand. alg.	$\frac{1}{2} \leq c \leq \frac{7k-15}{9k-27}$	$\frac{1}{2} \leq c \leq \frac{7k-15}{9k-27}$
Random	$\frac{3}{4} \leq c \leq \frac{3}{4} + \frac{1}{4k-4}$	$\frac{1}{2} \leq c \leq \frac{17k+14}{24k}$

Table 1: Simplified upper and lower bounds.

Some of the results in this paper were presented in [4].

2 Coloring Interval Graphs

Since we are only considering the unit price problem, the seat reservation problem is similar to the problem of coloring an interval graph on-line. This is easy to see. The route the train travels from station 1 through station k is the section of the real line considered. The part of the route a passenger travels is an open interval, and the seat the passenger is assigned is the color the interval is given.

Note that in the case where there are enough seats to accommodate all requests, the restriction that the optimal off-line algorithm be fair is in fact no restriction. Thus, the optimal fair off-line algorithm is polynomial time [10] since it is simply a matter of coloring an interval graph with the minimum number of colors. Recall that interval graphs are *perfect* [11], so the size of the largest clique is exactly the number of colors needed. Thus, when there is no pair of stations $(s, s + 1)$ such that the number of people who want to be on the train between stations s and $s + 1$ is greater than n, the optimal fair off-line algorithm will be able to accommodate all requests. The contrapositive is clearly also true; if there is a pair of stations such that the number of people who want to be on the train between those stations is greater than n, the optimal fair off-line algorithm will be unable to accommodate all requests. We will refer to the number of people who want to be on the train between two stations as the *density* between those stations.

3 Deterministic Algorithms

In this section, we investigate the competitive ratios of deterministic fair algorithms for the unit price seat reservation problem. We consider the cases $n = 2$ and $n \geq 3$ separately. Trivially, for $n = 1$, any fair on-line algorithm is 1-competitive on accommodating sequences.

3.1 A General Upper Bound for $n \geq 3$

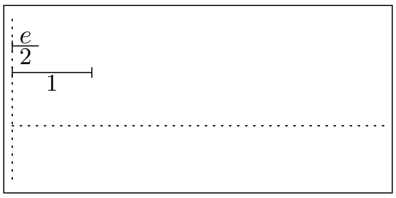
The upper bound on the competitive ratio on accommodating sequences is lowered to match the lower bound, for k large compared to n.

Theorem 3.1 The competitive ratio on accommodating sequences for the fair unit price seat reservation problem with 3 seats is at most $\frac{1}{2} + \frac{3}{k+5}$, where $k \geq 7$ and $k \equiv 1 \pmod 6$.

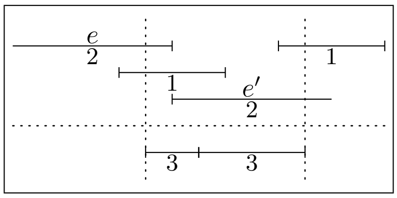
Proof The adversary gives a request sequence I in three phases. In Phase 1, the adversary gives intervals of length four with a spacing of two, except that the first interval only has length three, and the last has length one. The intervals are: $[1,4], [6,10], [12,16], \ldots, [k-1,k]$. These intervals are numbered 1 in Fig. 1.

In Phases 2 and 3, the adversary gives additional requests, determined by processing the already given requests from left to right, based on how the on-line algorithm $\mathcal{A}$ has placed the Phase 1 intervals. This processing is completed before the intervals are actually given. All Phase 2 intervals are given before all Phase 3 intervals. Within a phase, the intervals are given in order, based on their leftmost endpoint.

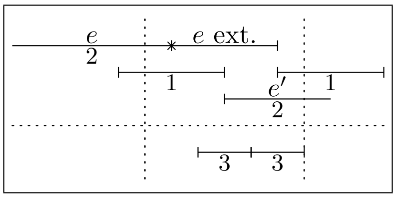
The proof is by induction on the number of intervals in Phase 1. In each step one such interval is processed. The processing is illustrated in Fig. 1. A step occurs between two dotted vertical lines. Since $\mathcal{A}$ is fair, all intervals from Phases 1 and 2 are accepted; these intervals are shown above the horizontal dotted line. During a step, we either extend an interval from Phase 2 or introduce a new interval for Phase 2, plus define some intervals for Phase 3. The intervals from Phase 3 are rejected by $\mathcal{A}$; these intervals are shown



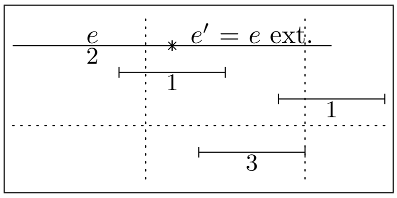
a) Invariant and base case.



b) Next interval with e.



c) Next interval with previous.



d) Next interval on free.

Figure 1: Cases of the proof.

below the horizontal dotted line. After each step, the invariant depicted in Fig. 1a will hold:

- Every interval from Phase 1 or 2 (except the first two all the way to the left) with its start station to the left of or on the vertical dotted line has a unique associated Phase 3 interval.
- The only intervals with start station strictly to the right of the vertical dotted line are the Phase 1 intervals.

Establishing the base case can be done by giving the Phase 2 interval $e = [1, 2]$ as illustrated in Fig. 1a. This interval and the Phase 1 interval $[1, 4]$ will be the only ones without associated Phase 3 intervals.

Note that this e interval as well as other Phase 2 intervals to be given later may be extended during the processing of the next Phase 1 interval. In the figure, this is marked with an asterisk.

For the induction step, we assume that we have processed up to a certain station and have been able to maintain the invariant. In processing the next Phase 1 interval, there are three cases: the interval could be on the same seat as e (Fig. 1b), on the same seat as the previous Phase 1 interval

(Fig. 1c), or on the third seat (Fig. 1d). In each case, we must reestablish the invariant six stations further to the right (the next vertical dotted line).

