The Maximum Resource Bin Packing Problem*

Joan Boyar [1] Leah Epstein [2] Lene M. Favrholdt [1] Jens S. Kohrt [1]
Kim S. Larsen [1] Morten M. Pedersen [1] Sanne Wøhlk [3]

[1] Department of Mathematics and Computer Science
University of Southern Denmark, Odense, Denmark
{joan,lenem,svalle,kslarsen,mortenm}@imada.sdu.dk

[2] Department of Mathematics
University of Haifa, 31905 Haifa, Israel
lea@math.haifa.ac.il

[3] Department of Accounting, Finance, and Logistics
Aarhus School of Business, Aarhus, Denmark
sanw@asb.dk

Abstract

Usually, for bin packing problems, we try to minimize the number of bins used or in the case of the dual bin packing problem, maximize the number or total size of accepted items. This paper presents results for the opposite problems, where we would like to maximize the number of bins used or minimize the number or total size of accepted items. We consider off-line and on-line variants of the problems.

For the off-line variant, we require that there be an ordering of the bins, so that no item in a later bin fits in an earlier bin. We find the approximation ratios of two natural approximation algorithms, First-Fit-Increasing and First-Fit-Decreasing for the maximum resource variant of classical bin packing.

For the on-line variant, we define maximum resource variants of classical and dual bin packing. For dual bin packing, no on-line algorithm is competitive. For classical bin packing, we find the competitive ratio of various natural algorithms.

We study the general versions of the problems as well as the parameterized versions where there is an upper bound of $\frac{1}{k}$ on the item sizes, for some integer k.

*The work of Boyar, Favrholdt, Kohrt, and Larsen was supported in part by the Danish Natural Science Research Council (SNF). The work of Epstein was supported in part by the Israel Science Foundation (ISF). A preliminary version of this paper appeared in the *Fifteenth International Symposium on Fundamentals of Computation Theory*, volume 3623 of Lecture Notes in Computer Science, pages 397–408. Springer-Verlag, 2005.

1 Introduction

Many optimization problems involve some resource, and the task for algorithm designers is typically to get the job done using the minimum amount of resources. Below, we give some examples.

Bin packing is the problem of packing items of sizes between zero and one in the smallest possible number of bins of unit size. Here, the bins are the resources. The traveling salesperson problem is the problem of finding a tour which visits each vertex in a weighted graph while minimizing the total weight of visited edges. Here the weight is the resource. Scheduling jobs on a fixed number of machines is the problem of minimizing the completion time of the last job. Here time is the resource.

Each of these problems comes in many variations and there are many more entirely different optimization problems. Since these problems are computationally hard assuming that P $\neq$ NP, the optimal solution can usually not be computed in reasonable time for large instances, so polynomial time approximation algorithms are devised. For many of these problems, there are interesting variants where the entire instance is not known when the computation must commence. The area of on-line algorithms deals with this problem scenario.

For detailed descriptions of many of these problems and their solutions in terms of approximation or on-line algorithms, see [4, 10, 13], for instance.

For all of these problems, minimizing the resources used seems to be the obvious goal. However, if the resource is not owned by the problem solver, but is owned by another party who profits from selling the resource, there is no longer agreement about the objective, since the owner of the resource wants to maximize the resources used, presumably under some constraints which could be outlined in a contract. Thus, many of the classical problems are interesting also when considered from the reverse perspective of trying to maximize the amount of resources that are used.

In [1], the Lazy Bureaucrat Scheduling Problem is considered. Here, tasks must be scheduled and processed by an office worker. The authors consider various constraints and objective functions. The flavor of the constraints is that the office worker cannot sit idle if there is work that can be done, and the office worker's objective is to schedule tasks under these constraints so as to minimize the work carried out; either total work, arranging to leave work as early as possible, or a similar goal. Though it is presented as a toy problem, it is an important view on some optimization problems, and many other problems are interesting in this perspective, provided that the constraints imposed on the problem are natural.

Also other problems have been investigated in this reverse perspective, e.g., longest path [16], maximum traveling salesperson problem [12] and lazy online interval coloring [9].

Maximum Resource Bin Packing

In this paper, we consider bin packing from the maximum resource perspective. We consider it as an approximation problem, but we also investigate two on-line variants of the problem. To our knowledge, this is the first time the reverse perspective of a problem has been considered in an

on-line setting. Note that the complexity status of the off-line problems studied in this paper is open.

The abstract problem of packing items of a given size into bins has numerous concrete applications, and for many of these, when the resource must be purchased, the reverse problem becomes interesting for one of the parties involved. We use the following concrete problem for motivation.

Assume that we hire a company to move some items by truck from one site, the origin, to another, the destination. Say that the price we must pay is proportional to the number of trucks used. Some companies may try to maximize the number of trucks used instead of trying to get the items packed in few trucks. To prevent the company from cheating us, the following constraint has been placed on the packing procedure:

Constraint 1: When a truck leaves the origin, none of the unpacked items remaining at the origin should fit into that truck.

In the off-line variant, *Off-Line Maximum Resource Bin Packing*, we are given an unlimited number of unit sized bins and a sequence of items with sizes in $(0, 1]$, and the goal is to maximize the number of bins used to pack all the items subject to Constraint 1. A set of items fits in a bin if the sum of the sizes of the items is at most one. In the off-line variant, there must be an ordering of the bins such that no item in a later bin fits in an earlier bin. Explained using the motivating example, Constraint 1 can be illustrated as follows: Trucks arrive at the origin one at a time. A truck is loaded, and may leave for its destination when none of the remaining items can fit into the truck. At this time, the next truck may arrive.

On-Line Maximum Resource Bin Packing is similar to the off-line version. However, the problem is on-line, meaning that items are revealed one at a time, and each item must be processed before the next item becomes available. Because of the on-line nature of the problem, instead of Constraint 1, the following modified constraint is used:

Constraint 2: The company is not allowed to begin using a new truck if the current item fits in a truck already being used.

Thus, the on-line algorithm is allowed to open a new bin every time the next item to be processed does not fit in any of the previous bins. The objective is still to use as many bins as possible. Thus, all partly loaded trucks are available all the time, and whenever an item does not fit, a new truck may pull up to join the others.

We also consider another on-line problem, *On-Line Dual Maximum Resource Bin Packing*. Here, the number of available bins is fixed. For each item, an algorithm has to accept it and place it in one of its bins, if it is possible to do so. Thus, here a fixed number of trucks have been ordered. In this case, neither Constraint 1 nor Constraint 2 is used; the objective is not to maximize the number of trucks used, since this number is fixed. There are two possible objective functions: the number of accepted items or the total size of the accepted items. In both cases, the objective is to minimize this value. Thus, the truck company wants to pack as few items or as little total size as possible

into the trucks, minimizing the fuel needed for each truck, or maybe hoping to get a new order of trucks for the items which do not fit into the fixed number of trucks which have been ordered.

