Alcom-FT Technical Report Series
ALCOMFT-TR-02-3

Seat Reservation Allowing Seat Changes

Joan Boyar* Susan Krarup Morten N. Nielsen*

Abstract

We consider a variant of the Seat Reservation Problem [6] in which seat changes are allowed. We analyze the model using the competitive ratio, the competitive ratio on accommodating sequences [6], and the accommodating function [5, 7]. A very promising family of algorithms considered in this paper is Min-Change, which will ask passengers to change seat, only if they would otherwise have been rejected. Min-Change belongs to a large class of *conservative* algorithms, which all have very high performance guarantees. For instance, if the optimal off-line algorithm can seat all of the passengers, $\frac{2}{3}$ of the passengers can be seated on-line using any conservative algorithm allowing only one seat change and $\frac{3}{4}$ will be seated if two seat changes are allowed. This should be compared to the asymptotically hardness result of $\frac{1}{2}$ for the best algorithm when no seat changes are allowed [2]. Another interesting algorithm, Modified-Kierstead-Trotter, is proposed and shown to seat all passengers if the optimal off-line algorithm could have accommodated them with only half as many seats. On this type of sequences, Modified-Kierstead-Trotter is strictly better than Min-Change-First-Fit which is strictly better than the Checkerboard Algorithm.

1 Introduction

In [6] the Seat Reservation Problem was analyzed. In this paper we consider an interesting variant of the Seat Reservation Problem, in which passengers are allowed to change seats during the trip. For this variant much more promising results can be shown. We use competitive analysis; the standard competitive ratio [9, 11, 13, 14], the competitive ratio on accommodating sequences [6] [1], and the accommodating function [5, 7]. Two different pricing policies are considered, the natural *proportional price* policy, and the *unit price* policy which is motivated by the train system in Denmark.

* Supported in part by the Danish Natural Science Research Council (SNF) and in part by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

[1] The *accommodating ratio* was defined in [6]. To follow common practice in the field the name was later changed to the competitive ratio on accommodating sequences.

1.1 Problem Definition

The set-up from [6] is as follows: A train with n seats travels from a start station to an end station, stopping at $k \geq 2$ stations, including the first and last. Reservations can be made for any trip from a station u to a station v. The passenger is given a single seat number when the ticket is purchased, which can be any time before departure. The algorithms (ticket agents) attempt to maximize income, i.e., the sum of the prices of the tickets sold. For political reasons, the problem must be solved in a *fair* manner, i.e., the ticket agent may not refuse a passenger if it is possible to accommodate him when he attempts to make his reservation. All algorithms, even the optimal off-line algorithm, OPT, must be fair.

The version considered in this paper allows the seat reservation system to assign seats so that a passenger may change seats a fixed number of times s during the trip. We use $\mathbb{A}$ to denote a fair on-line algorithm for the Seat Reservation Problem with at most s seat changes and use OPT to denote an optimal algorithm.

Competitive analysis is used to measure the performance quality. Let $\mathbb{A}(I)$ denote the benefit obtained by the algorithm $\mathbb{A}$ when run on I and let $\mathrm{OPT}(I)$ denote the maximal benefit which can be obtained by any algorithm. $\mathbb{A}$ is said to be c-competitive, if $\mathbb{A}(I) \geq c \cdot \mathrm{OPT}(I)$ for all input sequences I. The competitive ratio is the supremum of all c for which the algorithm is c-competitive.

A sequence in which all requests can be accepted within n seats is called an *accommodating sequence*. In this paper we are particularly interested in the competitive ratio when the input sequences are restricted to accommodating sequences.

1.2 Summary of Results

For the unit price problem on accommodating sequences a performance ratio of $\frac{1}{2}$ was shown to hold for the most natural deterministic algorithms in [6]. Later [2] (a preliminary version of the paper appeared as [3]) it was shown that $\frac{1}{2}$ is asymptotically best possible for all deterministic algorithms if $k >> n$. Also for our problem, there are algorithms obtaining a competitive ratio of $\frac{1}{2}$ on accommodating sequences, but we can show that a competitive ratio of $\frac{s+1}{s+2}$ on accommodating sequences can be promised for a very natural class of algorithms, where s denotes the number of seat changes allowed. This result can be compared directly with the ratio of $\frac{1}{2}$ from [2, 6], since for these two problems, the optimal algorithm has exactly the same power when considering accommodating sequences.

The competitive ratio on accommodating sequences has been extended to the accommodating function, which has been studied in a number of papers [4, 5, 7] (a preliminary version of the first citation appeared as [1]). This function was used to distinguish different algorithms having the same competitive ratio and competitive ratio on accommodating sequences. These papers give reason to believe that, other than the competitive ratio, accommodating sequences are the most important special case of the accommodating function. In particular, for a variant of Bin Packing a general theorem extending results obtained on accommodating sequences to results on the accommodating function is given [4].

For the unit price problem, we show the following results:

	Accommodating Sequences	All Sequences Competitive Ratio
General	$\geq 1/2$	$O(s/k)$, for $n = 2$
Conservative	$\geq (s+1)/(s+2)$	$\Theta(1/k)$ OPT general $O(s^2/k)$ OPT conservative
First-Fit*	$= \Theta((s+1)/(s+2))$	$\Theta(1/k)$ OPT general $O(s/k)$ OPT conservative
Min-Change-First-Fit	$\Theta((s+1)/(s+2))$	$\Theta(1/k)$ OPT general $O(s/k)$ OPT conservative

In addition, if one seat change is allowed, the algorithm, Modified-Kierstead-Trotter, is proposed and shown to seat all passengers if the optimal off-line algorithm could have accommodated them with only half as many seats. For such sequences we show that Modified-Kierstead-Trotter is strictly better than Min-Change-First-Fit which is strictly better than the Checkerboard Algorithm. The algorithms First-Fit* [2], Min-Change, Min-Change-First-Fit, the Checkerboard Algorithm and Modified-Kierstead-Trotter are defined later.

For the proportional price problem we prove a tight hardness result of $\Omega(\frac{s}{k})$ for all algorithms matched by the behavior of Min-Change. For First-Fit* it was not possible to find the exact bound, but we prove a hardness result of $O(\frac{s^2}{k})$ which is close to the general performance guarantee.

1.3 Definitions and Algorithms

The requests considered are intervals. Two (sub)intervals u and v *overlap*, if $u \cap v \neq \emptyset$. Note that our intervals are half-open, i.e., we do *not* consider it an overlap if one interval starts at the station where the other interval ends. Since we allow the algorithms to perform s seat changes, the intervals might be broken into (at most) $s+1$ subintervals. For any set S, we use $|S|$ to denote the number of elements in S. For a start station t, let $d(t, t+1)$ denote the number of intervals containing $[t, t+1)$. For a sequence I we define the *density* of the sequence to be $d(I) = \max_{1 \leq t \leq k-1} d(t, t+1)$.

The original Seat Reservation Problem is similar to the problem of coloring an interval graph on-line: Each interval is a node in the graph. If two intervals overlap, there is an edge between them. Interval graphs are *perfect* [10], so the size of the largest clique is exactly the number of colors needed. Thus, when there is no pair of stations $(t, t+1)$ such that the number of people who want to be on the train between stations t and $t+1$ is greater than n, the optimal

[2] We use First-Fit to denote the First-Fit algorithm for the original problem without seat changes. It is used here in some cases to describe the optimal algorithm.

off-line algorithm will be able to accommodate all intervals. We use this result on perfect graphs extensively. In order to define various fair algorithms for this problem, it is useful to define subintervals of the interval u which the algorithm may be trying to seat, relative to v, an interval which has already been seated.

Definition 1 For two (sub)intervals u and v, we define $\text{left}(u, v)$ to be the maximal subinterval of u which contains the leftmost endpoint of u and which can be seated on the same seat as v without overlapping v. Moreover, if u and v have the same start station, we let $u \setminus v$ denote the possible empty subinterval ranging from the end station of v to the end station of u. Otherwise $u \setminus v$ is undefined. □

It is natural to try to extend definitions of algorithms, for the original seat reservation problem, to the problem where seat changes are allowed. The following is one such possible extension of First-Fit; another is defined later.

Definition 2 Suppose a (sub)interval $u = [x, y)$ must be seated using at most t seat changes. *First-Fit** will consider the seats in numerical order, searching for the first which is free from x to $x+1$. Suppose seat i is found and v is the first (leftmost) overlapping (sub)interval for u on seat i, if such a (sub)interval exists. Then First-Fit* will seat $u' = \text{left}(u, v)$ on seat i and recursively seat $u \setminus u'$ (again starting from seat 1) using at most $t-1$ seat changes. If there is no (sub)interval v overlapping u on seat i, First-Fit* will seat all of u on seat i. First-Fit* must be fair. Thus, if $u \setminus u'$ cannot be seated using at most $t-1$ seat changes, First-Fit* continues its search for a seat which is free from x to $x+1$. □

Having decided to seat a (sub)interval on seat i, First-Fit* will seat as much as possible of the remaining journey on seat i before changing to another seat. This is a very natural property, which turns out to be a key in proving performance guarantees for First-Fit* and other algorithms.

Definition 3 Suppose an algorithm accepts a new interval u by seating $l \leq s+1$ subintervals. If the first $l-1$ subintervals are placed such that an already seated (sub)interval starts right after the subinterval ends, we call the algorithm *conservative*. □

First-Fit* is an example of a conservative algorithm, as is each algorithm in the following class:

Definition 4 For any algorithm $\mathbb{B}$ for the original seat reservation problem, the algorithm *Min-Change-*$\mathbb{B}$ places a (sub)interval $u = [x, y)$, which must be seated using at most $t \geq 0$ seat changes as follows: If the smallest number of seat changes possible is $j \leq t$, then Min-Change-$\mathbb{B}$ will find the largest $y' \leq y$ such that $[x, y')$ can be placed with no seat changes and $[y', y)$ with at most $j-1$ seat changes. Then, Min-Change-$\mathbb{B}$ places $[x, y')$ using $\mathbb{B}$ and places $[y', y)$ recursively. □

Note that the value y', which Min-Change-$\mathbb{B}$ finds, must equal y if $j = 0$, so for intervals which can be seated without splitting, the algorithm $\mathbb{B}$ is used. When the algorithm $\mathbb{B}$ can be arbitrary, we will simply refer to Min-Change-$\mathbb{B}$ as Min-Change.

Any variation of Min-Change will be conservative. It is natural to ask, if this is a restriction on the Min-Change strategy. It turns out not to be the case according to the following note (which is easily verified):

Note 1 If an interval can be seated using t seat changes, then there is a conservative way of seating the interval using at most t seat changes. □

The two algorithms defined below are mainly used to prove hardness results. The first, Next-Fit, is conservative; the second Max-Change-Next-Fit is not. Both algorithms are fair.