In Fig. 1b, we give a new Phase 2 interval e' which should serve as the e interval in the next step. Note that $\mathcal{A}$ will have no choice as to where to place the interval. Two Phase 3 intervals are defined, one for the Phase 1 interval processed and one for the new e'.

In Fig. 1c, we extend the already given e interval. If this e interval is the first interval of Phase 2, extending the interval might change the algorithm's decision as to where to place the interval. Note that this is no problem, since the only effect of this would be that e and e' are swapped. Two Phase 3 intervals are defined, one for the Phase 1 interval processed and one for the new e'. Again, e' will be the new e interval in the next step.

In Fig. 1d, we also extend the e interval, and that interval will also serve as the e interval in the next step. Thus, only one Phase 3 interval must be provided for the processed Phase 1 interval. Note that extending the e interval cannot influence its placement, since only one seat has room for it.

An optimal off-line algorithm can accommodate all requests, since the density is no more than three between any two stations, which implies that the clique number of the corresponding interval graph is at most three.

In summary, the adversary gives $\frac{k+5}{6}$ Phase 1 intervals and, except for the first of these, one Phase 3 interval for each. Additionally, it gives some number $x \geq 1$ of intervals in Phase 2, and $x - 1$ Phase 3 intervals for these. In total, we get the ratio

$$\frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} = \frac{\frac{k+5}{6} + x}{\frac{k+5}{6} + (\frac{k+5}{6} - 1) + x + (x - 1)}.$$

The maximum occurs at $x = 1$, so

$$\frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} \leq \frac{\frac{k+5}{6} + 1}{\frac{k+5}{6} + (\frac{k+5}{6} - 1) + 1} = \frac{k + 11}{2k + 10} = \frac{1}{2} + \frac{3}{k + 5}.$$

□

The above result can easily be extended to any $k \geq 7$, by giving the same sequence and ignoring the last few stations. Let $c \equiv k - 1 \pmod 6$. The upper bound is then $\frac{1}{2} + \frac{3}{k+(5-c)}$.

Corollary 3.1 The competitive ratio on accommodating sequences for the fair unit price seat reservation problem with $n \geq 3$ seats is at most $\frac{1}{2} + \frac{3n-3}{2k+6n-(8+2c)}$, where $k \geq 7$ and $c \equiv k-1 \pmod 6$.

Proof First we give $n-3$ intervals of the type $[1, k]$. Because of fairness, both the on-line as well as the optimal off-line algorithm must accept all $n-3$ requests. Now use the above theorem on the remaining three seats. The ratio becomes

$$\begin{aligned}\frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} &\leq \frac{\frac{(k-c)+5}{6} + x + (n-3)}{\frac{(k-c)+5}{6} + (\frac{(k-c)+5}{6} - 1) + x + (x-1) + (n-3)} \\ &\leq \frac{k+6n-(7+c)}{2k+6n-(8+2c)} = \frac{1}{2} + \frac{3n-3}{2k+6n-(8+2c)}.\end{aligned}$$

□

The technique of Corollary 3.1 which converts an upper bound for m seats to an upper bound for n seats ($n > m$) by adding some large requests in the beginning of the sequence, can be applied to any sequence. This can be done both to negative results for deterministic algorithms and to negative results for randomized algorithms. Thus, the asymptotic competitive ratio is a monotone non-increasing function of n, if there is no limit on the number of stations.

3.2 The Case $n = 2$

In this section, the case $n = 2$ is investigated. However, we note that in train systems, it is unlikely that a train has a small number of seats. So the bounds obtained here are probably irrelevant for this application, but they could be relevant for others such as assigning vacation bungalows.

The following theorem gives an upper bound on the competitive ratio on accommodating sequences for fair algorithms. This bound approaches $\frac{3}{5}$ as k approaches infinity.

Theorem 3.2 Let

$$f(k) = \begin{cases} \frac{3k-3(k \bmod 6)-6}{5k-5(k \bmod 6)-18}, & \text{when } (k \bmod 6) \in \{0,1,2\}; \\ \frac{3k-3(k \bmod 6)+6}{5k-5(k \bmod 6)+6}, & \text{when } (k \bmod 6) \in \{3,4,5\}. \end{cases}$$

If $n = 2$, any deterministic fair on-line algorithm for the unit price problem ($k \geq 9$) is at most $f(k)$-competitive, even on accommodating sequences.

Proof The adversary begins with one request for the interval $[3s+1, 3s+3]$ for each $s = 0, 1, \cdots, \lfloor \frac{k-3}{3} \rfloor$. After these requests are accommodated by the on-line algorithm, consider for each $i = 0, 1, \cdots, \lfloor \frac{k-9}{6} \rfloor$ how the three requests $[6i+1, 6i+3]$, $[6i+4, 6i+6]$, and $[6i+7, 6i+9]$ are accommodated. Suppose that the intervals $[6i+1, 6i+3]$, $[6i+4, 6i+6]$, and $[6i+7, 6i+9]$ are placed on the same seat. Then the adversary proceeds with a request for the interval $[6i+3, 6i+7]$ and then requests for each of the intervals $[6i+2, 6i+4]$ and $[6i+6, 6i+8]$. The on-line algorithm will accommodate the first request, but fail to accommodate the last two. In the second case, suppose only two adjacent intervals (among $[6i+1, 6i+3]$, $[6i+4, 6i+6]$, and $[6i+7, 6i+9]$) are placed on the same seat, say $[6i+1, 6i+3]$ and $[6i+4, 6i+6]$, then the adversary proceeds with three requests for the intervals $[6i+2, 6i+4]$, $[6i+3, 6i+5]$, and $[6i+5, 6i+8]$. The on-line algorithm will accommodate the first request but fail to accommodate the last two. In the last case, only the intervals $[6i+1, 6i+3]$ and $[6i+7, 6i+9]$ are placed on the same seat. Then the adversary proceeds with two requests for the intervals $[6i+2, 6i+5]$ and $[6i+5, 6i+8]$. The on-line algorithm will fail to accommodate both of them.

