On the sum minimization version of the online bin covering problem[⋆]

János Csirik[1], Leah Epstein[2], Csanád Imreh[1] and Asaf Levin[3]

[1] Department of Informatics, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary. {csirik,cimreh}@inf.u-szeged.hu
[2] Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il
[3] Chaya fellow. Faculty of Industrial Engineering and Management, The Technion, 32000 Haifa, Israel. levinas@ie.technion.ac.il

Abstract. Given a set of m identical bins of size 1, the online input consists of a (potentially, infinite) stream of items in $(0, 1]$. Each item is to be assigned to a bin upon arrival. The goal is to cover all bins, that is, to reach a situation where a total size of items of at least 1 is assigned to each bin. The cost of an algorithm is the sum of all used items at the moment when the goal is first fulfilled. We consider three variants of the problem, the online problem, where there is no restriction of the input items, and the two semi-online models, where the items arrived sorted by size, that is, either by non-decreasing size or by non-increasing size. The offline problem is considered as well.

keywords: online algorithms, bin covering.

1 Introduction

We consider the following model. We are given a list of items, arriving online, where each item j has a size $s_j \in (0, 1]$. There are m bins of size 1 and the goal is to cover all bins, that is, to assign items of total size at least 1 to each bin, minimizing the total size of the used items. In this online variant, the items arrive one by one, and at each step we have to assign a new item to one of the (uncovered) bins. Only after an item is assigned, the successor item (if it is required) is revealed. The process ends when all bins are covered.

We mention two possible applications. The first application is related to max-min allocations and fairness issues. A system of m machines relies on keeping all the machines productive for at least a given duration, as the entire system fails even in a case that just one of the machines ceases to be active (see [6] for details). Resources of variable sizes (fuel tanks) are presented and assigned to machines one by one. A resource cannot be split between machines. Once all machines received a sufficient amount of resources, this process can stop, and the total charge is proportional to the total size of resources which are used. The second application comes from the area of human resource management. Workers are to be assigned to perform m identical tasks. Each worker j is available for a given time s_j, and this time cannot be split between tasks. The workers are interviewed one by one, and need to be assigned to a task immediately. It is not allowed to refuse a worker. Once enough working time has been assigned to all tasks, no additional workers are interviewed. The goal is to minimize the total work time of the workers.

In this paper, we use competitive analysis and compare the cost of an online algorithm with the cost of an optimal offline algorithm, which can see the complete list of items at once.

[⋆] This research has been supported by the Hungarian National Foundation for Scientific Research, Grant F048587.

Though an offline algorithm knows the sequence in advance, it is not allowed to change the order of the items or to reject any item until all bins are covered. Similarly to the online algorithm, it has to cover the bins with a prefix of the item sequence, and its cost is the total size of the items in this prefix. In all proofs we consider a specific optimal offline algorithm OPT. For an algorithm $\mathcal{A}$, we denote its cost by $\mathcal{A}$ as well. For minimization problems, the asymptotic competitive ratio of an algorithm $\mathcal{A}$ is the infimum $\mathcal{R} \geq 1$ such that for any input, $\mathcal{A} \leq \mathcal{R} \cdot \text{OPT} + c$, where c is a constant independent of the input. We also use the absolute competitive ratio which is infimum $\mathcal{R} \geq 1$ such that for any input, $\mathcal{A} \leq \mathcal{R} \cdot \text{OPT}$.

Our problem is related to the *bin covering problem*, where the input consists of items of size at most 1, which need to be partitioned into sets (bins) so as to maximize the number of sets whose total sum is at least 1. The bin covering problem was investigated in the early 1980's by Assmann [1] and by Assmann et al. [2]. Later, further results have been achieved on that problem and its variants [3–5, 7]. An extensive overview on these results can be found in [8].

Another related problem is the online knapsack problem. In this problem, several knapsacks (bins) of an identical capacity are to receive subsets of input items, each having a weight and a value. The goal is to maximize the total value of the items which are packed into the knapsacks. Since weights can be arbitrarily large, the two types of competitive ratio result in the same measure for the online knapsack problem. Our problem can be considered as the dual of the knapsack problem, where the weight of every item is equal to its value. However, an additional restriction in our case is that items cannot be refused, that is, the packed set of items must be a prefix of the sequence of items. It is known that there exists no constant competitive algorithm for the online knapsack problem (no matter whether it is allowed to refuse to pack arriving items or not). A stochastic version where the weights and values are drawn according to a known distribution is investigated in [9] and [10], and a semi-stochastic version where the elements arrive in random order is studied in [11]. In the deterministic version, if it is allowed to remove items from the knapsack after they have been packed, then it is possible to achieve constant competitive ratio. Specifically, Iwama and Taketomi [12] designed an optimal online algorithm for this model for one knapsack, and studied a variant with multiple knapsacks. If the value of each item is 1, then the knapsack problem leads to a version of online bin packing where the number of bins is fixed and the goal is to maximize the number of the packed items. This problem is investigated in [13] for identical bins. Both options, the case where items cannot be refused (similarly to the problem studied in our paper), and the case that items can be refused, were considered. A generalization for variable sized bins is given in [14].

We define the load of a bin at a given time to be the sum of all items that are assigned to this bin. If the load of a bin is at least 1, we say that the bin is covered, and otherwise uncovered. We mention two natural algorithms for the problem. The algorithm LIST is defined as follows. LIST assigns a new item to the least loaded bin, and halts when all bins are covered. An additional natural algorithm is NEXT-FIT (NF). This algorithm assigns all items to an active bin, and moves on to the next bin only after this active bin is covered. We say that an algorithm $\mathcal{A}$ is thrifty if the following two conditions hold: First, $\mathcal{A}$ halts once all bins are covered, and second, $\mathcal{A}$ does not assign further items to any covered bins. Clearly, LIST is thrifty, and so is NF, if they are executed on a set of empty bins (or a set of bins that are not empty, but none of them had received an item after it had already been covered).

We study the parametric case in addition to the general results. In the parametric case, items sizes are bounded from above by a value $\frac{1}{p}$ for an integer p. The general case is identical to the case $p = 1$. For a thrifty algorithm, the total size assigned to each bin can exceed 1 by at most the size of the largest possible item. The following claim holds for the parametric case, and it implies an absolute competitive ratio of 2 for both LIST and NF, in the general case.

Claim. Let $\mathcal{A}$ be an arbitrary thrifty algorithm for the parametric case with items sizes in the interval $(0, \frac{1}{p}]$ for an integer $p \geq 1$. The absolute competitive ratio of $\mathcal{A}$ is at most $1 + \frac{1}{p}$. Furthermore, when $\mathcal{A}$ halts, the load of every bin of $\mathcal{A}$ is smaller than $1 + \frac{1}{p}$.

Our results. In this paper, we are interested in the online problem, where items arrive ordered arbitrarily, and in two semi-online variants, where items are sorted according to size (either in a non-increasing order or in a non-decreasing order).

We begin our study with the offline problem. In the offline variant, the algorithm receives a sequence of n items. The goal is to find a prefix of this sequence (of a minimum total size), and an allocation of the items in this prefix to bins, so that all bins are covered. In Section 2 we present an AFPTAS for the offline version of the problem. It is not difficult to see that the problem is NP-hard in the strong sense, using a simple reduction from 3-PARTITION. In this reduction, given $3M$ items which are an input to the 3-PARTITION problem, the input to our problem is created so that $m = M$, the first $3m$ items are based on the input to 3-PARTITION, having sizes in $(\frac{1}{4}, \frac{1}{2})$, and all additional items have some fixed size S. The strong NP-hardness holds even if the input sequence needs to be sorted according to one of the orders. The first $3M$ items can be sorted according to the required order, and the additional items have the size 1 in the case of non-decreasing item sizes, and $\frac{1}{8}$ in the case of non-increasing item sizes.

The remainder of the paper is devoted to the online and semi-online problems. We consider the case $m = 2$, and the case of arbitrary m. In the latter case, we study the asymptotic competitive ratio. In the case $m = 2$ the asymptotic competitive ratio is meaningless, so we study the absolute competitive ratio.

In Section 3 we provide a complete solution for $m = 2$ (in the general case as well as in the parametric case), that is, we present best possible online and semi-online algorithms. The absolute competitive ratios are $\frac{5}{3}$ (for arbitrary list of items), $\frac{3}{2}$ (for lists of items sorted by non-decreasing order) and $\frac{6}{5}$ (for lists of items sorted by non-increasing order).

In Section 4 we consider the case of an arbitrary number m of bins. We design two algorithms, which beat the easy upper bound of 2 on the asymptotic competitive ratio, in both semi-online cases. We first present an algorithm of asymptotic competitive ratio of at most 1.931215 for lists of items sorted by non-decreasing order. In addition, we present a simple algorithm for the other semi-online variant, of an asymptotic competitive ratio of $\frac{4}{3}$.

Finally, in Section 5 we present lower bounds for the possible asymptotic competitive ratios for all three variants of the problem. We prove a lower bound of 1.387 on the asymptotic competitive ratio of any algorithm which can process items given in an arbitrary order, a lower bound of 1.302 on the asymptotic competitive ratio of any algorithm which can process items which are sorted according to non-decreasing size, and a lower bound of 1.111 on the asymptotic competitive ratio of any algorithm which can process items which are sorted according to non-increasing size.