For all three problems, we study the general version as well as the parameterized version where there is an upper bound of $\frac{1}{k}$ on the item sizes, for some integer k.

A closely related problem is the Bin Covering Problem. In this problem, the algorithm is given a sequence of items and has to place them in bins, while trying to maximize the number of bins that contain items with a total size of at least one. This is quite similar to Off-Line Maximum Resource Bin Packing with bins twice as large and Constraint 1 replaced by the following weaker constraint:

Constraint 3: No pair of trucks leaving the origin may have a total load of items that could have been packed in one truck.

The problem is NP-complete but has an asymptotic fully polynomial time approximation scheme (AFPTAS) [14]. Further results on that problem can be found in [2, 7, 8].

Our Results

For Off-Line Maximum Resource Bin Packing, we show that no algorithm has an approximation ratio of more than $\frac{17}{10}$. For the parameterized version, the upper bound is $1 + \frac{1}{k}$ for $k \geq 2$. The algorithm First-Fit-Decreasing is worst possible in the sense that it meets this upper bound. First-Fit-Increasing is better; it has a competitive ratio of $\frac{6}{5}$ and a parameterized competitive ratio of $1 + \frac{k-1}{k^2+1}$ for $k \geq 2$. See Section 2 for a definition of the algorithms.

For On-Line Maximum Resource Bin Packing, we prove a general lower bound of $\frac{3}{2}$ on the parameterized competitive ratio for $k \leq 3$ and $1 + \frac{1}{k-1}$ for $k \geq 3$. We prove a general upper bound of 2 for $k \leq 2$ and $1 + \frac{1}{k-1}$ for $k \geq 2$. Hence, for $k \geq 3$, all algorithms have the same parameterized competitive ratio. We prove that First-Fit, Best-Fit, and Last-Fit all meet the general upper bound.

For On-Line Maximum Resource Dual Bin Packing, we prove that if the objective function is the total *number* of items packed, no deterministic algorithm is competitive; this also holds for any value of k for the parameterized problem. If the objective function is the total *size* of the packed items, no algorithm for the general problem is competitive. For the parameterized version, we prove general lower and upper bounds of $1 + \frac{1}{e(k-1)}$ and $1 + \frac{1}{k-1}$, respectively.

The proof of Theorem 3, below, showing that for Off-Line Maximum Resource Bin Packing, the approximation ratio of First-Fit-Increasing is $\frac{6}{5}$, uses a new variant of the standard weighting argument. That result also gives a connection between Off-Line Maximum Resource Bin Packing and the relative worst order ratio for on-line algorithms for the classical bin packing problem. The relative worst order ratio [5] is a new measure for the quality of on-line algorithms. Theorem 3 has been used to prove the upper bound on a result comparing First-Fit to Harmonic(k) using the relative worst order ratio [6]. Perhaps other "reverse" problems will have similar connections to the relative worst order ratio.

2 Notation and Algorithms

The input is a sequence of items, $I = \langle s_1, s_2, \ldots, s_n \rangle$. For convenience, we identify an item with its size and require that item s_i has size $0 < s_i \leq 1$ (or $0 < s_i \leq \frac{1}{k}$, for some integer k, for the parameterized problem). The items have to be placed in bins of size one.

For any input sequence I, let *ALG*(I) be both the packing produced when running *ALG* on this input sequence and the number of bins used for this packing. In particular, let *OPT* be an algorithm which produces an optimal packing and thus uses as many bins as possible subject to Constraint 1. Let *OPT*(I) be both this packing and the number of bins used.

Let *SMALL*(I) be a packing using the minimum number of bins that the items from I can be packed in without putting items with sizes totaling more than one in any bin, and let *SMALL* be an algorithm that creates this packing. Note that *SMALL* is an optimal algorithm from the classical bin packing problem, but for Maximum Resource Bin Packing, it is a worst possible algorithm.

An approximation algorithm *ALG* is a *c-approximation algorithm*, $c \geq 1$, if there is a constant b such that for all possible input sequences I, *OPT*$(I) \leq c$ *ALG*$(I) + b$. The infimum of all such c is called the *approximation ratio* of the algorithm, $\mathcal{R}_{ALG}$. For the parameterized problem, we consider the *parameterized approximation ratio*, $\mathcal{R}_{ALG}(k)$, which is the approximation ratio in the case where all items have size at most $\frac{1}{k}$ for some integer k.

An important algorithm in this context is First-Fit (*FF*), which places an item in the first bin in which it fits.

In this paper, we investigate two well known off-line variants of *FF* in detail:

- *First-Fit-Increasing* (*FFI*) handles items in non-decreasing order with respect to their sizes, placing them using First-Fit.
- *First-Fit-Decreasing* (*FFD*) handles items in non-increasing order with respect to their sizes, also placing them using First-Fit.

In the on-line variants of the problem, the algorithms receive the input, i.e., the items, one at a time and have to decide where to pack the item before the next item (if any) is revealed.

Similarly to the approximation ratio for approximation algorithms, the performance of deterministic on-line algorithms is measured in comparison with the optimal off-line algorithm *OPT* [11, 17, 18]. An on-line algorithm *ALG* is *c-competitive*, $c \geq 1$, if there is a constant b such that for all possible input sequences I, *OPT*$(I) \leq c$ *ALG*$(I) + b$. The infimum of all such c is called the *competitive ratio* of the algorithm, $\mathcal{C}_{ALG}$. For the parameterized problem, we consider the *parameterized competitive ratio*, $\mathcal{C}_{ALG}(k)$, which is the competitive ratio in the case where all items have size at most $\frac{1}{k}$ for some integer k.

For the on-line variants, we consider the following natural algorithms, all of which, except for Last-Fit, have been well studied in other contexts:

- *First-Fit* (*FF*) as defined previously.

- *Last-Fit* (*LF*) is the opposite of *FF*, i.e., it places a new item in the last opened bin in which it fits.
- *Best-Fit* (*BF*) places the item in a feasible bin which is as full as possible, i.e., a feasible bin with least free space.
- *Worst-Fit* (*WF*) is the opposite of *BF*, i.e., it places an item in a feasible bin with most free space.

3 Off-Line Maximum Resource Bin Packing

For Off-Line Maximum Resource Bin Packing, the goal is to maximize the number of bins used, subject to Constraint 1, so there must be an ordering of the bins such that no item placed in a later bin fits in an earlier bin. We show that no algorithm for the problem has an approximation ratio worse than $\frac{17}{10}$. Then, we use the proof that for classical Bin Packing, First-Fit's approximation ratio is $\frac{17}{10}$ [15] to prove that *FFD* has this worst possible approximation ratio. After that we show that *FFI* has a better ratio of $\frac{6}{5}$.