It then follows easily that, even on accommodating sequences, the competitive ratio of the on-line algorithm applied to this sequence of requests is at most $f(k)$ ($k \geq 9$). □

A specific on-line algorithm called *First-Fit* always processes a new request by placing it on the first seat which is unoccupied for the length of the journey. The following theorem shows that for $n = 2$, First-Fit is an asymptotically optimal on-line algorithm.

Theorem 3.3 First-Fit for the unit price problem is at least $\frac{3}{5}$-competitive on accommodating sequences, when $n = 2$.

Proof Consider any set of requests which the optimal off-line algorithm could accommodate with two seats. Let S be the subset of requests accommodated by First-Fit, and let U denote the subset of unaccommodated requests. The non-empty intervals between two consecutive requests accommodated on some seat (i.e., the durations in which the seat is empty) are called *gaps* on that seat. Since the sequence is accommodating, every request in U must have its starting station in a gap, and no two requests in U can have their starting stations in the same gap. Partition U into U_1 and U_2, where U_i denotes the subset of requests in U with starting station s in some gap on seat i. Intervals which have their starting station in a gap on both seats should be placed in U_1.

Sort the requests in U_i so that their starting stations are in increasing order, and consider them one-by-one in this order. For each request $r = [s, t] \in U_2$, let $r_1 = [s_1, t_1]$ denote the request, in S, for the first interval which prevents accommodating r on seat 2. Then seat 1 must be empty from station s_1 to station $\min\{t, t_1\}$. By the First-Fit rule, we have $t < t_1$, since otherwise request r_1 would be accommodated on seat 1. For the same reason, there should be some request $r_2 = [s_2, t_2] \in S$ which is accommodated on seat 1 and $t \leq s_2 < t_1$. We claim that there is no request $r' = [s', t'] \in U_1$ whose starting station s' is in the gap right before $[s_2, t_2]$. Otherwise, we would have $t' \leq s_1$, which ensures that request r' could be accommodated on seat 1. Conceptually, we assign requests r_1 and r_2 in S to request r. Notice that for different $r \in U_2$, the requests r_1 and r_2 in S are different.

After finishing the requests in U_2, we consider the requests in U_1. For each request $r = [s, t] \in U_1$, let $r_1 = [s_1, t_1]$ denote the request, in S, for the first interval which prevents accommodating r on seat 1. Then seat 2 must be empty from station s_1 to station $\min\{t, t_1\}$. Let $r_2 = [s_2, t_2]$ denote the last request that is accommodated on seat 2 before s_1. That is, seat 2 is empty from station t_2 to station $\min\{t, t_1\}$. Obviously, $t_2 \leq s_1$. Furthermore, there is no request $r' = [s', t'] \in U_2$ such that $t_2 \leq s' < s_1$. If there is no request $q \in U_2$ with its starting station in the gap (on seat 2) before s_2, or there is no such gap at all, then we assign requests r_1 and r_2 to r. In the case where there is a gap and there is some request $q = [u, v] \in U_2$ with starting station u in this gap, let q_1 and q_2 denote the two requests in S that were assigned to q. We then reassign requests r_1, q_1 and q_2 to requests r and q. Notice that for different r, the corresponding q must be different, and the same requests in S cannot be assigned to different requests from U. Thus, depending on which case we are dealing with, either two requests in S are assigned to one in U, or a group of three requests in S is assigned to a pair of requests in U. So the size of U is at most $\frac{2}{3}$ the size of S, which means that First-Fit accommodates at least three-fifths of the requests. □

4 Randomized Algorithms

In this section, we examine the competitive ratios on accommodating sequences for randomized fair on-line algorithms for the unit price problem, by comparing them with an oblivious adversary. Some results concerning randomized fair on-line algorithms for the proportional price problem can be found in [3].

Though the following theorem is about deterministic algorithms, and the result is worse than Theorem 3.1, it is included in this section because the structure of the proof allows for an easy transformation to a proof for the equivalent randomized problem. We believe it is easier to first understand the deterministic proof, and then verify the transformation in the subsequent corollary.

Theorem 4.1 Let

$$f(k) = \begin{cases} \dfrac{7k - 7(k \bmod 6) + 6}{9k - 9(k \bmod 6)}, & \text{when } (k \bmod 6) \in \{0, 1, 2\}; \\[2ex] \dfrac{14k - 14(k \bmod 6) - 15}{18k - 18(k \bmod 6) - 27}, & \text{when } (k \bmod 6) \in \{3, 4, 5\}. \end{cases}$$

Any deterministic fair on-line algorithm for the unit price problem ($k \geq 9$) is at most $f(k)$-competitive, even on accommodating sequences.

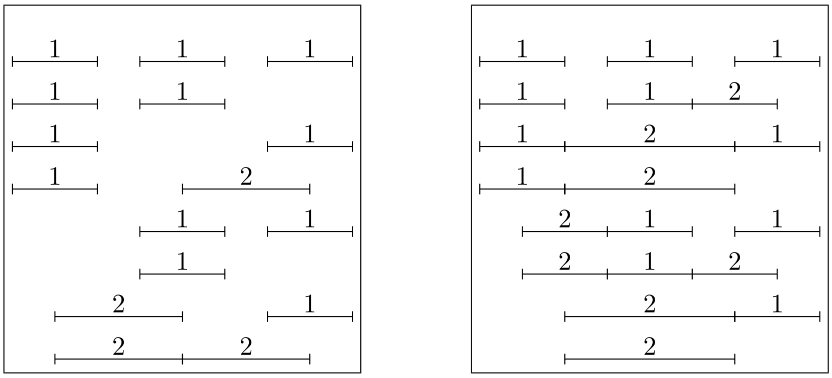
Proof The proof of this theorem is an adversary argument, which is a more dextrous design based on the idea in the proof of Theorem 1.2 in [6]. Assume that n is divisible by 2. The adversary begins the request sequence I with $\frac{n}{2}$ requests for the intervals $[3s+1, 3s+3]$ for $s = 0, 1, \cdots, \lfloor \frac{k-3}{3} \rfloor$. Any fair on-line algorithm $\mathcal{A}$ is able to accommodate this set of $\lfloor \frac{k}{3} \rfloor \cdot \frac{n}{2}$ requests. Suppose that after these requests are accommodated, there are q_i seats which contain both interval $[3i+1, 3i+3]$ and interval $[3i+4, 3i+6]$, $i = 0, 1, \cdots, \lfloor \frac{k-6}{3} \rfloor$. Then there are exactly q_i seats which are empty from station $3i+2$ to station $3i+5$.