Definition 5 Next-Fit behaves as follows: Having seated the last (sub)interval on seat i, the next (sub)interval $[x, y)$ is seated on the first seat j in the sequence $\langle i+1, i+2, \ldots, n, 1, 2, \ldots, i\rangle$, which is free from x to $x + 1$. If not all of the subinterval fits because it is blocked by an interval v, only $\mathrm{left}(u, v)$ is placed there, and this process is repeated, starting at seat $j + 1$, for the subinterval $u \setminus \mathrm{left}(u, v)$. If $u \setminus \mathrm{left}(u, v)$ cannot be seated with the required number of seat changes, the search for a seat for the initial subinterval is continued. □

Definition 6 Max-Change-Next-Fit breaks an interval into as many subintervals as possible and seats each subinterval in turn as Next-Fit would have done it. □

The algorithm above is well-defined if there is no choice as to how to break the intervals into smaller pieces. This property is satisfied for the sequences considered here, since each interval has size at most $s + 1$.

Clearly, unused space on seats can lead to the on-line algorithm rejecting intervals which OPT could have accepted.

Definition 7 We use the term *hole* to denote the empty space on a seat – a connected interval where no interval has been placed. The size of a hole is the number of consecutive trips of unit size it spans. □

2 A Worst Case Sequence

Later in the paper, we will use the sequences W and W' below as a first part of different worst case sequences.

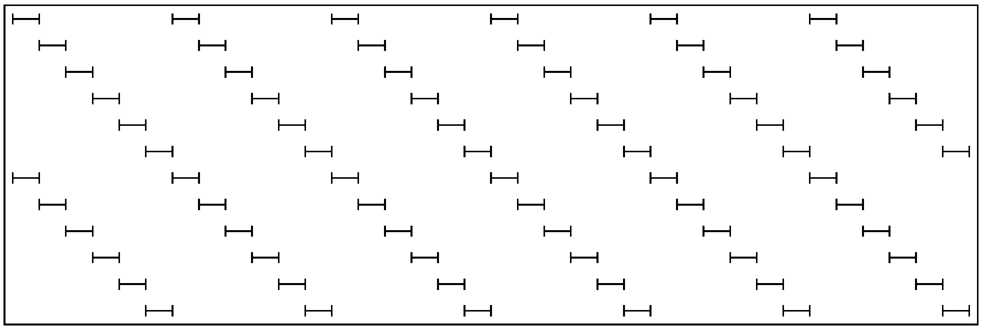


Figure 1: Placement of Next-Fit on W for $k = 37$, $n = 12$, and $s = 5$.

Assume $s+1$ divides $k-1$ and n and that $k-1$ divides $n(s+1)$. Let $r = \frac{k-1}{s+1}$. Let W denote the sequence

- For $i = 0, \ldots, s$ and for $j = 1 \ldots n$, let $p = ((ir + j - 1) \mod k) + 1$ and give $[p, p+1)$.

Under the same assumptions, let W' denote the sequence (having the same density as W on every seat):

- For $i = 0, \ldots, s$ and for $j = 0 \ldots \frac{n}{s+1} - 1$ let $p = ((ir + j(s+1)) \mod k) + 1$ and give $[p, p+s+1)$.

It is easy to verify the following:

Lemma 1 The following hold about the sequences W and W':

- $|W| = n(s+1)$.
- $|W'| = n$.
- After Next-Fit has been run on W every hole has size at most $\frac{k-1}{s+1} - 1$. (See Figure 1.)
- The density of W and W' is $d = \frac{n(s+1)}{k-1}$ on every seat.
- When Max-Change-Next-Fit seats W' it will end up in a configuration identical to the one Next-Fit has on W. (See Figure 1.)

3 Unit Price

Here we consider the unit price problem, so the goal is to maximize the number of intervals accepted.

3.1 Accommodating Sequences

In this subsection we prove upper and lower bounds (hardness results and performance guarantees, respectively) on the competitive ratio on accommodating sequences for different classes of algorithms for the unit price problem.

3.1.1 The Fairness Restriction is not Enough

It is tempting to believe that the fairness restriction is sufficient to ensure that an algorithm which is allowed s seat changes will perform better than an algorithm which is not. This is not the case. Recall that any fair algorithm for the original seat reservation problem has a performance guarantee of $\frac{1}{2}$ on accommodating sequences.

Theorem 1 There exists an algorithm for the unit price problem with a competitive ratio on accommodating sequences of at most $\frac{1}{2-\frac{2(s+1)}{k-1}}$.

Proof Assume n is even and $2(s+1)$ divides $k-1$. Given

- $\frac{n}{2}$ of each of the following intervals, for $i = 0, \ldots, \frac{k-1}{2(s+1)} - 1$, let $p = 2i(s+1)+1$: $[p, p+2(s+1))$,

there exists a deterministic algorithm which would split these intervals into $s+1$ intervals each of size 2, and seat them as shown in Figure 2. Then the adversary can give $\frac{n}{2}$ times the following:

- for $i = 0, \ldots, \frac{k-1}{2(s+1)} - 2$, let $p = 2i(s+1)+2$ and give $[p, p+2(s+1))$.

None of these intervals can be accepted by the algorithm using only s seat changes. OPT will behave as First-Fit and accept all intervals. The performance ratio is $\frac{\frac{n(k-1)}{4(s+1)}}{\frac{n(k-1)}{4(s+1)}+\frac{n}{2}(\frac{k-1}{2(s+1)}-1)} = \frac{k-1}{k-1+(k-1)-2(s+1)} = \frac{1}{2-\frac{2(s+1)}{k-1}}$. □

3.1.2 General Performance Guarantees

In order to prove a lower bound (positive result) on different classes of algorithms, we prove a lower bound on the number of intervals an on-line algorithm $\mathbb{A}$ accepts and an upper bound on how many intervals it rejects. First, we group the (sub)intervals $\mathbb{A}$ accepts into components.

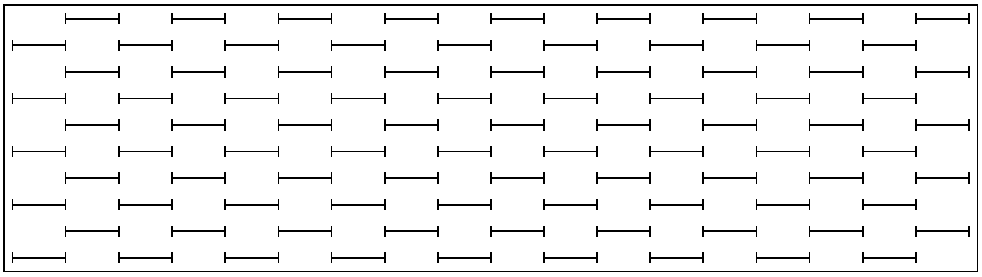


Figure 2: "Checkerboard" for $k = 37$, $n = 10$, and $s = 2$.

Considering a seating performed by an algorithm, $\mathbb{A}$, we will merge two (sub)intervals, if one starts right after the other ends. If this kind of merge is performed repeatedly seat by seat, we end up having larger (not connected) *components*. We use γ to denote the number of such components.

Theorem 2 Let $\mathbb{A}$ denote an algorithm for placing and possibly splitting intervals. If A denotes the set of intervals accepted by $\mathbb{A}$ and γ denotes the number of components in $\mathbb{A}$'s seating, then

1. $\gamma \leq (s+1)|A|$.
2. if $\mathbb{A}$ is a conservative algorithm, $\gamma \leq |A|$.

Proof To prove (1) it is enough to note that every interval accepted by $\mathbb{A}$ can be broken into at most $s+1$ subintervals (components), since at most s seat changes are allowed. To prove (2): When a new interval is considered, we mark the entire interval, if no seat changes are performed. If $l > 0$ seat changes are performed, we mark the last subinterval, while all other intervals are left unmarked. The number of marks is exactly $|A|$. We prove by induction on the number of elements accepted that every component contains at least one mark. This is clearly true before any interval has been given. Suppose $m \geq 0$ intervals have been accepted, and every component contains a marked (sub)interval. Consider the next interval to be accepted. If this interval is not broken into subinterval, it is marked and the claim is true. If the interval is broken into $l+1$ subintervals, the first l subintervals must be connected to an existing component, by the definition of a conservative algorithm, and the last subinterval, which might create a new component, is marked. Again the claim is true. □

Theorem 3 Consider a sequence in which all intervals could be accepted by OPT. If U denotes the set of intervals rejected by some algorithm $\mathbb{A}$, and γ denotes the number of components in $\mathbb{A}$'s seating, then $(s+1)|U| \leq \gamma$.

Proof Let $\mathbb{A}$ denote an arbitrary algorithm for the unit price problem. Let I denote a worst case sequence for $\mathbb{A}$ which can be accepted by an optimal algorithm on n seats. Define S to be the set of components in $\mathbb{A}$'s seating, then $\gamma = |S|$.

We will define a partial function $f : S \to U$ having the following properties:

- $|f^{-1}(u)| = s + 1$ for all $u \in U$.
- For all $u_1, u_2 \in U$ where $u_1 \neq u_2 : f^{-1}(u_1) \cap f^{-1}(u_2) = \emptyset$.

To each element in U, f assigns $s + 1$ different components from S, with no component in S used twice. This means that $\gamma = |S| \geq (s + 1)|U|$, proving the theorem.

During the construction of f, we maintain two sets U' and S'. Initially, let $S' = S$ and let $U' = \{(u, s) \mid u \in U\}$. The two sets should be interpreted as follows:

- An element $(v, t) \in U'$ represents a (sub)interval v which is allowed to change seat t times.
- S' will at any point represent a legal seating.

The (sub)intervals will be broken into smaller pieces, and the counter t will be decreased during the construction. Furthermore we will at any point maintain the following invariants:

1. No element (u, t) from U' can be accepted (added to S') even if t seat changes are allowed.
2. $d(U' \cup S') \leq n$.

