Algorithmica (2010) 56: 505–528
DOI 10.1007/s00453-008-9188-9

Bin Packing with Rejection Revisited

Leah Epstein

Received: 25 July 2007 / Accepted: 24 March 2008 / Published online: 8 April 2008
© Springer Science+Business Media, LLC 2008

Abstract We consider the following generalization of bin packing. Each item has a size in (0, 1] associated with it, as well as a rejection cost, that an algorithm must pay if it chooses not to pack this item. The cost of an algorithm is the sum of all rejection costs of rejected items plus the number of unit sized bins used for packing all other items.

We first study the offline version of the problem and design an APTAS for it. This is a non-trivial generalization of the APTAS given by Fernandez de la Vega and Lueker for the standard bin packing problem. We further give an approximation algorithm of an absolute approximation ratio $3/2$, where this value is best possible unless $\mathbf{P} = \mathbf{NP}$.

Finally, we study an online version of the problem. For the bounded space variant, where only a constant number of bins can be open simultaneously, we design a sequence of algorithms whose competitive ratios tend to the best possible asymptotic competitive ratio. These algorithms are generalizations of bounded space algorithms for standard bin packing. We show that our algorithms have the same asymptotic competitive ratios as these known for the standard problem, for which the sequence of the competitive ratios tends to $\Pi_\infty \approx 1.691$. Furthermore, we introduce an unbounded space algorithm which achieves a much smaller asymptotic competitive ratio. All our results improve upon previous results of Dósa and He.

Keywords Bin packing · Approximation schemes · Online algorithms

An extended abstract of this paper will appear in the Proceedings of the 4th Workshop on Approximation and Online Algorithms (WAOA 2006).

L. Epstein (✉)
Department of Mathematics, University of Haifa, 31905 Haifa, Israel
e-mail: lea@math.haifa.ac.il

1 Introduction

In the classical bin packing problem [5, 6, 23], a set (or sequence) of items, which are positive numbers no larger than 1, are to be packed into unit sized bins. The sum of items packed into one bin cannot exceed its size and the supply of such bins is unbounded. Each item must be packed into exactly one bin, minimizing the number of non-empty bins. However, in many applications, it is possible to *refuse* to pack an item. This rejection needs to be compensated, and costs some given amount for each item, which is called the "rejection cost" (or "rejection penalty") of the item. In an application where bins are disks and items are files to be saved on these disks, the rejection cost of a file is the cost of transferring it to be saved on alternative media. In another application, where bins are storage units, a rejection cost is paid to a disappointed customer whose goods cannot be stored.

We call the packing problem studied in this paper BIN PACKING WITH REJECTION. In this problem, an item has both a size and a rejection cost associated with it. Each item must be either assigned to a bin or rejected. A bin is *empty* if no item is assigned to it, otherwise it is *used*. Items that have been packed are referred to as *accepted*. Unlike the standard problem where the goal is to minimize the number of used bins, the target function in the problem with rejection is the sum of the following two amounts. The first one is the sum of all rejection costs of rejected items. The second one is the number of bins, which are used to pack the accepted items. The goal is to minimize this sum. Clearly, standard bin packing is a special case of bin packing with rejection, where all rejection costs are larger than 1.

We denote the set of items by I, where $|I| = n$. For an item $i \in I$, we denote its size by p_i and its rejection cost (or penalty) by r_i. In this paper we study both offline and online algorithms for bin packing with rejection. In online environments of bin packing problems, items are received as a sequence σ. In bin packing with rejection, every element in the sequence is a pair, consisting of the size and the rejection cost of this element. Thus, we get a sequence $(p_1, r_1), (p_2, r_2), \dots, (p_n, r_n)$, where the elements arrive one by one. Upon arrival, an item must be either assigned or rejected. Such a decision is irrevocable. Note that the set I contains the same elements as σ.

The bin packing problem with rejection was introduced and studied by Dósa and He [10]. They suggested an interesting application for the offline version of the problem which is related to caching. Items are files which would need to be used in a local system. Each file would be needed exactly once at a later time. A file can be downloaded in advance to this local system, and stored on local web servers. The process of downloading a file from a local server (when it is actually needed) is fast, but stored files consume space on the servers. In this case the incurred cost results from the cost of local servers. The second option is to download a file only when it is actually needed, without storing it first. In the last case, a rejection cost occurs which results from the communication time of downloading the file from an external server. An algorithm would need to have a cost as low as possible with respect to the sum of the two types of costs.