In the following, rather than considering each q_i at a time (as in [6]), we consider q_{2i}, q_{2i+1} together for $i = 0, 1, \cdots, \lfloor \frac{k-9}{6} \rfloor$. Let $p_i = q_{2i} + q_{2i+1} (\leq n)$. We distinguish between two cases:

- Case 1: $p_i \leq \frac{5n}{9}$; and
- Case 2: $p_i > \frac{5n}{9}$.

In the first case $p_i \le \frac{5n}{9}$, the adversary proceeds with $\frac{n}{2}$ requests for the interval $[6i+2, 6i+5]$ and $\frac{n}{2}$ requests for the interval $[6i+5, 6i+8]$. For these n additional requests, $\mathcal{A}$ can accommodate exactly p_i of them. Fig. 2a shows this configuration. The intervals marked with a "1" are the intervals which are given first, i.e., before deciding on Case 1 or 2. The intervals marked with a "2" are the ones given afterwards. Thus, for those $2n$ requests whose starting station $s \in [6i+1, 6i+6]$, $\mathcal{A}$ accommodates $n + p_i$ of them.

In the second case $p_i > \frac{5n}{9}$, the adversary proceeds with $\frac{n}{2}$ requests for the interval $[6i+3, 6i+7]$, followed by $\frac{n}{2}$ requests for interval $[6i+2, 6i+4]$ and $\frac{n}{2}$ requests for the interval $[6i+6, 6i+8]$. For these $\frac{3n}{2}$ additional requests, $\mathcal{A}$ can accommodate exactly $\frac{3n}{2} - p_i$ of them. Fig. 2b shows this configuration. Thus, of the $\frac{5n}{2}$ requests whose starting station $s \in [6i+1, 6i+6]$, $\mathcal{A}$ accommodates $\frac{5n}{2} - p_i$ of them.



a) $p_i \le \frac{5n}{9}$. b) $p_i > \frac{5n}{9}$.

Figure 2: Example configurations for the two cases.

In this way, the requests are partitioned into $\lfloor \frac{k-3}{6} \rfloor + 1$ groups; each of the first $\lfloor \frac{k-3}{6} \rfloor$ groups consists of either $2n$ or $\frac{5n}{2}$ requests and the last group consists of either n (if $(k \bmod 6) \in \{0,1,2\}$) or $\frac{n}{2}$ (if $(k \bmod 6) \in \{3,4,5\}$) requests. For each of the first $\lfloor \frac{k-3}{6} \rfloor$ groups, $\mathcal{A}$ can accommodate up to a fraction $\frac{7}{9}$ of the requests therein. This leads to the theorem. More precisely, let S denote the set of indices for which the first case happens, and let $\bar{S}$ denote the set of indices for which the second case happens. When $(k \bmod 6) \in \{0,1,2\}$, the ratio of the number of requests accepted by $\mathcal{A}$ to the number of requests accepted by an optimal off-line algorithm is

$$
\begin{aligned}
\frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} &\leq \frac{n + \sum_{i\in S}(n+p_i) + \sum_{i\in \bar{S}}(\frac{5n}{2} - p_i)}{n + \sum_{i\in S} 2n + \sum_{i\in \bar{S}} \frac{5n}{2}} \\
&\leq \frac{n + \sum_{i\in S} \frac{14n}{9} + \sum_{i\in \bar{S}} \frac{35n}{18}}{n + \sum_{i\in S} 2n + \sum_{i\in \bar{S}} \frac{5n}{2}} = \frac{1 + \sum_{i\in S} \frac{14}{9} + \sum_{i\in \bar{S}} \frac{35}{18}}{1 + \sum_{i\in S} 2 + \sum_{i\in \bar{S}} \frac{5}{2}} \\
&\leq \frac{1 + \frac{14}{9} \cdot \frac{k-3-(k \bmod 6)}{6}}{1 + 2 \cdot \frac{k-3-(k \bmod 6)}{6}} = \frac{7k - 7(k \bmod 6) + 6}{9k - 9(k \bmod 6)},
\end{aligned}
$$

where the last inequality holds because in general $\frac{a}{b} = \frac{c}{d} < 1$ and $a < c$ imply that $\frac{e+ax+cy}{e+bx+dy} \leq \frac{e+a(x+y)}{e+b(x+y)}$. When $(k \bmod 6) \in \{3,4,5\}$, the ratio is

$$
\begin{aligned}
\frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} &\leq \frac{\frac{n}{2} + \sum_{i\in S}(n+p_i) + \sum_{i\in \bar{S}}(\frac{5n}{2} - p_i)}{\frac{n}{2} + \sum_{i\in S} 2n + \sum_{i\in \bar{S}} \frac{5n}{2}} \\
&\leq \frac{\frac{n}{2} + \sum_{i\in S} \frac{14n}{9} + \sum_{i\in \bar{S}} \frac{35n}{18}}{\frac{n}{2} + \sum_{i\in S} 2n + \sum_{i\in \bar{S}} \frac{5n}{2}} = \frac{\frac{1}{2} + \sum_{i\in S} \frac{14}{9} + \sum_{i\in \bar{S}} \frac{35}{18}}{\frac{1}{2} + \sum_{i\in S} 2 + \sum_{i\in \bar{S}} \frac{5}{2}} \\
&\leq \frac{\frac{1}{2} + \frac{14}{9} \cdot \frac{k-3-(k \bmod 6)}{6}}{\frac{1}{2} + 2 \cdot \frac{k-3-(k \bmod 6)}{6}} = \frac{14k - 14(k-3-\bmod 6) - 15}{18k - 18(k \bmod 6) - 27}.
\end{aligned}
$$

This completes the proof. □

Corollary 4.1 Let

$$
f(k) = \begin{cases} \frac{7k - 7(k \bmod 6) - 15}{9k - 9(k \bmod 6) - 27}, & \text{when } (k \bmod 6) \in \{0,1,2\}; \\ \frac{14k - 14(k \bmod 6) + 27}{18k - 18(k \bmod 6) + 27}, & \text{when } (k \bmod 6) \in \{3,4,5\}. \end{cases}
$$

Any randomized fair on-line algorithm for the unit price problem ($k \geq 9$) is at most $f(k)$-competitive, even on accommodating sequences.