2 The offline problem

In this section we show that by using an AFPTAS for the bin covering problem we can obtain an AFPTAS for our problem. Previous results for the bin covering problem include an APTAS

[5] and an AFPTAS due to Jansen and Solis-Oba [7]. Such an AFPTAS provides the following performance guarantee (for a given function f). Denote by μ the value of the optimal solution, that is, the maximum number of bins that can be covered using the input. Then the scheme covers at least $(1-\varepsilon)\mu - f(\frac{1}{\varepsilon})$ bins.

To show the relation between bin covering and our problem, we define the following function $g : \mathbb{N} \to \mathbb{N}$, where $g(k)$ is the value of the optimal solution for bin covering on the first k items of input for our problem. Clearly, g is monotonically non-decreasing. Moreover, for any $k \geq 1$, we have $g(k) \leq g(k-1)+1$ (where $g(0)=0$). This last property holds since given a solution for bin covering using k items, the removal of one item can decrease the number of covered bins by at most 1. Therefore, finding an optimal solution to our problem can be reduced to finding an optimal solution to bin covering as follows: find the minimum k such that an optimal solution for the bin covering problem is exactly m. Recall that n denotes the number of items in the input sequence.

Our scheme works as follows. We guess the prefix of the items that the optimal solution uses. Note that for this guessing step there are at most n possible values and hence we can try all possibilities for this step in polynomial time, and output the best resulting solution. In the analysis of the scheme it suffices to consider the iteration in which we used the correct prefix of items (as the optimal prefix). For this prefix of items we use the AFPTAS of [7] to maximize the number of bins covered using the same set of items as the optimal solution used. Note that by our guessing we conclude that the bin covering instance has an optimal value equals to m and hence in this step we cover at least $(1-\varepsilon)m - f(\frac{1}{\varepsilon})$ bins. Therefore, the number of uncovered bins is at most $\varepsilon m + f(\frac{1}{\varepsilon})$. Now we use the next-fit heuristic to cover the remaining bins. Note that each such bin that is covered by next-fit, will have total load of at most two, and hence the total size of items that are used in this step is at most $2\varepsilon m + 2f(\frac{1}{\varepsilon})$. Therefore, the total size of items used by the algorithm is at most $\textsc{opt}+2\varepsilon m+2f(\frac{1}{\varepsilon}) \leq (1+2\varepsilon)\textsc{opt}+2f(\frac{1}{\varepsilon})$ where the inequality holds using $m \leq \textsc{opt}$, and so we prove the following result.

Proposition 1. *The offline problem has an AFPTAS.*

3 Two bins

We start with a complete solution for all variants in the case $m = 2$.

3.1 Arbitrary input sequences

We use the following thrifty algorithm for any integer $p \geq 1$.

```
Algorithm TwoBins.
0. Given a new item $j$, let $A_1 \geq A_2$ be the current loads of the two bins (where $A_2 < 1$).
1. If $1 \leq A_2 + s_j \leq \frac{2p+2}{2p+1}$, assign $j$ to the bin of load $A_2$. Go to Step 4.
2. If $A_1 < 1$ and $A_1 + s_j \leq \frac{2p+2}{2p+1}$, assign $j$ to the bin of load $A_1$. Go to Step 4.
3. Assign $j$ to the bin of load $A_2$.
4. If both bins are covered, halt, otherwise, go to Step 0.
```

Theorem 1. *The absolute competitive ratio of* TwoBins $p \geq 1$ *is* $\frac{(4p+1)(p+1)}{2p(2p+1)}$, *which is best possible.*

Proof. Consider the first time that some bin is covered. If this happens in Step 1 or Step 2, this bin will have a load of at most $\frac{2p+2}{2p+1}$. Since the algorithm is thrifty, the other bin would not exceed a load of $1+\frac{1}{p}$. We get $\textsc{TwoBins} \le 2+\frac{1}{p}+\frac{1}{2p+1}$ and $\textsc{opt} \ge 2$. In this case the absolute competitive ratio follows.

It is left to consider the case that a bin is covered in Step 3. In this case, let t be an index of the first item that ever covers a bin. At the time of assignment of item t, both bins were uncovered, but non-empty (since if an empty bin is covered using a single item, this happens already in Step 1). The item t is assigned to the least loaded bin. Consider next a case where at the time of arrival of t, at least one bin contains a total size of items that is at least $\frac{2p}{2p+1}$. After t is assigned to the least loaded bin, one bin is covered, and the other bin is loaded by at least $\frac{2p}{2p+1}$. Since it will not reach a load of more than $1+\frac{1}{p}$, the total size of additional items assigned to this bin, which are $t,\ldots,i$, where i is the last item which is used by $\textsc{TwoBins}$, is no larger than $\frac{1}{p}+\frac{1}{2p+1}$. We have $\textsc{TwoBins} = \sum_{k=1}^{i} s_k \le \sum_{k=1}^{t} s_k + \frac{1}{p}+\frac{1}{2p+1}$. Since $\sum_{k=1}^{t-1} s_k < 2$, $\textsc{opt}$ must use at least t items, and $\textsc{opt} \ge \sum_{k=1}^{t} s_k$. We get $\textsc{TwoBins} \le \textsc{opt} + \frac{1}{p}+\frac{1}{2p+1}$. Using $\textsc{opt} \ge 2$ we get the required absolute competitive ratio.

Finally, we examine the case where an item t covers a non-empty bin in Step 3, and both bins are loaded by less than $\frac{2p}{2p+1}$ (but more than zero) at the time that t arrives. We first state an important property of the algorithm. Starting the time that the second bin becomes non-empty, and until it receives at least p items (including the first and p-th items assigned to this bin), all item assigned to the second bin have sizes in the interval $(\frac{2}{2p+1},\frac{1}{p}]$. To prove this, first note that the first bin (i.e., the one used first) must have load of more than $\frac{2p+2}{2p+1}-\frac{1}{p} > \frac{p-1}{p}$ when the second bin is used first, otherwise the new item is assigned by one of the two first steps. Since its load is less than $\frac{2p}{2p+1}$ but a new item would increase it above $\frac{2p+2}{2p+1}$, the size of the new item is larger than $\frac{2}{2p+1}$. This is true at the assignment time of the first p items that are assigned to the second bin, since after $p-1$ such items, the load of the second bin is at most $\frac{p-1}{p}$. Moreover, during this process, no bin is covered in Step 3. The second bin cannot be covered since it has at most p items, and the total size of its items is at most 1. The first bin is more loaded during the process, and thus if it is covered, it can not happen in Step 3.

Consider the second bin after the process. It has p items, of total size larger than $\frac{2p}{2p+1}$. Thus the condition above is satisfied, and we are done.

To prove the lower bound, we fix a small value $\varepsilon > 0$. We present two items of size $\frac{1}{2p+1}$. If they are placed into the same bin, $2p$ items of size $\frac{2}{2p+1}$ are presented. At this time one bin has load $\frac{2p+2}{2p+1}$ and the other (uncovered) bin has load $\frac{2p}{2p+1}$. The last items have sizes $\frac{1}{2p+1}-\varepsilon$ and $\frac{1}{p}$. Clearly, both items are required to cover the uncovered bin. An optimal solution would use only the first $2p+2$ items, putting one item of size $\frac{1}{2p+1}$ and p larger items in each bin. We get an absolute competitive ratio of $1+\frac{\frac{1}{2p+1}+\frac{1}{p}-\varepsilon}{2}$ which is arbitrarily close to $\frac{(4p+1)(p+1)}{2p(2p+1)}$ for $\varepsilon \to 0$.

If the first two items are placed into different bins, the next $2p-1$ items have size $\frac{1}{p}$, they are followed by one item of size $\frac{1}{p(2p+1)}$. The items of size $\frac{1}{p}$ cause one bin to be covered, and the other bin reaches a load of $1-\frac{1}{p}+\frac{1}{2p+1} = 1-\frac{p+1}{p(2p+1)}$. The additional item brings this load up to $1-\frac{1}{2p+1}$. The sequence continues as in the previous case, and again an optimal solution uses all items but the last two. We get the same absolute competitive ratio in this case as well. $\square$

3.2 Input sequences with non-decreasing item sizes

In this case, we simply use List, where ties are broken in favor of the first bin. The same algorithm turns out to give an optimal result for the parametric case as well. We prove the following theorem for input sequences with non-decreasing item sizes. This shows that the optimal absolute competitive ratio for $p = 1$ is $\frac{3}{2}$.