For an algorithm $\mathcal{A}$, we denote its cost by $\mathcal{A}$ as well. The cost of an optimal offline algorithm that knows the complete sequence of items is denoted by OPT. In this paper we mostly consider the asymptotic competitive ratio and the asymptotic approximation ratio criteria. When we discuss performance guarantees of algorithms, we use the term *competitive* for online algorithms and the term *approximation* for offline algorithms. The asymptotic measures are standard measures of algorithm quality for bin packing problems. For a given input σ, let $\mathcal{A}(\sigma)$ be cost of algorithm $\mathcal{A}$ on σ. Let $\mathrm{OPT}(\sigma)$ be the minimum possible cost of serving all items in σ (i.e., the cost of packing a subset of the items plus the cost of rejecting all other items). The *asymptotic approximation ratio* (or *asymptotic competitive ratio*) for an algorithm $\mathcal{A}$ is defined to be $\mathcal{R}_{\mathcal{A}} = \limsup_{\nu\to\infty} \sup_{\sigma}\{\frac{\mathcal{A}(\sigma)}{\mathrm{OPT}(\sigma)} | \mathrm{OPT}(\sigma) = \nu\}$. We also consider the absolute approximation ratio in this paper. The absolute approximation ratio (or competitive ratio) of $\mathcal{A}$ is the infimum $\mathcal{R}$ such that for any input, $\mathcal{A} \le \mathcal{R} \cdot \mathrm{OPT}$. If the approximation (competitive) ratio of a polynomial time offline (respectively, online) algorithm is at most $\mathcal{R}$, we say that it is a $\mathcal{R}$-approximation (respectively, $\mathcal{R}$-competitive), this applies to both types of approximation and competitive ratios.

Previous Work In [10], Dósa and He studied four variants of bin packing with rejection. Specifically, these were the offline and the online variants of bin packing with rejection, studied with respect to the absolute measure and the asymptotic measure. For the offline problem, the approximation ratios of the algorithms shown in [10] are 2 and $\frac{3}{2}$, where the latter applies only to the asymptotic measure. Moreover, it is mentioned that unless $\mathbf{P} = \mathbf{NP}$, no algorithm can have absolute approximation ratio of less than $\frac{3}{2}$, due to a simple reduction from the PARTITION problem (see problem SP12 in [13]). Note that this reduction holds already for standard bin packing.

For the online problem, they design an algorithm of an absolute competitive ratio of $\frac{1+\sqrt{5}}{2} \approx 2.618$ and an algorithm of asymptotic competitive ratio $1.75 + \varepsilon$. They show a lower bound of 2.343 on the absolute competitive ratio of any algorithm for the first online variant, and mention that the lower bound of 1.5401 for the standard online bin packing problem, due to van Vliet [24] is the best lower bound known for the second variant.

As the standard bin packing problem is a special case of the problem with rejection, we next compare the above results with these known for the standard bin packing problem. The offline bin packing problem admits an APTAS (Asymptotic Polynomial Time Approximation Scheme), as was shown by Fernandez de la Vega and Lueker [8]. This scheme returns for every given value $\varepsilon > 0$ an algorithm with an asymptotic approximation ratio of $1 + \varepsilon$. The algorithm has polynomial running time if ε is seen as a constant. Karmarkar and Karp [18] designed an AFPTAS (Asymptotic Fully Polynomial Time Approximation Scheme) for the problem. They use a similar (but much more complex) algorithm, to achieve a running time which also depends on $\frac{1}{\varepsilon}$ polynomially. A powerful tool, which is used in the APTAS of [8], is linear grouping. The main idea of this method is to round the sizes of items into a small number of distinct sizes. Unlike rounding methods for scheduling [14], the output sizes resulting from the rounding must be representatives of real item sizes in the input. This method of rounding in [14], which rounds values from a given range to a closest number among a fixed sized set of values, such that no original value changes as a result by more than a small factor, is used in our paper for the rounding of rejection penalties. Both methods are typically used for sizes that are large enough, where small sizes are separated from the large ones and are treated separately. The resulting set of large items has a small number of sizes, which is a function of ε. Then, valid assignment configurations are defined for this set of sizes, and a solution to a packing problem can be described as a set of bins packed according to specific configurations.

As stated above, the absolute approximation ratio of an offline algorithm cannot be expected to be smaller than $\frac{3}{2}$. Several algorithms are known to achieve this bound. Specifically, the simple FIRST FIT DECREASING (FFD) algorithm, which sorts the items according to non-increasing size, and applies FIRST FIT (each item is packed to the earliest bin where it fits), is one of these algorithms. This result is implied by bounds on the performance of FFD, which are given e.g. by [9, 26] and also proved directly using a simple proof in [22]. Several other algorithms with the same approximation ratio are known (see e.g. [28]).

