# On-Line Maximizing the Number of Items Packed in Variable-Sized Bins


Leah Epstein[1⋆] and Lene M. Favrholdt[2⋆⋆]

[1] School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. `lea@idc.ac.il`.

[2] Department of Mathematics and Computer Science, University of Southern Denmark. `lenem@imada.sdu.dk`



keywords: On-Line, Bin Packing.



**Abstract.** We study an on-line bin packing problem. A fixed number $n$ of bins, possibly of different sizes, are given. The items arrive on-line, and the goal is to pack as many items as possible. It is known that there exists a legal packing of the whole sequence in the $n$ bins. We consider fair algorithms that reject an item, only if it does not fit in the empty space of any bin. We show that the competitive ratio of any fair, deterministic algorithm lies between $\frac{1}{2}$ and $\frac{2}{3}$, and that a class of algorithms including Best-Fit has a competitive ratio of exactly $\frac{n}{2n-1}$.


## 1 Introduction

*The Problem.* We consider the following bin packing problem. The input consists of $n$ bins, possibly of different sizes, and a sequence of positively sized items. The bins as well as the sizes of the bins are denoted by $B_1, B_2, \ldots, B_n$. The items arrive on-line, i.e., each item must be packed before the next item is seen, and packed items cannot be moved between bins. The goal is to pack as many items


⋆ Research supported in part by the Israel Science Foundation, (grant No. 250/01-1)

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

as possible into the $n$ bins. A bin is legally packed if the total size of the items assigned to it is at most the size of the bin. This problem of maximizing the number of items packed in a fixed number of bins is sometimes called *dual bin packing*, to distinguish it from the classical bin packing problem which is to pack *all* items in as *few* bins as possible. In [8] the problem is reported to have been named dual bin packing in [18]. Note that this name is also sometimes used for bin covering [2, 12, 13]. For a survey on classical bin packing in identical bins, see [14, 11].

Throughout the paper, we restrict the input sequences to be *accommodating* [6, 7], i.e., sequences that an optimal off-line algorithm, which knows all items in advance, can pack completely. The reason for this restriction is that, for general sequences, no on-line algorithm can pack a constant fraction of the number of items that can be packed by an optimal off-line algorithm.

The problem can also be seen as a scheduling problem with $n$ uniformly related machines. In the basic scheduling problem, each job is to be assigned to one of the machines so as to minimize the makespan. This problem was first studied for the case of identical machines by Graham [15], and for uniformly related machines by [1, 10, 4]. For a survey on on-line scheduling problems, see [20]. Consider a scheduling problem with a deadline and assume that the aim is to schedule as many jobs as possible before this deadline. If an optimal off-line algorithm can schedule all jobs of any input sequence before the deadline, this problem is equivalent to our problem. Our problem can also be seen as a special case of the multiple knapsack problem (see [19, 9]), where all items have unit profit. (This problem was mainly studied in the off-line environment.)

*The Algorithms.* In this paper we study fair algorithms [3]. A *fair* algorithm rejects an item, only if the item does not fit in the empty space of any bin.

Some of the algorithms that are classical for the classical bin packing problem (where the whole sequence of items is to be packed in as few bins as possible) can be adapted to our problem. Such an adaptation for identical bins was already done in [7]: the $n$ bins are all considered open from the beginning, and no new bin can be opened. We also use this adaptation. Since there is no unique way to define First-Fit for variable sized bins, we discuss this in Section 3.

*The Quality Measure.* The competitive ratio of an on-line algorithm $\mathbb{A}$ for the dual bin packing problem is the worst case ratio, over all possible input sequences, of the number of items packed by $\mathbb{A}$ to the number of items packed by an optimal off-line algorithm. Often an additive constant is allowed, yielding the following definition of the competitive ratio.

