Extending the Accommodating Function [*][†]

Joan Boyar, Lene M. Favrholdt, Kim S. Larsen[‡] and Morten N. Nielsen
University of Southern Denmark, Odense, Denmark
{joan,lenem,kslarsen,nyhave}@imada.sdu.dk

Abstract

The applicability of the accommodating function, a relatively new measure for the quality of on-line algorithms, is extended.

The standard quality measure for on-line algorithms is the competitive ratio, which is, roughly speaking, the worst case ratio of the on-line performance to the optimal off-line performance. However, for many on-line problems, the competitive ratio gives overly pessimistic results and/or fails to distinguish between algorithms that are known to perform very differently in practice. Many researchers have proposed variations on the competitive ratio to obtain more realistic results. These variations are often tailor-made for specific on-line problems.

The concept of the accommodating function applies to any on-line problem with some limited resource, such as bins, seats in a train, or pages in a cache. If a limited amount n of some resource is available, the accommodating function $\mathcal{A}(\alpha)$ is the competitive ratio when input sequences are restricted to those for which the amount αn of resources suffices for an optimal off-line algorithm. For all resource bounded problems, the standard competitive ratio is $\lim_{\alpha\to\infty}\mathcal{A}(\alpha)$.

The accommodating function was originally used only for $\alpha \geq 1$. We focus on $\alpha < 1$, observe that the function now appears interesting for a greater variety of problems, and use it to make new distinctions between known algorithms and to find new ones.

1 Introduction

An algorithm is said to be on-line if it receives its input in small pieces and handles each piece without any knowledge about the rest of the input. The standard quality measure for on-line algorithms is the competitive ratio which is similar to the approximation ratio for approximation algorithms, i.e., the profit/cost of the on-line solution is measured relative to the best possible solution that could be obtained, if the whole input was known from the beginning. However, often the difference in power between on-line and off-line algorithms seems to be too large for this to give meaningful results; the competitive ratio often gives results that are overly pessimistic and may fail to distinguish between very different algorithms. For

[*]A preliminary version of this paper appeared in the Eighth Annual International Computing and Combinatorics Conference, Lecture Notes in Computer Science 2387: 87–96, Springer-Verlag, 2002.

[†]Supported in part by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT) and in part by the Danish Natural Science Research Council (SNF).

[‡]Corresponding author. E-mail: kslarsen@imada.sdu.dk. Fax: +45 6593 2691.

this reason many variations on the competitive ratio have been proposed, some of which were introduced in [4, 6, 14, 15, 19, 21, 23].

While some of the previous variations are tailor made for specific problems, the accommodating function applies to any problem with a limited amount n of some resource. Informally, $\mathcal{A}(\alpha)$ is the competitive ratio when input sequences are restricted to those for which an optimal off-line algorithm does not benefit from having more than the amount αn of resources. These sequences are called α-sequences. We refer to on-line problems for which any input sequence is an α-sequence for some α as *resource bounded.* For such problems, the limit $\lim_{\alpha\to\infty} \mathcal{A}(\alpha)$ is the standard competitive ratio. All problems considered in this paper are resource bounded. The accommodating function was recently defined in [8], and it was applied to various problems in [8] and [1], but only for $\alpha \geq 1$. In this paper, values of $\alpha < 1$ are considered for the first time. The accommodating function is formally defined in Section 2.

1.1 Motivation and Background

The original motivation for considering this type of restriction of request sequences is from the Seat Reservation Problem [7], the problem of assigning seats to passengers in a train on-line, in a "fair" manner, to maximize earnings. For the unit price version, the competitive ratio for this problem is $\Theta(\frac{1}{k})$, where k is the number of stations where the train stops. This very discouraging performance cannot occur, however, for realistic request sequences. Based on data from previous years, the management is often able to judge approximately how many cars are necessary to accommodate all requests (given the entire set of seat reservations this is an easy problem). Hence, it is more realistic to consider only request sequences that can be fully accommodated by an optimal off-line algorithm. Such sequences, corresponding to $\alpha = 1$, are called *accommodating sequences.* For the unit price problem, $\mathcal{A}(1) \geq \frac{1}{2}$ [7].

The idea of restricting the adversary to only giving accommodating sequences carries over to any optimization problem with some limited resource, such as the seats in a train. Thus, for instance, it can be used on the k-Server Problem, where the servers constitute a limited resource since there are only k of them, or scheduling problems, where there is only a fixed number of machines.

In addition to giving rise to new interesting algorithmic and analytical problems, the accommodating function, compared to just one ratio, contains more information about the on-line algorithms. This information can be exploited in several ways. The shape of the function, for instance, can be used to warn against critical scenarios, where the performance of the on-line algorithm compared to the off-line can suddenly drop rapidly when fewer resources are available.

In [8], a variant of bin packing is investigated, for which the number of bins is fixed and the goal is to maximize the number of items packed. The bins have height k and the items are integer sized. It is shown that, in general, Worst-Fit has a strictly better competitive ratio than First-Fit, while, in the special case of accommodating sequences, First-Fit has a strictly better competitive ratio than Worst-Fit. In this case, the competitive ratio on accommodating sequences seems the more appropriate measure, since it is constant while, in the general case, the competitive ratio is $\Theta(\frac{1}{k})$, basically due to some sequences which seem very contrived. This shows that in addition to giving more realistic performance measures for some problems, the competitive ratio on accommodating sequences can be used to distinguish between algorithms, showing, not surprisingly, that the decision as to which algorithm should be used depends on what sort of request sequence is expected. The obvious question at this point was: Where is the cross-over point? When does First-Fit become better than Worst-Fit? This is another motivation for considering the accommodating function.

In [8], the accommodating function was investigated for $\alpha \geq 1$, spanning from the one known interesting case (accommodating sequences) to the other (the standard competitive ratio). In this paper we extend the definition of the accommodating function to include $\alpha < 1$.

1.2 Results

As example problems, we first consider two maximization problems. For Unrestricted Bin Packing, we analyze First-Fit and Best-Fit and investigate a new variant of Unfair-First-Fit [1], called Unfair-First-Fit$_\alpha$. In general, we do not assume that α is known to the algorithm, but Unfair-First-Fit$_\alpha$ is designed for the case where a good upper bound on α is known. This algorithm turns out to be better than First-Fit for α close to 1.

For Seat Reservation, we consider three deterministic algorithms, which asymptotically have the same competitive ratio on accommodating sequences, and show that the performance of these algorithms can be separated using the proposed extension of the accommodating function.

Finally, we consider well known on-line minimization problems to emphasize the broad applicability of the accommodating function for $\alpha < 1$.

1.2.1 The Accommodating Function for Two Maximization Problems

For the Seat Reservation Problem, considering the accommodating function for $\alpha < 1$ corresponds to the situation where the management has provided more cars than needed by an optimal off-line algorithm to accommodate all passenger requests. This seems to be a desirable situation, since the only way the train company can hope to accommodate all requests on-line is by having more resources than would be necessary if the problem was solved optimally off-line.

In this paper, we analyze the accommodating function for the problem in general and for First-Fit and Worst-Fit in particular. The results for First-Fit and Worst-Fit are depicted in Figure 1 to the left of the line $\alpha = 1$. To the right of this line, general results from [8] are shown.

Although the competitive ratio on accommodating sequences fails to distinguish between fair algorithms for the Seat Reservation Problem (they all have ratio $\frac{1}{2}$ in the limit), we investigate the accommodating function at $\alpha = \frac{1}{3}$ for three different algorithms and discover that Worst-Fit is the worst there, First-Fit is better, and an algorithm due to Kierstead and Trotter is optimal at $\alpha = \frac{1}{3}$.

We also consider another maximization problem, a variant of bin packing called Unrestricted Bin Packing. We analyze the accommodating function for the problem in general and for First-Fit and Best-Fit in particular. The results for First-Fit are depicted in Figure 2 to the left of the line $\alpha = 1$. The results to the right of the line are from [8]. Moreover, we show how performance guarantees for accommodating sequences can be extended in a fairly general

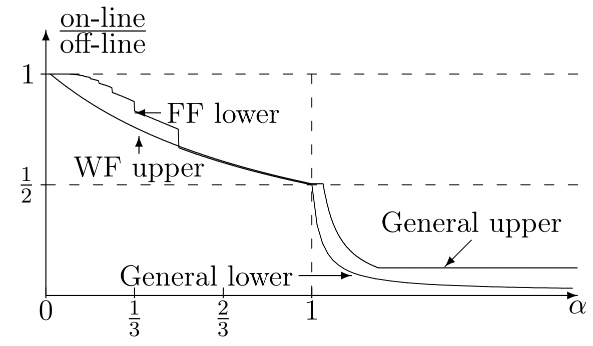

Figure 1: Bounds on $\mathcal{A}(\alpha)$ for the Seat Reservation Problem.

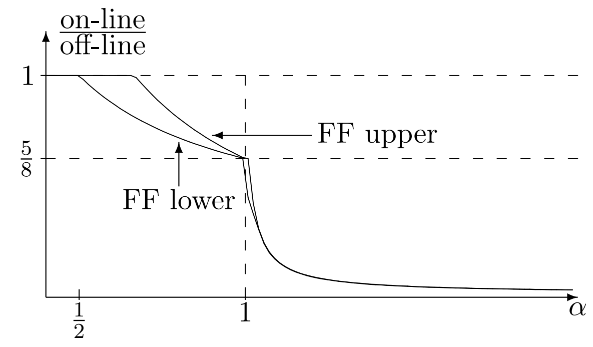

Figure 2: Bounds on $\mathcal{A}_{\mathrm{FF}}(\alpha)$ for Unrestricted Bin Packing.

manner to better bounds on the competitive ratio for more restricted sets of input sequences. Thus, accommodating sequences play a more important role than originally anticipated. One can also see from the graphs in Figures 1 and 2 that accommodating sequences have unique properties; the shape of the curve seems to often change significantly at $\alpha = 1$. In addition, for Unrestricted Bin Packing, we investigate an algorithm called Unfair-First-Fit$_\alpha$, which is shown to perform significantly better than First-Fit given that a good bound on α is known in advance.

1.2.2 The Connection to Resource Augmentation

Resource augmentation is another technique which is used to give more realistic results when the standard competitive ratio seems too negative [14]. With resource augmentation analysis, the on-line algorithm is given more resources than the off-line algorithm, but the performance ratio is still the worst case over all request sequences. There is clearly a similarity between resource augmentation and the accommodating function with $\alpha < 1$. However, for some problems there is a tremendous difference between the results one gets with the two techniques. In these cases, the accommodating function gives the more positive results, because there can exist sequences where the off-line algorithm could have done somewhat better with more resources (and thus would not be considered in the accommodating function analysis), while the on-line algorithm is unable to fully take advantage of its extra resources.

As an example of this phenomenon, for $\alpha \leq 1$, we prove a performance guarantee for the algorithm First-Fit for the Seat Reservation Problem, $\mathcal{A}_{\text{FF}}(\alpha) \geq 1 - \frac{1}{2^\ell}$, where $\ell = \lfloor \frac{1}{\alpha} \rfloor$, but even with $\frac{1}{\alpha}$ times as many seats as the off-line algorithm, First-Fit's competitive ratio can be as bad as $\frac{1+\alpha}{(\alpha-2/n)(k-1)}$; for fixed α, the ratio is $\Theta(\frac{1}{k})$.