Note that these invariants hold initially. Order the elements $(v, t) \in U'$ according to the left endpoint of v. This ordering will be maintained when new subintervals are added to U'. For the first element (v, t) in this ordering, let p denote the start station of the (sub)interval represented by (v, t). Since $d(U' \cup S') \leq n$, there is at least one free seat in the seating represented by S' from station p to station $p + 1$. By the first invariant, there must be a first (leftmost) element $s' \in S'$ on this seat blocking the (sub)interval. Define $f(s') = v$, let $v' = \text{left}(v, s')$, and let $v'' = v \setminus v'$. We update U' in the following way: Delete the pair representing v, and if $t > 0$, add a new pair $(v'', t - 1)$. We modify S' in the following way: Insert the item $v' \cup s'$ instead of s'. The net effect is that a counter has possible been decreased by one and a subinterval has moved from U' to S' which will not increase the density, $d(U' \cup S')$. Thus, the second invariant still holds. If v'' could be accepted using $t - 1$ seat changes, we could accept v using t seat changes because of the way v'' was constructed. But this cannot be the case, since the first invariant held before this step. Thus, it also holds after this step. Note that according to the ordering of U', no element from S' will be a first blocking (sub)interval twice. Since every element starts with a counter s and part of it is kept in U' until this counter is 0, it will be a first element in the ordering of U' exactly $s + 1$ times. Every time this happens a new element from S will be mapped to it. Therefore, $|f^{-1}(u)| = s + 1$, for all $u \in U$. $\square$

Corollary 1 Any algorithm for the unit price problem has a competitive ratio on accommodating sequences of at least $\frac{1}{2}$.

Proof Let $\mathbb{A}$ denote an algorithm for the unit price problem. According to Theorem 2, the number of components γ in $\mathbb{A}$'s seating is at most $(s+1)|A|$, where A denotes the set of intervals accepted by $\mathbb{A}$. From Theorem 3, we have that $(s+1)U \leq \gamma$. Inserting this into the performance ratio, we get $\frac{|A|}{|A|+|U|} \geq \frac{|A|}{|A|+|A|} = \frac{1}{2}$. $\square$

Theorem 1 shows that this result is tight.

Recall that conservative algorithms only split intervals when continuing on the same seat is impossible due to a previous interval. This natural restriction leads to algorithms with better performance bounds.

Corollary 2 Any conservative algorithm for the unit price problem has a competitive ratio on accommodating sequences of at least $\frac{s+1}{s+2}$.

Proof Using the same notation and arguments as in the previous corollary, we get $\gamma \leq |A|$ and $(s+1)|U| \leq \gamma$. This gives a performance ratio at least $\frac{|A|}{|A|+\frac{|A|}{s+1}} = \frac{s+1}{s+2}$. $\square$

3.1.3 Tight Hardness Results

We now show that this result is asymptotically tight for any $n \geq 3$ due to Min-Change-First-Fit's behavior.

Theorem 4 The competitive ratio on accommodating sequences for Min-Change-First-Fit is at most $\frac{(s+1)k'+4s+4+\frac{1}{n}(12s+12)}{(s+2)k'+2s+1-\frac{1}{n}(2sk'-4k'-4s-2)}$, where $k' \leq k$ is the largest integer such that $k'-4$ is divisible by 6.

Proof Assume that k' is chosen largest possible such that $k'-4$ is divisible by 6 and $k' \leq k$. We first consider the case where 3 divides n. Give $\frac{n}{3}$ of each interval in each of the following 3 phases (See Figure 3):

1. $[1,2)$, $[k'-1,k')$, and $[6i-2,6i+2)$ for $i=1,\ldots,\frac{k'-4}{6}$.
2. $[1,4)$, $[k'-4,k'-1)$, and $[6i,6i+4)$ for $i=1,\ldots,\frac{k'-4}{6}-1$.
3. $[k'-2,k')$ and $[6i-4,6i)$ for $i=1,\ldots,\frac{k'-4}{6}$.

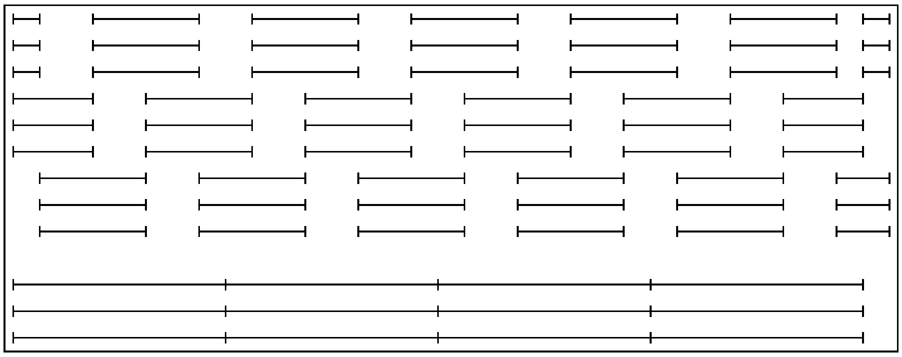


Figure 3: Placement for $k = 34$, $n = 9$ and $s = 3$.

When these intervals are accepted by Min-Change-First-Fit none of them have to change seats. Phase 1 is accepted on the first $\frac{n}{3}$ seats, Phase 2 is accepted on the next $\frac{n}{3}$ seats, and Phase 3 will use the remaining $\frac{n}{3}$ seats. Min-Change-First-Fit will accept these $\frac{n}{3}(3\frac{k'-4}{6}+4) = \frac{k'+4}{6}n$ intervals, but none of the following intervals

- $\frac{n}{3}$ times $[2(s+1)i-2s-1, 2(s+1)i+1)$ for $i = 1, \ldots, \lfloor \frac{k'-1}{2(s+1)} \rfloor$.

OPT uses $\frac{2n}{3}$ seats to accommodate the first three phases, since the density between any two stations is $\frac{2n}{3}$. OPT can use the remaining $\frac{n}{3}$ seats to accept the last phase. We get the following ratio:

$$\frac{\frac{k'+4}{6}n}{\frac{k'+4}{6}n + \frac{1}{3}n(\frac{k'-1}{2(s+1)}-1)} = \frac{k'+4}{k'+4+2(\frac{k'-1}{2(s+1)}-1)} = \frac{(s+1)k'+4s+4}{(s+2)k'+2s+1}.$$

If n is not divisible by 3, we let $c = n \bmod 3$ and give c intervals $[1, k)$ followed by the sequence above with $\frac{n}{3}$ replaced by $\lfloor \frac{n}{3} \rfloor$. In this case, the performance ratio is

$$\begin{aligned}
\frac{c+\frac{k'+4}{6}(n-c)}{c+\frac{k'+4}{6}(n-c)+\frac{1}{3}(n-c)(\frac{k'-1}{2(s+1)}-1)} &\le \frac{2+\frac{k'+4}{6}(n-2)}{2+\frac{k'+4}{6}(n-2)+\frac{1}{3}(n-2)(\frac{k'-1}{2(s+1)}-1)} \\
&\le \frac{2+\frac{k'+4}{6}n}{2+\frac{k'+4}{6}(n-2)+\frac{1}{3}(n-2)(\frac{k'-1}{2(s+1)}-1)} \\
&= \frac{(s+1)k'+4s+4+\frac{1}{n}(12s+12)}{(s+2)k'+2s+1-\frac{1}{n}(2sk'-4k'-16s-14)}.
\end{aligned}$$

$\square$

Clearly, the ratio could be made even closer to $\frac{s+1}{s+2}$ by shortening many of the holes to length 1.

The hardness result of $\frac{s+1}{s+2}$ asymptotically can be obtained for the much larger class of algorithms, Min-Change, for $n = 3$.

Consider the sequence in the proof above. Note that the intervals accepted by Min-Change-First-Fit are always given independently of s. Also note that for s arbitrary, an interval rejected by Min-Change-First-Fit is constructed by combining $s+1$ intervals that would have been given (and rejected by Min-Change-First-Fit) for the case $s = 0$. It is easy to see that such an interval will be rejected even if s seat changes are allowed. The reason is that the original rejected interval would require at least one seat change to be seated, and by combining $s + 1$ such intervals, at least $s + 1$ seat changes would be required for the resulting interval. We will use this observation below when extending a theorem from [2] for the original Seat Reservation problem.

Theorem 5 An algorithm $\mathbb{A}$ from the class Min-Change has a competitive ratio of at most $\frac{(s+1)(\frac{k'+5}{6}-s)}{(s+2)(\frac{k'+5}{6}-s)-2}$ on accommodating sequences for $n = 3$, where $k' \leq k$ is chosen largest possible such that 6 divides $k' + 5$.

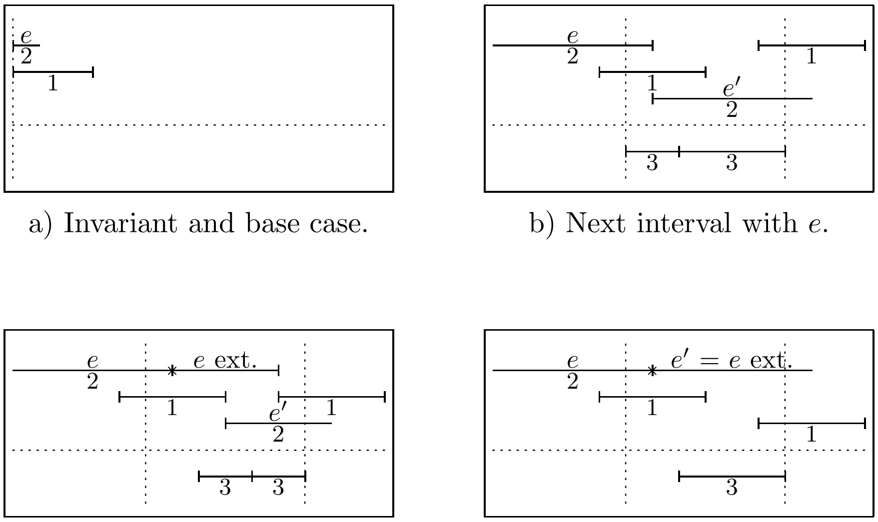
Proof Assume that k' is chosen largest possible such that $k' + 5$ is divisible by 6 and $k' \leq k$. Consider the sequence from Theorem 2.1 in [2] for $n = 3$: Note that in the original description which is summarized below, all Phase 3 intervals were rejected by the on-line algorithm, since no seat changes were allowed. Here these intervals will be combined into larger intervals as described later.

The adversary gives an input 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 Figure 4.

In Phases 2 and 3, the adversary gives additional intervals, determined by processing the already given intervals from left to right, based on how the on-line algorithm $\mathbb{A}$ 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 in [2] is by induction on the number of intervals in Phase 1. In each step one such interval is processed. The processing is illustrated in Figure 4. A step occurs between two dotted vertical lines. Since $\mathbb{A}$ is a Min-Change algorithm, all intervals from Phases 1 and 2 are accepted and none are split. These intervals are shown above the horizontal dotted line. During a step, either an interval from Phase 2 is extended, or a new interval for Phase 2 is defined, or both, and one or more intervals for Phase 3 are defined. The intervals from Phase 3 are shown below the horizontal dotted line. After each step, the invariant depicted in Figure 4a will hold:

- Every interval from Phases 1 or 2 (except the first two all the way to the left) with its



a) Invariant and base case.

b) Next interval with e.

c) Next interval with previous.

d) Next interval on free.

Figure 4: Cases of the proof of Theorem 5.