**Definition 1.1.** *For any algorithm $\mathbb{A}$ and any sequence $I$ of items, let $\mathbb{A}(I)$ be the number of items packed by $\mathbb{A}$ and let $OPT(I)$ be the number of items packed by an optimal off-line algorithm. Furthermore, let $0 \leq c \leq 1$. An on-line algorithm $\mathbb{A}$ is $c$-*competitive *if there exists a constant $b$ such that*

$$\mathbb{A}(I) \geq c \cdot OPT(I) - b, \text{ for any sequence } I \text{ of items.}$$

*The* competitive ratio *of $\mathbb{A}$ is*

$$C_{\mathbb{A}} = \sup\{c \mid \mathbb{A} \text{ is } c\text{-competitive}\}.$$

Note that since dual bin packing is a maximization problem, the competitive ratio lies between 0 and 1.

If the additive constant $b$ is zero or negative, the algorithm is called *strictly c-competitive*. The bounds given in this paper are valid for the strict competitive ratio as well as for the competitive ratio in general.

For randomized algorithms, the competitive ratio is defined similarly, but $\mathbb{A}(I)$ is replaced by the expected value of $\mathbb{A}(I)$, $E(\mathbb{A}(I))$.

*The Results.* We show the following results for fair algorithms on accommodating sequences.

- Any fair algorithm has a competitive ratio of at least $\frac{1}{2}$, and the competitive ratio of Worst-Fit is exactly $\frac{1}{2}$.
- A class of algorithms that give preference to smaller bins has a competitive ratio of exactly $\frac{n}{2n-1}$. This class contains Best-Fit as well as the variant of First-Fit that sorts the bins in order of non-decreasing sizes.
- Any fair, deterministic algorithm has a competitive ratio of at most $\frac{2}{3}$, and any fair, randomized algorithm has a competitive ratio of at most $\frac{4}{5}$.

*Previous Work.* Dual bin packing in identical bins has been studied both in the off-line version [17, 16] and in the on-line version for accommodating sequences [6, 7, 3]. Even for identical bins, a restriction on the input sequences is needed in order to be able to achieve a constant competitive ratio [7]. In [7], fair algorithms are considered and it is shown that First-Fit has a competitive ratio of at least $\frac{5}{8}$ on accommodating sequences. An upper bound of $\frac{6}{7}$ for any fair or unfair randomized algorithm is also given. In [3], a $(\frac{2}{3} - \frac{2}{4n+1})$-competitive unfair algorithm is given, the negative result for fair deterministic algorithms is improved to 0.809, and the bound of $\frac{5}{8}$ for First-Fit is shown to be asymptotically tight (the upper bound approaches $\frac{5}{8}$ as $n$ approaches infinity).

## 2 General Results on Fair Algorithms

In this section we show that, on accommodating sequences, the competitive ratio of any fair, deterministic algorithm lies between $\frac{1}{2}$ and $\frac{2}{3}$, and the competitive ratio of any randomized algorithm is at most $\frac{4}{5}$.

### 2.1 Positive Results

The main result of this section is that any fair algorithm is $\frac{1}{2}$-competitive on accommodating sequences. We need the following lemma which is adapted from a similar lemma for identical bins in [7].

**Lemma 2.1.** *For any fair algorithm, the number of rejected items is no larger than the number of accepted items, if the input sequence is accommodating.*

*Proof.* Given an instance of the dual bin packing problem with an accommodating sequence $I$, we define a sequence $I'$ as follows. Each accepted item of size $x$ is replaced by $\lfloor \frac{x}{s} \rfloor$ items of size $s$, where $s$ is the minimum size of any rejected item. Each rejected item is decreased to have size $s$. Clearly, a packing of all items of $I$ defines a legal packing of all items of $I'$, hence $I'$ is also an accommodating sequence.

Let $P$ be the on-line packing of $I$ and let $P'$ be the packing of $I'$ induced by $P$. Note that all items of $I'$ have the same size. Thus, to calculate an upper bound on the number of items rejected we just need to find an upper bound on the number of items of size $s$ that fit in the bins after doing the packing $P'$.