Proof The oblivious adversary behaves similarly to the adversary in the proof of Theorem 4.1. Instead of using the values $p_i = q_{2i} + q_{2i+1}$, which are defined in the proof of Theorem 4.1, the adversary uses the expected values of p_i to define the request sequence. Note that an oblivious adversary does not know the actual actions of the on-line algorithm. However, it does know the probabilities for the possible actions. Thus, it can compute the expected values of p_i. The oblivious adversary starts with the same sequence as the adversary in the proof of Theorem 4.1. Then, for each $i = 0, 1, \cdots, \lfloor \frac{k-9}{6} \rfloor$, it decides on Case 1 or Case 2, depending on the expected values $E[p_i]$ compared with $\frac{5n}{9}$. By generating corresponding requests, the linearity of expectations implies that the expected number of requests accommodated by the randomized algorithm is at most a fraction $f(k)$ of the total number of requests. □

Although it is straightforward to show that Theorem 4.1 holds for randomized algorithms, too, as shown above, one cannot use the same argument and show the same for the other theorems.

The most obvious randomized algorithm to consider for this problem is the one we call Random. When Random receives a new request and there exists at least one seat that interval could be placed on, Random chooses randomly among the seats which are possible, giving all possible seats equal probability. The next two theorems characterize Random's competitive ratio on accommodating sequences for $n = 2$, and show that in this case Random is nearly optimal.

Theorem 4.2 For $n = 2$, Random for the unit price problem is at least $\frac{3}{4}$-competitive on accommodating sequences.

Proof Given a request sequence S which could be accommodated with two seats, consider any optimal placement of the requests in S on two seats. Based on where they appear in this placement, we now refer to requests as Seat 1 and Seat 2 requests or intervals.

Based on the Seat 1 requests, we partition the requests into consecutive groups. We show for each group that, in an amortized sense, the expected number of requests accepted in that group is at least $\frac{3}{4}$.

The naming of the two seats is clearly arbitrary; we use the following numbering: Seat 2 is the seat containing the interval with smallest start station number. If there are two intervals with that same start station, then Seat 2 is the seat containing the longer of these two intervals. (If the two intervals are identical, they both will be accepted, so they can be ignored and the next intervals can be used instead.) The other seat is Seat 1.

In general, a group is defined as depicted in Fig. 3. It starts with a Seat 1 request I. If no Seat 1 request K overlaps I and extends beyond it to the right, then the group only includes I and some number x of Seat 1 intervals, which are contained in the interval I. The next group will be defined by considering requests that start no earlier than the end station of I and beginning as with the first group, possibly renaming the seats. This case gives no problem, so assume that this request K exists.

All of the requests which are subintervals of either I or K are included in this group, as are I and K. In the following, we assume that there are $x \geq 0$ subintervals of I and $y \geq 0$ subintervals of K in the request sequence. Thus, the entire group consists of $2+x+y$ requests, all of which are accepted by the optimal off-line algorithm. It may be the case that there is a Seat 1 request from the previous group overlapping I. Call that request L. Similarly, the interval K may overlap a Seat 2 request from the next group (the I interval from the next group), which we call J here.

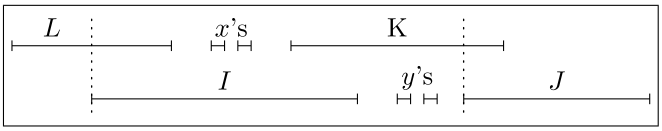


Figure 3: A group picture.

The proof is a lengthy case analysis based on where the relevant intervals occur in the request sequence. Since the $x + y$ intervals contained within I and K are always accepted, the ratio can only become worse if they are assumed to come before I and K, so we make that assumption. Similarly, we assume that if L comes before I, then L is accepted, and if J comes before K, then J is accepted. If L or J do not exist, the group is handled as if they came after I or K. The case analysis is done in the tables of Figures 2 and 3. The notation "I_1,I_2" indicates that I_1 occurs before I_2 in the request sequence. The notation "x's" indicates that $x > 0$ and "y's" indicates that $y > 0$. A mark, $\times$, in the table indicates that the given predicate is true; otherwise it is false. There are thirty-two cases. For each one, the probability that I is accepted, Prob(I), and the probability that K is accepted, Prob(K), are calculated, and a result, Result, is given. For the first of the two intervals I and K which is given, the probability of it being accepted is the probability that no two intervals which come before it and overlap it are placed on different seats. Since these other intervals are equally likely to be on Seat 1 as Seat 2, this probability is $\left(\frac{1}{2}\right)^{u-1}$, where u is the number of interfering intervals. The probability of acceptance of the second of I and K is calculated similarly, but by weighting the two possible cases of whether or not the first interval is accepted by the probability that it is accepted. The "Result", which is calculated as $\frac{\text{Prob}(I)+\text{Prob}(K)+x+y}{2+x+y}$, is the expected fraction of the intervals in that group which are accepted. All of the results are calculated using those values of x and y which give the minimum result (see below for how this minimum is defined). In most cases, setting them equal to 1 gives the minimum. The exceptions are cases 9 and 13 where $x = 1$ and $y = 2$; 17 and 21 where $x = 2$ and $y = 1$; 10, 18, 22, and 30 where $x = 2$; 11, 15, 23 and 27 where $y = 2$; 25 and 29 where $x = 2$ and $y = 2$; 26 where $x = 3$; and 31 where $y = 3$.