The first two theorems show a relation between the classical minimization problem for bin packing and this maximization problem. When considering a worst possible algorithm for Off-Line Maximum Resource Bin Packing, one considers the algorithm, *SMALL*, the optimal algorithm for the minimization problem.

Theorem 1 (General upper bound)**.** *For the Off-Line Maximum Resource Bin Packing Problem, any algorithm ALG has a parameterized approximation ratio of*

$$\mathcal{R}_{ALG}(k) \leq \begin{cases} \dfrac{17}{10}, & k = 1 \\ 1 + \dfrac{1}{k}, & k \geq 2. \end{cases}$$

Proof. Consider any multiset of requests, I. The minimum number of bins, m, *ALG* could use on I is no less than the number of bins used by *SMALL*.

Consider *OPT*'s packing of I, and create an ordered list I' containing the items in I, starting with the items *OPT* packed in the first bin, followed by those in the second bin, etc., until all items have been included.

By the restrictions on what an algorithm may do, First-Fit packs the items in I' exactly as *OPT* packed the items of I. By [15], the number of bins, n, used by First-Fit is at most $\frac{17}{10}m + 2$, if $k = 1$, and at most $(1 + \frac{1}{k})m + 2$, if $k \geq 2$. Thus, *OPT* uses at most $\frac{17}{10}m + 2$ bins, if $k = 1$, and at most $(1 + \frac{1}{k})m + 2$ bins, if $k \geq 2$, giving the stated ratio. □

Note that the packing by *SMALL* is the optimal result from the point of view of a client of the truck company. Thus, Theorem 1 implies that if a client checks that Constraint 1 is obeyed, it will pay

no more than about 1.7 times what it would pay if the truck company was serving the client's best interests. This gives a bound on how bad such a contract can be from the client's point of view.

We now show that Theorem 1 is tight: First-Fit-Decreasing has this approximation ratio.

Theorem 2. *For the Off-Line Maximum Resource Bin Packing Problem, the parameterized approximation ratio of FFD is*

$$\mathcal{R}_{FFD}(k) = \begin{cases} \frac{17}{10}, & k = 1 \\ 1 + \frac{1}{k}, & k \geq 2. \end{cases}$$

Proof. The upper bounds follow from the previous theorem. To prove the lower bounds, we use the sequences and proofs from [15]. We observe that *FFD* produces essentially the same packing as *SMALL* (*OPT* in the proofs in [15]). The packings produced by *FF* in [15] satisfy Constraint 1, so the results in [15] show that for every positive integer K there exist sequences where *FFD* uses at most $10K + 1$ bins while *OPT* uses $17K$ bins. For the parameterized problem, *FFD* uses K bins and *OPT* uses at least $K + \frac{K}{k}$ bins. The result follows. □

We now turn to the better algorithm, *FFI*.

Theorem 3. *The parameterized approximation ratio of FFI is*

$$\mathcal{R}_{FFI}(k) = \begin{cases} \frac{6}{5}, & k \leq 3 \\ \frac{k^2 + k}{k^2 + 1}, & k \geq 4. \end{cases}$$

We prove the lower bound first.

Lemma 4. *The parameterized approximation ratio of FFI is*

$$\mathcal{R}_{FFI}(k) \geq \begin{cases} \frac{6}{5}, & k \leq 3 \\ \frac{k^2 + k}{k^2 + 1}, & k \geq 4. \end{cases}$$

Proof. For $k \leq 2$ we use the following input: n items of size $\frac{1}{2}$ and n items of size $\frac{1}{3}$, where n is a large integer divisible by 6. The optimal packing is to put one item of size $\frac{1}{2}$ and one item of size $\frac{1}{3}$ in each bin. This makes *OPT* use n bins, each with a fraction of $\frac{1}{6}$ empty space. On the other hand, *FFI* packs $\frac{n}{3}$ bins, each containing three elements of size $\frac{1}{3}$, followed by $\frac{n}{2}$ bins, each with two items of size $\frac{1}{2}$. Hence, in total, *FFI* uses $\frac{5n}{6}$ bins, and the ratio follows.

For $k = 3$ (and $k = 2$) the fraction $\frac{k^2+k}{k^2+1}$ equals $\frac{6}{5}$, so we can prove this case with the $k \geq 4$ case. For $k \geq 3$ we use a slightly more complicated sequence. Let n be a large integer. The input contains

- $n(k^2-1)$ items of size $\frac{1}{k+1}$ and
- $n(k+1)$ items of size $\frac{1}{k}$.

FFI uses $n(k-1)$ bins for the smaller items and $n(k+1)/k$ bins for the larger ones, which is $n(k^2+1)/k$ in total. All the bins are completely full. An optimal packing would be to combine one larger item with $k-1$ smaller ones, using $n(k+1)$ bins. Each bin is thus full by a fraction of $\frac{1}{k}+\frac{k-1}{k+1} > \frac{k}{k+1}$, which makes the packing valid. The approximation ratio for this sequence is thus exactly $\frac{k^2+k}{k^2+1}$. □

Note that in this case, *FFI*'s packing is actually the same as the packing made by *SMALL*. This is not always the case, though.

Lemma 5. *The parameterized approximation ratio of FFI is*

$$\mathcal{R}_{FFI}(k) \leq \begin{cases} \frac{6}{5}, & k \leq 3 \\ \frac{k^2+k}{k^2+1}, & k \geq 4. \end{cases}$$

Proof. We first prove the case $k \leq 3$ which is slightly different from the other cases and has to be treated separately. For this part of the proof, we do not assume an upper bound on the item sizes.

We assign weights to items in the following way. For all items in the interval $(0, \frac{1}{6}]$ (small items), the weight is defined to be equal to the size. An item which belongs to an interval $(\frac{1}{i+1}, \frac{1}{i}]$ for some $i = 1, 2, 3, 4, 5$ (large items), is assigned the weight $\frac{1}{i}$.

The intuition for this weighting comes from considering a bin in the packing made by *FFI* that contains only items from a single interval $(\frac{1}{i+1}, \frac{1}{i}]$, $i \in \{1, 2, 3, 4, 5\}$. This bin contains at most i items, and therefore each item in such a bin can be thought of as contributing $\frac{1}{i}$ to the total size of items plus empty space in *FFI*'s packing.

Let W be the total weight of the items in a given input sequence. We prove that $\mathit{FFI} + 5 \geq W \geq \frac{5}{6}(\mathit{OPT} - 5)$, which implies the upper bound.