For each bin $B_i$, let $k_i$ denote the number of items in bin $B_i$ in the packing $P$. The empty space in $B_i$ in the packing $P'$ consists of the empty space in $B_i$ in the packing $P$ and the space freed by the rounding down of the items packed in $B_i$. The empty space in $B_i$ in $P$ is less than $s$, since the algorithm is fair, and the total size of each original item was decreased by less than $s$. Thus, the empty space in $B_i$ in $P'$ is strictly less than $s(k_i+1)$. We conclude that the number of rejected items is at most $\sum_{i=1}^{n} k_i$ which is the number of accepted items. □

**Corollary 2.2.** *Any fair algorithm has a competitive ratio on accommodating sequences of at least* $\frac{1}{2}$.

We close this section with an easy lemma that will be needed in Section 2.2 and Section 3. Let $C$ be the set of non-empty bins in the optimal off-line packing. Let $N = |C|$.

**Lemma 2.3.** *Given an accommodating input sequence, any fair algorithm rejects at most* $N-1$ *items.*

*Proof.* If the on-line algorithm does not reject any items, its packing is optimal. Assume now, that at least one item is rejected. Let $s$ be the minimum size of any rejected item. Since the algorithm is fair, the empty space in each bin is less than $s$. Another trivial upper bound on the empty space in any bin $B$ is the size $B$ of the bin. Thus, the total empty space in the on-line packing is strictly less than $Ns + \sum_{B \notin C} B$. The total empty space of OPT is at least $\sum_{B \notin C} B$. Hence, since OPT accepts all items, the total size of all rejected items is strictly less than $Ns$. Since all rejected items are of size at least $s$, there are at most $N-1$ rejected items. □

### 2.2 Negative Results

In this section we show an upper bound of $\frac{2}{3}$ for deterministic, fair algorithms and an upper bound of $\frac{4}{5}$ for randomized, fair algorithms.

We first prove the upper bound of $\frac{2}{3}$ for the strict competitive ratio. This is relatively easy for any $n \geq 2$. Consider for example the following instance with $n-2$ bins of size $\varepsilon$, $0 < \varepsilon < 1$, one bin of size 2, and one bin of size 3. The input sequence consists of two or three items that are all too large for the bins of size $\varepsilon$. The first item has size 1. If this first item is assigned to the bin of size 3, an item of size 3 arrives next. Otherwise, two items of size 2 will arrive. In the first case, only the first item is packed, since the second does not fit, and in the second case only two items are accepted, the third does not fit. It is easy to see that both sequences are accommodating. This gives an upper bound of $\frac{2}{3}$ on the strict competitive ratio, for $n \geq 2$. Applying Yao's inequality [21] as described in [5] on these two sequences gives an upper bound of $\frac{4}{5}$ on the strict competitive ratio for randomized algorithms. This can be seen in the following way. Consider the sequence where the first item of size 1 is followed by one item of size 3 with probability $p_1 = \frac{2}{5}$ and by two items of size 2 with probability $p_2 = \frac{3}{5}$. An algorithm that packs the first item in the bin of size 3 will have an expected performance ratio of at most $p_1 \cdot \frac{1}{2} + p_2 \cdot 1 = \frac{4}{5}$. Similarly, an algorithm that packs the first item in the bin of size 2 will have an expected performance ratio of at most $p_1 \cdot 1 + p_2 \cdot \frac{2}{3} = \frac{4}{5}$. Thus, no deterministic algorithm can have an expected performance ratio larger than $\frac{4}{5}$ on this sequence.

However, we are interested in negative results that hold for the competitive ratio in general, and not only for the strict competitive ratio. By Lemma 2.3, the number of rejected items is at most $n-1$. As long as there is only a constant number of bins, we can view the number of rejected items as just an additive constant, and hence any fair algorithm has competitive ratio 1. Thus, to prove the following theorem, we need to find arbitrarily long accommodating sequences with the property that only $\frac{2}{3}$ of the items are accepted.