For Unrestricted Bin Packing, one can also show that resource augmentation analysis gives results which are much more pessimistic than the corresponding accommodating function results. We prove a performance guarantee for First-Fit for $\frac{1}{2} \leq \alpha \leq 1$ of $\mathcal{A}_{\text{FF}}(\alpha) \geq \frac{3+2\alpha}{8\alpha}$, while the competitive ratio of First-Fit is at most $\frac{1}{\alpha k}$, even when First-Fit has $\frac{1}{\alpha}$ times as many bins as the off-line algorithm (again k is the height of the bins).

On the other hand, there are many cases where results or proofs from resource augmentation can be used to calculate the accommodating function for $\alpha < 1$. Performance guarantees about algorithms analyzed using resource augmentation directly give performance guarantees for the accommodating function. Impossibility results can also be applied if the original adversary only used accommodating sequences. We illustrate this with three minimization problems: the Paging Problem, the k-Server Problem, and Machine Scheduling (minimizing the makespan). The clear intuitive applicability of the accommodating function when the actual request sequences are expected to have the proper form makes these results and proofs from resource augmentation even more interesting than previously.

2 The Accommodating Function

We now define the accommodating function formally.

Definition 1 A *resource dependent problem* is an an on-line optimization problem P, where each problem instance, in addition to the input data given on-line, also has a parameter n, referred to as the amount of resources, such that for each input, the optimal solution depends monotonically on n.

We let OPT denote an optimal off-line algorithm for the resource dependent problem P. For any algorithm $\mathbb{A}$ and any input sequence I for P, $\mathbb{A}(I)$ denotes the profit/cost of running $\mathbb{A}$ on I, and $\text{OPT}_{n'}(I)$ denotes the profit/cost of OPT when the amount n' of resources is available.

For a given resource parameter n and any $\alpha > 0$, an input sequence I is said to be an *α-sequence*, if

$$\text{OPT}_{\alpha n}(I) = \text{OPT}_{n'}(I), \text{ for all } n' \geq \alpha n.$$

1-sequences are also called *accommodating sequences.*

A resource dependent problem is called *resource bounded* if every input sequence is an α-sequence for some α. □

If an input sequence is an α-sequence, then OPT does not benefit from having more than the amount αn of resources. In particular, if an input sequence is an accommodating sequence, then OPT does not benefit from having more resources than the amount already available. For many problems, the amount n of the resource is always a natural number. In that case, one should only consider values of α such that $\alpha n \in \mathbb{N}$. If not, we define $\mathcal{A}(\alpha)$ to have the same value as $\mathcal{A}(\frac{\lfloor \alpha n \rfloor}{n})$. Thus, $\mathcal{A}$ becomes a step function.

Definition 2 Let $\mathbb{A}$ be an on-line algorithm for a resource dependent maximization (minimization) problem. Then $\mathbb{A}$ is *c-competitive on α-sequences*, if there exists a constant b, such that

$$\mathbb{A}(I) \geq c \cdot \mathrm{OPT}(I) - b \quad \big(\mathbb{A}(I) \leq c \cdot \mathrm{OPT}(I) + b\big),$$

for any α-sequence I. The *accommodating function* is defined as

$$\mathcal{A}_{\mathbb{A}}(\alpha) = \sup\{c \mid \mathbb{A} \text{ is } c\text{-comp on } \alpha\text{-seq}\} \; (\mathcal{A}_{\mathbb{A}}(\alpha) = \inf\{c \mid \mathbb{A} \text{ is } c\text{-comp on } \alpha\text{-seq}\}).$$

□

Thus, for maximization problems, $\mathcal{A}(\alpha) \leq 1$, and for minimization problems, $\mathcal{A}(\alpha) \geq 1$.

The condition that the problems should be resource bounded is necessary to ensure that the limit of the competitive ratio on α-sequences for $\alpha \to \infty$ is the usual competitive ratio, since it is possible to create (very artificial) problems which are not resource bounded and where the limit is not the competitive ratio.

The condition implies that for every input sequence, when we increase the amount of resources which can be used, then at some point we have enough, so that extra resources will no longer improve the result. This seems very natural, and we have not found any natural problems which are not resource bounded. It is very easy to verify that all problems in this paper are resource bounded. As an example, for unrestricted bin packing, if we have an input sequence of length n, then when we allow n bins as resources, certainly no items will be rejected, so more bins cannot improve the result.

To illustrate the problem without the resource boundedness, consider the following artificial example:

Example 1 In this problem, we have one bin of size n. The size of this bin is our resource. The input is a sequence of items which have a size and a value. The objective is to maximize the total value.

There are two types of items, called ONE and STRETCH. ONE has size 1 and value 1. The size of STRETCH is all remaining non-zero space in the bin and its value is $1 - 1/n$.

Since the value of STRETCH grows with the bin size, OPT will always do better with a larger bin. This means that STRETCH cannot belong to any α-sequence, and therefore the problem is not resource bounded.

For this problem, the competitive ratio of any greedy algorithm on α-sequences is 1. However, the competitive ratio on all sequences is strictly smaller than 1. This is seen as follows: Give ONE and then STRETCH. If STRETCH is rejected, then stop. The ratio is $1/(2 - 1/n) < 1$ (here, OPT takes STRETCH). If STRETCH is accepted, continue by giving another ONE. The ratio is $(2 - 1/n)/2 < 1$ (here, OPT rejects STRETCH).

It is very easy to verify that all problems considered in this paper are resource bounded, as the example with the bins shows, and we will not mention this technicality again.

3 Unrestricted Bin Packing

In this section, we consider a maximization variant of Bin Packing, called Unrestricted Bin Packing, in which the objective is to maximize the number of items packed within n bins all of the same size $k \in \mathbb{N}$. An input sequence consists of items, the sizes of which are integers between 1 and k. The items are given one by one and an on-line algorithm must pack each item before the next item arrives. An α-sequence can be packed (by an optimal off-line algorithm) within αn bins.

We use the following notation for a given $\alpha \leq 1$. Assume that we have a numbering of the bins from 1 to n, according to the order in which they are taken into use. The first αn bins, denoted B_{I}, are called *internal* bins, and the remaining $n - \alpha n$ bins, denoted B_{X}, are called *external* bins. Whenever we consider a fixed request sequence, we let A denote the set of items accepted, i.e., packed in one of the n bins, by the on-line algorithm, and divide A into the set A_{I} of items packed in B_{I} and the set A_{X} of items packed in B_{X}. The set R contains the remaining (rejected) items. We sometimes use B'_{X} to denote the set of nonempty external bins.

Some well-known bin packing algorithms are First-Fit (FF), Best-Fit (BF), and Worst-Fit (WF). First-Fit packs each item in the first bin it fits in, and Best-Fit chooses a most full bin in which the item fits. Worst-Fit packs each item in a most empty bin, meaning that the first n items are packed in separate bins. The bounds on First-Fit found in this section are depicted in Figure 2 in Section 1.

3.1 Performance Guarantees

Definition 3 An algorithm for Unrestricted Bin Packing is called *fair* if it never rejects an item that it is able to pack. □

In this section, we give a performance guarantee for the class of fair algorithms and better guarantees for three specific algorithms, two of which are fair. In fact, the guarantees for the three algorithms are corollaries of a more general theorem extending results for accommodating sequences ($\alpha = 1$) to the α-sequences where α is less than 1.

Theorem 1 Any fair algorithm $\mathbb{A}$ for Unrestricted Bin Packing has

$$\mathcal{A}_{\mathbb{A}}(\alpha) \geq \begin{cases} \dfrac{1}{1+\alpha-\frac{1}{k}}, & \frac{1}{k} \leq \alpha \leq 1 \\ 1, & \alpha \leq \frac{1}{k} \end{cases}$$

Proof If $\alpha \leq \frac{1}{k}$, any α-sequence has at most n items, and hence any fair algorithm will pack all items of an α-sequence. This gives $\mathcal{A}(\alpha) = 1$, for $\alpha \leq \frac{1}{k}$.

Assume now that $\frac{1}{k} \leq \alpha \leq 1$ and let I be an arbitrary α-sequence. Let s denote the size of a smallest item in R. Since $\mathbb{A}$ is fair, the empty space in any bin is at most $s - 1$, meaning that the items accepted have a total size of at least $n(k - s + 1)$. Since I is an α-sequence, the total size of the items in I is at most αnk. Thus,

$$\begin{aligned}|R| &\le \frac{1}{s}(\alpha nk - n(k-s+1)) \\ &\le \frac{1}{s}(\alpha nk - \alpha n(k-s) - n), \text{ since } \alpha \le 1 \\ &= \alpha n - \frac{n}{s} \le \alpha n - \frac{n}{k}, \text{ since } s \le k\end{aligned}$$

If $|R| = 0$, the result of $\mathbb{A}$ is optimal. If $|R| > 0$, $|A| \ge n$ since $\mathbb{A}$ is fair. Therefore,

$$\frac{|A|}{|A|+|R|} \ge \frac{n}{n+\alpha n - \frac{n}{k}} = \frac{1}{1+\alpha-\frac{1}{k}}.$$

□

In Section 3.2, this performance guarantee is shown to be asymptotically tight due to the behavior of Worst-Fit.

The proof of Theorem 1 shows that fair algorithms will reject fewer than αn items. When considering First-Fit, a stronger fact can be shown, namely that αn is an upper bound on the number of items not packed in the internal bins, i.e., $|A_{\mathrm{X}}| + |R| < \alpha n$. Furthermore, for any α-sequence with $\alpha \le 1$, First-Fit packs at least the fraction $\mathcal{A}_{\mathrm{FF}}(1)$ of the items in the internal bins, and if it rejects an item no bin is empty. Similar or slightly weaker results can be shown for a broader class of algorithms. For such algorithms, Theorem 2 may be used to extend performance guarantees obtained on accommodating sequences to α-sequences with $\alpha < 1$. When reading Theorem 2, recall that B'_{X} denotes the set of nonempty external bins.

Theorem 2 Suppose that, for any $\alpha \le 1$, the following three conditions hold for an algorithm $\mathbb{A}$ for Unrestricted Bin Packing.

1. There exist constants β and b such that, for any α-sequence,

$$|A_{\mathrm{I}}| \ \ge\ \beta(|A_{\mathrm{I}}| + |B'_{\mathrm{X}}| + |R|) - b.$$

2. Whenever $|R| > 0$, no external bin is empty, i.e.,

$$|R| > 0 \ \Rightarrow\ |B'_{\mathrm{X}}| = (1-\alpha)n.$$

3. For any α-sequence, $|B'_{\mathrm{X}}| + |R| \le \alpha n$.

Then,

$$\mathcal{A}_{\mathbb{A}}(\alpha) \ge \begin{cases} 1 - (1-\beta)\left(2 - \dfrac{1}{\alpha}\right), & \dfrac{1}{2} \le \alpha \le 1 \\ 1, & \alpha \le \dfrac{1}{2} \end{cases}$$

Proof Assume that $|R| > 0$, because otherwise $\mathbb{A}$'s packing is optimal. Then, by Conditions 2 and 3, $(1-\alpha)n + |R| \le \alpha n$, which implies that $\alpha > \frac{1}{2}$. Hence, for $\alpha \le \frac{1}{2}$, $\mathcal{A}_{\mathbb{A}}(\alpha) = 1$.