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 Figure 4a. 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, it is assumed that all intervals up to a certain station have been processed, and that the invariant is maintained. In processing the next Phase 1 interval, there are three cases: the interval could be on the same seat as e (Figure 4b), on the same seat as the previous Phase 1 interval (Figure 4c), or on the third seat (Figure 4d). In each case, the invariant must be reestablished six stations further to the right (the next vertical dotted line).

In Figure 4b, a new Phase 2 interval e' which should serve as the e interval in the next step is given. Note that $\mathbb{A}$ 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 Figure 4c, an already given e interval is extended. 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 Figure 4d, an e interval is extended, and that interval will 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 and $\mathbb{A}$ is a Min-Change algorithm.

Having described the sequence from [2] the next step is to show how this can be used, when s seat changes are allowed. We will change Phase 3 to become Phase 3′ and depending on this Phase we will construct Phase 1′ and Phase 2′ from Phase 1 and Phase 2 respectively. The adversary should use the sequence consisting of Phase 1′, Phase 2′, and Phase 3′. The construction of Phase 3′: From left to right we combine $s+1$ *small* intervals from Phase 3, say, $\langle v_1, v_2, \ldots, v_{s+1}\rangle$ into one *large* interval in Phase 3′. The large interval starts where v_1 starts and ends where v_{s+1} ends. This process is repeated for each group of $s+1$ intervals. At the end, there are at most s intervals from Phase 3 which have not been combined into a large interval. Since these intervals cannot be used in Phase 3, we modify Phases 1 and 2 to obtain Phases 1′ and 2′ by removing the corresponding intervals from Phases 1 and 2. If $r \leq s$ intervals are not used from Phase 3, we remove r intervals in total from Phases 1 and 2.

The algorithm $\mathbb{A}$ will reject every large interval from Phase 3′ even if s seat changes are allowed, since for each of the small intervals at least one seat change is required.

An optimal off-line algorithm can accommodate all intervals, 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 (small) Phase 3 interval for each. Additionally, it gives some number $x \geq 1$ of intervals in Phase 2, and $x-1$ (small) Phase 3 intervals for these. The number of (large) intervals in Phase 3′ is therefore

$$\frac{\frac{k'+5}{6} - 1 + x - 1 - r}{s+1}.$$

Note that r is chosen such that this ratio is integer. The total number of intervals in Phases 1′ and 2′ is $\frac{k'+5}{6} + x - r$, so the performance ratio is

$$\begin{aligned}\frac{\frac{k'+5}{6} + x - r}{\frac{k'+5}{6} + x - r + \frac{\frac{k'+5}{6} - 2 + x - r}{s+1}} &= \frac{(s+1)(\frac{k'+5}{6} + x - r)}{(s+2)(\frac{k'+5}{6} + x - r) - 2} \\ &\leq \frac{(s+1)(\frac{k'+5}{6} + 1 - s)}{(s+2)(\frac{k'+5}{6} + 1 - s) - 2}.\end{aligned}$$

□

Theorem 6 The competitive ratio on accommodating sequences for First-Fit* is at most $\frac{(s+1)k'-s-1}{(s+2)k'-11s-14}$, where $k' \leq k$ is the largest integer such that $k'-4$ is divisible by 6.

Proof First assume that $s+3$ divides n. Let $r = \frac{1}{s+3}$. Consider the following sequence divided into three phases (See Figures 5 and 6 where the placement by First-Fit* is shown for $s = 1$ and $s = 3$ respectively):

1. rn times $[k'-2s-2, k')$, then rn times $[k'-2s, k'), \ldots,$ then rn times $[k'-2, k')$.
2. For $i = 0, \ldots, \lfloor \frac{k'-2s-9}{6} \rfloor$, let $s' = k'-2s-4-6i$ and let $l = 2(s+2)$. Give rn times $[s', s'+l)$, then rn times $[s'-4, s'-4+l)$, then rn times $[s'-2, s'-2+l,)$.
3. For $i = 0, \ldots, \lfloor \frac{k'-4s-7}{2(s+1)} \rfloor$, let $s' = k'-1-2(s+1)-2i(s+1)$ and give rn times $[s', s'+2(s+1))$.

First-Fit* will accept all intervals from Phase 1 and Phase 2, but then be forced to reject all intervals from Phase 3. Since the density is nowhere larger than n, OPT can accept all intervals, giving a performance ratio of

$$\begin{aligned}
\frac{rn(s+1) + 3rn(\lfloor \frac{k'-2s-9}{6} \rfloor + 1)}{rn(s+1) + 3rn(\lfloor \frac{k'-2s-9}{6} \rfloor + 1) + rn(\lfloor \frac{k'-4s-7}{2(s+1)} \rfloor + 1)} &\leq \\
\frac{rn(s+1) + 3rn(\frac{k'-2s-9}{6} + 1)}{rn(s+1) + 3rn\frac{k'-2s-9}{6} + rn\frac{k'-4s-7}{2(s+1)}} &= \\
\frac{2(s+1) + (k'-2s-9) + 6}{2(s+1) + k'-2s-9 + \frac{k'-4s-7}{s+1}} &= \\
\frac{k'-1}{k'-7+\frac{k'-4s-7}{s+1}} &= \\
\frac{(s+1)k'-s-1}{(s+2)k'-11s-14} &
\end{aligned}$$

If n is not divisible by $s+3$, let $c = m \bmod (s+3)$. The adversary can give c intervals $[1, k)$ followed by the sequence described above with rn replaced by $\lfloor rn \rfloor$. In this case, the performance ratio is

$$\frac{c + r(n-c)(s+1) + 3r(n-c)(\lfloor \frac{k'-2s-9}{6} \rfloor + 1)}{c + r(n-c)(s+1) + 3r(n-c)(\lfloor \frac{k'-2s-9}{6} \rfloor + 1) + r(n-c)(\lfloor \frac{k'-4s-7}{2(s+1)} \rfloor + 1)}$$

Similarly to earlier proofs, we remove c from $n-c$ in the enumerator. Using that c is at most $s+2$, we end up with the following upper bound on the performance ratio:

$$\frac{(s+2) + rn(s+1) + 3rn(\frac{k'-2s-9}{6} + 1)}{c + r(n-(s+2))(s+1) + 3r(n-(s+2))\frac{k'-2s-9}{6} + r(n-(s+2))\frac{k'-4s-7}{2(s+1)}}$$

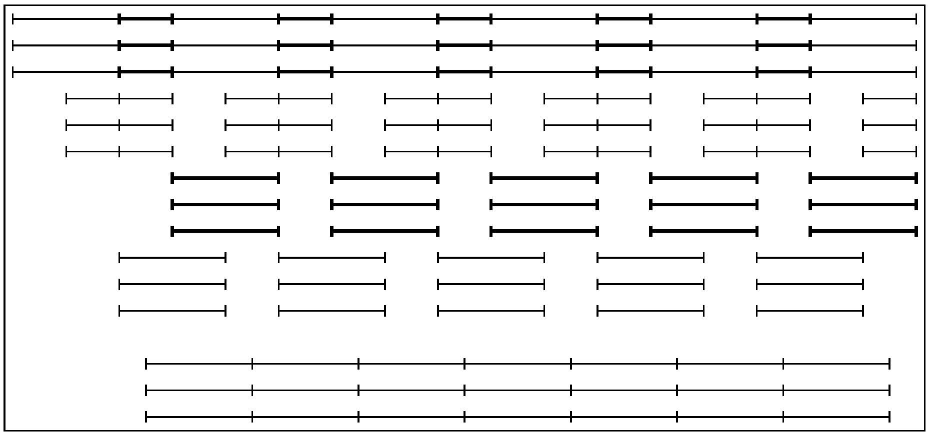


Figure 5: Placement by First-Fit* for $k = 35$, $n = 12$, and $s = 1$. From Phase 2 the first interval is emphasized for all i.

Inserting $r = \frac{1}{s+3}$ and simplifying, we get

$$\frac{(s+1)k' - s - 1 + \frac{1}{n}(s+1)(2s^2 + 10s + 12)}{(s+2)k' - 11s - 14 - \frac{1}{n}(s^2k' + 4sk' + 4k' - 9s^2 - 28s - 22)}.$$

□

3.2 Competitive Ratio

Above we showed that the competitive ratios are constant, when the input sequences are restricted to accommodating sequences. In this section we consider the competitive ratio (on all sequences) obtaining much more pessimistic results. We believe that the restriction to conservative algorithm is very natural and therefore we find competitive ratios both against any optimal algorithm and against the class of conservative off-line algorithms.

3.2.1 Performance Guarantees

In the class of conservative algorithms every algorithm has a competitive ratio of at least $\frac{1}{k}$ as the following lemma shows.

Theorem 7 The competitive ratio of any conservative algorithm is at least $\frac{1}{k}$.

Proof Let $\mathbb{A}$ denote a conservative algorithm. Assume that R is non empty, since otherwise the algorithm is optimal on the input sequence. As a consequence of Theorem 2, a conservative

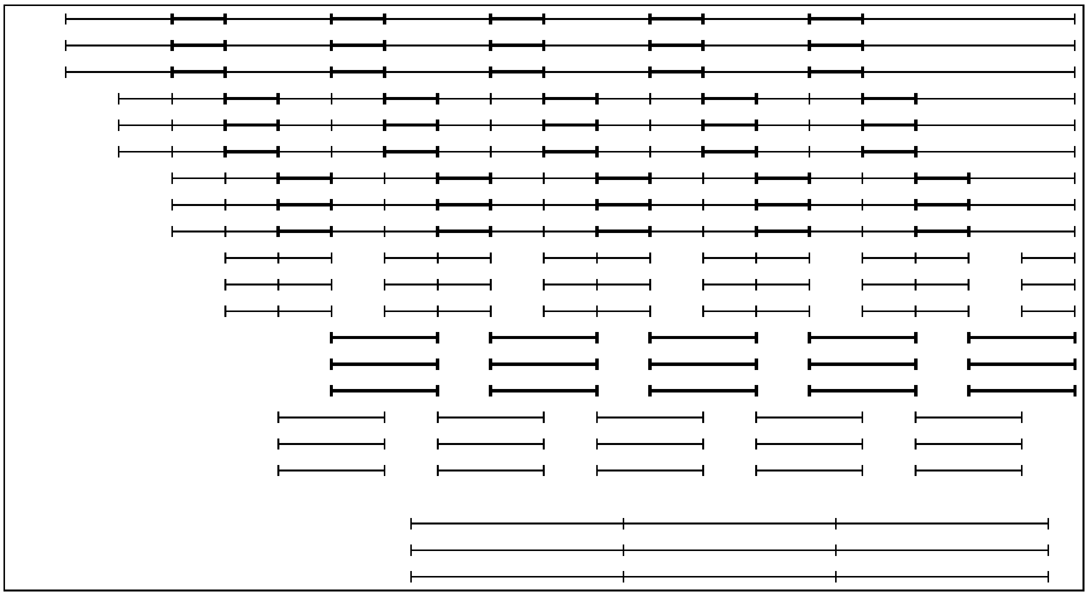


Figure 6: Placement by First-Fit* for $k = 41$, $n = 18$, and $s = 3$. From Phase 2 the first interval is emphasized for all i.

algorithm creates at most one new component for each interval accepted. To reject an interval, there must be at least one component on every seat, so $\mathbb{A}$ accepts at least n intervals. Since no algorithm can accept more than $(k-1)n$ intervals, the bound follows. $\square$

Theorem 10 in the next section shows that this result is tight; the competitive ratio of any conservative algorithm is $O(\frac{1}{k})$ if OPT is not restricted to being conservative.