**Theorem 2.4.** *Any fair, deterministic on-line algorithm for the dual bin packing problem has a competitive ratio of at most* $\frac{2}{3}$ *on accommodating sequences.*

*Proof.* For $\ell = 1, \ldots, \lfloor \frac{n}{2} \rfloor$, we give the pair of bins

$$B_{2\ell-1} = 2\ell + 4^{\ell}\varepsilon \text{ and } B_{2\ell} = 2\ell + 2 \cdot 4^{\ell}\varepsilon,$$

where $\varepsilon < \frac{1}{4^n}$ is a positive constant. Thus, $4^{\ell}\varepsilon < 1$, $1 \leq \ell \leq \lfloor \frac{n}{2} \rfloor$. If $n$ is odd, the last bin is of size $\frac{\varepsilon}{2}$ (so that no items are packed in that bin for the sequence we define). The sequence contains $3 \cdot \lfloor \frac{n}{2} \rfloor$ items and is constructed so that exactly $2 \cdot \lfloor \frac{n}{2} \rfloor$ of them are accepted.

The sequence is defined inductively in steps $\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor - 1, \ldots, 1$. In step $k$, two large items are given and one small item is defined. The small items are given after all large items and are defined such that they will be rejected by the on-line algorithm. The sizes of the two large items are defined such that

- the on-line algorithm will pack them in $B_{2k}$ and $B_{2k-1}$, one in each bin, and
- after packing the two items, the empty space in the two bins have the same size denoted $E_k$.

For convenience we define $E_{\lfloor \frac{n}{2} \rfloor + 1} = 0$. As will be seen later, $E_{k+1} < E_k$, $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$. Furthermore, we will prove that $E_1 < 1$.

The first large item given in step $k$ has size $2k - E_{k+1}$. Thus, the very first item has size $2 \cdot \lfloor \frac{n}{2} \rfloor$, and the size of the first large item of each of the later steps depends on the empty space created in the previous step. Since $2k - E_{k+1} > 2k - 1$ and all previous bins

$B_n, \ldots, B_{2k+1}$ have less than one unit of empty space, this item fits only in $B_{2k}$ and $B_{2k-1}$. What happens next depends on which of these two bins the algorithm chooses.

*Case 1: The first large item is packed in $B_{2k-1}$.* In this case, the next large item has size $2k - E_{k+1} + 4^k\varepsilon$. This item will be packed in $B_{2k}$. Now, the empty space in each of the bins $B_{2k}$ and $B_{2k-1}$ is $E_k = E_{k+1} + 4^k\varepsilon$. The small item defined in this step has size $S_k = E_k + 4^k\varepsilon$. Note that this item does not fit in $B_{2k}$ or $B_{2k-1}$, but the off-line algorithm can pack the first large item in $B_{2k}$ together with the small item and put the second large item in $B_{2k-1}$.

*Case 2: The first large item is packed in $B_{2k}$.* In this case, the next large item has size $2k - E_{k+1} - 4^k\varepsilon$. For $k \geq 2$, this item does not fit in $B_{2k-2}$, since $2k - E_{k+1} - 4^k\varepsilon > 2k - 1 - 4^k\varepsilon \geq 2k - 2 + 3 \cdot 4^k\varepsilon$, for $n \geq 2$, and $B_{2k-2} = 2k - 2 + 2 \cdot 4^{k-2}\epsilon$. Hence, this item must be packed in $B_{2k-1}$. Now, the empty space in each of the bins $B_{2k}$ and $B_{2k-1}$ is $E_k = E_{k+1} + 2 \cdot 4^k\varepsilon$. The small item defined in this step has size $S_k = E_k + 4^k\varepsilon$. This item does not fit in $B_{2k}$ or $B_{2k-1}$, but the off-line algorithm can pack the first large item in $B_{2k-1}$ and put the second large item in $B_{2k-1}$ together with the small item.