We assume now that $|R| > 0$ and $\frac{1}{2} \leq \alpha \leq 1$ and split the remaining part of the proof in two cases depending on the size of A_{I} compared to β and αn.

Case 1: $|A_I| \geq \frac{\beta}{1-\beta}\alpha n$. In this case, the result follows from Conditions 2 and 3 alone:

$$\begin{aligned}
\frac{|A_{\mathrm{I}}|+|A_{\mathrm{X}}|}{|A_{\mathrm{I}}|+|A_{\mathrm{X}}|+|R|} &\geq \frac{|A_{\mathrm{I}}|+|B'_{\mathrm{X}}|}{|A_{\mathrm{I}}|+|B'_{\mathrm{X}}|+|R|} \\
&\geq \frac{|A_{\mathrm{I}}|+(1-\alpha)n}{|A_{\mathrm{I}}|+\alpha n}, && \text{by Conditions 2 and 3} \\
&\geq \frac{\beta\alpha n+(1-\beta)(1-\alpha)n}{\beta\alpha n+(1-\beta)\alpha n}, && \text{since } |A_{\mathrm{I}}| \geq \frac{\beta\alpha n}{1-\beta} \\
&= \frac{\beta\alpha+(1-\beta)(1-\alpha)}{\alpha} \\
&= 1-(1-\beta)\left(2-\frac{1}{\alpha}\right)
\end{aligned}$$

Case 2: $|A_I| < \frac{\beta}{1-\beta}\alpha n$.

$$\begin{aligned}
\frac{|A_{\mathrm{I}}|+|A_{\mathrm{X}}|}{|A_{\mathrm{I}}|+|A_{\mathrm{X}}|+|R|} &\geq \frac{|A_{\mathrm{I}}|+|B'_{\mathrm{X}}|}{|A_{\mathrm{I}}|+|B'_{\mathrm{X}}|+|R|} \\
&\geq \beta-\varepsilon+\frac{|B'_{\mathrm{X}}|}{|A_{\mathrm{I}}|+|B'_{\mathrm{X}}|+|R|}, && \text{by Condition 1 with } \varepsilon=\frac{b}{|A_{\mathrm{I}}|+|B'_{\mathrm{X}}|+|R|} \\
&\geq \beta-\varepsilon+\frac{(1-\alpha)n}{|A_{\mathrm{I}}|+\alpha n}, && \text{by Conditions 2 and 3} \\
&> \beta-\varepsilon+\frac{(1-\beta)(1-\alpha)}{\beta\alpha+(1-\beta)\alpha}, && \text{since } |A_{\mathrm{I}}| < \frac{\beta\alpha n}{1-\beta} \\
&= \beta+\frac{(1-\beta)(1-\alpha)}{\alpha}-\varepsilon \\
&= 1-(1-\beta)\left(2-\frac{1}{\alpha}\right)-\varepsilon
\end{aligned}$$

□

The constant b in the first condition of Theorem 2 is actually not needed for any of the results in this paper; it is there to make the theorem more generally applicable with $\beta = \mathcal{A}_{\mathbb{A}}(I)$.

The first corollaries to the theorem concern the fair algorithms First-Fit and Best-Fit. They extend results from [8] saying that the two algorithms are both $\frac{5}{8}$-competitive on accommodating sequences.

Corollary 1 For Unrestricted Bin Packing, $\mathcal{A}_{\mathrm{FF}}(\alpha) \geq \begin{cases} \dfrac{3+2\alpha}{8\alpha}, & \frac{1}{2} \leq \alpha \leq 1 \\ 1, & \alpha \leq \frac{1}{2}. \end{cases}$

Proof Let $\alpha \leq 1$ and consider any α-sequence I. Assume that First-Fit is given only αn bins, and let A_α be the set of items packed by First-Fit in these αn bins. By [8], $\mathcal{A}_{\mathrm{FF}}(1) \geq \frac{5}{8}$ (and the additive constant in the definition of the competitive ratio is not used in the proof).

Hence, $|A_\alpha| \geq \frac{5}{8} \cdot |I|$. Assume now that First-Fit is given the same sequence I and n bins to pack it in. Since each item is packed in the first bin in which it fits, exactly the items in A_α will be packed in B_{I}. Hence, the first condition is satisfied with $\beta = \frac{5}{8}$.

Condition 2 holds since First-Fit is a fair algorithm.

None of the items in $A_{\text{X}} \cup R$ fit in B_{I}, i.e., $|A_{\text{X}}| + |R| \geq |B'_{\text{X}}| + |R|$ items do not fit in any of the αn internal bins. By the same argument as in the proof of Theorem 1, this means that Condition 3 is satisfied. □

Corollary 2 For Unrestricted Bin Packing, $\mathcal{A}_{\text{BF}}(\alpha) \geq \begin{cases} \dfrac{3+2\alpha}{8\alpha}, & \frac{1}{2} \leq \alpha \leq 1 \\ 1, & \alpha \leq \frac{1}{2}. \end{cases}$

Proof Again, we just need to verify the three conditions in Theorem 2.

Consider any α-sequence I and let A'_{X} be the set of items in I which Best-Fit places as the first item in some external bin. No item in $A'_{\text{X}} \cup R$ fits in B_{I} at the time it is given. The subsequence $f(I)$ of I consisting of the items in $A_{\text{I}} \cup A'_{\text{X}} \cup R$ is clearly also an α-sequence, and given the subsequence $f(I)$ and only αn bins, Best-Fit will accept exactly those items in A_{I}. This shows that the first condition is satisfied with $\beta = \mathcal{A}_{\text{BF}}(1)$. By [8], $\mathcal{A}_{\text{BF}}(1) \geq \frac{5}{8}$ (and the additive constant in the definition of the competitive ratio is not used in the proof).

Condition 2 holds because Best-Fit is fair.

Condition 3 holds because none of the items in $A'_{\text{X}} \cup R$ fit in B_{I}. □

The third corollary to the theorem above concerns the following slight variation of the algorithm Unfair-First-Fit [1]. Unfair-First-Fit behaves just like First-Fit unless the given item has size larger than $\frac{k}{2}$. In this case, the item is rejected on purpose, if the number of currently accepted items is at least $\frac{2}{3}$ of the number of items in the entire sequence seen so far. The new algorithm given in Figure 3, called Unfair-First-Fit$_\alpha$ (UFF$_\alpha$), assumes that α is known in advance. Using this knowledge, the algorithm divides the bins into B_{I} and B_{X}. Note that the ratio considered in the if-statement of the algorithm is the number of items in the internal bins, compared to all items given. If this ratio is at least $\frac{2}{3}$, the item is rejected if it does not fit in an external bin.

```
Input: $S = \langle o_1, o_2, \ldots, o_n \rangle$
while $S \neq \langle\rangle$
  $o := \text{hd}(S); S := \text{tail}(S)$
  if $\text{size}(o) \leq \frac{k}{2}$ or $\frac{|A_{\text{I}}|}{|A_{\text{I}}|+|A_{\text{X}}|+|R|+1} < \frac{2}{3}$
    "Accept" - Try to pack $o$ using First-Fit
  else
    "Reject" - Try to pack $o$ in $B_{\text{X}}$ using First-Fit
  Update $A_{\text{I}}, A_{\text{X}}, R$ accordingly
```

Figure 3: The algorithm Unfair-First-Fit$_\alpha$

The actions "Accept" and "Reject" should be understood with respect to the internal bins, in that they distinguish between whether the internal bins are considered or not. In the "Accept" case, the item is placed in the internal bins according to the First-Fit packing rule, if there is enough space. If there is not enough space in any of the internal bins, the First-Fit rule is tried against the external bins. If there is not enough space in either an internal or an external bin, the item is rejected (added to R). The "Reject" case is similar, but the internal bins are not considered.

If a good upper bound α' on α is known, $\mathrm{UFF}_{\alpha'}$ can be used, giving a performance guarantee of $\mathcal{A}_{\mathrm{UFF}_{\alpha'}}(\alpha')$.

Corollary 3 For $\alpha n \geq 9$,

$$\mathcal{A}_{\mathrm{UFF}_\alpha}(\alpha) \geq \begin{cases} \dfrac{1+\alpha}{3\alpha} - \varepsilon, & \dfrac{1}{2} \leq \alpha \leq 1, \quad \text{where } \varepsilon = \dfrac{4\alpha - 2}{(4n+1)\alpha} < \dfrac{1}{n} \\ 1, & \alpha \leq \dfrac{1}{2} \end{cases}$$

Proof Let $\alpha \leq 1$ and consider any α-sequence I and the algorithm Unfair-First-Fit. Assume that Unfair-First-Fit is given only αn bins, and let A_α be the set of items packed by Unfair-First-Fit in these αn bins. By [1], $\mathcal{A}_{\mathrm{UFF}}(1) \geq \frac{2}{3} - \frac{2}{4n+1}$ (and the additive constant in the definition of the competitive ratio is not used in the proof). Hence, $|A_\alpha| \geq \beta|I|$, where $\beta = \frac{2}{3} - \frac{2}{4n+1}$. Assume now that UFF_α is given the same sequence I and n bins to pack it in. By the order in which the sets B_{I}, B_{X}, and R are tried, UFF_α packs exactly the items in A_α in B_{I}. Hence, the first condition of Theorem 2 is satisfied with $\beta = \frac{2}{3} - \frac{2}{4n+1}$.

An item is not rejected if it fits in B_{X}, which proves the second condition.

To show the third condition, let s denote the size of a smallest item in $A_X \cup R$. If this smallest item was packed by an "Accept" attempt, every bin in B_{I} has empty space less than s. Since all items could be packed in B_{I}, this means that $|B_{\mathrm{X}}| + |R| < \alpha n$. If the smallest item was packed by a "Reject" attempt, this item has size larger than $\frac{k}{2}$, which then all items in $A_X \cup R$ must have. Since no two items of this size can be packed together, there are at most αn such items.

Finally, assuming $\alpha \geq \frac{1}{2}$, we derive the bound

$$\begin{aligned} \mathcal{A}_{\mathrm{UFF}_\alpha}(\alpha) &\geq 1 - \left(1 - \left(\frac{2}{3} - \frac{2}{4n+1}\right)\right)\left(2 - \frac{1}{\alpha}\right) = 1 - \left(\frac{1}{3} + \frac{2}{4n+1}\right)\left(2 - \frac{1}{\alpha}\right) \\ &= 1 - \left(\frac{2}{3} + \frac{4}{4n+1} - \frac{1}{3\alpha} - \frac{2}{(4n+1)\alpha}\right) = \frac{1+\alpha}{3\alpha} - \frac{4\alpha - 2}{(4n+1)\alpha}. \end{aligned}$$

$\square$

3.2 Impossibility Results

The theorem below extends a result in [8] and it shows that any (even randomized) on-line algorithm $\mathbb{A}$ for Unrestricted Bin Packing has $\mathcal{A}_{\mathbb{A}}(\alpha) < 1$, for $\alpha > \frac{4}{5}$.

Theorem 3 Let $\alpha \leq 1$ and $k \geq 5$. Then, for any on-line algorithm $\mathbb{A}$ for Unrestricted Bin Packing,

$$\mathcal{A}_{\mathbb{A}}(\alpha) \leq \begin{cases} \dfrac{2\alpha+4}{7\alpha}, & \text{if } \alpha n \text{ is even.} \\ \dfrac{2\alpha+4}{7\alpha} + O\left(\dfrac{1}{n}\right), & \text{otherwise.} \end{cases}$$

Proof Note that, for $\alpha \leq \frac{4}{5}$, the bound is larger than 1. Hence, we may safely assume that $\alpha \geq \frac{4}{5}$. The adversary gives two phases of items.