Consider first the optimal solution *OPT*. We show that the total weight of items is at least $W \geq \frac{5}{6}(\mathit{OPT} - 5)$. To show that, we claim that all bins in *OPT*, except for at most five bins, have items of weight at least $\frac{5}{6}$. First, consider the bins containing at least one small item. Due to Constraint 1, there is at most one such bin whose total sum of item sizes is less than $\frac{5}{6}$. Note that the weight of an item is at least its size. A bin which contains items of total size of at least $\frac{5}{6}$ has weight at least that amount. A bin which contains an item larger than $\frac{1}{2}$ has weight at least 1. Therefore, we only need to consider bins containing only items in $(\frac{1}{6}, \frac{1}{2}]$. We define a pattern to be a multiset of numbers in $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}$ whose sum is at most 1. The type of a pattern is the inverse of the smallest number in it. A pattern P of type j is a maximal pattern if adding another instance of $\frac{1}{j}$ to P, would not result in a pattern, i.e. $P \cup \{\frac{1}{j}\}$ is not a pattern. The pattern of a bin is the multiset of the weights of its items.

For each $j = 2, 3, 4, 5$, the packing has at most one bin whose pattern is of type j but is not maximal. We show that a bin of any maximal pattern has weight at least $\frac{5}{6}$ [1]. The only maximal pattern of type 2 is $\{\frac{1}{2}, \frac{1}{2}\}$. The maximal patterns of type 3 are $\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$ and $\{\frac{1}{2}, \frac{1}{3}\}$. Consider a maximal pattern of type 4. We need to show that the sum of elements in the pattern is at least $\frac{5}{6}$. Let a, b, c be the amounts of $\frac{1}{2}, \frac{1}{3}$, and $\frac{1}{4}$ in the pattern. If the sum is less than $\frac{5}{6}$, we have $\frac{3}{4} < \frac{a}{2} + \frac{b}{3} + \frac{c}{4} < \frac{5}{6}$. This gives $9 < 6a + 4b + 3c < 10$. Since a, b, c are integers, this is impossible. Similarly, consider a maximal pattern of type 5. Let a, b, c, d be the amounts of $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}$, and $\frac{1}{5}$ in the pattern. If the sum is less than $\frac{5}{6}$, we have $\frac{4}{5} < \frac{a}{2} + \frac{b}{3} + \frac{c}{4} + \frac{d}{5} < \frac{5}{6}$. This gives $48 < 30a + 20b + 15c + 12d < 50$ or $30a + 20b + 15c + 12d = 49$. Since a, b, c, d are non-negative integers, this combination is impossible.

Consider now the packing of *FFI*. We show that the total weight of items is at most $W \leq \mathit{FFI} + 5$. Note that the algorithm actually acts as Next Fit Increasing and never assigns an item to an old bin once a new bin is opened. A bin of *FFI* is called a transition bin if it has both at least one small item and at least one other item, or it has only large items, but it contains items of distinct weights. The last case means that the algorithm is done packing all items of weight $\frac{1}{j}$ for some $5 \geq j \geq 2$ and has started packing items of weight $\frac{1}{j-1}$. Therefore, there are at most five transition bins. In any other bin, the sum of the weights of the items is at most one; this is clear if there are only small items whose weights are equal to their sizes. For other items, there are j items of weight $\frac{1}{j}$ in a bin containing only such items. For the transition bins, the total weight of items whose size is at most one can be at most 2, so for each of these five bins there is excess weight of 1, giving $W \leq \mathit{FFI} + 5$. Hence $\mathit{OPT} \leq \frac{6}{5}\mathit{FFI} + 11$.

We now prove the lemma for $k \geq 4$. We slightly revise the definitions. Items are small if they are in the interval $(0, \frac{1}{k+3}]$. The weight of a small item is its size. An item which belongs to an interval $(\frac{1}{i+1}, \frac{1}{i}]$ for some $i = k, k+1, k+2$, is assigned the weight $\frac{1}{i}$.

Consider the optimal solution *OPT*. We show that the total weight of items is at least $W \geq (k^2+1)(\mathit{OPT} - 4)/(k^2+k)$. To show that, we claim that all bins except for at most four bins have items of weight at least $(k^2+1)/(k^2+k)$. First, consider the bins containing at least one small item. Due to Constraint 1, there is at most one such bin whose total sum of items is less than $1 - \frac{1}{k+3} = \frac{k+2}{k+3} \geq \frac{k^2+1}{k^2+k}$ (which holds for all $k \geq 3$).

Note that the weight of an item is at least its size. A bin which contains items of total size at least $\frac{k^2+1}{k^2+k}$ has weight at least that amount. We need to consider bins containing only items in $(\frac{1}{k+3}, \frac{1}{k}]$. We define a pattern to be a multiset of numbers in $\frac{1}{k+2}, \frac{1}{k+1}, \frac{1}{k}$ whose sum is at most 1. Again, we define the type of a pattern as the inverse of the smallest item in it.

For each $j = k, k+1, k+2$, the packing has at most one bin whose pattern is of type j but is not maximal. We show that a bin of any maximal pattern has weight at least $\frac{k^2+1}{k^2+k}$. The only maximal pattern of type k is $\{\frac{1}{k}, \ldots, \frac{1}{k}\}$.

Consider a maximal pattern of type $k+1$. We need to show that the sum of elements in the pattern is at least $\frac{k^2+1}{k^2+k}$. Let a and b be the amounts of $\frac{1}{k}$ and $\frac{1}{k+1}$ in the pattern. If the sum is less than

[1] The paper [3] showed a similar result on patterns for a different purpose.

$\frac{k^2+1}{k^2+k}$, we have $\frac{k}{k+1} < \frac{a}{k} + \frac{b}{k+1} < \frac{k^2+1}{k^2+k}$. This gives $k^2 < (k+1)a + kb < k^2+1$. Since a and b are integers, this is impossible. Similarly, consider a maximal pattern of type $k+2$. Let a, b, and c be the amounts of $\frac{1}{k}$, $\frac{1}{k+1}$, and $\frac{1}{k+2}$ in the pattern. If the sum is less than $\frac{k^2+1}{k^2+k}$, we have $\frac{k+1}{k+2} < \frac{a}{k}+\frac{b}{k+1}+\frac{c}{k+2} < \frac{k^2+1}{k^2+k}$. This gives $k(k+1)^2 < (k+2)(k+1)a+k(k+2)b+k(k+1)c < (k^2+1)(k+2)$ or $a(k^2+3k+2)+b(k^2+2k)+c(k^2+k) = k(k+1)^2+1$. We need to exclude the existence of an integer solution for a, b, and c. Assume that such a solution exists. Note that $a+b+c < k+2$, since otherwise the left hand side is at least $k(k+1)(k+2) > k(k+1)^2+1$. Also note that $a+b+c > k-1$, since otherwise the left hand side is at most $(k+1)(k+2)(k-1) < k(k+1)^2+1$. If $a+b+c = k+1$ we get $(a+b+c)(k^2+k)+2a(k+1)+kb = k(k+1)^2+1$. Simplifying, we have $2a(k+1)+kb = 1$, which is clearly impossible. If $a+b+c = k$, we need that $2a(k+1)+kb = k^2+k+1$. Rewriting, $2a-1 = k(k+1-b-2a)$, so $2a-1$ must be divisible by k. Clearly, a cannot be zero. Since $0 < a \leq k$ and a is an integer, the only value it can have is $\frac{k+1}{2}$, but this gives $b = -1$ which is impossible.