If the algorithm is not required to be conservative, we cannot prove a performance guarantee better than $\frac{1}{(s+1)k}$.

Theorem 8 The competitive ratio of any algorithm is at least $\frac{1}{(s+1)k}$.

Proof let $\mathbb{A}$ denote an arbitrary algorithm. Assume $R \neq \emptyset$. Again using Theorem 2, to get at least n components, $\mathbb{A}$ must accept at least $\frac{n}{s+1}$ intervals. The bound follows using the same arguments as in the previous proof. $\square$

3.2.2 General Hardness Results

Now we turn to upper bounds on the competitive ratio for the unit price problem.

Theorem 9 If $n = 2$ and $s \in o(k)$, the competitive ratio of any algorithm is $O(\frac{s}{k})$, even if OPT is restricted to being conservative.

Proof The adversary gives $s+2$ unit intervals starting at the first stations except station 3.

- $[i, i+1)$ for $i = 1, \ldots s+3$ and $i \neq 3$.

We divide the proof into two cases depending of the placement of these intervals.

- Case 1: There exists two consecutive intervals on the same seat. (We also consider $[2, 3)$ and $[4, 5)$ to be consecutive.)
- Case 2: All intervals were placed alternating between the two seats.

If Case 1 occurs, the adversary gives $[1, k)$, which can be accepted by $\mathbb{A}$ using at most s seat changes (see Figure 7). Note that from station $s+4$ to k, this interval will change seat at most s times. Therefore, the next interval $[s+4, k)$ can be accepted using at most s seat changes (it would change seat at the same positions). At this point $\mathbb{A}$ has filled both seats except between station 3 and 4, where only one interval is given. Therefore, the following unit size intervals must be rejected by $\mathbb{A}$:

- $[i, i+1)$ for $i = 1, \ldots, k-1$ and $i \neq 3$.

OPT can arrange the first $s+2$ intervals alternating between the two seats. Therefore the interval of length $k-1$ is rejected and there is room for the remaining sequence.

If Case 2 occurs, the adversary gives $[3, 4)$. Now consider the $s+2$ intervals starting at stations $2, \ldots, s+3$. Since the interval just given must be placed on the same seat as either $[2, 3)$ or $[4, 5)$, we are in a situation similar to Case 1, in which two consecutive intervals are on the same seat. Instead of giving $[1, k)$ as in Case 1, we give $[2, k)$. The remaining intervals are the same except that $[1, 2)$ should be left out, and [3,4) can be included.

In Case 2, the algorithm will get the best performance ratio. In this case its performance is $s+2+1+1+1 = s+5$. OPT has a performance of $s+2+1+1+(k-2) = k+s+2$. □

A stronger negative result holds in the case where OPT is not restricted to being conservative, even if the algorithm is conservative. This matches the performance guarantee of Theorem 7 to within a constant factor.

Theorem 10 For $s \geq 1$ and n arbitrary, any conservative algorithm has a competitive ratio of $O(\frac{1}{k})$ for the unit price problem, if OPT is not restricted to being conservative.

Proof Let $\mathbb{A}$ denote a conservative algorithm. The adversary gives

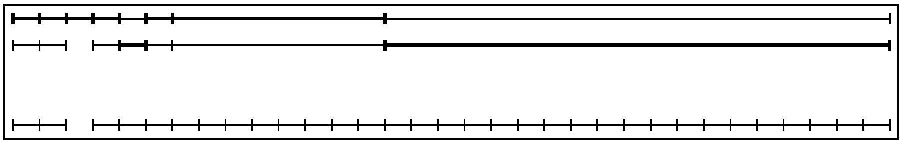


Figure 7: Non-conservative placement of Case 1 in Theorem 9 for $k = 34$ and $s = 3$. The interval $[1, k)$ is emphasized.

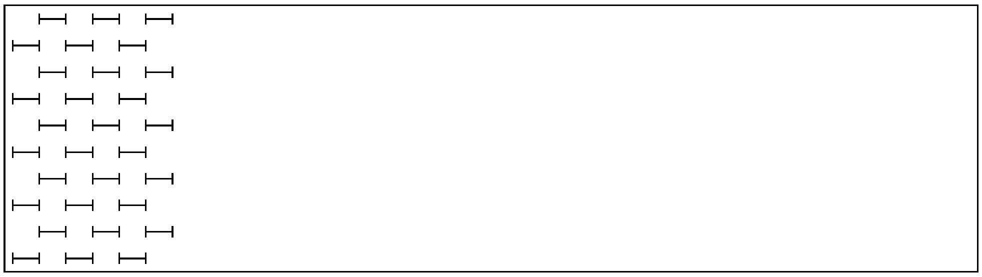


Figure 8: "Checkerboard" for $k = 37$, $n = 10$ and $s = 4$.

- $\frac{n}{2}$ times $[1, s + 2)$.
- $\frac{n}{2}$ times $[s + 2, s + 3)$.

Since $\mathbb{A}$ is conservative, no seat changes are performed for the first $\frac{n}{2}$ intervals. Therefore, the algorithm must accept the following:

- $\frac{n}{2}$ times $[1, k)$.

OPT will behave as in Figure 8 and reject all the long intervals, and therefore make room for:

- $\frac{n}{2}$ times $[s + 3, k)$.
- $\frac{n}{2}$ times $[i, i + 1)$ for $i = 0, \ldots, k - 1$.

The small intervals given at the end are rejected by $\mathbb{A}$. The performance ratio is $\frac{\frac{n}{2}+\frac{n}{2}+\frac{n}{2}+\frac{n}{2}}{\frac{n}{2}+\frac{n}{2}+\frac{n}{2}+\frac{n}{2}(k-1)} = \frac{4}{2+k}$. □

Adding the restriction that OPT also be conservative, gives a slightly less pessimistic negative result.

Theorem 11 Any conservative algorithm has a competitive ratio of $O(s^2/k)$ for the unit price problem, if OPT is conservative and $s \in o(k)$.

Proof Let $\mathbb{A}$ denote a conservative algorithm for the unit price problem. The adversary gives

- $\frac{n}{2}$ times $[i, i+1)$ for $i = 1, \ldots, s+2$.

Let q denote the number of holes of size at least 2. If an interval containing $[1, s+3)$ is given, it will be accepted if there is at least one hole of size at least 2 (and the density is smaller then n). On the other hand, since an interval is allowed to change seat at most s times, at most $s+1$ of these holes will be used (and either disappear or reduce their size to 1). $\mathbb{A}$ will therefore have to accept x intervals containing $[1, s+3)$, where $\frac{q}{s+1} \leq x \leq q$. We divide the proof into two cases depending on q:

- Case 1: $q > \frac{s+1}{2(s+2)}n$.
- Case 2: $q \leq \frac{s+1}{2(s+2)}n$.

If Case 1 occurs, the adversary gives

- q times $[1, k)$.

In this case, we let x count the number of intervals of length $k-1$ accepted by $\mathbb{A}$. The adversary continues with

- $n-x$ times $[s+3, k)$.
- x times $[i, i+1)$ for $i = s+3, \ldots, k-1$.

OPT behaves as in Figure 8 and rejects all intervals of length $k-1$. The performance ratio is $\frac{\frac{n}{2}(s+2)+x+(n-x)}{\frac{n}{2}(s+2)+(n-x)+x(k-1)} \leq \frac{\frac{n}{2}(s+2)+n}{\frac{n}{2}(s+2)+n+\frac{q}{s+1}(k-2)} \leq \frac{\frac{n}{2}(s+2)+n}{\frac{n}{2}(s+2)+n+\frac{n}{2(s+2)}(k-2)} = \frac{(s+2)(s+4)}{(s+2)(s+4)+k-2}$

If Case 2 occurs, the adversary gives

- $\frac{n}{2}$ times $[1, s+3)$.

In this case, we let x count the number of intervals of length $s+2$ accepted by $\mathbb{A}$. The adversary continues with

- $\frac{n}{2} - x$ times $[2, k)$.
- $\frac{n}{2} + x$ times $[s+3, k)$.

- $\frac{n}{2} - x$ times $[i, i+1)$ for $i = s+3, \ldots, k-1$.

When the intervals of the form $[2, k)$ are received, at most s seat changes can be necessary. So they are all accepted. The [s+3,k) intervals are accepted without seat changes, so the last group of intervals must all be rejected. OPT can arrange to accept all of the [1,s+3) intervals and thus reject all of the [2,k) intervals, leaving space for the short intervals at the end.

The performance ratio is

$$\begin{aligned}
&\frac{\frac{n}{2}(s+2) + x + \frac{n}{2} - x + \frac{n}{2} + x}{\frac{n}{2}(s+2) + \frac{n}{2} + \frac{n}{2} + x + (\frac{n}{2} - x)(k - (s+3))} \\
&= \frac{\frac{n}{2}(s+2) + n + x}{\frac{n}{2}(s+2) + \frac{3n}{2} + (\frac{n}{2} - x)(k - s - 4)} \\
&\leq \frac{\frac{n}{2}(s+2) + n + q}{\frac{n}{2}(s+2) + \frac{3n}{2} + (\frac{n}{2} - q)(k - s - 4)} \\
&\leq \frac{\frac{n}{2}(s+2) + n + \frac{s+1}{2(s+2)}n}{\frac{n}{2}(s+2) + \frac{3n}{2} + (\frac{n}{2} - \frac{s+1}{2(s+2)}n)(k - s - 4)} \\
&= \frac{(s+2)(s+4) + s + 1}{(s+2)(s+5) + k - s - 4}.
\end{aligned}$$

□

3.2.3 Hardness Result for First-Fit*

For the specific conservative algorithms First-Fit* and Min-Change-First-Fit, it is possible to prove a stronger upper bound.

Theorem 12 If OPT is conservative, First-Fit* and Min-Change-First-Fit will have competitive ratios of $O(\frac{s}{k})$.