Note that $E_{k+1} + 4^k\varepsilon \leq E_k \leq E_{k+1} + 2 \cdot 4^k\varepsilon$, $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$. The first inequality tells us that, to prove that none of the small items will be accepted, it suffices to prove that $S_k > E_1$, $2 \leq k \leq \lfloor \frac{n}{2} \rfloor$. This is easily done using the second inequality. For $2 \leq k \leq \lfloor \frac{n}{2} \rfloor$,

$$E_1 \leq E_k + 2 \cdot \sum_{i=1}^{k-1} 4^i\varepsilon < E_k + 4^k\varepsilon = S_k.$$

Finally,

$$E_1 \leq E_{\lfloor \frac{n}{2} \rfloor + 1} + 2 \cdot \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} 4^i \varepsilon < 4^{\lfloor \frac{n}{2} \rfloor + 1} \varepsilon < 4^{\lfloor \frac{n}{2} \rfloor + 1 - n} \leq 1.$$

□

We move on to randomized algorithms. Since the previous sequence was built step by step, we need to give a simpler sequence in order to prove the following theorem.

**Theorem 2.5.** *Any fair randomized algorithm has a competitive ratio on accommodating sequences of at most* $\frac{4}{5}$.

*Proof.* We use $\lfloor \frac{n}{2} \rfloor$ bins of size $1+\varepsilon$ and $\lfloor \frac{n}{2} \rfloor$ bins of size $2-\varepsilon$, where $0 < \varepsilon < \frac{1}{2}$. If $n$ is odd, the last bin is of size $\varepsilon$. The sequence starts with $\lfloor \frac{n}{2} \rfloor$ items of size 1. We describe a proof for deterministic algorithms first. Since the algorithm is fair, all $\lfloor \frac{n}{2} \rfloor$ items are accepted. Let $x$ be the number of bins of size $1+\varepsilon$ that received an item (no bin can receive more than one item). Then, exactly $x$ bins of size $2-\varepsilon$ are empty. What happens next depends on the size of $x$.

*Case* $x \leq \frac{3}{5} \cdot \lfloor \frac{n}{2} \rfloor$. In this case, the sequence continues with $\lfloor \frac{n}{2} \rfloor$ items of size $2-\varepsilon$, and the on-line algorithm accepts $\lfloor \frac{n}{2} \rfloor + x$ items in total out of the $2\lfloor \frac{n}{2} \rfloor$. This gives a fraction of

$$\frac{\lfloor \frac{n}{2} \rfloor + x}{2\lfloor \frac{n}{2} \rfloor} \leq \frac{1+\frac{3}{5}}{2} = \frac{4}{5}.$$

*Case* $x > \frac{3}{5} \cdot \lfloor \frac{n}{2} \rfloor$. In this case, the sequence continues with $\lfloor \frac{n}{2} \rfloor$ items of size $1+\varepsilon$ followed by $\lfloor \frac{n}{2} \rfloor$ items of size $1-\varepsilon$. After the arrival of items of size 1, there are $\lfloor \frac{n}{2} \rfloor$ empty bins. Thus, all items of size $1+\varepsilon$ are accepted and now each bin has exactly one item. Items of size

$1-\varepsilon$ can only be assigned to bins of size $2-\varepsilon$ that contain an item of size 1, hence $\lfloor \frac{n}{2} \rfloor - x$ of them are accepted. Thus, the fraction

$$\frac{3\lfloor \frac{n}{2} \rfloor - x}{3\lfloor \frac{n}{2} \rfloor} < \frac{3-\frac{3}{5}}{3} = \frac{4}{5}$$

of the items is accepted.

To get a randomized result, let $x$ be the expectation of the number of bins of size $1+\varepsilon$ that got an item. The bound follows by linearity of expectation. □

## 3 Results on Specific Fair Algorithms

We now analyze specific algorithms. Some natural fair algorithms are First-Fit, Best-Fit, and Worst-Fit. The algorithm First-Fit is not a single algorithm, but a class of algorithms that give an order to the bins, and use the algorithm according to this order, i.e., assign an item to the first bin (in the ordered set of bins) that the item fits in. Among the various versions of First-Fit, two are most natural; *Smallest-Fit* assigns an item to the smallest bin it fits into, and *Largest-Fit* assigns an item to the largest bin it fits into. The other algorithms are used in their classical version, i.e., Best-Fit packs each item in a bin where it will leave the smallest possible empty space, and Worst-Fit packs it in the bin where it leaves the largest empty space. We refer to these four algorithms as SF, LF, BF, and WF.