Now consider the packing of *FFI*. The total weight of items is at most $W \leq \mathit{FFI} + 4$ since there can be only four transition bins. This implies the upper bound on the approximation ratio. □

4 On-Line Maximum Resource Bin Packing

For *On-Line Maximum Resource Bin Packing*, the goal is to maximize the number of bins used subject to Constraint 2, so the algorithm is only allowed to open a new bin if the current item does not fit in any open bin. Note that the optimal algorithm must process the requests in the same order, obeying Constraint 2, even though it "knows" the entire sequence in advance. We have matching lower and upper bounds for most algorithms, and we conjecture that the algorithm Worst-Fit (*WF*) has an optimal competitive ratio of $\frac{3}{2}$.

Theorem 6 (General upper bound)**.** *For the On-Line Maximum Resource Bin Packing Problem, any algorithm ALG has a parameterized competitive ratio of*

$$\mathcal{C}_{\mathit{ALG}}(k) \leq \begin{cases} 2, & k = 1 \\ \dfrac{k}{k-1}, & k \geq 2. \end{cases}$$

Proof. For $k = 1$, this is proven using the fact that for any algorithm, all bins, except possibly one, are at least half full. For $k \geq 2$, we use the fact that all bins, except possibly one, are full to at least $\frac{k-1}{k}$. □

For $k \geq 3$, Theorem 6 is tight for deterministic algorithms:

Theorem 7 (General lower bound)**.** *Any deterministic on-line algorithm ALG has a parameterized competitive ratio of*

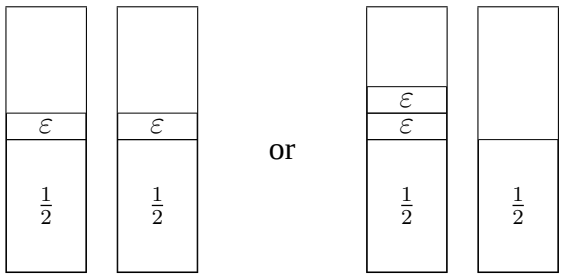
$$\mathcal{C}_{\mathit{ALG}}(k) \geq \begin{cases} \dfrac{3}{2}, & k \leq 2 \\ \dfrac{k}{k-1}, & k \geq 3. \end{cases}$$

The theorem follows from Lemmas 8 and 9 handling the cases $k \leq 2$ and $k \geq 3$ respectively.

Lemma 8. *There exists a family of sequences I_n with items no larger than $\frac{1}{2}$ such that OPT$(I_n) \to \infty$ for $n \to \infty$ and, for any deterministic on-line algorithm ALG,*

$$OPT(I_n) \geq \frac{3}{2} ALG(I_n).$$

Proof. The sequence is given in phases. Each phase begins with $\langle \frac{1}{2}, \varepsilon, \frac{1}{2}, \varepsilon \rangle$, where $\varepsilon < \frac{1}{12}$. For this sequence, there are two possible packings:



If the on-line algorithm chooses the first packing, the sequence continues with $\langle 2\varepsilon, \frac{1}{2} - \varepsilon, \frac{1}{2} - 3\varepsilon \rangle$, filling up the two on-line bins. An optimal off-line algorithm chooses the second packing, places the 2ε-item in the second bin, and opens a new bin for the last two items. Thus, *OPT* uses three bins, and has all bins filled to at least $\frac{1}{2} + 2\varepsilon$.

If the on-line algorithm chooses the second packing, the sequence continues with $\langle \frac{1}{2}, \frac{1}{2} - 4\varepsilon, \varepsilon, \varepsilon \rangle$. In this case, an optimal off-line algorithm chooses the first packing, and thus opens a new bin for the first two of the last four items. The last two items are placed in the first two bins. Again, *OPT* uses three bins and has all bins filled to at least $\frac{1}{2} + 2\varepsilon$.

Since each on-line bin is filled completely and each off-line bin is filled to at least $\frac{1}{2} + 2\varepsilon$, this can be repeated arbitrarily many times, with the result that *ALG* uses two bins per phase and *OPT* uses three bins per phase. □

Lemma 9. *For $k \geq 3$, there exists a family of sequences I_n with items no larger than $\frac{1}{k}$ such that OPT$(I_n) \to \infty$ for $n \to \infty$ and, for any deterministic on-line algorithm ALG,*

$$OPT(I_n) \geq \frac{k}{k-1} ALG(I_n) - \frac{1}{k-1}.$$

Proof. The sequence consists of an initial subsequence followed by n phases. Let $0 < \varepsilon < \frac{1}{4k^2(n+1)}$ be a very small constant. The sequence is constructed in such a way that after the initial subsequence, and also after every phase, *OPT* has the first bin filled to $\frac{k-1}{k} + 2k\varepsilon$ and all other open bins filled to exactly $\frac{k-1}{k} + 2\varepsilon$. The on-line algorithm has all open bins fully occupied, except the first bin that is always filled to at least $\frac{k-1}{k} + 2k\varepsilon$.

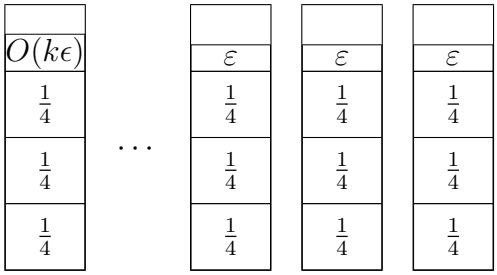
The sequence starts with $\langle (k-1) \times \frac{1}{k}, 2k\varepsilon \rangle$. After this, any algorithm has a single open bin (hereafter denoted the first bin) which is full up to $\frac{k-1}{k} + 2k\varepsilon$. The purpose of this initial subsequence is

to let the on-line algorithm have a bin where it puts one or more items of total size of up to $4k\varepsilon$ in each phase.

The rest of the sequence is given in n phases. Each phase starts with $\left\langle (k-1) \times \frac{1}{k}, \varepsilon \right\rangle$ repeated $k-1$ times. No algorithm can open more than $k-1$ bins for these $k(k-1)$ items. Actually, any algorithm uses exactly $k-1$ bins for the sequence, since the total size of the $\frac{1}{k}$ items is larger than $k-2$.

We consider two cases, according to how the on-line algorithm distributes the first items of the current phase.