As for the online problem, the currently best known upper bound on the asymptotic competitive ratio is 1.58889 due to Seiden [21]. The online problem has been extensively studied with respect to the asymptotic competitive ratio. Previous results include the following sequence of improvements. The online bin packing problem was first investigated by Ullman [23]. He showed that the FIRST FIT algorithm has an asymptotic competitive ratio of $\frac{17}{10}$. This result was then published in [17]. Johnson [16] showed that the NEXT FIT algorithm has an absolute competitive ratio of 2. Yao [25] designed an algorithm called REVISED FIRST FIT and showed that it has an asymptotic competitive ratio of $\frac{5}{3}$.

An important version of online bin packing (which is not studied in [10]) is the bounded space model. Bounded space algorithms can only have a constant number of bins available to accept items at any point during processing. The available bins are also called "open bins". The bounded space assumption is a quite natural one. Essentially the bounded space restriction guarantees that output of packed bins is steady, and that the packer does not accumulate an enormous backlog of bins which are only output at the end of processing. The algorithm NEXT FIT is bounded space since it uses a single open bin at every time.

Lee and Lee [19] developed a bounded space algorithm called the HARMONIC algorithm. This algorithm is actually a sequence of algorithms. Each such algorithm performs an online partition of the input into classes, where each class contains items of similar sizes and it is packed independently of the other classes. The depth of the partition into classes depends on the exact algorithm in the sequence. In particular, items larger than $\frac{1}{2}$ are separated from the other input items, and are packed into dedicated bins, one such item per bin. Specifically, the m-th HARMONIC algorithm (for $m > 1$) partitions the items into m classes and uses bounded space of at most $m - 1$ open bins. For any $\varepsilon > 0$, there is a number m such that the HARMONIC algorithm that uses m classes has a competitive ratio of at most $(1 + \varepsilon)\Pi_\infty$ [19], where $\Pi_\infty \approx 1.69103$ is the sum of series (see Sect. 3.2).

Lee and Lee [19] adapted HARMONIC into the REFINED HARMONIC algorithm, for which they showed an asymptotic competitive ratio of $\frac{273}{228} < 1.63597$. The next improvements were MODIFIED HARMONIC and MODIFIED HARMONIC 2. Ramanan, Brown, Lee and Lee showed that the first algorithm has an asymptotic competitive ratio of at most $\frac{538}{333} < 1.61562$ and claimed that the second algorithm has an asymptotic competitive ratio of at most $\frac{239091}{148304} < 1.61217$ [20]. The main idea of the improved algorithms is to allow combination of the items of size in $(\frac{1}{2}, 1]$ with smaller items.

Lee and Lee [19] showed there is no bounded space algorithm with a performance ratio below Π_∞. Note that the algorithms mentioned above REFINED HARMONIC, MODIFIED HARMONIC and MODIFIED HARMONIC 2 are all unbounded space adaptations of HARMONIC. The algorithm of Dósa and He [10], which has an asymptotic competitive ratio which tends to 1.75, does not use bounded space, as it uses FIRST FIT as a sub-routine. However, it is achieved by a sequence of algorithms, whose sequence of asymptotic competitive ratios tends to 1.75 from above.

There is much less study of the absolute competitive ratio, and the existent study focuses on the performance of simple algorithms such as NEXT FIT (which is discussed above) and FIRST FIT. Simchi-Levi [22] proved an upper bound of 1.75 on the absolute competitive ratio of FIRST FIT. A lower bound of $\frac{5}{3}$ on the absolute competitive ratio of any algorithm was given by Zhang [27].

There has been a fair amount of research on variants of well known problems, where a notion of rejection is introduced. Such studies include research on variants of various important scheduling problems (see [1, 2, 11, 15]). Since scheduling is strongly related to bin packing, this gives another motivation to the study of the bin packing problem with rejection.

Our Results We first study the offline problem. We design an APTAS for bin packing with rejection which uses techniques from [8] but also from [14] and [4]. For a given value of ε, the APTAS has cost of at most $(1+\varepsilon)\text{OPT}+1$. The APTAS uses linear programming for the packing of items that we call *small*. Next, we design an algorithm with absolute approximation ratio $\frac{3}{2}$. To do that, we apply the APTAS using a constant value of ε, and combine this with a different approach and additional arguments for cases where the value OPT is small. We use linear programs for these cases as well. Note that here the costs do not always take integer values unlike in standard bin packing. Our $(1+\varepsilon)$-approximation (in the asymptotic case) and $\frac{3}{2}$-approximation (in the absolute case) improve the previous results of [10] for the two measures which are $\frac{3}{2}$ and 2 respectively.