Theorem 2. *Given an upper bound $\frac{1}{p}$ on the sizes of items, where $p \geq 1$ is an integer, the cost of* List *is at most* $\textsc{opt} + \frac{1}{p}$*, and this is best possible. The absolute competitive ratio of* List *is $\frac{2p+1}{2p}$, which is best possible as well.*

Proof. We claim that List acts as Round-Robin for non-decreasing item sizes. Assume that item 1 is assigned to the first bin. We show by induction that all odd-indexed items are assigned to the first bin, and all even-indexed items are assigned to the second bin. Consider first an item of index $2k$ for $k \geq 1$. By the induction hypothesis, the load of the first bin is $\sum_{j=1}^{k} s_{2j-1}$ and the load of the second bin is $\sum_{j=1}^{k-1} s_{2j} = \sum_{j=2}^{k} s_{2j-2}$. Since $s_{2j-2} \leq s_{2j-1}$ for $2 \leq j \leq k$, and $s_1 > 0$, the load of the first bin is strictly larger than the load of the second bin, and item $2k$ is assigned to the second bin.

Next, consider an item of index $2k+1$ for $k \geq 1$. By the induction hypothesis, the load of the first bin is $\sum_{j=1}^{k} s_{2j-1}$ and the load of the second bin is $\sum_{j=1}^{k} s_{2j}$. Since $s_{2j} \geq s_{2j-1}$ for $1 \leq j \leq k$, the load of the first bin is no larger than the load of the second bin, and item $2k+1$ is assigned to the first bin.

As a result, we can see that after one bin is covered, the other bin is covered immediately after that with the next item. Let i be the index of the item that covers the first bin that is covered (this can be the first or the second bin). Then $\text{List} = \sum_{k=1}^{i+1} s_k$. Since $\sum_{k=1}^{i-1} s_k < 2$, we have $\textsc{opt} \geq \sum_{k=1}^{i} s_k$. Since $s_{i+1} \leq \frac{1}{p}$, we get $\text{List} \leq \textsc{opt} + \frac{1}{p}$. Since $\textsc{opt} \geq 2$, this also implies an absolute competitive ratio of at most $1 + \frac{1}{2p}$.

For the lower bound, let $\varepsilon > 0$ be a small number. We construct a sequence which starts with two items of size $p\varepsilon$. If they are assigned to the same bin, we continue with items of size $\frac{1}{p} - \varepsilon$, and otherwise, we give another item of size $2p\varepsilon$, and items of size $\frac{1}{p} - 2\varepsilon$. In the first case, an optimal solution assigns one item of size $p\varepsilon$ and p items of size $\frac{1}{p} - \varepsilon$ to each bin. In the second case, an optimal solution assigns one item of size $2p\varepsilon$ and p items of size $\frac{1}{p} - 2\varepsilon$ to one bin, and two items of size $p\varepsilon$ and p items of size $\frac{1}{p} - 2\varepsilon$ to the other bin. Therefore the optimal cost is 2. The algorithm however, does not cover both bins already using $2p$ larger items. In the first case, the bin with no small items requires $p+1$ larger items, and in the second case, the bin which receives only one of the three first items requires $p+1$ larger items. The cost is therefore larger than $(2p+1)(\frac{1}{p} - 2\varepsilon)$, which gives a lower bound of $\frac{2p+1}{2p}$ for $\varepsilon \to 0$. Note that the cost of the algorithm exceeds $\textsc{opt}$ by at least $\frac{1}{p} - 2\varepsilon$. $\square$

3.3 Input sequences with non-increasing item sizes

In this last case, we again use List (ties are broken arbitrarily). The resulting algorithm in this case does not act as Round-Robin, but it is still possible to show optimality. We prove the following theorem for input sequences with non-increasing item sizes. We later discuss the parametric case for $p \geq 2$, which for this input behaves differently from the case $p = 1$, and to get the best result a different algorithm is required.

Theorem 3. *The absolute competitive ratio of* List *is* $\frac{6}{5}$*, which is best possible.*

Proof. Let t be the index of the first item that covers a bin. If $t = 1$ it means that $s_1 = 1$. In this case we are left with a single bin, and therefore the solution is optimal. If $t = 2$ we get that $s_2 = 1$ and so $s_1 = 1$, therefore this case is impossible (it means that already the first bin is covered, and thus $t = 1$). If $t = 3$, we get that $s_2 + s_3 \geq 1$. An optimal solution must have two items in $\{1, 2, 3\}$ assigned to the same bin. We may assume that these items are $2, 3$ (since they are the smallest but still cover a bin), and we get that the solution obtained by List is optimal.

If the $(t + 1)$-th item covers the second bin, and $t \geq 5$, we have $\text{OPT} \geq \sum_{k=1}^{t} s_k$ (since $\sum_{k=1}^{t-1} s_k < 2$), and the cost of the algorithm is $\sum_{k=1}^{t+1} s_k$. Since $s_{t+1} \leq s_i$ for $1 \leq i \leq t$, we have $s_{t+1} \leq \frac{1}{t} \sum_{k=1}^{t} s_k$, thus the cost of the algorithm is at most $\frac{t+1}{t}\text{OPT} \leq \frac{6}{5}\text{OPT}$.

If the $(t+1)$-th item does not cover the other bin, we recall that $t \geq 4$, and let $x = 1 - L$, where L is the load of this uncovered bin, at the time of assignment of item t. We denote the load of the covered bin, just before item t is assigned there, by M. By the definition of List, $L \geq M$. We have $\text{OPT} \geq M + L + s_t$. Since item $t + 1$ does not cover the bin it is assigned to, we have $s_{t+1} < x$. All future items are of size no larger than s_{t+1}, therefore the final load of this bin will be no larger than $1 + s_{t+1} < 1 + x$. The cost of the algorithm is therefore at most $M + s_t + 1 + x \leq \text{OPT} + 1 + x - L = \text{OPT} + 2x$, if $x \leq \frac{1}{5}$, we are done since $\text{OPT} \geq 2$. Otherwise, since $t \geq 5$, at least one bin received at least two items just before item t is assigned, and its load was at most $\max\{M, L\} = L$, thus $s_t < \frac{L}{2} = \frac{1-x}{2} \leq \frac{2}{5}$. The cost of the algorithm is at most $M + s_t + 1 + x \leq L + x + 1 + s_t = 2 + s_t \leq \frac{12}{5}$. Since $\text{OPT} \geq 2$ we are done in this case as well.

We are left with the case $t = 4$ where the 5-th item covers the other bin. We show that the solution obtained by List is optimal. Assume by contradiction that OPT can use at most four items. If OPT has a bin with only one item, it means that $s_1 = 1$ and $t = 1$. Therefore, OPT has two items in each bin. This means that the bin which does not contain the first item is covered by two items in $\{2, 3, 4\}$ and therefore $s_2 + s_3 \geq 1$. However, List assigns items $2, 3$ to the second bin and $t > 3$ so we get $s_2 + s_3 < 1$. Contradiction.

For the lower bound, we fix a small value $\varepsilon > 0$. We construct a sequence which starts with two items of size $\frac{1}{2} + \varepsilon$. If the algorithm assigns them to the same bin, all future items have size $\frac{1}{2} - \varepsilon$ and otherwise, all future items have size $\frac{1}{2} - 2\varepsilon$. In the first case, an optimal packing would be to pack every item of size $\frac{1}{2} + \varepsilon$ together with an item of size $\frac{1}{2} - \varepsilon$, and get a cost of 2. In the second case, an optimal packing would be to pack the two items of size $\frac{1}{2} + \varepsilon$ together, and three items of size $\frac{1}{2} - 2\varepsilon$ in an additional bin, and get a cost of $\frac{5}{2} - 4\varepsilon$.

In the first case, the algorithm has covered one bin using the first two items, and it needs three additional items to cover the other bin. This gives a cost of $\frac{5}{2} - \varepsilon$. In the second case, the algorithm needs two additional items to cover each bin. This gives a cost of $3-6\varepsilon$. Letting ε tend to zero, we get ratios of $\frac{5}{4}$ and $\frac{6}{5}$ in the two cases, which implies a lower bound of $\frac{6}{5}$ on the absolute competitive ratio. □

We next define the algorithm TwoBinsDecreasing (TBD). The first p items are assigned to the first bin. The next p items are assigned to the second bin. Every additional item is assigned to the least loaded bin, until both bins are covered.

Theorem 4. *The absolute competitive ratio of* TBD, *for* $p \geq 2$, *is* $\frac{2p+3}{2p+2}$ *which is best possible.*

Proof. Let t be the index of the first item that covers a bin. If $t = p$, we have a bin covered by exactly p items of size $\frac{1}{p}$, the algorithm is left with one bin and the solution is optimal. Otherwise, we get that no p items can cover a bin, and thus any solution needs at least $2p+2$ items and for the algorithm, the first $2p$ items cannot cover a bin and so $t \geq 2p+1$.

If $t = 2p+1$ we show the TBD gives an optimal solution. The $(2p+1)$-th item is assigned to the least loaded bin, thus at the time of assignment, there is a covered bin which contains a total size equal to the size of items $p+1, \ldots, 2p+1$. OPT must assign a subset of at least $p+1$ items out of the first $2p+1$ items to one bin. Since the smallest such $p+1$ items already cover a bin, we may assume that these are indeed the items that OPT assigns to one bin. Therefore, the other bin of OPT will have items of the same sizes as the other bin of TBD, and TBD obtains an optimal solution,

Consider now the case where after the t-th item covers one bin, the $(t+1)$-th item covers the other bin. We have $t \geq 2p+2$, and $\text{OPT} \geq \sum_{i=1}^{t} s_i$ since $\text{OPT} \geq 2$ and $\sum_{i=1}^{t-1} s_i < 2$. The algorithm has a cost of $\sum_{i=1}^{t+1} s_i$. Since $s_{t+1} \leq s_i$ for $1 \leq i \leq t$ we get $\text{TBD} \leq \frac{t+1}{t}\text{OPT} \leq \frac{2p+3}{2p+2}\text{OPT}$.