- *Case A: ALG assigns exactly the amount $\frac{k-1}{k} + \varepsilon$ to each new bin.* In this case, *ALG* assigns the items exactly as Worst-Fit would do, for example, the packing for $k = 4$ looks as follows.



  Note that *ALG* has placed some number of $O(k\epsilon)$-items in the first bin. All bins between the first bin and the new $k-1$ bins are completely full.

  In this case, we add $\langle (k-3) \times \varepsilon, 2\varepsilon \rangle$ to the phase. Since there are only $k-2$ additional items, at least one bin opened by *ALG* in the current phase is full exactly up to $\frac{k-1}{k} + \varepsilon$. *OPT* places the items such that all of its bins are filled to exactly $\frac{k-1}{k} + 2\varepsilon$ except for the first bin which is filled to $\frac{k-1}{k} + 2k\varepsilon$.

  Next, we order the bins used by *ALG* in this phase (not including the first bin if it was used) in order of how much space they have left in non-increasing order. Let s_i, $1 \leq i \leq k-1$ be the space left in bin i of this phase by this ordering. Then $\frac{1}{k} - k\varepsilon \leq s_i \leq \frac{1}{k} - \varepsilon$ for $1 \leq s_i \leq k-1$. The first bin has at most $\frac{1}{k} - 2k\varepsilon$ space left. All remaining bins of *ALG* are completely full. We now add $k-1$ items with sizes $\langle s_1, s_2, \ldots, s_{k-1} \rangle$ to the phase. These will fit exactly in the $k-1$ bins. After this all the bins of *ALG*, except for the first bin, are completely full.

  After the $k-2$ additional items, all of *OPT*'s bins have at most $\frac{1}{k} - 2\varepsilon$ space left. Consequently using the first item of size $\frac{1}{k} - \varepsilon$, *OPT* can open a new bin and it places this and all the subsequent items in this bin.

  The last item of this phase consists of one item of size $\frac{k-1}{k} + 2\varepsilon - \sum_{i=1}^{k-1} s_i$. This is between $(k+1)\varepsilon$ and $2k\varepsilon$. *ALG* has to place this in the first bin, whereas *OPT* can place it in its new bin such that all of *OPT*'s bins except the first are exactly $\frac{k-1}{k} + 2\varepsilon$ full.

- *Case B: After the first part of the phase, ALG does not have $\frac{k-1}{k}+\varepsilon$ in each new bin.* In this case, *OPT* has exactly $\frac{k-1}{k}+\varepsilon$ in each new bin. No matter how *ALG* distributes the items, it has $k-1$ new bins, one of which is filled to at most $\frac{k-1}{k}$. Similar to Case A, we order the bins according to the space they have left. If $s_1 > \frac{1}{k}$, then some of the $k-1$ new bins must be completely full, and we first give some items of size $\frac{1}{k}$ until $s_1 \leq \frac{1}{k}$ before we proceed as in case A. Otherwise we simply do as in Case A, giving $k-1$ items with sizes $\langle s_1, s_2, \ldots, s_{k-1}\rangle$. In either case the first item has size $\frac{1}{k}$ and in general the size of items in this subphase is between $\frac{1}{k}-(k-1)\varepsilon$ and $\frac{1}{k}$. After these $k-1$ items all $k-1$ new bins of *ALG* are completely full.

  Since *OPT* opens a new bin for the item of size $\frac{1}{k}$, it can place all the items arriving afterwards in the same bin after which the bin is between $\frac{k-1}{k}-(k-1)\varepsilon$ and $\frac{k-1}{k}$ full. The final items of this phase are $k-1$ items of size ε, and one item of size $\frac{k-1}{k}+2\varepsilon$ minus the space used in *OPT*'s new bin. The first $k-1$ bins of *OPT* opened in this phase receive an item of size ε to achieve a total of $\frac{k-1}{k}+2\varepsilon$, and the last one receives the larger item for the same purpose. These items are placed in the first bin by *ALG* (since all its other bins are full).

Note, that in either case we place one or more items of a total size of at most $4k\varepsilon$ in *ALG*'s first bin. Since $\varepsilon < \frac{1}{4k^2(n+1)}$, the space needed in the first bin is $\frac{k-1}{k}+4k(n+1)\varepsilon < 1$.

We get $ALG(I_n) = (k-1)n+1$ whereas $OPT(I_n) = kn+1$. Hence, $ALG(I_n) = \frac{k-1}{k}\cdot OPT(I_n)+\frac{1}{k}$, or $OPT(I_n) = \frac{k}{k-1}\cdot ALG(I_n) - \frac{1}{k-1}$. □

By Theorems 6 and 7, any on-line algorithm has the same competitive ratio for $k \geq 3$, i.e., the only possible gap is for $k \leq 2$. For this case, we consider the algorithms First-Fit, Best-Fit, Last-Fit, and Worst-Fit.

Theorem 10. *For the On-Line Maximum Resource Bin Packing Problem, the parameterized competitive ratio of FF and BF is*

$$\mathcal{C}_{FF}(k) = \mathcal{C}_{BF}(k) = \begin{cases} 2, & k = 1 \\ \dfrac{k}{k-1}, & k \geq 2. \end{cases}$$

Proof. The upper bound follows from Theorem 6, and the lower bound for $k \geq 3$ follows from Theorem 7. Thus, we only need to prove the lower bound of 2 for $k \leq 2$. To this end, consider the sequence $\langle \frac{1}{2}, \varepsilon\rangle^n$, where n is a large odd integer and $\varepsilon \leq \frac{1}{2n}$. *FF* as well as *BF* puts all the small items in the first bin, using $1+\frac{n-1}{2}$ bins in total. *OPT* on the other hand distributes the small items one per bin, using n bins. This gives a ratio arbitrarily close to 2 for n arbitrarily large. □

Theorem 11. *For the On-Line Maximum Resource Bin Packing Problem, the parameterized competitive ratio of LF is*

$$\mathcal{C}_{LF}(k) = \begin{cases} 2, & k = 1 \\ \dfrac{k}{k-1}, & k \geq 2. \end{cases}$$

Proof. Again, we only need to prove the lower bound of 2 for $k \leq 2$. Let n be a large integer and $\varepsilon \leq \frac{1}{8n-4}$. The input is given in three phases:

1. $\langle \frac{1}{2}, \varepsilon \rangle^n$
2. $\langle \varepsilon \rangle^n$
3. $\langle \frac{1}{2} - \varepsilon, 3\varepsilon \rangle^{n-1}$

Both algorithms, *LF* and *OPT*, use n bins for the first $2n$ items, putting one item of size $\frac{1}{2}$ and one item of size ε in each bin.

LF puts all items from phase two in the last bin. It then packs the large items of phase three in the first $n-1$ bins and the small items of phase three in the last bin, using only n bins. *OPT*, on the other hand, distributes the items of phase two evenly in the n open bins, and is able to open a new bin for each of the $n-1$ pairs of items in phase three.