In the first phase, αn items of size $\lfloor \frac{k-1}{2} \rfloor$ are given. Since $k \geq 5$, at most two such items can be packed in one bin. Let q denote the fraction of bins containing two items in $\mathbb{A}$'s packing. Depending on the expected value of q, $E[q]$, the next phase is different.

Case 1: $E[q] \leq \frac{3\alpha-1}{7}$. The adversary gives $\lfloor \frac{\alpha n}{2} \rfloor$ items of size k. By linearity of expectation,

$$E\left[\frac{|A|}{|A|+|R|}\right] \leq \frac{n+E[q]n}{\lfloor \frac{3}{2}\alpha n \rfloor} .$$

If αn is even, this reduces to

$$E\left[\frac{|A|}{|A|+|R|}\right] \leq \frac{2+2E[q]}{3\alpha} \leq \frac{14+2(3\alpha-1)}{21\alpha} = \frac{2\alpha+4}{7\alpha} .$$

Otherwise, we get

$$\begin{aligned} E\left[\frac{|A|}{|A|+|R|}\right] &\leq \frac{n+E[q]n}{\frac{3}{2}\alpha n - \frac{1}{2}} = \frac{2n+2E[q]n}{3\alpha n - 1} \leq \frac{14n+2(3\alpha-1)n}{21\alpha n - 7} = \frac{6\alpha n + 12n}{21\alpha n - 7} \\ &= \frac{(6\alpha n + 12n)(1+\frac{7}{21\alpha n - 7})}{(21\alpha n - 7)(1+\frac{7}{21\alpha n-7})} = \frac{(6\alpha n+12n)(1+\frac{1}{3\alpha n-1})}{21\alpha n} \\ &= \frac{6\alpha+12}{21\alpha} + \frac{6\alpha+12}{21\alpha(3\alpha n-1)} = \frac{2\alpha+4}{7\alpha} + \frac{2+4/\alpha}{21\alpha n-7} \\ &\leq \frac{2\alpha+4}{7\alpha} + \frac{5}{12n-5} , \text{ since } \alpha \geq \frac{4}{5} . \end{aligned}$$

Case 2: $E[q] > \frac{3\alpha-1}{7}$. The adversary gives αn items of size $\lceil \frac{k+1}{2} \rceil$. In this case, $\mathbb{A}$ accepts at most $\alpha n + n - qn$. Thus,

$$E\left[\frac{|A|}{|A|+|R|}\right] \leq \frac{\alpha n + n - E[q]n}{2\alpha n} < \frac{7\alpha+7-(3\alpha-1)}{14\alpha} = \frac{2\alpha+4}{7\alpha} .$$

□

The following theorem extends an impossibility result for First-Fit on accommodating sequences in [1] and shows that, when n is sufficiently large,

$$\mathcal{A}_{\mathrm{UFF}_\alpha}(\alpha) > \mathcal{A}_{\mathrm{FF}}(\alpha), \text{ for } 0.90 \approx \frac{7+\sqrt{85}}{18} < \alpha \leq 1.$$

Theorem 4 For Unrestricted Bin Packing, for $\frac{2}{3} + \frac{1}{3n} \leq \alpha \leq 1$,

$$\mathcal{A}_{\mathrm{FF}}(\alpha) \leq \begin{cases} \dfrac{5}{9\alpha - 1} + O\left(\dfrac{1}{n}\right), & \text{if } n = 9 \cdot 2^i - 5 \text{ and } k \geq 12 \cdot 2^{3i}, i \in \mathbb{N}_0, \text{ and } 3 \mid k \\ \dfrac{5}{9\alpha - 1} + O\left(\dfrac{1}{\sqrt{n}}\right), & \text{otherwise.} \end{cases}$$

Proof We first show the bound for the special values of n. Then we extend to all values of n. Assume that, for some $i \in \mathbb{N}_0$, n is of the form $n = 9 \cdot 2^i - 5$ and that k is at least $12 \cdot 2^{3i}$ and is divisible by 3. An adversary can give the following request sequence, divided into $i+3$ phases:

Phase 0: 3 items of size $A = \frac{k}{3} - 2^{3i}$.

Phase $j = 1 \ldots i$: $3 \cdot 2^j$ pairs, each with one item of size $B_j = \frac{k}{3} + 2^{3i-3j+2}$, followed by one of size $C_j = \frac{k}{3} - 2^{3i-3j}$.

Phase $i+1$: $3 \cdot 2^i$ items of size $D = \frac{2k}{3} + 1$.

Phase $i+2$: $3 \cdot (\alpha n - (6 \cdot 2^i - 3))$ items of size $E = \frac{k}{3}$.

The number of items given in Phase $i+2$ is non-negative, since $\alpha \geq \frac{2}{3} + \frac{1}{3n}$. First-Fit will pack the items of Phase 0 in one bin. The assumption on k ensures that these three items will remain alone, since the smallest item given after Phase 0 has size $\frac{k}{3} - 2^{3i-3}$ and therefore the requirement is that $3 \cdot 2^{3i} < \frac{k}{3} - 2^{3i-3}$ giving that $k > (9 + \frac{3}{8})2^{3i}$. For each phase j, $1 \leq j \leq i$, First-Fit will pack one pair consisting of an item of size B_j and an item of size C_j in each bin, using $3 \cdot 2^j$ bins. After such a pair is packed, all future items are too large to join a pair. The number of bins used in the first $i+1$ phases is $1 + \sum_{j=1}^{i} 3 \cdot 2^j = 6 \cdot 2^i - 5$. In the next phase, each item will be placed in its own bin, using the last $3 \cdot 2^i$ bins. There will be no space for items from the last phase. Adding up the number of items from the first $i+2$ phases, First-Fit will accept $15 \cdot 2^i - 9$ items.

OPT can pack each item from Phase 1 with two of the items of size B_1 from Phase 2, using a total of three bins for this. Then, for $1 \leq j \leq i-1$, it can place every pair of items of size B_{j+1} together with one item of size C_j. This occupies $3 \cdot 2^j$ bins. Then it can pack one item of size C_i together with one item of size D using a total of $3 \cdot 2^i$ bins for this. The number of bins used is $3 + 3\sum_{j=1}^{i-1} 2^j + 3 \cdot 2^i = 6 \cdot 2^i - 3$. From the last phase, three items can be packed together in each bin, and the entire sequence will in this way occupy αn bins.

The ratio is thus

$$\frac{15 \cdot 2^i - 9}{15 \cdot 2^i - 9 + 3 \cdot (\alpha n - (6 \cdot 2^i - 3))} = \frac{5 \cdot 2^i - 3}{\alpha n - 2^i} = \frac{5 - \frac{2}{n}}{9\alpha - 1 - \frac{5}{n}} = \frac{5}{9\alpha - 1} + O\left(\frac{1}{n}\right).$$

For n general, we choose integers $n' \in \Theta(\sqrt{n}), c \leq n'$, and $s \in \Theta(\sqrt{n})$ satisfying $n = sn' + c$ and $n' = 9 \cdot 2^i - 5$. By giving c items of size k followed by s times the sequence described above (with n' substituted for n), we get the following performance ratio

$$\frac{c + s(15 \cdot 2^i - 9)}{c + s(3\alpha n' - 3 \cdot 2^i)} \leq \frac{c + s(5 \cdot 2^i - 3)}{s(\alpha n' - 2^i)} = \frac{5}{9\alpha - 1} + O\left(\frac{c}{n} + \frac{1}{n'}\right) = \frac{5}{9\alpha - 1} + O\left(\frac{1}{\sqrt{n}}\right).$$

□

The following theorem establishes that Theorem 1, showing that any fair algorithm has accommodating function at least $\frac{1}{1+\alpha-1/k}$, is asymptotically tight.

Theorem 5 For Unrestricted Bin Packing,

$$\mathcal{A}_{\mathrm{WF}}(\alpha) < \frac{1}{1+\alpha-\frac{1}{k}-\frac{1}{n}}, \text{ for } \alpha \leq 1.$$

Proof The adversary gives two phases of items:

- n items of size 1
- $\alpha n - \lceil \frac{n}{k} \rceil$ items of size k.

Worst-Fit will accept only the first phase, whereas the optimal algorithm can behave like First-Fit and pack all items in αn bins. The ratio is

$$\mathcal{A}_{\mathrm{WF}}(\alpha) \leq \frac{n}{n+(\alpha n - \lceil \frac{n}{k} \rceil)} < \frac{n}{n+(\alpha n - \frac{n+k}{k})} = \frac{1}{1+\alpha-\frac{1}{k}-\frac{1}{n}}.$$

□

4 Unit Price Seat Reservation

The other maximization problem we consider in detail is the Unit Price Seat Reservation Problem. 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 s to a station t. The passenger is given a single seat number when the ticket is purchased, which can be any time before departure. 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. For this problem an α-sequence can be packed by an optimal off-line algorithm using αn seats.

The algorithms (ticket agents) attempt to maximize income, i.e., the sum of the prices of the tickets sold. In this paper, we consider only the pricing policy in which all tickets have the same price, the *unit price problem*, since the results in [7] for the proportional price problem show that its competitive ratio, even on accommodating sequences, is $\Theta(\frac{1}{k})$.

The seat reservation problem is similar to the problem of coloring an interval graph on-line (the seats corresponding to the colors), which has been well studied because of applications to dynamic storage allocation. The difference is that with graph coloring, all vertices must be given a color and the goal is to minimize the number of colors used. With the seat reservation problem, there is a fixed number n of colors, and the goal is to maximize the number of vertices that are colored.

We use the following result from graph theory: Interval graphs are *perfect* [13], so the size of the largest clique is exactly the number of colors needed to color the whole graph. 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 the number n of seats, the optimal 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 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.

For the Seat Reservation Problem, First-Fit and Worst-Fit are defined similarly as for Unrestricted Bin Packing; First-Fit places each request on the first seat it fits on, and Worst-Fit places each request on a seat where the sum of the gap sizes on both sides of the request is largest possible. The results found in this section for these two algorithms are depicted in Figure 1 in Section 1.

4.1 Performance Guarantees

The performance guarantee for fair algorithms on accommodating sequences found in [7] can be extended to a general performance guarantee for $\alpha \leq 1$. Consider any α-sequence and any fair on-line algorithm $\mathbb{A}$. Let A be the set of requests which are accepted and R be the set rejected. For any subset S of the n seats, let A_S denote the subset of the intervals in A which $\mathbb{A}$ places on seats in S. The following lemma shows that if many requests are rejected, then the number of intervals in A_S must be large. The lemma is used to prove a performance guarantee on all fair algorithms. The main idea of the proof is to focus on some specific subset, consisting of αn of the n seats.

Lemma 1 Let S be any subset of the seats, consisting of αn seats, $\alpha \leq 1$ and let I be any α-sequence. For any fair algorithm $\mathbb{A}$, $|A_S| \geq |R|$.

Proof Since $\mathbb{A}$ is fair, none of the requests in R can be placed on any of the seats in S. Consider the following request sequence I', consisting of all the intervals in $A_S \cup R$, with the intervals in A_S coming before those in R. Clearly, there exists a fair algorithm $\mathbb{A}'$ which given I' and a train with αn seats places the intervals in A_S exactly as $\mathbb{A}$ did, except that it uses all αn seats instead of αn selected seats out of n. Algorithm $\mathbb{A}'$ accepts all the intervals in A_S, and, as a consequence, rejects all intervals from R.