If the $(t+1)$-th item does not cover the other bin, we use the same notations as in the proof of Theorem 3. Since $t \geq 2p+2$, both bins are covered after TBD switched to assigning items to the least loaded bin. Thus, the same properties holds, that is, $\text{TBD} \leq \text{OPT} + 2x \leq (1+x)\text{OPT}$ and $\text{TBD} \leq \text{OPT} + s_t \leq \text{OPT}(1 + \frac{s_t}{2})$. Since at least $2p+1$ items were assigned before any bin was covered, there exists at least one that received at least $p+1$ items before item t was assigned and thus $s_t \leq \frac{L}{p+1} = \frac{1-x}{p+1}$. If $x \leq \frac{1}{2p+2}$, we are done using the first bound. Otherwise, $s_t \leq \frac{2p+1}{(2p+2)(p+1)} < \frac{1}{p+1}$, which implies the required bound.

For the lower bound, we fix a small value $\varepsilon > 0$. We construct a sequence which starts with two items of size $\frac{1}{p+1} + p\varepsilon$. If they are assigned to the same bin, the next items are all of size $\frac{1}{p+1} - \varepsilon$. Otherwise, there are $2p-1$ items of size $\frac{1}{p+1}$ followed by items of size $\frac{1}{p+1} - 2p\varepsilon$. The optimal solution in the first case assigns one of the initial items, and p of the following items to each bin. In the second case it assigns the first two items, $p-2$ of the next batch of items, and one item of size $\frac{1}{p+1} - 2p\varepsilon$, to one bin, and $p+1$ items of size $\frac{1}{p+1}$ to another bin. The algorithm, in the first case, needs at least $p-1$ items for the bin that contains both largest items, and at least $p+2$ items for the other bin, getting a cost of at least $2(\frac{1}{p+1} + p\varepsilon) + (2p+1)(\frac{1}{p+1} - \varepsilon) = \frac{2p+3}{p+1} - \varepsilon$. In the second case, clearly at least $2p+2$ items are needed in total, at least $p+1$ for each bin. We next argue that $2p+3$ items are used by the algorithm. After the arrival of first $2p+1$ items, there exists a bin which received at most p items so far. Let a be the number of items on the less loaded bin at this time, its load at this time is $\frac{1}{p+1} + p\varepsilon + (a-1)\frac{1}{p+1} = \frac{a}{p+1} + p\varepsilon$. If it receives just one additional item, its load becomes $\frac{a}{p+1} + p\varepsilon + \frac{1}{p+1} - 2p\varepsilon < 1$, for $a \leq p$. Therefore, at least one bin receives at least $p + 2$ items in total, and we get a cost of at least $2 + \frac{1}{p+1} - 2p\varepsilon$. □

4 An arbitrary number of bins

In this section we consider online algorithm for the general problem in the two semi-online cases (of monotone non-decreasing item sizes and monotone non-increasing item sizes). We define two algorithms, PackDecreasing (PD) and PackIncreasing (PI).

4.1 Input sequences with non-decreasing item sizes

In this section, the algorithm uses several parameters. We give initial bounds on their values and later fix them exactly.

Items are partitioned into three types. Items of size at most $\frac{1}{2}$ are called *small.* Items of size in $(\frac{1}{2}, \alpha]$, where $\frac{3}{4} \leq \alpha \leq \frac{5}{6}$, are called medium, and items of size larger than α are called large. Since items arrive sorted by non-decreasing size, all small items arrive before all medium items, which are followed by large items. In particular, if OPT uses any large items for packing, then the further items which are used by the algorithm (but not by OPT) are all large.

A second parameter is β, where $\frac{1}{8} \leq \beta \leq \frac{1}{4}$ is used as follows. The first $\lfloor \beta m \rfloor$ bins are called *reserved bins*, the other bins are called *unreserved.* We use the following algorithm.

Algorithm PI
Phase 1. As long as the new arriving items are small, and there exists at least one reserved bin of load smaller than $1 - \alpha$, small items are assigned using List into reserved bins. If this process is stopped due to the arrival of a medium or a large item, go to Phase 2. Otherwise, go to Phase 4.
Phase 2. Use NF on the unreserved bins. Continue to Phase 3.
Phase 3. Use List on the reserved bins, until all of them are covered. Halt.
Phase 4. Assign arriving items using NF, first to the unreserved bins and then to reserved bins, until all bins are covered.

To analyze PI, denote the prefix of the input, which is used by OPT to cover the bins, by I. We first state some simple facts regarding the action of PI.

Proposition 2. *1. At the end of Phase 1, the load of every reserved bin is at most $\frac{3}{2} - \alpha$.*
2. After Phase 2 (if it is applied) every unreserved bin contains at most two items, each of size at least $\frac{1}{2}$. Therefore the number of items assigned at this step is at most $2(m - \lfloor \beta m \rfloor)$.
3. OPT $\geq m$, and if I contains $M > m$ large items, then OPT $\geq m + (M - m)(2\alpha - 1)$.

Proof. The first property holds since a reserved bin does not receive additional items after its load reaches or exceeds $1 - \alpha$. The second property holds since medium and large items have size larger than $\frac{1}{2}$. The only case in which a bin receives a single item is if the item has a size of 1. The third property holds since at least $M - m$ bins must receive two large items in OPT. □

Lemma 1. *If the algorithm performs Phase 4, then its asymptotic competitive ratio is at most $2 - \alpha\beta + \frac{\beta}{2}$.*

Proof. We consider two cases. If all unreserved bins received only small or medium items, we prove that the load of each such bin, except for possibly one bin, is at most 2α. If no unreserved bin received a medium item, then the load of each such bin is at most $\frac{3}{2} \le 2\alpha$. Otherwise, consider the first unreserved bin which received a medium item. All further unreserved bins receive exactly two medium items. The load of every reserved bin is at most 2. Thus the total size of items used is at most $2\alpha(m - \lfloor \beta m \rfloor - 1) + 2(\lfloor \beta m \rfloor + 1) = 2\alpha m + 2\lfloor \beta m \rfloor (1-\alpha) + 2 - 2\alpha \le m(2\alpha - 2\alpha\beta + 2\beta) + 2(1-\alpha)$.

Otherwise, if some unreserved bin received a large item, then all items which were assigned to reserved bins in Phase 4 are large. Since every reserved bin contained a total load of at least $1-\alpha$ after Phase 1, then a single large item is sufficient to cover it. Thus, the final load of every reserved bin is at most $\frac{5}{2} - \alpha$. The total size of items used is therefore at most $2(m - \lfloor \beta m \rfloor) + (\frac{5}{2} - \alpha)\lfloor \beta m \rfloor = 2m + (\frac{1}{2} - \alpha)\lfloor \beta m \rfloor \le 2m + (\frac{1}{2} - \alpha)(\beta m - 1) = 2m + (\frac{1}{2} - \alpha)(\beta m) + \alpha - \frac{1}{2}$.

The asymptotic competitive ratio is at most $\max\{2\alpha - 2\alpha\beta + 2\beta, 2 - \alpha\beta + \frac{\beta}{2}\}$. Since $2\alpha - 2\alpha\beta + 2\beta \le 2 - \alpha\beta + \frac{\beta}{2}$ is equivalent to $(\frac{3}{2} - \alpha)(2 - \beta) \ge 1$, which holds for our potential choices of α and β, the claim follows. □

We further consider the case where Phases 2,3 are performed, and thus the total size of small items is relatively small. Let γ be a parameter which is used only for the analysis, such that $\frac{1}{8} \le \gamma \le \frac{1}{5}$.

Lemma 2. *If I contains at least $2\lfloor \gamma m \rfloor$ medium items, then the asymptotic competitive ratio is at most $2(\gamma\alpha + 1 - \gamma)$.*

Proof. In Phase 2, at least $\lfloor \gamma m \rfloor$ unreserved bins receive two medium items each, and thus the total size of items used is at most $2\alpha\lfloor \gamma m \rfloor + 2(m - \lfloor \gamma m \rfloor) = 2m + 2\lfloor \gamma m \rfloor(\alpha - 1) \le 2m + 2\gamma\alpha m - 2\gamma m$. □

We are left with the case where the total size of both the medium and the small items in I is relatively small. Specifically, the total size of small items is at most $\lfloor \beta m \rfloor(\frac{3}{2} - \alpha)$, and the total size of medium items is at most $2\alpha\lfloor \gamma m \rfloor$.