Proof Assume $s+1$ divides $k-1$ and n and that $k-1$ divides $n(s+1)$. Let $r = \frac{k-1}{s+1}$. Let $d = \frac{n(s+1)}{k-1}$. An adversary will give the sequence W followed by

- $n - d$ times $[1, k)$.

which First-Fit* and Min-Change-First-Fit will accept. OPT will behave just like Next-Fit on W and reject all these intervals. The adversary then gives

- $n - d$ times $[i, i+1)$ for $i = 1, \ldots, k-1$.

which will be accepted by OPT and rejected by First-Fit* and Min-Change-First-Fit. The performance ratio is $\frac{(s+1)n+(n-d)}{n(k-1)} \leq \frac{s+2}{k-1}$ $\square$

3.3 Distinguishing Between Algorithms Using the Accommodating Function

The restriction to accommodating sequences has been extended to a function of restrictions, the accommodating function, in [5] and [7]. In [5], this function was used to distinguish between three different algorithms for the seat reservation problem, all of which have the same competitive ratio on accommodating sequences. Here, we use the accommodating function to distinguish between three algorithms for the seat reservation problem allowing one seat change.

For completeness, we now define the accommodating function for the seat reservation problem (allowing seat changes). We say that I is an *α-sequence*, if all the intervals in the sequence I could be accommodated on αn seats by an optimal off-line algorithm. We investigate the competitive ratio on such restricted sequences. To be precise, an on-line algorithm $\mathbb{A}$ is *c-competitive on α-sequences* if $c \leq 1$ and there exists a constant b, such that for every α-sequence I, $\mathbb{A}(I) \geq c \cdot \mathrm{OPT}(I) - b$. The *accommodating function* $\mathcal{A}$ is defined as $\mathcal{A}(\alpha) = \sup\{c \mid \mathbb{A}$ is c-competitive on α-sequences$\}$.

Thus, the accommodating function for an algorithm is the competitive ratio of that algorithm on α-sequences as a function of α. Note that 1-sequences are *accommodating sequences*, since the optimal algorithm can accommodate all intervals in such a sequence.

3.3.1 The Modified-Kierstead-Trotter Algorithm

The first algorithm we consider is a modification of Kierstead and Trotter's algorithm [12]. First we review the original algorithm which can be applied to the original seat reservation problem when the density of the input sequence is at most $\frac{n+2}{3}$. Then we describe the modified algorithm which achieves $\mathcal{A}(\alpha) = 1$ for $\alpha = 1/2$, when one seat change is allowed.

The Kierstead-Trotter algorithm uses d different buckets, $B_i, 1 \leq i \leq d$, to hold the items:

Kierstead-Trotter Algorithm
The input sequence I is assumed to have density d, and the intervals in I are processed in the order given. Initially, the sets B_i, $1 \leq i \leq d$, are all empty. To determine where to place a new interval x, find the smallest j such that the set of intervals $\{x\} \cup \bigcup_{i=1}^{j} B_i$ has density j. Set $B_j = B_j \cup \{x\}$. If $j = 1$, place x on seat 1. If $j \geq 2$, use First-Fit to place x on one of the seats in $\{3j-4, 3j-3, 3j-2\}$.

To show that this algorithm works, Kierstead and Trotter prove that each of the sets B_i has density at most two, and each interval x in B_i overlaps at most two other intervals in B_i. This fact is used to show that $3d-2$ seats are sufficient to seat a sequences of density d without seat changes. We use the same fact to show that by allowing one seat change $2d-2$ seats suffices when $d \geq 2$, since in this case only two seats are necessary for $B_1 \cup B_2$ and for any other B_i, if First-Fit* is used. This is shown in the following lemma.

Lemma 2 Let I be a sequence of intervals with density at most two. Then, First-Fit* accommodates the entire sequence I using only two seats.

Proof Consider such a sequence I with density at most two. The proof is by induction on the number of intervals accepted by First-Fit*. The induction hypothesis says that for every interval $x = [x_1, x_2)$ seen so far

1. x has been accepted by First-Fit*.
2. Let z denote the earliest station among x_2 and the starting stations of intervals overlapping x. If there is no interval $y = [y_1, y_2)$ overlapping x with $y_1 < x_1$, the portion of x from x_1 to z is seated on seat 1.

The first interval clearly satisfies this condition. Assume that l intervals have been accepted and $x = [x_1, x_2)$ is the $(l+1)$'st to be considered by First-Fit*. To show that the properties are satisfied, we consider intervals $y = [y_1, y_2)$ overlapping x. We first show that for such intervals with $x_1 \leq y_1$, the portion overlapping x will be seated on seat 1. Next, we consider two cases depending on whether or not there is an interval $y = [y_1, y_2)$ overlapping x with $y_1 < x_1$.

Suppose $y = [y_1, y_2)$ overlaps x and $x_1 \leq y_1$. If $y \subset x$, no other interval overlaps y, since the density of I is at most 2, therefore y is placed on seat 1 by First-Fit*. If $y_2 > x_2$, the portion $[y_1, x_2)$ of y does not overlap any other interval, therefore according to the induction hypothesis it is seated on seat 1 by the First-Fit* strategy. With z defined as in property 2, it follows that seat 2 is free from z to x_2 before x is seated.

Assume that there exists an interval $y = [y_1, y_2)$ overlapping x such that $y_1 < x_1$. If the portion of y overlapping x is seated on seat 1, there is no interval on seat 2 overlapping x, therefore x can be seated on seat 2. If the portion of y overlapping x is seated on seat 2, x can be seated on seat 1 as long as it overlaps y, and from then on seat 2 is free. This justifies condition 2 of the hypothesis. Note that First-Fit* will not change to seat 2 before it is necessary.

If no y exists with $y_1 < x_1$, we have to verify condition 2 of the hypothesis. As noted above, seat 2 will be free for the entire trip requested by x. First-Fit* will seat as much of x as possible on seat 1 and then change to seat 2 when necessary.

In any case x is accepted satisfying condition 1. □

This makes it clear that the following algorithm, Modified-Kierstead-Trotter, will accommodate all intervals in any α-sequence with $\alpha \leq 1/2$, if one or more seat changes are allowed.

Modified-Kierstead-Trotter Algorithm
The input sequence I is assumed to have density d, and the intervals in I are processed in the order given. Initially, the sets B_i, $1 \leq i \leq d$, are all empty. To determine where to place a new interval x, find the smallest j such the set of intervals $\{x\} \cup \bigcup_{i=1}^{j} B_i$ has density j. Set $B_j = B_j \cup \{x\}$. If $j = 1$ or $j = 2$, use First-Fit* to accommodate x on seats 1 or 2. If $j \geq 3$, use First-Fit* to accommodate x on one of the seats in $\{2j-3, 2j-2\}$.

Corollary 3 Modified-Kierstead-Trotter is 1-competitive if $n \geq 2$ and the density of the input sequence is no more than $\frac{n+2}{2}$.

3.3.2 The Checkerboard Algorithm

The second algorithm we consider, the Checkerboard algorithm, is the worst of the three for $\alpha = \frac{1}{2}$.

Theorem 13 There exists a fair, conservative algorithm, the Checkerboard algorithm, such that for $\frac{s+1}{k-1} < \alpha \leq 1$, the accommodating function for this algorithm is bounded above by $\mathcal{A}(\alpha) \leq \frac{1}{1+\frac{\alpha}{s+1}-\frac{1}{k-1}}$, when s seat changes are allowed.

Proof Assume that $k-1$ divides $(s+1)n$, and consider the following input sequence:

- $\frac{(s+1)n}{k-1}$ requests for $[i, i+1)$ for $1 \leq i \leq k-1$.
- $\alpha n - \frac{(s+1)n}{k-1}$ requests for $[1, k)$.

Let the Checkerboard Algorithm be one of the algorithms which places $s+1$ of the short intervals on each of the n seats, with no two intervals adjacent to each other. Clearly this is possible, if $k \geq 2(s+1)$. Thus, all of the long intervals will be rejected. OPT will place all of the short intervals on the first $\frac{(s+1)n}{k-1}$ seats and thus can also accept the long intervals. Hence, the performance ratio is $\frac{(s+1)n}{(s+1)n+\alpha n-\frac{(s+1)n}{k-1}}$, giving the required upper bound on $\mathcal{A}(\alpha)$. □

Notice that for $\alpha = \frac{1}{2}$ and $s = 1$, this hardness result on the accommodating function for the Checkerboard algorithm approaches $\frac{4}{5}$, asymptotically.

3.3.3 Min-Change-First-Fit

The third algorithm we consider is Min-Change-First-Fit. First, we separate Min-Change-First-Fit from the Checkerboard algorithm, showing that for one seat change, Min-Change-First-Fit is strictly better than the Checkerboard algorithm at $\alpha = 1/2$, when $k \geq 20$. The proof uses some of the techniques used in [5], showing that for no seat changes First-Fit's accommodating function is bounded below by $\mathcal{A}_{FF}(\alpha) \geq 1 - \frac{1}{2^l}$, where $l = \lfloor \frac{1}{\alpha} \rfloor$ and $\alpha \leq 1$.

Theorem 14 For Min-Change-First-Fit, the accommodating function $\mathcal{A}(\alpha) \geq \frac{2^{l-1}(1-l\alpha+\alpha)(s+1)+(2^{l-1}-1)\alpha}{2^{l-1}(1-l\alpha+\alpha)(s+1)+2^{l-1}\alpha} \geq 1 - \frac{1}{2^{l-1}(s+2)}$, where $l = \lfloor \frac{1}{\alpha} \rfloor$ and $\alpha \leq 1$.

Proof Consider an α-sequence I. Let A be the set of intervals which are accepted by Min-Change-First-Fit and R be the set of intervals rejected.

Divide the seats up into l blocks, where $l = \lfloor \frac{1}{\alpha} \rfloor$, such that the first αn seats are in block 1, the next αn seats in block 2, etc. Any leftover seats will be added to the lth block, so that it has $n - l\alpha n + \alpha n$ seats. Let B_i denote the set of connected components of intervals which Min-Change-First-Fit places in the ith block.

We consider the last block first and compute a lower bound on $|B_l|$, the number of connected components of intervals from I which Min-Change-First-Fit places in B_l. Choose the αn seats S from among the last $n - l\alpha n + \alpha n$ which contain the fewest of these components. Using Theorem 3 gives that the number of these components in S is at least $(s+1)|R|$. Since Min-Change-First-Fit places at least as many components on each seat in $B_l \setminus S$ as it does on any seat in S, $|B_l| \geq \left(\frac{(1-l\alpha+\alpha)(s+1)}{\alpha}\right)|R|$.