The request sequence I' is an α-sequence, since it is constructed from a subset of requests from an α-sequence, so every request could be accommodated in those αn seats by an optimal off-line algorithm. By [7], the competitive ratio of any fair algorithm on accommodating sequences is at least $\frac{1}{2}$ and the additive constant in the definition of the competitive ratio is not used in the proof, so $|A_S| \geq |R|$. □

Theorem 6 Any fair algorithm for Unit Price Seat Reservation is $\frac{1}{1+\alpha}$-competitive on α-sequences, for $\alpha \leq 1$.

Proof Choose S to be a subset containing αn seats which minimizes $|A_S|$. By Lemma 1, $|A_S| \geq |R|$. Since S was chosen to minimize $|A_S|$, $\mathbb{A}$ places at least as many intervals on each seat not in S as it does on any seat in S. Thus, $|A| \geq n\frac{|A_S|}{\alpha n} \geq \frac{|R|}{\alpha}$. This gives a performance ratio of

$$\frac{|A|}{|A|+|R|} \geq \frac{\frac{1}{\alpha}|R|}{\frac{1}{\alpha}|R|+|R|} = \frac{1}{1+\alpha}.$$

□

This result is asymptotically tight due to Worst-Fit's behavior, as shown in the next subsection.

It was shown in [7] that First-Fit's competitive ratio on accommodating sequences is not strictly better than $\frac{1}{2}$, the performance guarantee for any fair algorithm. It is interesting that for $\alpha \leq \frac{1}{2}$, it is possible to prove that First-Fit's performance is better than Worst-Fit's.

Theorem 7 For Unit Price Seat Reservation,

$$\mathcal{A}_{\mathrm{FF}}(\alpha) \geq \frac{2^{\ell-1}(1-\ell\alpha)+(2^{\ell}-1)\alpha}{2^{\ell-1}(1-\ell\alpha)+(2^{\ell})\alpha} \geq 1-\frac{1}{2^{\ell}}, \text{ where } \ell = \left\lfloor \frac{1}{\alpha} \right\rfloor \text{ and } \alpha \leq 1.$$

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

Divide the seats up into ℓ blocks, where $\ell = \lfloor \frac{1}{\alpha} \rfloor$, 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 $n - \ell\alpha n + \alpha n$ seats. Let A_i denote the set of intervals which First-Fit places in the ith block.

We consider the last block first and compute a lower bound on $|A_\ell|$, the number of intervals First-Fit places in it. Using the techniques from the proof of Theorem 6, one can choose the αn seats from among the last $n - \ell\alpha n + \alpha n$ which contain the fewest intervals. Those seats must contain at least $|R|$ intervals, so $|A_\ell| \geq \frac{1-\ell\alpha+\alpha}{\alpha}|R|$.

First-Fit could not place any of the intervals it placed in the last block 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. Again using the result that any fair algorithm is at least $\frac{1}{2}$-competitive on accommodating sequences, we have that $|A_{\ell-1}| \geq |R| + |A_\ell|$. Similarly, $|A_i| \geq |R| + \sum_{j=i+1}^{\ell} |A_j|$. 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 $C_\ell \geq \frac{2^{\ell-1}(1-\ell\alpha)+(2^{\ell}-1)\alpha}{\alpha}|R|$. Thus, the performance ratio

$$\mathcal{A}_{\mathrm{FF}}(\alpha) \geq \frac{|A|}{|A|+|R|} = \frac{C_\ell}{C_\ell+|R|} \geq \frac{2^{\ell-1}(1-\ell\alpha)+(2^{\ell}-1)\alpha}{2^{\ell-1}(1-\ell\alpha)+(2^{\ell})\alpha}.$$

□

This performance guarantee is not tight. Kierstead and Qin [16] have shown that First-Fit colors every interval graph using at most 25.72 times as many colors as necessary, so the competitive ratio on α-sequences is 1 when $\alpha \leq \frac{1}{25.72} \approx 0.039$.

4.2 Impossibility Results

Our first impossibility result for the Unit Price Seat Reservation Problem shows that the general performance guarantee of Theorem 6 is asymptotically tight by giving a matching impossibility result for Worst-Fit.

Theorem 8 For Unit Price Seat Reservation,

$$\mathcal{A}_{\mathrm{WF}}(\alpha) < \frac{1}{1+\alpha-\frac{1}{k-1}-\frac{1}{n}}, \text{ when } \alpha \leq 1.$$

Proof Since this upper bound is at least 1 for $\alpha n < \lceil \frac{n}{k-1} \rceil$, we assume that $\alpha n \geq \lceil \frac{n}{k-1} \rceil$. Consider the sequence consisting of n intervals each of size 1 "evenly" distributed among the intervals of the form $[i, i+1)$, for $1 \leq i \leq k-1$, i.e., the number of intervals for each i is either $\lfloor \frac{n}{k-1} \rfloor$ or $\lceil \frac{n}{k-1} \rceil$, followed by $\alpha n - \lceil \frac{n}{k-1} \rceil$ copies of the interval $[1, k)$. The unit-length intervals can all be placed on $\lceil \frac{n}{k-1} \rceil$ seats, so the sequence is an α-sequence. Worst-Fit will put the unit-length intervals on separate seats and reject all of the long intervals. OPT will accept all of the intervals since $\alpha \leq 1$, so the ratio is

$$\mathcal{A}_{\mathrm{WF}}(\alpha) \leq \frac{n}{n + \alpha n - \lceil \frac{n}{k-1} \rceil} < \frac{n}{n + \alpha n - \frac{n+k-1}{k-1}} = \frac{1}{1 + \alpha - \frac{1}{k-1} - \frac{1}{n}}.$$

□

Theorem 9 below says that if k can be arbitrarily large, no fair on-line (deterministic or randomized) algorithm for the Unit Price Seat Reservation Problem is better than $\frac{8-\alpha}{9\alpha}$-competitive on α-sequences. Thus, for $\alpha > \frac{4}{5}$, $\mathcal{A}_{\mathbb{A}}(\alpha) < 1$ for any algorithm $\mathbb{A}$ for the problem. The proof of this is based on Theorem 3.1 in [3], which in turn is based on a proof from [7].

Theorem 9 Let $\alpha \leq 1$ and $k \geq 9$. Then, for any fair on-line algorithm $\mathbb{A}$ for Unit Price Seat Reservation,

$$\mathcal{A}_{\mathbb{A}}(\alpha) \leq \frac{8-\alpha}{9\alpha} + O\left(\frac{1}{k}\right).$$

Proof Assume that αn is even, $\alpha \leq 1$ and $k \geq 9$. Let $k' \leq k$ be chosen largest possible such that 6 divides $k' - 3$. The adversary begins with $\frac{\alpha n}{2}$ requests for each of the intervals $[3s+1, 3s+3)$ for $s = 0, 1, \ldots, \frac{k'-3}{3}$. We refer to these intervals as the *original* intervals. Any fair on-line algorithm will accept this set of $\frac{k'}{3} \cdot \frac{\alpha n}{2}$ requests. Suppose that after these requests are accepted, there are q_i seats which are empty from station $3i+2$ to station $3i+5$. For $i = 0, 1, \ldots, \frac{k'-9}{6}$, let $p_i = q_{2i} + q_{2i+1}$.

If $E[p_i] \leq \frac{16-11\alpha}{9} n$, the adversary proceeds with $\frac{\alpha n}{2}$ requests for the interval $[6i+2, 6i+5)$ and $\frac{\alpha n}{2}$ requests for the interval $[6i+5, 6i+8)$. For these αn additional requests, the on-line algorithm can accommodate exactly p_i of them. Thus, for the total of $2\alpha n$ requests, the starting station of which is in $[6i+1, 6i+6)$, the on-line algorithm accommodates $\alpha n + p_i$.

If $E[p_i] > \frac{16-11\alpha}{9} n$, the adversary proceeds with $\frac{\alpha n}{2}$ requests for the interval $[6i+3, 6i+7)$, followed by $\frac{\alpha n}{2}$ requests for the interval $[6i+2, 6i+4)$ and $\frac{\alpha n}{2}$ requests for the interval $[6i+6, 6i+8)$. The $[6i+3, 6i+7)$ requests are all accepted. Out of the next αn requests, the on-line algorithm can accommodate exactly $(4-3\alpha)n - p_i$ of them. This is seen as follows. Consider the $[6i+2, 6i+4)$ intervals. The on-line algorithm can place $n - q_{2i} - \frac{\alpha n}{2}$ of these intervals on the same seat as a $[6i+4, 6i+6)$ interval, namely where there is no $[6i+1, 6i+3)$ interval at the same time. On the $n - \frac{\alpha n}{2}$ seats where there are no $[6i+4, 6i+6)$ intervals, one of the $\frac{\alpha n}{2}$ intervals of type $[6i+3, 6i+7)$ would prevent us from placing a $[6i+2, 6i+4)$ interval. Thus, the on-line algorithm can place at most $n - \frac{\alpha n}{2} - \frac{\alpha n}{2}$ intervals of type $[6i+2, 6i+4)$ on seats with no $[6i+4, 6i+6)$ interval. Adding up, the total number of accepted requests are at most $(2 - \frac{3\alpha}{2})n - q_{2i}$. Similarly, the on-line algorithm accepts at most $(2 - \frac{3\alpha}{2})n - q_{2i+1}$ of the $[6i+6, 6i+8)$ intervals, totaling $(4-3\alpha)n - p_i$. Thus, of the $\frac{5\alpha n}{2}$ requests, the starting station of which is in $[6i+1, 6i+6)$, the on-line algorithm accommodates at most $\frac{3\alpha n}{2} + (4-3\alpha)n - p_i = (4 - \frac{3}{2}\alpha)n - p_i$ of them.

Clearly for both cases, the density is nowhere more than αn, so the optimal off-line algorithm can accommodate all requests on αn seats.

In this way, the requests are partitioned into $\frac{k'-3}{6} + 1$ groups; each of the first $\frac{k'-3}{6}$ groups consists of either $2\alpha n$ or $\frac{5\alpha n}{2}$ requests and the last group consists of $\frac{\alpha n}{2}$ requests. For each of the first $\frac{k'-3}{6}$ groups, the on-line algorithm can accommodate up to a fraction $\frac{8-\alpha}{9\alpha}$ of the requests in the group. 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. By linearity of expectation,

$$\begin{aligned}
E\left[\frac{|A|}{|A|+|R|}\right] &\leq \frac{\frac{\alpha n}{2} + \sum_{i\in S}(\alpha n + E[p_i]) + \sum_{i\in\bar{S}}((4-\frac{3}{2}\alpha)n - E[p_i])}{\frac{\alpha n}{2} + \sum_{i\in S} 2\alpha n + \sum_{i\in\bar{S}} \frac{5\alpha n}{2}} \\
&\leq \frac{\frac{\alpha}{2} + \sum_{i\in S}(\alpha + \frac{16-11\alpha}{9}) + \sum_{i\in\bar{S}}(4 - \frac{3}{2}\alpha - \frac{16-11\alpha}{9})}{\frac{\alpha}{2} + \sum_{i\in S} 2\alpha + \sum_{i\in\bar{S}} \frac{5\alpha}{2}} \\
&\leq \frac{\frac{\alpha}{2} + (\alpha + \frac{16-11\alpha}{9}) \cdot \frac{k'-3}{6}}{\frac{\alpha}{2} + 2\alpha \cdot \frac{k'-3}{6}} \\
&= \frac{(8-\alpha)2k' + 33\alpha - 48}{9\alpha \cdot 2k' - 27\alpha},
\end{aligned}$$

where the last inequality holds because in general $\frac{a}{b} = \frac{c}{d} < 1$ and $a < c$ imply that $\frac{e+ax+cy}{e+bx+dy} \leq \frac{e+a(x+y)}{e+b(x+y)}$. This completes the proof. □

We consider a class of fair algorithms called Any-Fit, which will use an empty seat only if there is not enough space on partially used seats. Any-Fit includes both First-Fit and Best-Fit.