Then the total size of the large items in I is at least $m - \lfloor \beta m \rfloor(\frac{3}{2} - \alpha) - 2\alpha\lfloor \gamma m \rfloor \ge m(1 - \beta(\frac{3}{2} - \alpha) - 2\alpha\gamma)$, which also gives the lower bound $(1 - \beta(\frac{3}{2} - \alpha) - 2\alpha\gamma)m$ on the number of large items in I. We would like to show that for the ranges of parameters which we allow, I contains at least $2\beta m$ large items. Indeed, we have $\beta(\frac{3}{2} - \alpha) + 2\alpha\gamma + 2\beta = (\frac{7}{2} - \alpha)\beta + 2\alpha\gamma \le (\frac{7}{2} - \alpha)\beta + \frac{2}{5}\alpha = (\frac{7}{2} - \alpha)(\beta - 0.4) + 1.4 \le 1$.

A bin of OPT is called *good*, if it contains a single large item and no medium items (but it may contain some small items). Let n_1 be the number of good bins in OPT, and n' the total number of large items in OPT.

Claim. $\text{OPT} \ge (n' - n_1)(\alpha - \frac{1}{2}) + m$

Proof. In OPT, there may be two types of bins which contain large items, and are not good. One type contains a large item together with a medium item, and the total size of items in such a bin is at least $\frac{1}{2} + \alpha = 1 + (\alpha - \frac{1}{2})$. The other type contains two large items, and total size of items in such a bin is at least $2\alpha = 1 + 2(\alpha - \frac{1}{2})$. Thus, every large item which is not packed in a good bin, contributes at least $\alpha - \frac{1}{2}$ to the total sum of items, in addition to the size of at least 1, which is packed in each bin. □

Lemma 3. *If* $n_1 \leq 2\beta m$*, then the asymptotic competitive ratio is at most* $\frac{2}{1+(\alpha-\frac{1}{2})(1-\frac{7}{2}\beta+\alpha\beta-2\alpha\gamma)}$.

Proof. The cost of PI is at most $2m$, therefore, by Claim 4.1, and by the calculation of the number of large items, the competitive ratio is at most $\frac{2m}{((1-\beta(\frac{3}{2}-\alpha)-2\alpha\gamma)m-2\beta m)(\alpha-\frac{1}{2})+m}$. □

Lemma 4. *If* $n_1 > 2\beta m$*, then the asymptotic competitive ratio is at most* $\max\{\frac{40}{21}, 2-\alpha\beta+\frac{\beta}{2}\}$.

Proof. Consider the last item assigned by PI to an unreserved bin, and denote it and its size by T. We show that this item must be large. If this is not an item of I, then it is no smaller than any item of I, while we assume that I contains some large items. Otherwise, since I contains at most $2\lfloor\gamma m\rfloor$ medium items, at least $m - \lfloor\beta m\rfloor - \lfloor\gamma m\rfloor \geq m(1-\beta-\gamma) > \frac{m}{2}$ unreserved bins received only large items. Moreover, if $T < 1$, then every unreserved bin which received large items, received two such items.

Assume first that $T \in I$ and $T < 1$. Then OPT contains at least $2m(1-\beta-\gamma) > m$ large items. Therefore, it has at least $m - 2m\beta - 2m\gamma$ bins which contain two large items. We get OPT $\geq m + (m - 2m\beta - 2m\gamma)(2\alpha - 1) \geq 1.05m$. Using PI $\leq 2m$, we get a competitive ratio of at most $\frac{40}{21} \approx 1.905$.

We show that in the other cases, every reserved bin receives a single large item. If $T = 1$, then all large items assigned to reserved bins are of size 1, so the claim is clear. Otherwise, we show the claim for $T \notin I$. It is enough to show that the total size of small items of every unreserved bin is at least $1 - T$.

Consider the good bins of OPT. Each such bin contains a large item of size at most T, and thus small items of total size at least $1 - T$.

Let S denote the total size of small items excluding one item of every bin, which is the last item assigned to this bin. We call these items *last items*. Since the small items are packed using List, the load of every bin (including the last item of each bin) is at least $\frac{S}{\lfloor\beta m\rfloor}$. Consider the list of good bins of OPT, and remove all such bins which contain at least one last item. This leaves at least $n_1 - \lfloor\beta m\rfloor > \beta m$ good bins. Therefore, we get $S \geq (\beta m)(1-T)$. Therefore, each unreserved bin is loaded by at least $1 - T$, before it received a large item (but including its last item).

Since every reserved bin receives one large item, it has a total size of items of at most $\frac{5}{2} - \alpha$, and we get the same situation as in Lemma 1, and thus the same bound. □

We use the parameters $\alpha = 0.826113$, $\beta = 0.211$ and $\gamma = 0.19778$. In order to find the values of the parameters, we used Matlab to solve a set of nonlinear equations, requiring the different upper bounds (which are found in the different cases) to be equal, and minimizing the resulting upper bound.

By Lemmas 1, 2, 3 and 4, we conclude that the asymptotic competitive ratio of the algorithm is at most 1.931215.

Theorem 5. *There exists an online algorithm whose asymptotic competitive ratio is at most* 1.931215 *for instances where items arrive sorted by non-decreasing sizes.*

4.2 Input sequences with non-increasing item sizes

Denote by h the number of items with sizes in the interval $\left(\frac{2}{3}, 1\right]$, and by ℓ the number of items with sizes in the interval $\left[\frac{1}{2}, \frac{2}{3}\right]$. Note that the values of h and ℓ are revealed to the algorithm online. Our algorithm has several cases based on the values of h and ℓ.

We next define the action of PD. The algorithm acts (without loss of generality) under the assumption that the input sequence does not contain unit sized items. In order to be able to assume this, if there are any unit sized items, then the algorithm starts by placing the unit sized items in separate bins. There exists an optimal solution which packs each such item into a separate bin, so the bins which are covered by unit sized items can be neglected.

Algorithm PD

The first $\min\{h, m\}$ items are packed into different bins.
Case 1: $h \geq m$. In this case, place the next m items one in each bin, such that bin i contains item i and item $2m + 1 - i$. Afterwards, pack the uncovered bins using NF.
Case 2: $h < m$, $2(m-h) \leq \ell$. Pack the next $2(m - h)$ items in pairs (of consecutive items) in the bins of indices $h + 1, \ldots, m$, that is, into new bins that do not contain items larger than $\frac{2}{3}$. Continue to pack the next h items one in each bin (these items are added to the first h items), and then use NF to cover all uncovered bins.
Case 3: $h < m$, $2(m-h) > \ell$. Let $h' = \lfloor \frac{\ell}{2} \rfloor$. Pack the next $2h'$ items in pairs (of consecutive items) in the bins of indices $h+1, \ldots, h+h'$, that is, into new bins that do not contain items larger than $\frac{2}{3}$. If $\ell > 2h'$ (i.e., ℓ is odd), pack one item into the bin of index $h + h' + 1 \leq m$. Note that h' is the number of bins that the algorithm covers using a pair of items with sizes in $\left[\frac{1}{2}, \frac{2}{3}\right]$. Pack the next $2(m - h) - \ell$ items such that each of the m bins contains either one item of size greater than $\frac{2}{3}$ or exactly two items of the first $2m-h$ items. Pack the next $m-h'$ items one in each uncovered bin, starting with the first h bins, and the uncovered bins at the end of the packing of the first $3m - h' - h$ items, are covered using NF.

Note that in the last two cases, PD acts in the same way until the value of ℓ is revealed, that is, $\min\{2(m-h), \ell\}$ items are packed into new bins, so that each bin (except for possibly the last one, if $\ell < 2(m - h)$ and ℓ is odd) receives two items.

We analyze the three cases in Lemmas 5, 6, 7. The proofs of the first two lemmas are similar, while the third one is more complicated.

Lemma 5. *The asymptotic competitive ratio of* PD *in the first case is at most* $\frac{4}{3}$.

Proof. Note that since we assume that the input sequence does not contain a unit size item, then each bin of the optimal solution has at least two items, and therefore OPT uses the first $2m$ items. Denote by t the size of the $2m$-th item, then since each bin of the algorithm has at least one item of size at least $\frac{2}{3}$ and the second item we put in the bin has size at least t, we conclude that after the first $2m$ items are packed by the algorithm each bin has load of at least $\frac{2}{3} + t$ and the size of each additional item is at most t. Then, we are using NF to cover a gap of a total size of at most $1 - (\frac{2}{3} + t) = \frac{1}{3} - t$ in each bin, using items of size at most t. Hence the total size of the items, which are added to such a bin, by the phase where NF is applied, is at most $\frac{1}{3} - t + t = \frac{1}{3}$. Hence the total size of the items that the algorithm uses is at most the total size of the first $2m$ items plus at most $\frac{m}{3}$. The claim follows since the total size of the first $2m$ items is at most OPT and $m \leq$ OPT. □

Lemma 6. *The asymptotic competitive ratio of* PD *in the second case is at most* $\frac{4}{3}$.