Now consider the algorithm First-Fit, which does not allow seat changes, and a modified input sequence I'. For each component which Min-Change-First-Fit placed on some seat, there is an interval in I' covering exactly the same subintervals as the component. The intervals in I' are ordered according to the numbers of the seat on which Min-Change-First-Fit placed them, with those placed on the first seat coming first, etc. The intervals in R come at the end of I'. Note that since Min-Change-First-Fit is conservative, by Theorem 2, I' contains at most as many intervals as I does.

First-Fit will place the intervals in I' on exactly the same seats as Min-Change-First-Fit placed the corresponding components. Clearly, First-Fit could not place any of the intervals in block B_l or R in any previous block. Thus, one can look at the intervals in the second to last block, in the last block, and in R, and note that they must form an α-sequence. Using the result from [6] that any fair algorithm is at least $\frac{1}{2}$-competitive on accommodating sequences, we have that $|B_{l-1}| \geq |R| + |B_l|$. Similarly, $|B_i| \geq |R| + \sum_{j=i+1}^{l} |B_j|$. If C_i denotes the total number of intervals First-Fit places in the last i blocks, then $C_i \geq 2C_{i-1} + |R|$, so $C_l \geq \left(\frac{2^{l-1}(1-l\alpha+\alpha)(s+1)+(2^{l-1}-1)\alpha}{\alpha}\right)|R|$. Thus, the performance ratio for Min-Change-First-Fit is $\frac{|A|}{|A|+|R|} = \frac{C_l}{C_l+|R|} \geq \frac{2^{l-1}(1-l\alpha+\alpha)(s+1)+(2^{l-1}-1)\alpha}{2^{l-1}(1-l\alpha+\alpha)(s+1)+(2^{l-1})\alpha}$. □

For $\alpha = 1/2$ and $s = 1$, since $l = 2$, Min-Change-First-Fit has a performance guarantee of at least $\frac{5}{6}$. However, Min-Change-First-Fit's performance ratio at $\alpha = 1/2$ is not 1, so Min-Change-First-Fit is not as good as the Modified-Kierstead-Trotter algorithm described above at $\alpha = 1/2$, when one seat change is allowed. This can be shown using Chrobak and Ślusarek's result [8] that there exists a sequence where First-Fit uses more than 4.4 times as many seats (colors) as OPT, and thus is not 1-competitive on $\frac{1}{2}$-sequences.

Theorem 15 Min-Change-First-Fit is not 1-competitive on α-sequences, for $\alpha > 5/22$ and k sufficiently large.

Proof Consider the sequence which Chrobak and Ślusarek use to prove their result of 4.4. Let n be the number of seats which First-Fit would use to pack their sequence. Then, their sequence is a $\frac{5}{22}$-sequence. Create $s+1$ copies of their sequence as follows: Suppose the last station they use is station l. For every interval $[u, v)$ in their sequence, create $s+1$ copies, $[u + il, v + il)$ for $0 \leq i \leq s$. At the end of the sequence place $\alpha n - \frac{5}{22}n$ copies of the interval $[1, k)$. Since Min-Change-First-Fit would pack all but the last $[1, k)$ intervals exactly as First-Fit would, it will have at least $s+1$ intervals on every seat, with none adjacent to each other. Thus, Min-Change-First-Fit will reject the last intervals, exactly as First-Fit would. Hence, the performance ratio is not 1. $\square$

Note that any result, such as Chrobak and Ślusarek's, giving a construction showing that some algorithm $\mathbb{A}$ uses r times as many seats as OPT, will show that the variant of Min-Change based on $\mathbb{A}$ is not 1-competitive on α sequences for $\alpha > 1/r$.

4 Proportional Price

Here we consider the proportional price problem, so the goal is to maximize the sum of the lengths of the intervals accepted. We assume that the stations are evenly spaced. In this section we let $T(v)$ denote the length of the (sub)interval v.

4.1 The Competitive Ratio on Accommodating Sequences

The proportional price problem allowing seat changes is similar to the original proportional price problem in that there is no dramatic difference between the competitive ratio and the competitive ratio on accommodating sequences. Both are quite bad.

4.1.1 Performance Guarantee

Theorem 16 Any conservative algorithm for the proportional price problem has a competitive ratio on accommodating sequences of $\Omega(s/k)$, if $s \in o(k)$.

Proof Let $\mathbb{A}$ denote an arbitrary conservative algorithm for the proportional price problem, let S denote the set of components accepted by $\mathbb{A}$, and let U denote the set of intervals rejected. From Theorem 2 and Theorem 3 we know that $|S| \geq |U|(s+1)$. Since every component in S has size at least 1 and every element in U has size at most $k-1$ we obtain the claimed ratio: $\frac{T(S)}{T(S)+T(U)} \geq \frac{|U|(s+1)}{|U|(s+1)+|U|(k-1)}$ □

4.1.2 Matching Hardness Result

Negative results on Next-Fit's and Min-Change-First-Fit's performances show that the result in Theorem 16 is asymptotically tight for $s \in o(k)$.

Theorem 17 Next-Fit has a competitive ratio of at most $\frac{s+1}{k-1}$ on accommodating sequences for the proportional price problem.

Proof Assume $s+1$ divides $k-1$ and n and that $k-1$ divides $n(s+1)$. Let $r = \frac{k-1}{s+1}$. Let $d = \frac{n(s+1)}{k-1}$. The adversary gives the sequence W. According to Lemma 1, Next-Fit will have to reject the next intervals:

- $n-d$ times $[1,k)$.

which OPT can accept if it behaves like First-Fit. Since OPT has filled every seat completely, the performance ratio is $\frac{n(s+1)}{n(k-1)} = \frac{s+1}{k-1}$. □

Theorem 18 If $s \leq k-5$, the competitive ratio on accommodating sequences of Min-Change-First-Fit for the proportional price problem is at most $\frac{2s+10}{k+2s+9}$.

Proof Let $p \geq s+3$ be chosen smallest possible such that 3 divides $p+1$. The adversary uses a sequence very similar to the sequence in Theorem 4 giving $\frac{n}{3}$ of each of the intervals below (See Figure 9):

- $[1,2)$, $[p-1,p)$, and $[3i, 3i+2)$ for $i = 1, \dots, \frac{p-4}{3}$.
- $[3i+1, 3i+3)$ for $i = 0, \dots, \frac{p-4}{3}$.
- $[3i+2, 3i+4)$ for $i = 0, \dots, \frac{p-4}{3}$.

The sequence above is contained in [1,p), where $s+3 \leq p \leq s+5$. For each station the sequence has a density of $\frac{2n}{3}$, giving Min-Change-First-Fit a performance of $p(\frac{2n}{3})$. Min-Change-First-Fit will occupy all n seats such that no seat is free for 2 consecutive trips of unit size between station 1 and station p. This means that an element spanning all intervals given above, must be rejected. The adversary continues giving

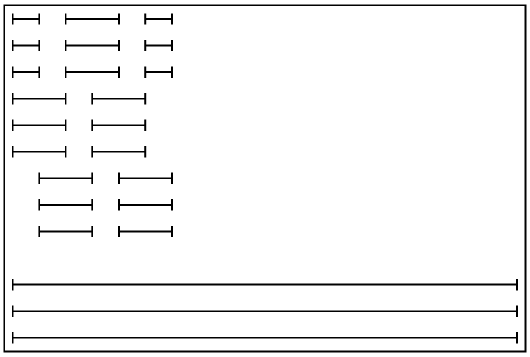


Figure 9: Placement by Min-Change-First-Fit for $k = 20$, $n = 9$ and $s = 3$.

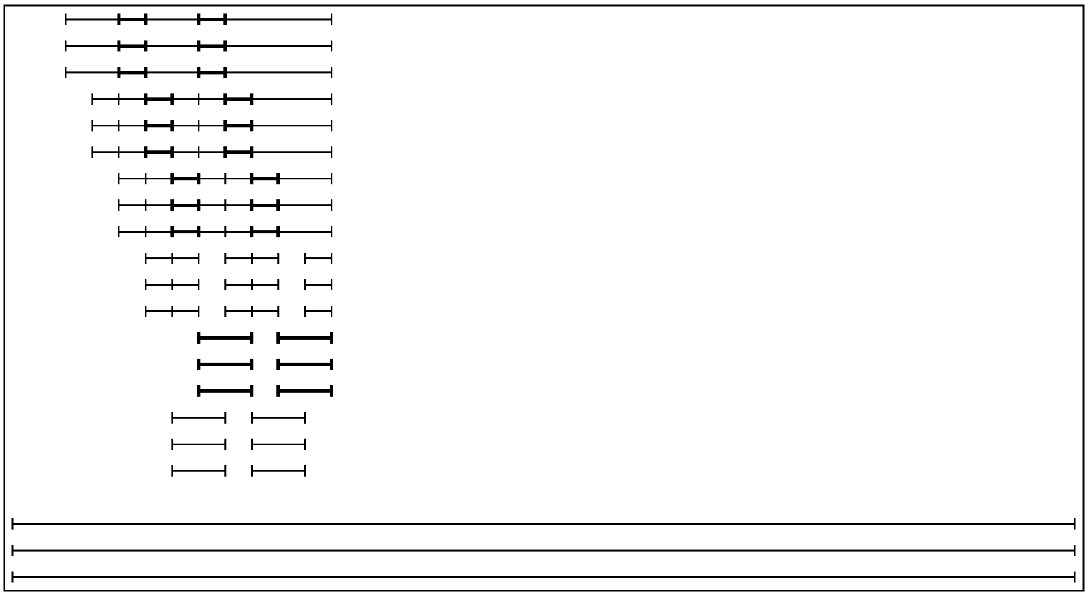


Figure 10: Placement by First-Fit* for $k = 41$, $n = 18$ and $s = 3$.

- $\frac{n}{3}$ times [1,k).

which can all be accepted by OPT, since the density is nowhere more than n. OPT will have a performance of $p\frac{2n}{3} + \frac{n}{3}(k-1)$. The performance ratio is at most $\frac{p(\frac{2n}{3})}{p(\frac{2n}{3})+\frac{n}{3}(k-1)} \leq \frac{2(s+5)}{2(s+5)+(k-1)} = \frac{2s+10}{k+2s+9}$. □

The theorem below extends a result in [6].

Theorem 19 For the proportional price problem, any Min-Change variant has a competitive ratio on accommodating sequences of $O(\frac{s}{k})$.

Proof The idea is to describe a sequence of intervals where all intervals end up within a block spanning four stations. After the block is given, the on-line algorithm is forced to perform (at least) one seat change to accepts an interval spanning the entire block. After $s+1$ disjoint blocks are given, the adversary will give the *long* interval $[1,k)$ a number of times. None of these long intervals can be accepted by the on-line algorithm, since the interval spans $s+1$ blocks all requiring the interval to change seat at least once within each block.