Theorem 10 For any Any-Fit algorithm $\mathbb{A}$ for Unit Price Seat Reservation,

$$\mathcal{A}_{\mathbb{A}}(\alpha) \leq \frac{1}{3\alpha - 1} + O\left(\frac{1}{k}\right), \text{ for } \frac{1}{3} < \alpha \leq 1.$$

Proof Since the upper bound is at least 1 for $\alpha \leq \frac{2}{3}$, we assume that $\alpha \geq \frac{2}{3}$. Assume first that 3 divides n and 6 divides $k-4$. Give $\frac{n}{3}$ requests of each of the following types (see Figure 4, where we have illustrated the placement by $\mathbb{A}$):

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

Since $\mathbb{A}$ is fair, it accepts all these requests, amounting to $\frac{n}{3}(3\frac{k-4}{6} + 4) = \frac{k+4}{6}n$.

OPT uses $\frac{2n}{3}$ seats to accommodate all these requests, since the density between any two stations is $\frac{2n}{3}$. OPT can use the space from $\frac{2n}{3}$ up to αn to accept the following intervals, which $\mathbb{A}$ will be unable to accommodate:

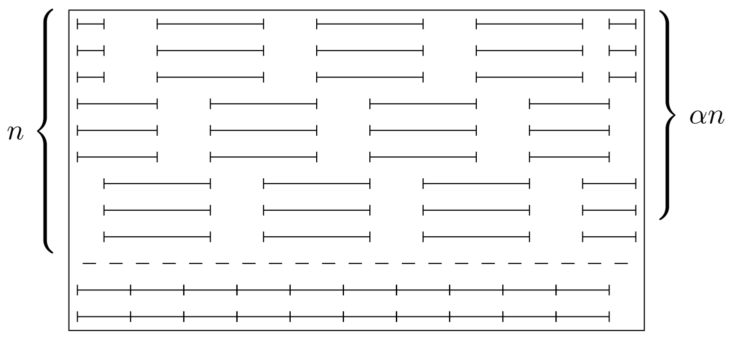

Figure 4: Placement for $k = 22$ and $n = 9$.

- For $i \in \{1, \ldots, \frac{k}{2} - 1\}$: $[2i-1, 2i+1)$.

We get the following ratio:

$$\mathcal{A}_{\mathbb{A}}(\alpha) \leq \frac{\frac{k+4}{6}n}{\frac{k+4}{6}n + (\alpha - \frac{2}{3})n(\frac{k}{2} - 1)} = \frac{k+4}{(3\alpha-1)k + 8 - 6\alpha} = \frac{1}{3\alpha - 1} + O\left(\frac{1}{k}\right).$$

Let $c = n \bmod 3$ and $d = (k-4) \bmod 6$. If $c = 0$ and $d > 0$, the sequence above with $\frac{k-4}{6}$ replaced by $\lfloor \frac{k-4}{6} \rfloor$ can be used. If $c > 0$, the adversary first gives $c \leq 2$ intervals $[1, k)$ and then gives the sequence above with $\frac{n}{3}$ replaced by $\lfloor \frac{n}{3} \rfloor$ and $\frac{k-4}{6}$ replaced by $\lfloor \frac{k-4}{6} \rfloor$. The ratio is then

$$\begin{aligned}
\mathcal{A}_{\mathbb{A}}(\alpha) &\leq \frac{c + \frac{k-d+4}{6}(n-c)}{c + \frac{k-d+4}{6}(n-c) + (\alpha - \frac{2}{3})(n-c)(\frac{k-d}{2} - 1)} \\
&= \frac{k + 4 - \frac{k-2}{n}c - (1 - \frac{c}{n})d}{(3\alpha-1)k + 8 - 6\alpha - \frac{(3\alpha-1)k+2-6\alpha}{n}c - (1 - \frac{c}{n})(3\alpha-1)d} \\
&\leq \frac{k+4}{(3\alpha-1)k + 8 - 6\alpha - \frac{(6\alpha-2)k+4-12\alpha}{n} - 15\alpha + 5} \\
&= \frac{(k+4)\left(1 + \frac{\left((6\alpha-2)k+4-12\alpha\right)/n + 15\alpha - 5}{(3\alpha-1)k+8-6\alpha-\left((6\alpha-2)k+4-12\alpha\right)/n - 15\alpha + 5}\right)}{(3\alpha-1)k + 8 - 6\alpha} \\
&= \frac{k+4}{(3\alpha-1)k + 8 - 6\alpha} + O\left(\frac{1}{nk} + \frac{1}{k^2}\right) \\
&= \frac{1}{3\alpha-1} + O\left(\frac{1}{k}\right).
\end{aligned}$$

□

4.3 Distinguishing Between Algorithms

As mentioned above, First-Fit has a competitive ratio on accommodating sequences which is not strictly better than $\frac{1}{2}$. This result is from [7]. Later, in [3], it was shown that $\frac{1}{2}$ is an asymptotic (for k much larger than n) upper bound on the performance of any fair algorithm. Similarly, First-Fit's competitive ratio is no better than $\frac{2}{k-1} - \frac{1}{(k-1)^2}$, while the performance guarantee for any fair algorithm is $\frac{2}{k}$ [7]. Theorems 7 and 8 together show that in contrast to these results for $\alpha \geq 1$, when $\alpha < \frac{1}{2}$, First-Fit is not as bad as any other possible fair algorithm. Specifically, Worst-Fit has an asymptotic competitive ratio of at most $\frac{3}{4}$ on $\frac{1}{3}$-sequences, while First-Fit has a ratio of at least $\frac{7}{8}$.

Kierstead and Trotter [17] have shown that every interval graph with clique number $\omega(G)$ can be $(3\omega(G) - 2)$-colored on-line. Since colors correspond to seat assignments, this means that their algorithm has a competitive ratio of 1 on $\frac{1}{3}$-sequences. Their algorithm solves the problem of minimizing the number of seats (colors) used; it is undefined for values of $\alpha > \frac{1}{3}$ when applied to the maximization problem where the number of seats is limited. However, their algorithm can obviously be extended in many ways so that it solves this maximization problem. For example, one could use their algorithm whenever it assigns a seat number which exists and use First-Fit otherwise.

Chrobak and Ślusarek [11] have shown 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 $\frac{1}{3}$-sequences. Thus, Kierstead and Trotter's algorithm has a better competitive ratio on $\frac{1}{3}$-sequences than First-Fit.

Kierstead and Trotter's algorithm is not, however, better than First-Fit on 1-sequences, since it has been shown [7] that no fair algorithm has a competitive ratio on accommodating sequences which is strictly better than $\frac{1}{2}$ asymptotically.

In conclusion, we have that the competitive ratio on α-sequences, where $\alpha < 1$, can be useful in distinguishing between algorithms.

Theorem 11 Kierstead and Trotter's algorithm, First-Fit, and Worst-Fit, all have asymptotic competitive ratio $\frac{1}{2}$ on accommodating sequences, but have different competitive ratios on $\frac{1}{3}$-sequences.

Note 1 It is tempting to try to use the techniques in Chrobak and Ślusarek [11] to improve the impossibility result on First-Fit or the techniques in Kierstead and Trotter [17] to improve the general impossibility result. Unfortunately, the number of requests accepted is so large compared to the number rejected that the impossibility results which can be obtained are essentially 1. □

5 Resource augmentation in comparison with the accommodating function

The concept of the accommodating function should not be confused with resource augmentation introduced in [14]. Resource augmentation analysis gives the on-line algorithm more resources than the optimal off-line algorithm that it is compared to, but there is no restriction on the input sequences.

Note 2 In the resource augmentation setting, the on-line algorithm has the amount m of resources and the optimal off-line algorithm has the amount $n \leq m$ of resources. Performance guarantees obtained in this setting are also valid for the case where all input sequences are $\frac{n}{m}$-sequences. This is because, even though the optimal off-line algorithm in our setting has the same amount of resources as the on-line algorithm, the result it obtains cannot be improved with extra resources beyond n when considering $\frac{n}{m}$-sequences. $\square$

The contrapositive of the observation above gives that when considering impossibility results, it is the other way around: impossibility results for the accommodating function carry over to the resource augmentation setting.

Resource augmentation does not always give as realistic or optimistic results as the accommodating function for $\alpha < 1$. In order to obtain the same results, one needs to restrict to accommodating sequences while also doing resource augmentation. To see that resource augmentation alone can be insufficient, consider the algorithm First-Fit for Unit Price Seat Reservation. Theorem 7 gives a performance guarantee for First-Fit for $\alpha \leq 1$: $\mathcal{A}_{\mathrm{FF}}(\alpha) \geq 1 - \frac{1}{2^\ell}$, where $\ell = \lfloor \frac{1}{\alpha} \rfloor$. In contrast, resource augmentation cannot give a constant performance guarantee for First-Fit.

Theorem 12 For $\alpha > \frac{2}{n}$, the competitive ratio of First-Fit for Unit Price Seat Reservation is at most $\frac{1+\alpha}{(\alpha - \frac{2}{n})(k-1)}$, when First-Fit has at most $\frac{1}{\alpha}$ times as many seats as OPT.

Proof Let $\frac{2}{n} \leq \alpha < 1$ be given. We allow First-Fit to have n seats, while OPT has $\lfloor \alpha n \rfloor$ seats. Consider the sequence beginning with $\lfloor \frac{\alpha n}{k-1} \rfloor$ requests for $[i, i+1)$ for $1 \leq i \leq k-1$, one request for $[i, i+1)$ for $1 \leq i \leq \lfloor \alpha n \rfloor - (k-1)\lfloor \frac{\alpha n}{k-1} \rfloor$, $n - 1 - \lfloor \frac{\alpha n}{k-1} \rfloor$ requests for $[1, k)$, and finally the request $[\lfloor \alpha n \rfloor + 1 - (k-1)\lfloor \frac{\alpha n}{k-1} \rfloor, k)$. First-Fit will accept all of these and the train will be full. The sequence continues with $\lfloor \alpha n \rfloor - 1 - \lfloor \frac{n}{k-1} \rfloor$ requests for $[i, i+1)$ for $1 \leq i \leq k-1$, and First-Fit must reject them all. Note that OPT can place one of the first short intervals on each of its $\lfloor \alpha n \rfloor$ seats and thus reject all of the long intervals, except possibly the last "long" interval. Thus, it can accept all of the unit-length intervals at the end. The performance ratio is at most

$$\frac{|A|}{|A| + |R|} = \frac{\lfloor \alpha n \rfloor + n - \lfloor \frac{\alpha n}{k-1} \rfloor}{\lfloor \alpha n - 1 \rfloor (k-1)} \leq \frac{1 + \alpha}{(\alpha - \frac{2}{n})(k-1)} \quad .$$

$\square$

For Unrestricted Bin Packing, one can also show that resource augmentation analysis alone gives results which are much more pessimistic than the corresponding accommodating function results. Recall that the performance guarantee from Corollary 1 for First-Fit for $\alpha \leq 1$ is $\mathcal{A}_{\mathrm{FF}}(\alpha) \geq \frac{3+2\alpha}{8\alpha}$.