The amortization is used to handle the problem that the worst case occurs when both L and J occur before I and K, but this cannot happen for two consecutive groups. If, for example, a group of type 27 occurs immediately before a group of type 1, the extra expectation of $\frac{1}{4}$ from case 27 can be used to (more than) cover the deficit of $\frac{1}{8}$ from case 1, so that overall the expectation is high enough.

Call the groups that fall into the first eight cases (in Table 2) *late groups*, since their I and K intervals occur later in the request sequence than the relevant intervals from the two surrounding groups. Similarly, call the groups that fall into the last eight cases (in Table 3) *early groups*, those that fall into cases 9 through 16 *late-early groups*, and those that fall into cases 17 through 24 *early-late groups*. In the "Result" column of Tables 2 and 3, fractions which are less than $\frac{3}{4}$ are expressed as $\frac{\frac{3}{4}w-z}{w}$, where $w = 2 + x + y$ is the number of intervals in the group. We refer to the value z as the deficit for the group. Notice that there is never a deficit greater than $\frac{1}{8}$ of an interval, and these only occur for late groups. For the early groups, the "Result" is expressed as $\frac{\frac{3}{4}w+z}{w}$, and here the value z is the surplus for the group. All of the early groups have a surplus of at least $\frac{1}{4}$, which more than covers the deficit of any late group. (Note that when the minimums were calculated, they were calculated to maximize the deficit or minimize the surplus, rather than to minimize the expected fraction accepted. It is only groups 25 and 29 where this makes a difference.) Clearly, the first group defined has no request L, so it is either an early group or an early-late group. Although one cannot assume that each late group has an early group immediately preceding it, it is easy to see that for each late group there must be some some early group before it in the request sequence. The surplus of this early group can more than cover the deficit of the late group. All of the early-late and late-early groups are such that the expected fraction of the intervals in those groups accepted is at least $\frac{3}{4}$, so the total expected fraction of intervals accepted is at least $\frac{3}{4}$. □

No.	L,I	J,K	I,K	x's	y's	Prob(I)	Prob(K)	Result
1	×	×	×	×	×	$\frac{1}{2}^{x}$	$\frac{1}{2}^{y} - \frac{1}{2}^{x+y+1}$	$\frac{3-\frac{1}{8}}{4}$
2	×	×	×	×		$\frac{1}{2}^{x}$	$1 - \frac{1}{2}^{x+1}$	$\frac{3}{4}$
3	×	×	×		×	1	$\frac{1}{2}^{y+1}$	$\frac{3}{4}$
4	×	×	×			1	$\frac{1}{2}$	$\frac{3}{4}$
5	×	×		×	×	$\frac{1}{2}^{x} - \frac{1}{2}^{x+y+1}$	$\frac{1}{2}^{y}$	$\frac{3-\frac{1}{8}}{4}$
6	×	×		×		$\frac{1}{2}^{x+1}$	1	$\frac{3}{4}$
7	×	×			×	$1 - \frac{1}{2}^{y+1}$	$\frac{1}{2}^{y}$	$\frac{3}{4}$
8	×	×				$\frac{1}{2}$	1	$\frac{3}{4}$
9	×		×	×	×	$\frac{1}{2}^{x}$	$\frac{1}{2}^{y-1} - \frac{1}{2}^{x+y}$	$\frac{31}{40}$
10	×		×	×		$\frac{1}{2}^{x}$	1	$\frac{13}{16}$
11	×		×		×	1	$\frac{1}{2}^{y}$	$\frac{13}{16}$
12	×		×			1	1	1
13	×			×	×	$\frac{1}{2}^{x} - \frac{1}{2}^{x+y}$	$\frac{1}{2}^{y-1}$	$\frac{31}{40}$
14	×			×		$\frac{1}{2}^{x+1}$	1	$\frac{3}{4}$
15	×				×	$1 - \frac{1}{2}^{y}$	$\frac{1}{2}^{y-1}$	$\frac{13}{16}$
16	×					$\frac{1}{2}$	1	$\frac{3}{4}$

Table 2: Different groups (1–16).

This value of $\frac{3}{4}$ is, in fact, a very tight lower bound on Random's competitive ratio on accommodating sequences when there are $n = 2$ seats.

Theorem 4.3 Let

$$f(k) = \frac{3}{4} + \frac{1}{4(k - ((k-1) \bmod 2))}.$$

For $n = 2$, Random for the unit price problem ($k \geq 3$) is at most $f(k)$-competitive on accommodating sequences.

Proof The adversary begins the request sequence I with the interval $[1, 2]$ followed by the intervals $[2i, 2i + 2]$ for $i \in \{1, \ldots, \lfloor \frac{k-2}{2} \rfloor\}$. If k is odd, it then gives the request $[k - 1, k]$.