Proof. Note that since we assume that the input sequence does not contain a unit size item, then each bin of the optimal solution has at least two items, and therefore OPT uses the first $2m$ items. Denote by t the size of the $2m$-th item, then since each bin of the algorithm that is not covered by the first $2m$ items, has at least one item of size at least $\frac{2}{3}$ and the second item we put in the bin has size at least t, we conclude that after the first $2m$ items are packed by the algorithm each bin that is still uncovered, has load of at least $\frac{2}{3}+t$ and the size of each item is at most t. Then, we are using NF to cover a gap of a total size of at most $1-(\frac{2}{3}+t)=\frac{1}{3}-t$ in each bin, using items of size at most t. Hence the total size of the items, which are added to such a bin, by the phase where NF is applied, is at most $\frac{1}{3}-t+t=\frac{1}{3}$ for each bin which is uncovered prior to this step. Hence the total size of the items that the algorithm uses is at most the total size of the first $2m$ items plus at most $\frac{m}{3}$. The claim follows since the total size of the first $2m$ items is at most OPT and $m \le$ OPT. □

Lemma 7. *The asymptotic competitive ratio of* PD *in the third case is at most* $\frac{4}{3}$.

Proof. Recall that we assume that the input sequence does not contain a unit size item. Hence each bin in OPT must have at least two items. Moreover, each bin of OPT that has only two items must have an item of size at least $\frac{1}{2}$. Hence there are at most $h+\ell \le h+2h'+1$ such bins, and each other bin must have at least three items. Therefore, OPT uses the first $2(h+2h'+1)+3(m-h-2h'-1)=3m-h-2h'-1$ items. Clearly, OPT uses at least $2m$ items.

The number of items that the algorithm uses until the phase where NF is applied, is $2m+(m-h-h')=3m-h-h'$. That is, at most $h'+1$ items that are not used by OPT. Another upper bound on the number of items that are not used by OPT among these items is $m-h-h'$. Denote the size of the $(2m-h)$-th item by s. Clearly, $s \le \frac{1}{2}$. Therefore, since $(2m-h)+(m-h-h')=3m-2h-h' \le 3m-h-h'$, the additional items appear in the sequence no earlier than the $(2m-h)$-th item and so every one of the additional items has a size of at most s. Bins that received two items of size at least $\frac{1}{2}$ are covered by these two items, and so, the number of uncovered bins prior to the phase of NF is at most $m-h'$.

If $s \le \frac{1}{3}$, then at the time of assignment of the $(2m-h)$-th item, all bins contain two items each, except for the first h bins that contain just one item each, no bin contains a load of more than $\frac{4}{3}$ (since the only covered bins contain two items of size at most $\frac{2}{3}$). Therefore, this would be the situation also after all bins are covered, and the asymptotic competitive ratio would be at most $\frac{4}{3}$.

Assume therefore that $s > \frac{1}{3}$, and hence at the time, when $(2m-h)$-th has just been assigned, all bins contain a load of at least $2s > \frac{2}{3}$. Denote by t the size of the $(3m-h-h')$-th item, then each bin out of the first h received an item of size at least t in addition to the load of at least $\frac{2}{3}$. Therefore, each of these bins is filled with a load of at least $\frac{2}{3}+t$ and the size of the remaining items is at most t. The final loads of these bins would be no larger than $1+t$, thus, the application of NF adds a total load of at most $\frac{1}{3}$ to each such bin. As for the other uncovered bins, after receiving three items per bin, each one of them contain a load of at least $2s+t$. Since their final loads would be no larger than $1+t$, each one of them receives an additional load of $1-2s$.

Therefore, the total cost of the algorithm is at most the total size of the first $3m-h-2h'-1$ items plus $s(h'+1)+\frac{h}{3}+(1-2s)(m-h-h')$. The first term is at most OPT, and $m \le$ OPT, so we get a total of at most $(2-2s)\text{OPT}+(3s-1)h'+(2s-\frac{2}{3})h+s$.

We next compute an additional upper bound on the cost of the algorithm in this case. The algorithm covers h' bins with a total load of at most $\frac{4}{3}$ (these are the bins with a pair of items of size in the interval $\left[\frac{1}{2},\frac{2}{3}\right]$). After $2m$ items have been assigned, each one of the first h bins contains a total load of at least $\frac{2}{3}+t$. Therefore, the application of NF can increase the load of such a bin by at most $\frac{1}{3}$. The last $m-h-h'$ bins contain at least a load of $2s$

at this time, and finally at most $1+t$, so they gain at most $1+t-2s \leq 1-s$, since $t \leq s$. Hence the total cost of the resulting solution is at most the size of the first $2m$ items, plus at most $\frac{h}{3}+(1-s)(m-h-h')$. Thus the cost is at most $(2-s)\text{OPT}+(s-1)h'+(s-\frac{2}{3})h$. We multiply the first bound on the cost by $\frac{1-s}{2s}>0$, and the other bound by $\frac{3s-1}{2s}\geq 0$, and get the following upper bound on the cost: $\frac{1}{2s}\left((3s-s^2)\text{OPT}+(s^2-\frac{s}{3})h+s-s^2\right)$.

Using $h \leq m \leq \text{OPT}$, we get a cost of at most $\frac{4\text{OPT}}{3}+\frac{1-s}{2} \leq \frac{4\text{OPT}}{3}+\frac{1}{3}$. □

By Lemmas 5, 6 and 7, we conclude that the asymptotic competitive ratio of the algorithm is at most $\frac{4}{3}$.

Theorem 6. *There exists an online algorithm whose asymptotic competitive ratio is at most $\frac{4}{3}$ for instances where items arrive sorted by non-increasing sizes.*

5 Lower bounds

In this section we prove lower bounds on the asymptotic competitive ratio of any algorithm for m bins, for all three variants.

5.1 A lower bound for input sequences with non-decreasing items

Theorem 7. *For every $p, \alpha' > 1$, the asymptotic competitive ratio of any algorithm for input sequences with non-decreasing items is at least*

$$\frac{\alpha'(p+1)-1}{\alpha'(p+1)-1-\ln\alpha'}.$$

Proof. Let κ be an integer and let $\alpha \leq \alpha'$ be a rational number such that $\kappa\cdot\alpha$ is an integer. Let $\delta>0$ be an arbitrarily small positive value, such that $\delta<\frac{1}{(p+1)\kappa\cdot m}$. We choose m to be divisible by $(\kappa\cdot\alpha)!$. The sequence consists of $m\kappa$ items of size δ, followed by items of size $\frac{1-t\delta}{p}$ for some integer t such that $\kappa \leq t \leq \kappa\cdot\alpha$. By the choice of δ, every bin must receive at least p items of size $\frac{1-t\delta}{p}$ to be covered.

We compute the cost of an optimal solution OPT for a given value of t. OPT has $\frac{m\kappa}{t}$ bins, each of which contains t items of size δ, and p items of size $\frac{1-t\delta}{p}$. All other bins contain $p+1$ items of size $\frac{1-t\delta}{p}$. Note that $(p+1)\frac{1-t\delta}{p}=1+\frac{1}{p}-\frac{t\delta(p+1)}{p}\geq 1$, by the definition of δ and since $\kappa\alpha<m$. Letting δ tend to zero, we get a cost of $\frac{m\kappa}{t}+\frac{p+1}{p}(m-\frac{m\kappa}{t})=\frac{p+1}{p}m-\frac{m\kappa}{pt}$.

Given an assignment of an algorithm $\mathcal{A}$, of the items of size δ, we define X_0 to be the number of bins that have at most $\kappa-1$ items. We can assume that these bins are empty, since each such bin requires $p+1$ items of size $\frac{1-t\delta}{p}$ to be covered, no matter what the actual value of t is. Let X_ℓ be the number of bins which contain ℓ items of size δ, for $\kappa\leq\ell\leq\kappa\cdot\alpha$. We may assume that no bin contains more than $\kappa\cdot\alpha$ items of size δ, since for any value of t, adding p items of size $\frac{1-t\delta}{p}$ to a bin with $\kappa\cdot\alpha$ items of size δ covers this bin, and no bin with at most $p-1$ items of size $\frac{1-t\delta}{p}$ is covered. We get $\sum\limits_{\ell=\kappa}^{\alpha\kappa}\ell\cdot X_\ell=m\kappa$. Given a specific value of t, such that the second part of the input consists of items of size $\frac{1-t\delta}{p}$, bins with at least t items of size δ require p additional items, while all other bins require $p+1$ additional items. For $\delta\to 0$, this gives a cost of $\frac{p+1}{p}(X_0+\sum\limits_{i=\kappa}^{t-1}X_i)+\sum\limits_{i=t}^{\kappa\cdot\alpha}X_i=\frac{p+1}{p}m-\frac{1}{p}\sum\limits_{i=t}^{\kappa\cdot\alpha}X_i$.

Let $\mathcal{R}$ be the asymptotic competitive ratio of $\mathcal{A}$. We neglect a possible additive constant in the definition since we can let $m \to \infty$.