Theorem 13 For Unrestricted Bin Packing, the competitive ratio of First-Fit is at most $\frac{1}{\alpha k}$, when First-Fit has at most $\frac{1}{\alpha}$ times as many bins as OPT.

Proof We allow First-Fit to have n bins, while OPT has $\lfloor \alpha n \rfloor$ bins. The adversary can give n items of size k followed by $\alpha n k$ items of size 1. First-Fit will accept the first n items, but none of the small items. OPT will reject the first n items and accept all the small items, giving a ratio of $\frac{n}{\alpha n k} = \frac{1}{\alpha k}$. $\square$

Next, we consider three concrete minimization problems to demonstrate how the accommodating function is applicable to them. These problems show the subtleties in the definition of the accommodating sequences. They also demonstrate how resource augmentation results can be applied to give accommodating function results.

5.1 Paging

We consider the Paging Problem in the page fault model, i.e., the algorithm maintains a fast memory (cache) consisting of k pages of memory and the input is a sequence of page requests. If a page in cache is requested, no cost is incurred; otherwise the requested page must be transferred from the slow memory at a cost of 1 and another page must be evicted from the cache. It is usually assumed that the page to be evicted must be chosen before any of the following requests are revealed. The goal is to choose an eviction strategy which minimizes cost. Before the first page request is served, the cache is empty. In this problem, the limited resource is the cache.

When an optimal off-line algorithm serves an α-sequence, it will have the same cost for every cache size $k' \geq \alpha k$. To see why such a sequence can consist of more than αk different pages, consider the following example: For $k = 6$, the following sequence is a $\frac{1}{2}$-sequence, $S = \langle 1, 2, 3, 4, 5, 6, 7, 1, 2, 8, 9, 10, 1, 2, 11, 12 \rangle$. Keeping the two pages 1 and 2 in the cache will give a cost of 12 with a cache of size k' for all $k' \geq 3$.

In the (h, k)-Paging Problem, the on-line algorithm is assumed to have a cache of size k as in the usual Paging Problem, but the performance of the on-line algorithm is measured relative to an optimal off-line algorithm whose cache has size $h \leq k$. (See [5] for a survey on (h, k)-Paging.)

Sleator and Tarjan [21] prove that for the (h, k)-Paging Problem LRU (Least-Recently-Used) and FIFO (First-In/First-Out) are both $\frac{k}{k-h+1}$-competitive. Using Note 2, this implies that FIFO and LRU are $\frac{k}{(1-\alpha)k+1}$-competitive. Sleator and Tarjan also prove that their performance guarantee is tight, and a very similar proof shows that it is also tight for α-sequences. The proof is given in detail here to give further intuition for α-sequences.

Theorem 14 Any deterministic Paging algorithm $\mathbb{A}$ has $\mathcal{A}_{\mathbb{A}}(\alpha) \geq \frac{k}{(1-\alpha)k+1}$, when $\alpha \leq 1$.

Proof Let $\mathbb{A}$ be a deterministic Paging algorithm. Assume that OPT has a cache of size αk. The requests are given in phases $P_0, P_1, \ldots, P_n$. In phase P_0, αk different pages are requested once. The rest of the phases consist of k requests each. For each phase P_i, $1 \leq i \leq n$, the first $k - \alpha k + 1$ requests are to pages that have not been requested before. Let S_i be the set consisting of these $k - \alpha k + 1$ pages and the αk pages that were in OPT's cache at the beginning of P_i. Then, $|S_i| = k + 1$. Each of the remaining $\alpha k - 1$ requests in phase P_i is to a page in S_i that is currently not in $\mathbb{A}$'s cache.

The first $k - \alpha k + 1$ pages of phase P_i , $1 \leq i \leq n$, cost both algorithms $k - \alpha k + 1$, since they have not been requested earlier. During these $k - \alpha k + 1$ requests, OPT keeps the pages to be requested during the last $\alpha k - 1$ requests of the phase in its cache. Thus, the optimal cost of each phase P_i, $1 \leq i \leq n$, is $(1 - \alpha)k + 1$, while the on-line cost is k. Since OPT has faults only on requests to pages that have not been requested earlier, the sequence is an α-sequence. $\square$

In [22] Young considers the randomized paging algorithm MARK for (h,k)-Paging. When a page is requested it is marked, and when a page must be evicted, MARK chooses uniformly at random among the unmarked pages in the cache. If all pages in the cache are marked, all marks are erased. Young proves that MARK is 2-competitive, when $\frac{k}{k-h} < e$, and $2(\ln \frac{k}{k-h} - \ln\ln \frac{k}{k-h} + \frac{1}{e-1})$-competitive when $\frac{k}{k-h} \geq e$. Combining Young's result with Note 2 directly gives us that, for $\alpha < 1$,

$$\mathcal{A}_{\text{MARK}}(\alpha) \leq \begin{cases} 2, & \text{for } \frac{1}{1-\alpha} < e \\ 2\left(\ln \frac{1}{1-\alpha} - \ln\ln \frac{1}{1-\alpha} + \frac{1}{e-1}\right), & \text{for } \frac{1}{1-\alpha} \geq e. \end{cases}$$

Using only $\frac{h}{k}$-sequences, Young [22] also proves that any randomized (h,k)-Paging algorithm has a competitive ratio of at least $\ln \frac{k}{k-h} - \ln\ln \frac{k}{k-h} - \frac{3}{k(1-\alpha)}$ when $\frac{k}{k-h} \geq e$. Thus, any randomized algorithm $\mathbb{A}$ has $\mathcal{A}_{\mathbb{A}}(\alpha) \geq \ln \frac{1}{1-\alpha} - \ln\ln \frac{1}{1-\alpha} - \frac{3}{k(1-\alpha)}$ when $\frac{1}{1-\alpha} \geq e$, so MARK is within a factor of approximately 2 of optimal.

5.2 The k-Server Problem

The k-Server Problem [20] is a natural generalization of the Paging Problem which was discussed in the previous subsection. For the k-Server Problem [20], the algorithm controls k servers, which are located on points of a metric space (M,d), where M is a set of points with $|M| > k$ and d is a metric for M. The request sequence is a sequence of points in M, and the algorithm must service all requests sequentially, by having a server at the point requested or moving one there. The goal is to minimize the total distance the servers move. When considering the accommodating function for the k-Server Problem, we use the servers as the resource; this is the natural generalization from the last section where the pages in cache were the resource. Hence, on an α-sequence, OPT does not benefit from having more than αk servers.

Since the k-Server Problem is a generalization of the Paging Problem, by Theorem 14, no k-Server algorithm can be better than $\frac{k}{(1-\alpha)k+1}$-competitive in general. In [20] it is proven that this impossibility result is actually valid for any metric space.

5.2.1 The Work Function Algorithm

The Work Function Algorithm (WFA) is believed to be an optimal on-line algorithm for the k-Server Problem. The algorithm keeps track of the optimal solution to the part of the input sequence seen so far, and tries to minimize the cost of each move without getting too far from the optimal configuration. More precisely, for each request to some point r, it moves a server from a point $x \in C$ minimizing $w(C-x+r) + d(x,r)$, where C is the set of points occupied by WFA's servers, $d(x,r)$ is the distance between x and r, and $w(C-x+r)$ is the optimal cost of serving the requests seen so far and ending up in the configuration $C-x+r$.

In [18], the Work Function Algorithm for the k-Server Problem is investigated in the case where the on-line algorithm has k servers and the off-line algorithm has $h \leq k$ servers. Using Note 2, the performance guarantees found in [18] are also valid for the k-Server Problem on α-sequences. Letting $h = \alpha k$, one of these guarantees can be expressed as

$$\text{WFA}_k(I) \leq k \cdot \text{OPT}_{\alpha k}(I) + (\alpha k - 1) \cdot \text{OPT}_k(I) + \text{const},$$

which reduces to $\mathrm{WFA}_k(I) \leq (k+\alpha k-1) \cdot \mathrm{OPT}_k(I) + \mathrm{const}$, in the case of α-sequences. Thus, we arrive at the following theorem.

Theorem 15 For $\alpha \leq 1$, $\mathcal{A}_{\mathrm{WFA}}(\alpha) \leq (1+\alpha)k-1$.

5.2.2 Balance and Greedy

The algorithm Balance [10] serves each request r with a server s_i such that $D_i + d_i(r)$ is minimized, where D_i is the total distance traveled by the server s_i before the request r and $d_i(r)$ is the distance it will have to travel to serve r. In general, Balance is not competitive, i.e., there is no constant c such that Balance is c-competitive. A simple modification of the proof of this gives that for $\alpha \geq \frac{2}{k}$, Balance is not competitive, even on α-sequences. For $\alpha = \frac{1}{k}$, however, Balance is k-competitive.

The algorithm Greedy serves each request r with a server s_i, such that the distance moved, $d_i(r)$, is minimized. Like Balance, Greedy is not competitive on α-sequences, as long as $\alpha \geq \frac{2}{k}$. This can be shown using only three points. On the other hand, it is easy to see that, in contrast to Balance, Greedy is 1-competitive on $\frac{1}{k}$-sequences.

5.3 Machine Scheduling Minimizing Makespan

We study the problem of assigning jobs to machines such that the makespan is minimized, i.e., the completion time of the latest completing job is minimized. The jobs are characterized by their length, and there are m identical machines. Each job must be assigned to a machine before the next job is seen.

5.3.1 List Scheduling

The List Scheduling algorithm (LS) [12] schedules each new job on a least loaded machine.

Theorem 16 For $\alpha \leq 1$, $\mathcal{A}_{\mathrm{LS}}(\alpha) = 1 + \alpha - \frac{1}{m}$.

Proof In [12] it is shown that the competitive ratio of List Scheduling is at most $1+\alpha-\frac{1}{m}$, if LS has m machines and the off-line algorithm has only αm machines. Thus, according to Note 2 in Section 2, $\mathcal{A}_{\mathrm{LS}}(\alpha) \leq 1+\alpha-\frac{1}{m}$.

The sequence used in [9] to match this performance guarantee consists of $(\alpha m-1)m$ jobs of length 1, followed by one job of length m. Clearly, this sequence is an α-sequence. Thus, $\mathcal{A}_{\mathrm{LS}}(\alpha) \geq 1+\alpha-\frac{1}{m}$. □

5.3.2 General Impossibility Results

In [2] the scheduling problem is investigated in the resource augmentation setting. For the case where OPT has only α times as many machines as the on-line algorithm, a general impossibility result of $1+e^{-\frac{1}{\alpha}(1-o(1))}$, valid for both deterministic and randomized algorithms, is given using α-sequences. They also give an impossibility result of $\frac{5}{4}$ for deterministic algorithms for $\alpha = \frac{1}{2}$. The following impossibility result is a generalization of this result. Our bound is tighter than the bound of $1+e^{-\frac{1}{\alpha}(1-o(1))}$ from [2], but it is only valid for deterministic algorithms.