No.	L,I	J,K	I,K	x's	y's	Prob(I)	Prob(K)	Result
17		$\times$	$\times$	$\times$	$\times$	$\frac{1}{2}^{x-1}$	$\frac{1}{2}^{y} - \frac{1}{2}^{x+y}$	$\frac{31}{40}$
18		$\times$	$\times$	$\times$		$\frac{1}{2}^{x-1}$	$1 - \frac{1}{2}^{x}$	$\frac{13}{16}$
19		$\times$	$\times$		$\times$	1	$\frac{1}{2}^{y+1}$	$\frac{3}{4}$
20		$\times$	$\times$			1	$\frac{1}{2}$	$\frac{3}{4}$
21		$\times$		$\times$	$\times$	$\frac{1}{2}^{x-1} - \frac{1}{2}^{x+y}$	$\frac{1}{2}^{y}$	$\frac{31}{40}$
22		$\times$		$\times$		$\frac{1}{2}^{x}$	1	$\frac{13}{16}$
23		$\times$			$\times$	1	$\frac{1}{2}^{y}$	$\frac{13}{16}$
24		$\times$				1	1	1
25			$\times$	$\times$	$\times$	$\frac{1}{2}^{x-1}$	$\frac{1}{2}^{y-1} - \frac{1}{2}^{x+y-1}$	$\frac{\frac{18}{4}+\frac{3}{8}}{6}$
26			$\times$	$\times$		$\frac{1}{2}^{x-1}$	1	$\frac{\frac{15}{4}+\frac{1}{2}}{5}$
27			$\times$		$\times$	1	$\frac{1}{2}^{y}$	$\frac{3+\frac{1}{4}}{4}$
28			$\times$			1	1	$\frac{\frac{3}{2}+\frac{1}{2}}{2}$
29				$\times$	$\times$	$\frac{1}{2}^{x-1} - \frac{1}{2}^{x+y-1}$	$\frac{1}{2}^{y-1}$	$\frac{\frac{18}{4}+\frac{3}{8}}{6}$
30				$\times$		$\frac{1}{2}^{x}$	1	$\frac{3+\frac{1}{4}}{4}$
31					$\times$	1	$\frac{1}{2}^{y-1}$	$\frac{\frac{15}{4}+\frac{1}{2}}{5}$
32						1	1	$\frac{\frac{3}{2}+\frac{1}{2}}{2}$

Table 3: Different groups (17–32).

Random will place each of these requests, and, since there is no overlap, they are placed on the first seat with probability $\frac{1}{2}$.

Now the adversary continues the sequence with $[2i+1, 2i+3]$ for $i \in \{0, \ldots, \lfloor \frac{k-3}{2} \rfloor\}$.

Each interval in this last part of the sequence overlaps exactly two intervals from earlier and can therefore be accommodated if and only if these two intervals are placed on the same seat. This happens with probability $\frac{1}{2}$.

Thus, all requests from the first part of the sequence, and expected about half of the requests for the last part, are accepted. More precisely we obtain:

$$\frac{\text{Random}(I)}{\text{OPT}(I)} \leq \frac{\frac{k+1-((k-1) \bmod 2)}{2} + \frac{1}{2} \cdot \frac{k-1-((k-1) \bmod 2)}{2}}{k - ((k-1) \bmod 2)} = f(k).$$

□

The competitive ratio of $\frac{3}{4}$ on accommodating sequences for Random with $n = 2$ seats does not extend to more seats. In general, one can show that Random's competitive ratio on accommodating sequences is bounded from above by approximately $\frac{17}{24} = 0.7083\bar{3}$.

Theorem 4.4 Even on accommodating sequences, the competitive ratio for Random is at most $\frac{17k+14}{24k}$, for the unit price problem, when $k \equiv 2 \pmod 4$.

Proof Assume that n is divisible by 3. The request sequence I is as follows:

- $[1, 2]$ — $\frac{n}{3}$ times.
- $[4s+2, 4s+6]$ — $\frac{n}{3}$ times — for $s = 0, 1, ..., \frac{k-6}{4}$.
- $[1, 4]$ — $\frac{n}{3}$ times.
- $[4s, 4s+4]$ — $\frac{n}{3}$ times — for $s = 1, 2, ..., \frac{k-6}{4}$.
- $[k-2, k]$ — $\frac{n}{3}$ times.
- $[2s+1, 2s+3]$ — $\frac{n}{3}$ times — for $s = 0, 1, ..., \frac{k-4}{2}$.
  These will be referred to as the *extra* intervals.

If First-Fit was applied on this sequence, all $\frac{kn}{3}$ requests would be accommodated. Random will accommodate everything except some of the extra

Combinations				Expected
$[4s-2,4s+2]$	$[4s+2,4s+6]$	$[4s+6,4s+10]$	$[4s,4s+4]$	Number
$\times$	$\times$	$\times$		$n/27$
$\times$	$\times$			$2n/27$
$\times$		$\times$		$2n/27$
	$\times$	$\times$		$2n/27$
$\times$				$4n/27$
	$\times$			$4n/27$
		$\times$		$n/27$
		$\times$	$\times$	$n/9$
			$\times$	$2n/9$
				$2n/27$

Table 4: The expected number of seats with various combinations of the intervals.

intervals. In what follows, the intervals of length shorter than 4, which are not extra intervals, will be thought of as if they had length 4 and thus extended before the first station or after the last. Notice that m extra intervals of the form $[4s+1,4s+3]$ will be accepted if and only if exactly m seats which receive the interval $[4s-2,4s+2]$ also receive the interval $[4s+2,4s+6]$. Similarly, m extra intervals of the form $[4s+3,4s+5]$ will be accepted if and only if exactly m seats which receive the interval $[4s,4s+4]$ also receive the interval $[4s+4,4s+8]$. Let us consider the types of extra intervals in pairs $([4s+1,4s+3],[4s+3,4s+5])$ to calculate the expected number which are accommodated by Random. Consider all but the last pair of these extra intervals. The intervals which can interfere with whether or not these extra intervals are accommodated are those of the forms $[4s-2,4s+2]$, $[4s+2,4s+6]$, $[4s,4s+4]$, $[4s+4,4s+8]$. The only other intervals which can affect where any of these are placed, relative to each other, are those of the form $[4s+6,4s+10]$. When the intervals $[4s-2,4s+2]$ are placed, the probability for each one that it will have an interval $[4s+2,4s+6]$ immediately after it is $\frac{1}{3}$. Thus one expects that $\frac{1}{3}$ of the $[4s+1,4s+3]$ intervals will be accepted. The intervals of the form $[4s,4s+4]$ cannot be on the same seat as any $[4s-2,4s+2]$, or $[4s+2,4s+6]$ interval. Table 4 shows the expected number of seats which will be assigned the various combinations of the intervals $[4s-2,4s+2]$, $[4s+2,4s+6]$, $[4s+6,4s+10]$, and $[4s,4s+4]$. An "$\times$" indicates the presence of an interval of that type.