We have
$$\frac{p+1}{p}m - \frac{1}{p}\sum_{i=t}^{\kappa\cdot\alpha} X_i \le \mathcal{R}\cdot\left(\frac{p+1}{p}m - \frac{m\kappa}{pt}\right).$$

We sum up all the inequalities, where the inequality for $t=\kappa$ is multiplied by κ, and we conclude that
$$m\alpha\kappa\frac{p+1}{p} - \frac{1}{p}\sum_{\ell=\kappa}^{\alpha\kappa}\ell\cdot X_\ell \le \mathcal{R}\left(m\alpha\kappa\frac{p+1}{p} - \frac{m\kappa}{p} - \sum_{\ell=\kappa+1}^{\alpha\kappa}\frac{m\kappa}{p\ell}\right).$$

Using $\sum\limits_{\ell=\kappa}^{\alpha\kappa}\ell\cdot X_\ell = m\kappa$, this gives
$$\alpha(p+1)-1 \le \mathcal{R}\left(\alpha(p+1)-1-\sum_{\ell=\kappa+1}^{\alpha\kappa}\frac{1}{\ell}\right).$$

For large enough κ, we have $\sum\limits_{\ell=\kappa+1}^{\alpha\kappa}\frac{1}{\ell} \approx \ln\alpha$, and the lower bound follows, by letting $\alpha \to \alpha'$. □

Corollary 1. *Theorem 7 implies a lower bound of* 1.302017 *for* $p=1$ *(using* $\alpha' = 2.1555664$*), a lower bound of* 1.16445 *for* $p=2$ *(using* $\alpha' = 2.3603$*), a lower bound of* 1.11326 *for* $p=3$ *(using* $\alpha' = 2.4551$*), and a lower bound strictly larger than 1 for every value of* p *(using, e.g.,* $\alpha' = e$*).*

5.2 A lower bound for input sequences with non-increasing items

Theorem 8. *For every* $p \ge 1$*, the asymptotic competitive ratio of any algorithm for input sequences with non-increasing items is at least*
$$\frac{p^2+3p+1}{p^2+3p+2-H_{p+1}},$$
where H_i *is the* i*-th harmonic number, i.e.,* $H_i = \sum\limits_{j=1}^{i}\frac{1}{j}$.

Proof. We choose m to be divisible by $(p+1)!$. The sequence consists of m items of size $\frac{1}{p+1}+\varepsilon$ (for some $0<\varepsilon<\frac{1}{(p+2)^2}$, followed by a sequence consisting of items of size $\frac{1}{p+1}-\frac{p+1-i}{i}\varepsilon$ for some $1 \le i \le p$, or items of size $\frac{1}{p+1}-(p+1)\varepsilon$ (this is defined as the case $i=0$).

Note that for a given value of i, $p+1-j$ items of size $\frac{1}{p+1}+\varepsilon$ and j items of size $\frac{1}{p+1}-\frac{p+1-i}{i}\varepsilon$ cover a bin if and only if $j \le i$, since $(p+1-j)(\frac{1}{p+1}+\varepsilon)+j(\frac{1}{p+1}-\frac{p+1-i}{i}\varepsilon) = 1+\varepsilon(p+1-\frac{j}{i}(p+1))$. For items of size $\frac{1}{p+1}-(p+1)\varepsilon$, even a combination of p items of size $\frac{1}{p+1}+\varepsilon$ together with one item of size $\frac{1}{p+1}-(p+1)\varepsilon$ is not enough to cover a bin, thus the same property holds for $i=j=0$ as well, that is, a bin which contains $p+1$ items is covered if and only if all these items are of size $\frac{1}{p+1}+\varepsilon$.

Consider an optimal solution for a given value of i, which we denote by OPT_i. Such a solution consists of $\frac{m}{p+1-i}$ bins with $p+1-i$ items of size $\frac{1}{p+1}+\varepsilon$ and i items of size $\frac{1}{p+1}-\frac{p+1-i}{i}\varepsilon$ and additional $m-\frac{m}{p+1-i}$ bins with $p+2$ items of size $\frac{1}{p+1}-\frac{p+1-i}{i}\varepsilon$. For $i=0$ the first set of bins contains only the items of size $\frac{1}{p+1}+\varepsilon$, and in the second set of bins, the additional items are of size $\frac{1}{p+1}-(p+1)\varepsilon$. Thus for $\varepsilon \to 0$, we have $\text{OPT}_i = \frac{m}{p+1-i}+\frac{p+2}{p+1}(m-\frac{m}{p+1-i}) = \frac{p+2}{p+1}m - \frac{1}{(p+1)(p+1-i)}m$.

Let X_j (for $0 \le j \le p+1$) be the number of bins of the algorithm with j items after the arrival of the items of size $\frac{1}{p+1}+\varepsilon$. Clearly, $\sum_{j=0}^{p+1} X_j = m$ and $\sum_{j=0}^{p+1} j \cdot X_j = m$.

For a given value of i, such that the second part of the input contains items of size $\frac{1}{p+1}-\frac{p+1-i}{i}\varepsilon$, or $\frac{1}{p+1}-(p+1)\varepsilon$ if $i=0$, the algorithm needs to add j items to bins that contain $p+1-j$ items such that $j \le i$, and $j+1$ items to bins that contain $p+1-j$ items such that $i < j \le p+1$. Therefore, for $\varepsilon \to 0$ the algorithm will contain a total size of 1 in $X_{p+1}+\ldots+X_{p+1-i}$ bins, and a total size of $\frac{p+2}{p+1}$ in $X_{p-i}+\ldots+X_0$ bins. This gives the cost $m\frac{p+2}{p+1}-\frac{1}{p+1}\sum_{j=p+1-i}^{p+1} X_j$.

Let $\mathcal{R}$ be the asymptotic competitive ratio. We have

$$m\frac{p+2}{p+1} - \frac{1}{p+1}\sum_{j=p+1-i}^{p+1} X_j \le \mathcal{R}(\frac{p+2}{p+1}m - \frac{1}{(p+1)(p+1-i)}m) \ .$$

Multiplying by $\frac{p+1}{m}$ and taking the sum over $0 \le i \le p$, we get,

$$(p+1)(p+2) - \frac{1}{m}\sum_{j=0}^{p+1} j \cdot X_j \le \mathcal{R}((p+2)(p+1) - H_{p+1}) \ .$$

Using $\sum_{j=0}^{p+1} j \cdot X_j = m$ we get the claimed lower bound. □

Corollary 2. *Theorem 8 implies a lower bound of* $\frac{10}{9} \approx 1.111$ *for* $p=1$*, a lower bound of* 1.081967 *for* $p=2$*, a lower bound of* 1.06 *for* $p=3$*, and a lower bound strictly larger than 1 for every value of* p*.*

5.3 A lower bound for arbitrary input sequences

We combine the methods used in the two previous lower bounds to obtain an improved lower bound for the case where there is no restriction on the order in which items arrive.

Theorem 9. *For every* $\alpha' > 1$*, the asymptotic competitive ratio of any algorithm for arbitrary input sequences is at least* $\frac{7\alpha'-5+3\ln\alpha'}{6\alpha'-4}$.

Proof. Let κ be an integer and let $\alpha \le \alpha'$ be a rational number such that $\kappa \cdot \alpha$ is an integer. The input starts as in the proof of Theorem 7, with $m\kappa$ very small items of size δ, followed by items of size $1-t\delta$ for some integer t such that $\kappa \le t \le \alpha\kappa$, but the number of items of size $1-t\delta$ is exactly $\frac{m\kappa}{t}$. If $t > \kappa$, additional items are introduced, using a construction similar to the proof of Theorem 8. First, $m-\frac{m\kappa}{t}$ items of size $\frac{1}{2}+\varepsilon$, followed by either $m-\frac{m\kappa}{t}$ items of size $\frac{1}{2}-\varepsilon$ or $\frac{m}{2}-\frac{m\kappa}{2t}$ items of size 1. If additional items are needed, the last size of the sequence is repeated until all bins are covered. The value ε satisfies the following properties. Roughly, it is defined to be small but much larger than δ, so that when the input continues to items of size $\frac{1}{2}+\varepsilon$, the items of size δ in uncovered bins can be neglected. Thus we have $\varepsilon > m\kappa\delta$. On the other hand, we assume $\varepsilon < \frac{1}{6}$, and let ε tend to zero later.

An optimal solution has $\frac{m\kappa}{t}$ bins that contains one item of size $1-t\delta$ and t items of size δ. If the sequence ends with items of size $\frac{1}{2}-\varepsilon$, then each of the $m-\frac{m\kappa}{t}$ additional bins contains one item of size $\frac{1}{2}+\varepsilon$ and one item of size $\frac{1}{2}-\varepsilon$. Otherwise, there are $\frac{m}{2}-\frac{m\kappa}{2t}$ bins with two items of size $\frac{1}{2}+\varepsilon$ and $\frac{m}{2}-\frac{m\kappa}{2t}$ bins with a single item of size 1. Letting $\varepsilon \to 0$, we get that $\text{OPT} = m$ in all cases.

To compute the cost of the algorithm for a given value of t, we first note that it can be assumed, without loss of generality, that all bins which contain at least t items of size δ, receive exactly one item of size $1-t\delta$. Note first that the number of such bins is at most $\frac{m\kappa}{t}$, thus the number of items of size $1-t\delta$ is large enough. Assigning one such larger item to each bin with at least t items of size δ covers the bin, therefore the algorithm clearly never assigns more than one such larger item to these bins. Consider the situation of the algorithm after the arrival of the first two types of items. Assume that there exists a bin b with at least t smaller items and with no larger item. If there exists a bin with at most $t-1$ smaller items and one larger item, then moving the larger item to b clearly improves the situation of the algorithm; it now has one empty bin and one covered bin instead of one empty bin and one bin which requires one additional item to be covered. The uncovered bins containing only items of size δ can be viewed as empty at this time since the items of size δ would not contribute towards covering the bin in the future. If there exists a bin with at most $t-1$ smaller items and two larger items, then moving one larger item to b again improves the situation of the algorithm; it now has one covered bin and one uncovered bin that will get covered using any future item, instead of one covered bin and one empty bin.