We describe the block spanning the stations 1 through 4. An adversary can give $\frac{n}{2}$ of each of the following intervals:

- [1,2)
- [3,4)

Let q denote the number of seats in the on-line algorithm's seating without any intervals. We divide the proof into two cases:

- Case 1: $q < \frac{n}{3}$.
- Case 2: $q \geq \frac{n}{3}$.

If Case 1 occurs, the adversary continues with q times the interval $[1,4)$. The on-line algorithm is forced to accept all of these intervals, each on its own seat. Moreover, due to the construction of the sequence, an additional interval $[1,4)$ cannot be accepted without performing a seat change. Note that for the sequence given so far, the density anywhere in the block is at most $\frac{n}{2} + q < \frac{5n}{6}$. For Case 1, the on-line algorithm accepts a total volume of $2\frac{n}{2} + 3q < 2n$.

If Case 2 occurs, the adversary continues with $\frac{n-q}{2}$ of each of the intervals $[1,3)$ and $[2,4)$. Now, the on-line algorithm has filled up all seats in the sense that no interval of size $[1,4)$ can be accepted without performing a seat change, and the density anywhere in the block is at most $\max(\frac{n}{2} + \frac{n-q}{2}, 2\frac{n-q}{2}) = n - \frac{q}{2} \leq \frac{5n}{6}$. The total volume accepted by the on-line algorithm is larger than in the previous case: $2\frac{n}{2} + 2\frac{n-q}{2}2 = 3n - 2q \leq \frac{7n}{3}$.

The above sequence is given for each of the four stations in $[3i+1, 3i+4)$ for $i = 0, 1, \ldots s$ each time translated such that $3i+1$ corresponds to station 1 above. This is clearly possible as long as $3s + 4 \leq k$.

Next, the adversary gives $\frac{n}{6}$ of the interval $[1,k)$ which must be rejected by the on-line algorithm as explained above, whereas the optimal algorithm can accept every one of them, since the maximal density of the entire sequence is n, no matter which case is used for each block. The performance ratio is at most $\frac{\frac{7n}{3}(s+1)}{\frac{n}{6}(k-1)} \leq \frac{14(s+1)}{k-1}$.

$\square$

It is unknown if First-Fit*'s competitive ratio on accommodating sequences is quite as bad as Min-Change-First-Fit's or Next-Fit's.

Theorem 20 For the proportional price problem, if $s \leq \frac{k-3}{2}$, the competitive ratio on accommodating sequences of First-Fit* is $O(\frac{s^2}{k})$.

Proof First assume that $s+3$ divides n. Let $p = 2s+7$ and let $r = \frac{1}{s+3}$. The adversary uses the sequence from Theorem 6 slightly modified (See Figure 10):

- rn times $[p-s-1, p)$, then rn times $[p-s, p)$, ... , then rn times $[p-1, p)$.
- For $i = 0, \ldots, \lfloor \frac{p-s-5}{3} \rfloor$, let $s' = p-s-2-3i$ and let $l = s+2$. Give rn times $[s', s'+l)$, $[s'-2, s'-2+l)$, and then $[s'-1, s'-1+l)$.

All of the intervals above are contained in $[1, p)$. For each i, three intervals are given, creating holes of size (at least) one at three new stations and increasing the number of connected components by one on each of the last $3rn$ seats. The number of stations with a hole of size at least one is therefore at least $3 \cdot \lfloor \frac{p-s-5}{3} + 1 \rfloor \geq p-s-5 = s+2$, by the choice of p. With at least $s+2$ holes arranged in this manner, any intervals spanning the entire region must be rejected. The adversary continues with

- rn times [1,k).

which can all be accepted by OPT, since the density is nowhere more than n. First-Fit* will have a performance of at most $rn\sum_{i=1}^{s+1} i + rn3(\lfloor \frac{p-s-5}{3} \rfloor + 1)(s+2) \leq rn\frac{(s+1)(s+2)}{2} + rn(2s+7-s-2)(s+2) = \frac{n}{2(s+3)}(s+2)(3s+11)$. OPT will have a performance of at least $rn(k-1) = \frac{n}{s+3}(k-1)$. The performance ratio is at most $\frac{\frac{n}{2(s+3)}(s+2)(3s+11)}{\frac{n}{s+3}(k-1)} = \frac{(s+2)(3s+11)}{2(k-1)}$.

If n is not divisible by $s+3$, we let $c = n \bmod (s+3)$. In this case, the performance ratio is

$$\begin{aligned}
\frac{c + \frac{n-c}{2(s+3)}(s+2)(3s+11)}{c + \frac{n-c}{s+3}(k-1)} &\leq \frac{c + \frac{n}{2(s+3)}(s+2)(3s+11)}{\frac{n-c}{s+3}(k-1)} \\
&\leq \frac{(s+2) + \frac{n}{2(s+3)}(s+2)(3s+11)}{\frac{n-(s+2)}{s+3}(k-1)} \\
&= \frac{(s+2)(3s+11) + \frac{2}{n}(s+3)(s+2)}{2(k-1) - \frac{2}{n}(s+2)(k-1)}.
\end{aligned}$$

□

4.2 Competitive Ratio

Now we turn to the standard competitive ratio and consider all possible request sequences. Since the results when restricting to accommodating sequences are so pessimistic, the results here are not dramatically worse.

4.2.1 General Hardness Result

Theorem 21 Any algorithm will have a competitive ratio of $O(\frac{s}{k})$ for the proportional price problem, even if OPT is conservative.

Proof Let $\mathbb{A}$ denote an arbitrary algorithm for the proportional price problem. The adversary gives

- $\frac{n}{2}$ times $[i, i+1)$ for $i = 1, \ldots, s+2$.

Let q be the number of times $[1, s+3)$ can be accepted by $\mathbb{A}$ after these intervals. We divide the proof into two cases:

- Case 1: $q \leq \frac{n}{4}$.
- Case 2: $q > \frac{n}{4}$.

In both cases the adversary gives

- q times $[1, s+3)$.

If Case 1 occurs, the adversary gives

- $\frac{n}{2} - q$ times $[1, k)$.

$\mathbb{A}$ must reject the long intervals. If OPT behaves like First-Fit, it will accept all the given items. The performance ratio will be at most $\frac{\frac{n}{2}(s+2)+q(s+2)}{\frac{n}{2}(s+2)+q(s+2)+(\frac{n}{2}-q)(k-1)} \leq \frac{\frac{n}{2}(s+2)+\frac{n}{4}(s+2)}{\frac{n}{2}(s+2)+\frac{n}{4}(s+2)+\frac{n}{4}(k-1)} = \frac{3(s+2)}{3(s+2)+(k-1)}$.

If Case 2 occurs, the adversary gives

- $\frac{n}{2} - q$ times $[2, s+3)$.
- q times $[2, k)$.

Also in this case $\mathbb{A}$ will accept everything except the long intervals. OPT should pack the first intervals as in Figure 8, reject the intervals $[1, s+3)$, and accept the remaining intervals. The performance ratio is $\frac{\frac{n}{2}(s+2)+q(s+2)+(\frac{n}{2}-q)(s+1)}{\frac{n}{2}(s+2)+(\frac{n}{2}-q)(s+1)+q(k-2)} \leq \frac{n(s+2)-\frac{n}{4}}{\frac{3n}{4}(s+1)+\frac{n}{4}(k-2)} = \frac{4(s+2)-1}{3(s+1)+k-2}$. □

5 Conclusion and Open Problems

For the unit price problem, there is a significant difference between the competitive ratio and the competitive ratio on accommodating sequence. For the latter, in order to find a difference due to more seat changes being allowed, conservative algorithms are considered, so that seat changes are only made when necessary. This seems to be a natural requirement for an algorithm and leads to improved competitive ratios on accommodating sequences with more allowed seat changes.

New algorithms have been discovered: We have discussed Min-Change-First-Fit and First-Fit* which were shown to have a competitive ratio on accommodating sequences of $\frac{s+1}{s+2}$ allowing s seat changes. Modified-Kierstead-Trotter was shown to seat the entire sequence on n seats when restricting to sequences which could be seated by an optimal off-line on $\frac{n}{2}$ seats. Moreover, for such sequences we showed that Modified-Kierstead-Trotter was strictly better than Min-Change-First-Fit, which was strictly better than the Checkerboard Algorithm.

For accommodating sequences, the best algorithm considered in this paper has a performance guarantee of $\frac{s+1}{s+2}$. It is an open problem whether there exists a better algorithm. Also a general hardness result is left open.

For all sequences, the hardness results for conservative algorithms hold for any n, whereas the hardness result for any algorithm holds for $n = 2$ only. We leave a general hardness result for all algorithms open.

Another open problem is to determine whether or not First-Fit is better than Min-Change-First-Fit for the proportional price problem when considering accommodating sequences.

6 Acknowledgments

We would like to thank Thore Husfeldt for suggesting the problem with one seat change, with the motivation that Sweden's train system allows it.

References

[1] Y. Azar, J. Boyar, L. M. Favrholdt, K. S. Larsen, and M. N. Nielsen. Fair versus Unrestricted Bin Packing. In *Proceedings of the 7th Scandinavian Workshop on Algorithm Theory*, volume 1851 of *Lecture Notes in Computer Science*, pages 200–213. Springer-Verlag, 2000.

[2] E. Bach, J. Boyar, L. Epstein, L. M. Favrholdt, T. Jiang, K. S. Larsen, G.-H. Lin, and R. van Stee. Tight bounds on the competitive ratio on accommodating sequences for the seat reservation problem. *Journal of scheduling.* To appear.

[3] E. Bach, J. Boyar, T. Jiang, K. S. Larsen, and G.-H. Lin. Better bounds on the accommodating ratio for the seat reservation problem. In *Proceedings of the Sixth Annual International Computing and Combinatorics Conference*, volume 1858 of *Lecture Notes in Computer Science*, pages 221–231. Springer-Verlag, 2000.

[4] J. Boyar, L. M. Favrholdt, K. S. Larsen, and M. N. Nielsen. Fair versus Unrestricted Bin Packing. *Nordic Journal of Computing*. To appear.

[5] J. Boyar, L. M. Favrholdt, K. S. Larsen, and M. N. Nielsen. Extending the Accommodating Function. Submitted, 2002.

[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. *SIAM Journal on Computing*, 31(1):233–258, 2001.

[8] Marek Chrobak and Maciej Ślusarek. On some packing problems relating to Dynamical Storage Allocation. *RAIRO Theoretical Informatics and Applications*, 22:487–499, 1988.

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

[10] T. R. Jensen and B. Toft. *Graph Coloring Problems*. John Wiley & Sons, 1995.

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

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

[13] M. M. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for on-line problems. In *CMGA: Graphs and Algorithms*. Contemporary Mathematics Volume 89, AMS, 1989, 1989.

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