This gives a ratio of $\frac{2n-1}{n}$ which is arbitrarily close to 2 for n arbitrarily large. □

Investigation of Worst-Fit seems to indicate that it works very well in comparison with the other algorithms studied here. However, the gap between the lower bound of $\frac{3}{2}$ and the upper bound of 2 remains. Based on our investigation, we conjecture the following:

Conjecture 12. *For the On-Line Maximum Resource Bin Packing Problem, the competitive ratio of WF is* $\frac{3}{2}$.

5 On-Line Maximum Resource Dual Bin Packing

For this problem, there are exactly n bins. An item cannot be rejected if it fits in some bin, but there are no constraints as to which bins the algorithm may use, except that no bin may be filled to more than 1. As mentioned in the introduction we consider two objective functions: the number of accepted items and the total size of the accepted items. For both objective functions, no deterministic algorithms is competitive if no bounds are put on the item sizes.

Theorem 13. *For the On-Line Maximum Resource Dual Bin Packing Problem with accepted total size as cost function, no deterministic algorithm is competitive in general.*

Proof. Let $n \geq 2$ be the number of available bins, and let *ALG* be any deterministic algorithm. The input sequence is constructed in up to n rounds. In round i, for $1 \leq i \leq n-1$, n items of size ε are given, for some small $\varepsilon > 0$. If, after the ith round, *ALG* has one or more bins with fewer than i items, then an item of size $1-\varepsilon(i-1)$ is given. *OPT* distributes all $i \cdot n$ ε-items with i items in each bin, and can thus reject this large item. The performance ratio is then

$$\frac{ALG(I)}{OPT(I)} = \frac{in\varepsilon + (1-\varepsilon(i-1))}{in\varepsilon} = \frac{in - i + 1 + \frac{1}{\varepsilon}}{in}$$

For ε arbitrarily small, this ratio can be arbitrarily large.

If, after $n-1$ rounds, *ALG* has $n-1$ items in each of its n bins, we give an item of size $n\varepsilon$, and then $n-1$ items of size $1-\varepsilon(n-1)$. *ALG* has to accept all these items. *OPT* arranges the first items, including the item of size $n\varepsilon$, such that all bins are filled to $n\varepsilon$. It can then reject all the large items. This gives a performance ratio of

$$\frac{ALG(I)}{OPT(I)} = \frac{n^2\varepsilon + (n-1)(1-\varepsilon(n-1))}{n^2\varepsilon} = \frac{2n-1+(n-1)\frac{1}{\varepsilon}}{n^2}$$

This can again be arbitrarily large for ε arbitrarily small. □

We note that the situation for the parameterized problem for $k > 1$ is very different from the situation for the general problem. For every $k > 1$, it is not hard to show that any algorithm has competitive ratio of at most $k/(k-1)$. The reason for this is that if *OPT* rejected any item at all, then its bins are full up to at least $1 - 1/k$.

The following lower bound tends to $1 + \frac{1}{e(k-1)}$ as n tends to infinity and is at least $1 + \frac{1}{3(k-1)}$ for any $n \geq 2$.

Theorem 14. *Consider the On-Line Parameterized Maximum Resource Dual Bin Packing Problem with accepted total size as cost function. For $k \geq 2$, any deterministic algorithm ALG for this problem has*

$$C_{ALG}(k) \geq 1 + \frac{m}{n(k-1)}, \quad \textit{where } m = \max\left\{ j \;\middle|\; \sum_{i=j}^{n} \frac{1}{i} > 1 \right\} \in \left\{ \left\lfloor \frac{n}{e} \right\rfloor, \left\lceil \frac{n}{e} \right\rceil \right\}.$$

Proof. Let n be the number of bins and let $\varepsilon > 0$ be a very small constant. Let m be the largest number $1 \leq m < n$ such that $\sum_{i=m}^{n} \frac{1}{i} > 1$. Since $1/x$ is a monotonically decreasing function for $x > 0$, we get a lower bound of $\left\lfloor \frac{n}{e} \right\rfloor$ on m:

$$\sum_{i=\left\lfloor \frac{n}{e} \right\rfloor}^{n} \frac{1}{i} > \int_{\left\lfloor \frac{n}{e} \right\rfloor}^{n+1} \frac{1}{x} dx = [\ln x]_{\left\lfloor \frac{n}{e} \right\rfloor}^{n+1} = \ln \frac{n+1}{\left\lfloor \frac{n}{e} \right\rfloor} > \ln \frac{n+1}{\frac{n}{e}} = \ln \frac{n+1}{n} + 1 > 1.$$

For the upper bound, m can be at most $\left\lceil \frac{n}{e} \right\rceil$, since

$$\sum_{i=\left\lceil \frac{n}{e} \right\rceil+1}^{n} \frac{1}{i} < \int_{\left\lceil \frac{n}{e} \right\rceil}^{n} \frac{1}{x} dx = [\ln x]_{\left\lceil \frac{n}{e} \right\rceil}^{n} = \ln \frac{n}{\left\lceil \frac{n}{e} \right\rceil} < \ln \frac{n}{\frac{n}{e}} = 1.$$

Hence, depending on the exact value of n, m is either $\left\lfloor \frac{n}{e} \right\rfloor$ or $\left\lceil \frac{n}{e} \right\rceil$.

The initial input is $n!$ items of size ε. We first prove that, for any packing of these items, there exists an integer i, $m \leq i \leq n$, such that at least i bins receive strictly less than $\frac{n!}{n+m-i}$ items. Assume for the purpose of contradiction that this is not the case. Then, at least one bin has at least

$\frac{n!}{m}$ small items, and for each $i = n-1, n-2, \ldots, m$, there is at least one additional bin that receives at least $\frac{n!}{n+m-i}$ items. Since the total number of items is $n!$, we get that $\sum_{i=m}^{n} \frac{n!}{n+m-i} \leq n!$, which is equivalent to $\sum_{i=m}^{n} \frac{1}{i} \leq 1$. By the definition of m, this cannot be the case.