Denote the number of bins that contain j items of size δ by X_j $(j = 0, \kappa, \kappa+1, \ldots, \alpha\kappa-1, \alpha\kappa)$. We have $\sum\limits_{j=\kappa}^{\alpha\kappa} X_j + X_0 = m$ and $\sum\limits_{j=\kappa}^{\alpha\kappa} j \cdot X_j = m\kappa$. After the arrival of items of size $1-t\delta$ and the items of size $\frac{1}{2}+\varepsilon$, we have several types of bins. As shown above, there are $\sum\limits_{j=t}^{\kappa\alpha} X_j$ covered bins where the total size of items is 1 (for $\delta \to 0$). There are additional six types of bins, according to number of items of sizes $1-t\delta$ and $\frac{1}{2}+\varepsilon$ in these bins. The numbers of such bins are denoted by a_t, b_t, c_t, d_t, e_t and f_t. The types are bins with one item of size $1-t\delta$, bins with one item of size $1-t\delta$ and one item of size $\frac{1}{2}+\varepsilon$, bins with two items of size $\frac{1}{2}+\varepsilon$, bins with one item of size $\frac{1}{2}+\varepsilon$, bins with no items and bins with two items of size $1-t\delta$. Clearly, we have

$$a_t + b_t + c_t + d_t + e_t + f_t = m - \sum_{j=t}^{\alpha\kappa} X_j \ , \tag{1}$$

$a_t + b_t + 2f_t = \frac{m\kappa}{t}$ and $b_t + 2c_t + d_t = m - \frac{m\kappa}{t}$. Subtracting half the second equality from the first one gives,

$$a_t + \frac{b_t}{2} - c_t - \frac{d_t}{2} + 2f_t = \frac{3m\kappa}{2t} - \frac{m}{2}. \tag{2}$$

If the last items are of size $\frac{1}{2}-\varepsilon$, the cost of the algorithm (if $\varepsilon \to 0$, which implies also $\delta \to 0$) is $m+\frac{a_t}{2}+\frac{b_t}{2}+\frac{e_t}{2}+f_t$. If the last items are of size 1, the cost of the algorithm (if $\varepsilon \to 0$) is $m+a_t+\frac{b_t}{2}+\frac{d_t}{2}+f_t$. Let $\mathcal{R}$ be the asymptotic competitive ratio, then $m+\frac{a_t}{2}+\frac{b_t}{2}+\frac{e_t}{2}+f_t \le \mathcal{R}m$

and $m + a_t + \frac{b_t}{2} + \frac{d_t}{2} + f_t \leq \mathcal{R}m$. Multiplying the first inequality by 2 and adding the second one, we get, $3m + 2a_t + \frac{3b_t}{2} + \frac{d_t}{2} + e_t + 3f_t \leq 3\mathcal{R}m$. Using (1) and subsequently (2), we get, $3m + a_t + \frac{b_t}{2} - c_t - \frac{d_t}{2} + 2f_t + m - \sum_{j=t}^{\alpha\kappa} X_j \leq 3\mathcal{R}m$ and $\frac{7m}{2} + \frac{3m\kappa}{2t} - \sum_{j=t}^{\alpha\kappa} X_j \leq 3\mathcal{R}m$, for $\kappa < t \leq \alpha\kappa$.

For $t = \kappa$, the calculation is different. The cost of the algorithm for $\delta \to 0$ is $m + X_0 = 2m - \sum_{j=\kappa}^{\alpha\kappa} X_j$, so we have $2m - \sum_{j=\kappa}^{\alpha\kappa} X_j \leq \mathcal{R}m$. We multiply this inequality by κ and sum up with all other inequalities, which gives,

$$2m\kappa + \frac{7m}{2}(\alpha\kappa - \kappa) + \frac{3m\kappa}{2} \sum_{j=\kappa+1}^{\alpha\kappa} \frac{1}{j} - \sum_{j=\kappa}^{\alpha\kappa} jX_j \leq \mathcal{R}m(3(\alpha\kappa - \kappa) + \kappa) \ .$$

Using $\sum_{j=\kappa}^{\alpha\kappa} j \cdot X_j = m\kappa$ and dividing the inequality by $\frac{m\kappa}{2}$ we have $4 + 7(\alpha - 1) + 3 \sum_{j=\kappa+1}^{\alpha\kappa} \frac{1}{j} - 2 \leq 2\mathcal{R}(3\alpha - 2)$.

For large enough κ, we have $\sum_{\ell=\kappa+1}^{\alpha\kappa} \frac{1}{\ell} \approx \ln \alpha$. We get $\mathcal{R} \geq \frac{7\alpha - 5 + 3\ln\alpha}{6\alpha - 4}$ by letting $\alpha \to \alpha'$ the lower bound follows. □

Corollary 3. *Theorem 9 implies a lower bound of* 1.387667 *(using* $\alpha' = 2.2624$*).*

Remark 1. In this section we focused only on the case $p = 1$. Lower bounds for the case $p > 1$ follow from the previous sections. In order to prove improved lower bounds for $p \geq 2$, constructions that combine ideas from Theorems 7 and 8, similarly to the result for $p = 1$, can be used.

6 Conclusion

In this paper, we presented online and semi-online algorithms for a minimization version of a bin covering problem. We gave tight results for the case of two bins for three types of inputs, general inputs, inputs where items are sorted by non-decreasing size, and inputs where items are sorted by non-increasing size.

For multiple bins, a class of greedy algorithms (which we call thrifty algorithms) have an asymptotic competitive ratio of at most 2 (in fact, they have an absolute competitive ratio of at most 2). We designed algorithms which perform better for non-increasing inputs, and for non-decreasing inputs. The design of such an algorithm for arbitrarily ordered inputs remains as an open problem.

References

1. S.F. Assmann, *Problems in discrete applied mathematics*, PhD thesis, Mathematics Department, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1983.
2. S.F. Assmann, D.S. Johnson, D.J. Kleitman, J.Y.-T. Leung, On a dual version of the one-dimensional bin packing problem, *Journal of Algorithms*, 5 (1984) 502-525.
3. J. Csirik, V. Totik, On-line algorithms for a dual version of bin packing, *Discrete Applied Mathematics*, 21 (1988) 163-167.
4. N. Alon, Y. Azar, J. Csirik, L. Epstein, S.V. Sevastianov, A.P.A. Vestjens, G.J. Woeginger, On-line and off-line approximation algorithms for vector covering problems, *Algorithmica*, 21(1998) 104-118.

5. J. Csirik, D. S. Johnson, C. Kenyon, Better approximation algorithms for bin covering, in: S. Rao Kosaraju (Ed.), *Proc. of the 12th Annual Symposium on Discrete Algorithms (SODA2001)*, SIAM, Philadelphia, 2001, pp. 557-566.
6. D. K. Friesen, B. L. Deuermeyer, Analysis of greedy solutions for a replacement part sequencing problem, *Mathematics of Operations Reasearch*, 6 (1981) 74-87.
7. K. Jansen, R. Solis-Oba, An asymptotic fully polynomial time approximation scheme for bin covering, *Theoretical Computer Science*, 306 (2003) 543-551.
8. E. G. Coffman, J. Csirik, and J. Leung, Variants of classical one-dimensional bin packing, in T. Gonzalez (Ed.), *Handbook of Approximation Algorithms and Meta-Heuristics*, Francis and Taylor Books (CRC Press), London, 2007, Chapter 33.
9. G. Lueker, Average-case analysis of off-line and on-line knapsack problems, *Journal of Algorithms*, 29 (1998) 277-305.
10. A. Marchetti-Spaccamela, C. Vercellis, Stochastic on-line knapsack problems, *Mathematical Programming*, 68 (1995) 73-104.
11. M. Babaioff, N. Immorlica, D. Kempe, R. Kleinberg, A knapsack secretary problem with applications, in: M. Charikar, K. Jansen, O. Reingold, J. D. P. Rolim (Eds.), *Proc. of the 10th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX2007)*, Springer, Berlin, 2007, pp. 16-28.
12. K. Iwama, S. Taketomi, Removable on-line knapsack problems, in: P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. Eidenbenz, R. Conejo (Eds.), *Proc. of the 29th International Colloquium on Automata, Languages and Programming (ICALP2002)*, Springer, Berlin, 2002, pp. 293-305.
13. Y. Azar, J. Boyar, L. Epstein, L. M. Favrholdt, K. S. Larsen, M. N. Nielsen, Fair versus unrestricted bin packing, *Algorithmica*, 34 (2002) 181-196.
14. L. Epstein, L. M. Favrholdt, On-Line maximizing the number of items packed in variable-sized bins, *Acta Cybernetica*, 16 (2003) 57-66.