The intervals of the form $[4s+4, 4s+8]$ can only go where there is neither a $[4s+2, 4s+6]$ nor a $[4s+6, 4s+10]$ interval. One can see from Table 4 that the expected number of seats like this is $\frac{4n}{9}$. The expected number of them that have a $[4s, 4s+4]$ interval is $\frac{2n}{9}$, so one expects $\frac{1}{2}$ of the intervals of the form $[4s+3, 4s+5]$ to be accommodated. A similar, but simplified argument gives exactly the same expectations for the last two types of extra intervals. This gives that the expected number of extra intervals accommodated by Random is $\frac{n}{3}(\frac{1}{3} + \frac{1}{2})\frac{k-2}{4} = \frac{5n}{3}\frac{k-2}{24}$. Hence,

$$\frac{E[\text{Random}(I)]}{\text{OPT}(I)} = \frac{\frac{n}{3}(\frac{k+2}{2} + 5\frac{k-2}{24})}{\frac{kn}{3}} = \frac{17k+14}{24k} \quad .$$

□

For other k, not congruent to 2 modulo 4, and other $n \geq 3$, not congruent to 0 modulo 3, similar results hold. Giving first $n \bmod 3$ $[1, k]$ requests, and then using the same sequence of requests as in the previous proof (and thus not using the last stations), gives upper bounds of the form $\frac{17k-c_1}{24k-c_2}$ for constants c_1 and c_2 which depend only on the value of $k \bmod 4$.

5 Concluding Remarks

We have shown that any fair deterministic algorithm for the unit price seat reservation problem has an asymptotic competitive ratio of $\frac{1}{2}$ on accommodating sequences. The most interesting open problem remaining here is whether or not there exists a randomized algorithm which does better. In particular, what is the competitive ratio of the algorithm Random on accommodating sequences? We have shown that it is $\frac{3}{4}$ when $n = 2$, and no more than $\frac{17}{24}$ for $n \geq 3$. However, the best known lower bound on its performance is still $\frac{1}{2}$ for $n \geq 3$.

Acknowledgments

Eric Bach was supported in part by NSF Grant CCR-9510244.

Joan Boyar would like to thank Faith Fich for interesting discussions regarding the seat reservation problem with $n = 2$ seats.

Joan Boyar and Kim S. Larsen carried out part of this work while visiting the Department of Computer Sciences, University of Wisconsin – Madison.

They were supported in part by SNF (Denmark), in part by NSF (U.S.) grant CCR-9510244, in part by the ESPRIT Long Term Research Programme of the EU under project number 20244 (ALCOM-IT).

Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen were supported in part by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

Tao Jiang and Guo-Hui Lin were supported in part by NSERC Research Grant OGP0046613 and a CITO grant.

Tao Jiang was supported in part by a UCR startup grant.

Rob van Stee was supported by the Netherlands Organization for Scientific Research (NWO), project number SION 612-30-002.

References

[1] B. Awerbuch, Y. Bartal, A. Fiat, S. Leonardi and A. Rosén, On-Line Competitive Algorithms for Call Admission in Optical Networks, *Proceedings of the 4th Annual European Symposium on Algorithms*, Lecture Notes in Computer Science 1136: 431–444, Springer-Verlag, 1996.

[2] B. Awerbuch, Y. Bartal, A. Fiat and A. Rosén, Competitive Non-Preemptive Call Control, *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 312–320, ACM Press, 1994.

[3] E. Bach, J. Boyar and K. S. Larsen, The Accommodating Ratio for the Seat Reservation Problem, PP-1997-25, Department of Mathematics and Computer Science, University of Southern Denmark, 1997.

[4] E. Bach, J. Boyar, T. Jiang, K. S. Larsen, G.-H. Lin, Better Bounds on the Accommodating Ratio for the Seat Reservation Problem, *Proceedings of the Sixth Annual International Computing and Combinatorics Conference*, Lecture Notes in Computer Science 1858: 221–231, Springer-Verlag, 2000.

[5] A. Bar-Noy, R. Canetti, S. Kutten, Y. Mansour and B. Schieber, Bandwidth Allocation with Preemption, *Proceedings of the 27th Annual ACM Symposium on Theory of Computing* pp. 616–625, ACM Press, 1995.

[6] J. Boyar and K. S. Larsen, The Seat Reservation Problem, *Algorithmica* 25: 403–417, 1999.

[7] J. Boyar, K. S. Larsen and M. N. Nielsen, The Accommodating Function — a generalization of the competitive ratio, *Proceedings of the Sixth International Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science 1663: 74–79, Springer-Verlag, 1999. Full paper to appear in *SIAM Journal on Computing.*

[8] M. C. Carlisle and E. L. Lloyd, On the k-Coloring of Intervals, *Advances in Computing and Information*, Lecture Notes in Computer Science 497: 90–101, Springer-Verlag, 1991.

[9] J. A. Garay, I. S. Gopal, S. Kutten, Y. Mansour and M. Yung, Efficient On-Line Call Control Algorithms, *Journal of Algorithms* 23: 180–194, 1997.

[10] F. Gavril, Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques, and Maximum Independent Set of a Chordal Graph, *SIAM Journal on Computing* 1: 180–187, 1972.

[11] T. R. Jensen and B. Toft, *Graph Coloring Problems* Wiley, 1995.

[12] H. A. Kierstead and W. T. Trotter, Jr., An Extremal Problem in Recursive Combinatorics, *Congressus Numerantium* 33: 143–153, 1981.

[13] R. J. Lipton and A. Tomkins, On-Line Interval Scheduling, *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 302–311, ACM Press, 1994.

[14] P. Raghavan and E. Upfal, Efficient Routing in All-Optical Networks, *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pp. 134–143, ACM Press, 1994.

[15] L. van Wassenhove, L. Kroon and M. Salomon, Exact and Approximation Algorithms for the Operational Fixed Interval Scheduling Problem, Working paper 92/08/TM, INSEAD, Fontainebleau, France, 1992.