Now, pick an i, $m \leq i \leq n$, such that at least i bins receive strictly less than $\frac{n!}{n+m-i}$ items in *ALG*'s packing. Give

1. $\left\langle \dfrac{n!}{n+m-i}\,\varepsilon \right\rangle^{i-m}$
2. $\left\langle \dfrac{1}{k} \right\rangle^{n(k-1)}$
3. $\left\langle \dfrac{1}{k} - \dfrac{n!-1}{n+m-i}\,\varepsilon \right\rangle^{m}$

After packing the first $i-m$ of these items, *ALG* still has at least m bins filled to strictly less than $\frac{n!}{n+m-i}\,\varepsilon$. Let r be the number of *ALG*'s bins which are completely empty. Since all bins are less than $\frac{1}{k}$ full, there is room for exactly $n(k-1)+r$ items of size $\frac{1}{k}$ and at least $\max\{m-r, 0\}$ items of size $\frac{1}{k} - \frac{n!-1}{n+m-i}\,\varepsilon$. Thus, *ALG* is able to pack the remaining items as well, giving a total size of

$$n!\varepsilon + (i-m)\,\frac{n!}{n+m-i}\,\varepsilon + \frac{n(k-1)+m}{k} - \frac{n!-1}{n+m-i}\,m\varepsilon.$$

Before the arrival of the size $\frac{1}{k}$ items, *OPT* can pack the items from phase one in one bin each and distribute the initial $n!$ items in the remaining bins to fill all bins up to exactly $\frac{n!}{n+m-i}\,\varepsilon$. Each bin gets $k-1$ items of size $\frac{1}{k}$, and no further items can be packed. The total size packed by *OPT* is

$$n!\varepsilon + (i-m)\,\frac{n!}{n+m-i}\,\varepsilon + \frac{n(k-1)}{k}.$$

As ε decreases, the ratio converges to $1 + \frac{m}{n(k-1)}$. □

The lowest possible value of $\frac{m}{n}$ is $\frac{1}{3}$ which is obtained when n equals 3, 6 or 9.

For the case where the objective function is the number of accepted items, the situation is even worse.

Theorem 15. *For the On-Line Parameterized Maximum Resource Dual Bin Packing Problem with the number of accepted items as cost function, no deterministic algorithm is competitive, for any k.*

Proof. Let $n \geq 2$ be the number of bins available, and let *ALG* be any deterministic algorithm.

The input sequence begins with $2kn - 2$ items of size $\frac{1}{2k}$. *ALG* fills all but at most two bins completely, and the remaining two bins are either both filled to $1 - \frac{1}{2k}$, or one is filled completely and the other to $1 - \frac{1}{k}$.

In the first case, the sequence continues with one item of size $\frac{1}{k}$ and $\left\lfloor \frac{1}{k\varepsilon} \right\rfloor$ items of size ε. *ALG* rejects the first of these and accepts all of the small ones. *OPT*, on the other hand, arranges the items of size $\frac{1}{2k}$, so that all but one bin is full, the item of size $\frac{1}{k}$ fits in that last bin, and all the small items are rejected.

In the second case, the sequence continues with one item of size $\frac{1}{2k} + \varepsilon$, two items of size $\frac{1}{2k}$, and $\left\lfloor \frac{1}{2k\varepsilon} \right\rfloor - 1$ items of size ε. *ALG* accepts the first of these items, rejects the next two, and accepts all the small items. *OPT*, on the other hand, rejects the first of these items, accepts the next two, and rejects all the small items.

By making ε arbitrarily small, the number of items accepted by *ALG* can be made arbitrarily large, while the number of items accepted by *OPT* is either $2nk - 1$ or $2nk$. □

6 Concluding Remarks

The most interesting open problem is to prove that the off-line maximum resource bin packing problem is NP-hard (or to find a polynomial time algorithm for it).

For the off-line version of the problem, we have investigated First-Fit-Decreasing, which is worst possible, and First-Fit-Increasing, which performs better and obtains an approximation ratio of $\frac{6}{5}$. It would be interesting to establish a general lower bound on the problem, and, if it is lower than $\frac{6}{5}$, to determine the optimal algorithm for the problem. Does there exist a polynomial time approximation scheme for the off-line version?

For the on-line version, we have considered the two standard bin packing problems from the literature. For dual bin packing, no algorithm is competitive in general, independent of whether the cost measure is the total size or the total number of accepted items. With the total accepted size as cost function, the situation is completely different for the parameterized version; for $k \geq 2$, any algorithm has a parameterized competitive ratio between $1 + \frac{1}{k-1}$ and about $1 + \frac{1}{e(k-1)}$.

For the classic variant of on-line bin packing, we have established general upper and lower bounds and proved that First-Fit, Best-Fit, and Last-Fit perform worst possible. The behavior of Worst-Fit seems very promising, but we leave it as an open problem to determine its competitive ratio.

7 Acknowledgments

First of all, we would like to thank Michael Bender for suggesting the problem and for interesting initial discussions. We would also like to thank Morten Hegner Nielsen for interesting discussions at the 3rd NOGAPS, where the work on the problem was initiated. In addition, we are thankful to

Gerhard Woeginger for comments on an earlier version of this paper and the anonymous referees for their comments.

References

[1] E.M. Arkin, M.A. Bender, J.S.B. Mitchell, and S. Skiena. The Lazy Bureaucrat scheduling problem. *Information and Computation*, 184(1):129–146, 2003.

[2] S.F. Assman, D.S. Johnson, D.J. Kleitman, and J.Y-T. Leung. On a dual version of the one-dimensional bin packing problem. *J. Alg.*, 5(4):502–525, 1984.

[3] A. Bar-Noy, R.E. Ladner, and T. Tamir. Windows scheduling as a restricted version of bin packing. In *Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 224–233, 2004.

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

[5] J. Boyar and L.M. Favrholdt. The relative worst order ratio for on-line algorithms. In *Algorithms and Complexity: 5th Italian Conference*, volume 2653 of *Lecture Notes in Computer Science*, pages 58–69. Springer-Verlag, 2003.

[6] J. Boyar and L.M. Favrholdt. The relative worst order ratio for on-line bin packing algorithms. Technical report PP–2003–13, Department of Mathematics and Computer Science, University of Southern Denmark, 2003.

[7] J. Csirik, J.B.G. Frenk, M. Labbé, and S. Zhang. Two simple algorithms for bin covering. *Acta Cybernetica*, 14(1):13–25, 1999.

[8] J. Csirik, D.S. Johnson, and C. Kenyon. Better approximation algorithms for bin covering. In *Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 557–566, 2001.

[9] L. Epstein and M. Levy. Online interval coloring and variants. In *Proc. 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 602–613. Springer-Verlag, 2005.

[10] M.R. Garey and D.S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[11] R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal*, 45:1563–1581, 1966.

[12] R. Hassin and S. Rubinstein. An approximation algorithm for the maximum traveling salesman problem. *Information Processing Letters*, 67(3):125–130, 1998.

[13] D.S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1997.

[14] K. Jansen and R. Solis-Oba. An asymptotic fully polynomial time approximation scheme for bin covering. *Theoretical Computer Science*, 306(1–3):543–551, 2003.

[15] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, and R.L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comp.*, 3:299–325, 1974.

[16] D.R. Karger, R. Motwani, and G.D.S. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18(1):82–98, 1997.

[17] A.R. Karlin, M.S. Manasse, L. Rudolph, and D.D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.

[18] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.