Theorem 17 Let $\alpha \leq 1$ and let $\ell \leq \alpha m$ be an integer such that $i \mid \alpha m$ for all $i \in \{1, 2, \ldots, \ell\}$. Then, for any deterministic algorithm $\mathbb{A}$ for the scheduling problem,

$$\text{if } \alpha \geq \frac{m+1}{m} \left(\sum_{i=1}^{\ell} \frac{1}{i} \right)^{-1}, \text{ then } \mathcal{A}_{\mathbb{A}}(\alpha) \geq \frac{\ell+1}{\ell} \enspace .$$

Proof The jobs are given in phases starting with Phase 1. In the ith phase, $\frac{\alpha m}{i}$ jobs of size i are given. The sequence is ended as soon as two jobs have been assigned the same machine. This happens at the latest in Phase ℓ, since $\sum_{i=1}^{\ell} \frac{\alpha m}{i} \geq m+1$. Let $r \leq \ell$ be the number of phases.

To prove that the sequence is an α-sequence, we must prove that the jobs can be scheduled on αm machines such that the last job on each machine has completion time r. Note that, for $1 \leq i \leq r$, the total length of jobs of length i is αm and the total length of all of the jobs is $\alpha m r$. For the purpose of the proof, we now schedule the jobs (without idle time) in non-increasing order of length. Assume to the contrary that a job of length i cannot be scheduled such that its completion time is no later than r. This means that all machines are busy until time $r - i + 1$. Thus, a total volume of $(r - i + 1)\alpha m$ has been given. However, this is a contradiction, since this volume will not be reached until all the jobs of length i have been processed.

The makespan of $\mathbb{A}$ is at least $r+1$, and the optimal makespan is r. Thus,

$$\mathcal{A}_{\mathbb{A}}(\alpha) \geq \frac{r+1}{r} \geq \frac{\ell+1}{\ell}.$$

$\square$

Example:

$$\begin{aligned}
\alpha &\geq \frac{2}{3}\frac{m+1}{m} &&\Rightarrow \mathcal{A}(\alpha) \geq \frac{3}{2}\\
\alpha &\geq \frac{6}{11}\frac{m+1}{m} &&\Rightarrow \mathcal{A}(\alpha) \geq \frac{4}{3}\\
\alpha &\geq \frac{12}{25}\frac{m+1}{m} &&\Rightarrow \mathcal{A}(\alpha) \geq \frac{5}{4}
\end{aligned}$$

For $\alpha \geq \frac{4}{5} + \frac{2}{m}$, the following theorem supplies a stronger impossibility result than Theorem 17. Specifically, for $i = 2$ and $p = 1 + \sqrt{2}$, it says that

$$\alpha \geq \frac{8}{9} + \frac{16}{9m} \text{ gives } \mathcal{A}_{\mathbb{A}}(\alpha) \geq 1 + \frac{1}{\sqrt{2}} \approx 1.707,$$

and for $i = 3$ and $p = \frac{1}{2}(1 + \sqrt{3})$,

$$\alpha \geq \frac{24}{29} + \frac{52}{29m} \text{ gives } \mathcal{A}_{\mathbb{A}}(\alpha) \geq 1 + \frac{1}{\sqrt{3}} \approx 1.577.$$

Theorem 18 If ($i = 2$ and $p \geq 1 + \sqrt{2}$) or ($i = 3$ and $p \geq \frac{1}{2}(1 + \sqrt{3})$), then

$$\alpha \geq \frac{2 + \frac{5}{m} - \frac{2}{mi}}{3 - \frac{1}{i} - \frac{1}{\lfloor ip \rfloor}} \text{ gives } \mathcal{A}_{\mathbb{A}}(\alpha) \geq \frac{i+1}{i} + \frac{1}{ip}$$

for any deterministic algorithm $\mathbb{A}$ for the scheduling problem.

Proof Note that $n_3 > 0$, since $i \geq 2$ and $\lfloor ip \rfloor \geq 4$.

Consider the following sequence consisting of three phases.

1. $n_1 = \alpha m$ jobs of length 1
2. $n_2 = \alpha m - 2$ jobs of length p
3. 1 job of length $p+1$, or $n_3 = \alpha m - \lceil \frac{n_2}{i} \rceil - \lceil \frac{n_1}{\lfloor ip \rfloor} \rceil$ jobs of length ip

The jobs given in the first phase form an α-sequence with optimal makespan 1. Thus, if two of these jobs are scheduled on the same machine, $\mathcal{A}_{\mathbb{A}}(\alpha) \geq 2$. Hence, we assume that no two jobs from Phase 1 are placed on the same machine.

If two jobs from Phase 2 are scheduled together with a job from Phase 1, or if three Phase 2 jobs are scheduled together, the on-line makespan is at least $2p+1$. In this case, the adversary gives one job of length $p+1$ in Phase 3. The sequence is now an α-sequence with optimal makespan $p+1$, giving a competitive ratio of at least $\frac{2p+1}{p+1}$. Assuming that ($i = 2$ and $p \geq 1 + \sqrt{2}$) or ($i = 3$ and $p \geq \frac{1}{2}(1+\sqrt{3})$), this ratio is at least $\frac{i+1}{i} + \frac{1}{ip}$.

Assume now that no two jobs from Phase 2 are scheduled together with a job from Phase 1, and no three jobs from Phase 2 are scheduled together. In this case, the adversary gives $\alpha m - \lceil \frac{n_2}{i} \rceil - \lceil \frac{n_1}{\lfloor ip \rfloor} \rceil$ jobs of length ip in Phase 3. At most $m - n_1$ machines could receive two jobs from Phase 2, so at least $m_2 = n_2 - (m - n_1)$ machines receive at least one job from Phase 2. Assuming that $\alpha \geq (2 + \frac{5}{m} - \frac{2}{mi})/(3 - \frac{1}{i} - \frac{1}{\lfloor ip \rfloor})$, we get that $m_2 + n_3 \geq m+1$, and hence, some job from Phase 3 must be placed with a job from Phase 2 plus another job from either Phase 1 or Phase 2. Thus, the on-line makespan is at least $1 + p + ip$.

Since the jobs from Phase 1 can be scheduled on $\lceil \frac{n_1}{\lfloor ip \rfloor} \rceil$ machines, the Phase 2 jobs on $\lceil \frac{n_2}{i} \rceil$ machines, and the Phase 3 jobs on $\alpha m - \lceil \frac{n_2}{i} \rceil - \lceil \frac{n_1}{\lfloor ip \rfloor} \rceil$ machines, giving a makespan of ip and using only αm machines, the sequence is an α-sequence. Thus,

$$\mathcal{A}_{\mathbb{A}}(\alpha) \geq \frac{1+p+ip}{ip} = \frac{i+1}{i} + \frac{1}{ip}.$$

□

6 Conclusion

The accommodating function for $\alpha \leq 1$ seems to be interesting for a variety of on-line problems, possibly for a greater variety than when $\alpha > 1$. For example, the accommodating function for deterministic algorithms for the Paging Problem has a very uninteresting shape for $\alpha > 1$; the value is constant at k, while for $\alpha \leq 1$, $\mathcal{A}(\alpha) = \frac{k}{(1-\alpha)k+1}$.

The study of the accommodating function in general and in particular for $\alpha \leq 1$ has given rise to new algorithms, Unfair-First-Fit and Unfair-First-Fit$_\alpha$.

In addition, the Seat Reservation Problem demonstrates the utility of the accommodating function with $\alpha < 1$ in distinguishing between different algorithms. Three algorithms, which all have competitive ratio close to $\frac{1}{2}$ on accommodating sequences, have different competitive ratios on $\frac{1}{3}$-sequences.

The proofs of performance guarantees seem, in general, to be more interesting than the proofs of impossibility results. This appears less true for those problems where performance guarantees concerning resource augmentation can be used directly, giving performance guarantees for the accommodating function with $\alpha \leq 1$. However, the original proofs in the resource augmentation setting tend to be interesting, and those results become even more interesting given their application to this setting where the set of request sequences is limited. On the other hand, the accommodating function with $\alpha \leq 1$ sometimes gives much more useful information than resource augmentation. Examples of this were given for the Seat Reservation Problem and Unrestricted Bin Packing, using First-Fit.

7 Acknowledgment

We would like to thank an anonymous referee who helped improving the presentation of the paper.

References

[1] Y. Azar, J. Boyar, L. Epstein, L. M. Favrholdt, K. S. Larsen, and M. N. Nielsen. Fair versus Unrestricted Bin Packing. *Algorithmica*, 34(2):181–196, 2002.

[2] Y. Azar, L. Epstein, and R. van Stee. Resource Augmentation in Load Balancing. In *Proceedings of the 7th Scandinavian Workshop on Algorithm Theory*, volume 1851 of *Lecture Notes in Computer Science*, pages 189–199. Springer-Verlag, 2000.

[3] E. Bach, J. Boyar, L. Epstein, L. M. Favrholdt, T. Jiang, K. S. Larsen, G.-H. Lin, and R. van Stee. Tight Bounds on the Competitive Ratio on Accommodating Sequences for the Seat Reservation Problem. *Journal of Scheduling*, 6(2):131–147, 2003.

[4] S. Ben-David and A. Borodin. A New Measure for the Study of On-Line Algorithms. *Algorithmica*, 11(1):73–91, 1994.

[5] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[6] A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive Paging with Locality of Reference. *Journal of Computer and System Sciences*, 50(2):244–258, 1995. Also in *STOC 91*, pages 249–259.

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

[8] J. Boyar, K. S. Larsen, and M. N. Nielsen. The Accommodating Function: A Generalization of the Competitive Ratio. *SIAM Journal on Computing*, 31(1):233–258, 2001.

[9] M. Brehop, E. Torng, and P. Uthaisombut. Applying Extra Resource Analysis to Load Balancing. *Journal of Scheduling*, 3:273–288, 2000.

[10] M. Chrobak, H. Karloff, T. Payne, and S. Vishwanathan. New Results on Server Problems. *SIAM Journal on Discrete Mathematics*, 4(2):172–181, 1991.

[11] M. Chrobak and M. Ślusarek. On Some Packing Problems Related to Dynamic Storage Allocation. *RAIRO Informatique Théoretique et Applications*, 22:487–499, 1988.

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

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

[14] B. Kalyanasundaram and K. Pruhs. Speed is as Powerful as Clairvoyance. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, pages 214–221, 1995.

[15] C. Kenyon. Best-Fit Bin-Packing with Random Order. In *7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 359–364, 1996.

[16] H. A. Kierstead and J. Qin. Coloring Interval Graphs with First-Fit. *Discrete Mathematics*, 144:47–57, 1995.

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

[18] E. Koutsoupias. Weak Adversaries for the k-Server Problem. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 444–449, 1999.

[19] E. Koutsoupias and C. H. Papadimitriou. Beyond Competitive Analysis. In *35th Annual Symposium on Foundations of Computer Science*, pages 394–400, 1994.

[20] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive Algorithms for Server Problems. *Journal of Algorithms*, 11(2):208–230, 1990.

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

[22] N. Young. On-Line Caching as Cache Size Varies. In *Proceedings of the 2nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 241–250, 1991.

[23] N. E. Young. On-Line File Caching. In *Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 82–86, 1998.