We start the analysis by showing that $\frac{1}{2}$ is indeed the exact competitive ratio of WF and LF.

**Theorem 3.1.** *The competitive ratio of Worst-Fit and Largest-Fit on accommodating sequences is* $\frac{1}{2}$.

*Proof.* Let $\varepsilon > 0$ be a constant such that $\varepsilon \leq \frac{1}{n}$. Consider the following set of bins. One large bin of size $n$ and $n-1$ small bins of size

1. The sequence consists of $n-1$ items of size 1 followed by $n-1$ items of size $1+\varepsilon$. Both algorithms LF and WF assign all items of size 1 to the large bin. As a result, all bins have a free space of size 1, hence none of the items of size $1+\varepsilon$ can be accepted. The optimal algorithm assigns each small item to a small bin, and all other items to the large bin; they all fit since

$$(1+\varepsilon)(n-1) \leq \frac{(n+1)(n-1)}{n} < n \ .$$

This example in combination with Corollary 2.2 proves the theorem. □

We further analyze a class of fair algorithms called *Smallest-Bins-First* to which SF and BF belong. This is the class of fair algorithms that whenever an item is assigned to an empty bin, this is the smallest bin in which the item fits. There are no additional rules, and the algorithm may use an empty bin even if the item fits in a non-empty bin, as long as it uses the smallest empty bin for that. SF belongs to this class according to its definition. BF belongs to this class since, among the empty bins that an item fits into, it fits better into the smaller bins than the larger bins. We give a tight analysis of this class as a function of $n$. Specifically we prove the following.

**Theorem 3.2.** *The competitive ratio of any Smallest-Bins-First algorithm on accommodating sequences is* $\frac{n}{2n-1}$.

*Proof.* If, after running the algorithm, all bins of the on-line algorithm are non-empty, then there are at least $n$ accepted items and at most $n-1$ rejected items (by Lemma 2.3). Thus, in this case, the competitive ratio is at least $\frac{n}{2n-1}$.

Otherwise, consider the largest (last) bin $b$ that remained empty after running the on-line algorithm. We consider items of size smaller than or equal to $b$, and items larger than $b$ separately. Since a bin of size $b$ is empty and no bin larger than $b$ is empty, according to the definition of the class of algorithms, each bin of size more than $b$ contains at least one item larger than $b$, namely the first item packed in the bin. Moreover, all items of size at most $b$ are accepted. Let $x_s$ be the number of items in bins of size at most $b$ and let $n_\ell$ be the number of bins larger than $b$. Let $N_s$ be the number of non-empty bins of OPT of size at most $b$ and $N_\ell$ its number of non-empty bins larger than $b$. Clearly, $x_s \geq N_s$ (all those bins are of size at most $b$ and contain at least one item). We get that the number of accepted items is at least $x_s + n_\ell \geq N_s + N_\ell = N$. Thus, by Lemma 2.3, the competitive ratio is at least $\frac{N}{2N-1} \geq \frac{n}{2n-1}$.

To show that the result is tight for this class of algorithms, let $\varepsilon < \frac{1}{n}$ be a positive constant. Consider the set of bins $B_i = 1+\varepsilon i$, $i = 1, \ldots, n$. The sequence consists of $n$ items, one of size $1+\varepsilon(i-1)$ for each $i = 1, \ldots, n$, followed by $n-1$ items of size $\frac{n\varepsilon}{n-1}$. All algorithms in the class assign the item of size $1+\varepsilon(i-1)$ to $B_i$. All other items are rejected. The optimal off-line algorithm assigns each large item except the first one to a bin of its size. The first item and the $n-1$ small items are assigned to $B_n$. $\square$

Note that when $n = 2$, the lower bound of $\frac{n}{2n-1}$ matches the general upper bound of $\frac{2}{3}$.

## 4 Conclusion

We have proven an upper bound of $\frac{2}{3}$ for all fair algorithms. We have also shown that any fair algorithm accepts at least half of the items, and that some algorithms do significantly better for very small $n$. It is left as an open problem to design a fair algorithm with a competitive ratio significantly larger than $\frac{1}{2}$ for any $n$, or prove that this is not possible. It is also unknown how much unfair algorithms can be better; the best negative result for those is $\frac{6}{7}$, which holds even for identical bins [7].

**Acknowledgment:** We would like to thank Joan Boyar for reading and commenting on the paper. We also thank the editor Gerhard Woeginger for suggesting the title.

## References

1. J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-Line Routing of Virtual Circuits with Applications to Load Balancing and Machine Scheduling. *Journal of the ACM*, 44(3):486–504, 1997. Also in *Proc. 25th ACM STOC*, 1993, pp. 623-631.
2. S. F. Assmann, D. S. Johnson, D. J. Kleitman, and J. Y. Leung. On a Dual Version of the One-Dimensional Bin Packing Problem. *Journal of Algorithms*, 5:502–525, 1984.
3. Y. Azar, J. Boyar, L. Epstein, L. M. Favrholdt, K. S. Larsen, and M. N. Nielsen. Fair versus Unrestricted Bin Packing. *Algorithmica (to appear)*. Preliminary version at SWAT 2000, volume 1851 of *LNCS*: 200-213, Springer-Verlag, 2000.
4. P. Berman, M. Charikar, and M. Karpinski. On-Line Load Balancing for Related Machines. *Journal of Algorithms*, 35:108–121, 2000.
5. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
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. J. L. Bruno and P. J. Downey. Probabilistic Bounds for Dual Bin-Packing. *Acta Informatica*, 22:333–345, 1985.
9. C. Chekuri and S. Khanna. A PTAS for the Multiple Knapsack Problem. In *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 213–222, 2000.
10. Y. Cho and S. Sahni. Bounds for List Schedules on Uniform Processors. *SIAM Journal on Computing*, 9:91–103, 1988.

11. E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation Algorithms for Bin Packing: A Survey. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 2, pages 46–93. PWS Publishing Company, 1997.
12. J. Csirik and J. B. G. Frenk. A Dual Version of Bin Packing. *Algorithms Review*, 1:87–95, 1990.
13. J. Csirik and V. Totik. On-Line Algorithms for a Dual Version of Bin Packing. *Discr. Appl. Math.*, 21:163–167, 1988.
14. J. Csirik and G. Woeginger. On-Line Packing and Covering Problems. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms*, volume 1442 of *LNCS*, chapter 7, pages 147–177. Springer-Verlag, 1998.
15. R. L. Graham. Bounds for Certain Multiprocessing Anomalies. *Bell Systems Technical Journal*, 45:1563–1581, 1966.
16. E. G. Coffman Jr. and J. Y. Leung. Combinatorial Analysis of an Efficient Algorithm for Processor and Storage Allocation. *SIAM Journal on Computing*, 8(2):202–217, 1979.
17. E. G. Coffman Jr. J. Y. Leung, and D. W. Ting. Bin Packing: Maximizing the Number of Pieces Packed. *Acta Informatica*, 9:263–271, 1978.
18. J. Y. Leung. *Fast Algorithms for Packing Problems.* PhD thesis, Pennsylvania State University, 1977.
19. S. Martello and P. Toth. *Knapsack Problems.* John Wiley and Sons, Chichester, 1990.
20. J. Sgall. On-Line Scheduling. In *A. Fiat and G. J. Woeginger, editors,* Online Algorithms: The State of the Art, volume 1442 of *LNCS*, pages 196–231. Springer-Verlag, 1998.
21. A. C. Yao. Towards a Unified Measure of Complexity. *Proc. 12th ACM Symposium on Theory of Computing*, pages 222–227, 1980.