We continue with a study of the online problem. To be able to prove upper bounds for online algorithms, we generalize the notion of weighting [21, 23] to algorithms which allow rejection. We establish the best asymptotic competitive ratio for bounded space algorithms, and show it is the same as for the problem without rejection. For this, we adapt the HARMONIC algorithm of Lee and Lee [19] to be able to handle the notion of rejection. We show that the adapted algorithms still have the same asymptotic competitive ratios, and thus, achieve the best possible performance. Finally we show an improved unbounded space algorithm which is a modification of MODIFIED HARMONIC which can handle rejections. Both our algorithms, the rejective variants of HARMONIC and MODIFIED HARMONIC, achieve better asymptotic competitive ratios than the algorithm of [10]. Their ratios are approximately 1.69103 and 1.61562, whereas the algorithm of [10] has a competitive ratio $1.75+\varepsilon$.

Tables 1 and 2 summarize the known results and the new results of this paper, for offline and online bin packing with rejection, with comparison to offline and online standard bin packing.

Table 1 Summary of results for online algorithms

Bin packing variant	Asymptotic performance	Absolute performance
Standard	Upper bound: 1.58889 [21] Lower bound: 1.5401 [24] Bounded space algorithms: 1.691 [19]	Upper bound: 1.75 [22] Lower bound: $\frac{5}{3}$ [27]
With rejection (previous work)	Upper bound: $1.75 + \varepsilon$ [10] Lower bound: 1.5401 [24]	Upper bound: $\frac{1+\sqrt{5}}{2} \approx 2.618$ [10] Lower bound: 2.343 [10]
With rejection (this paper)	Upper bound: 1.61562 Bounded space algorithms: 1.691	

Table 2 Summary of results for offline algorithms

Bin packing variant	Asymptotic performance	Absolute performance
Standard	APTAS [8] AFPTAS [18]	Upper bound: $\frac{3}{2}$ [22] Lower bound: $\frac{3}{2}$ [13]
With rejection (previous work)	Upper bound: 1.5 [10]	Upper bound: 2 [10] Lower bound: $\frac{3}{2}$ [13]
With rejection (this paper)	APTAS	Upper bound: $\frac{3}{2}$

2 Offline Bin Packing with Rejection

2.1 An APTAS

To design an APTAS, we use methods similar to the well known APTAS for the classical bin packing problem, given by Fernandez de la Vega and Lueker [8]. The adaptation that we design here has some similarities with [4], however there are many differences due to the different natures of the problems. In order to be able to deal with rejection costs, we also use rounding methods which are similar to ones used for scheduling, as in [14]. These methods allow to reduce the number of large enough rejection penalties into a constant number. The structure of the scheme consists of a partition of the items into large items and small items. Rounding of the large items into a constant number of sizes is done using two steps, where the second step removes a small fraction of large items from the instance. After the rounding is completed, all possible packings of the large items are created. For every such packing, small items are combined into the packing, using some number of additional bins that are used for small items only. In order to combine the small items, linear programming is invoked. The solution of a linear program is fractional, and therefore it needs to be modified into a valid solution. Clearly, the sizes of large items are modified into their original sizes. Some large items are removed from the input in the process of rounding. These items are packed into new bins. We show that the rounding steps, together with the additional bins that handle the small items packed fractionally, and the large items that were removed in the second rounding step, harm the solution by a small enough factor.

We assume that without loss of generality, each rejection cost r_i satisfies $r_i \le 1$. We can make this assumption since an item of rejection cost at least 1, which is rejected in some solution, can be placed in a bin of its own instead, and the cost of the solution does not increase. We also assume $\mathrm{OPT} \ge 1$. In order to be able to assume this, note that if $\mathrm{OPT} < 1$ this means that all jobs are rejected, since any solution which uses at least one bin has cost of at least 1. Therefore, we can compute the sum of all rejection costs. If this sum is smaller than 1, then we output this solution, and otherwise, we run the APTAS. We can always check the solution which rejects all jobs. We give it as the output, if it turns out to be better than the result of the APTAS. This will also be useful to get a better approximation for small values of OPT which is done later, in Sect. 2.2.

As in [8], a first partition is done into “large items” and “small items”. Let δ be a function of ε defined later. We require of δ to be an inverse of an integer. An item j is considered to be large if both $r_j > \delta$ and $p_j > \delta$. All other items are small. We denote the multiset of large items by L and the multiset of small items by M. We have $I = L \cup M$.

The first step is to construct a set of possible packings of the large items. For each such packing of large items only, we add the other items in a near optimal way. The number of packings of large items would be polynomially bounded, yet, packings are enumerated in a way that at least one packing, which is close enough for our purposes to an optimal packing (restricted to large items only), is tested.

Let N be the number of large items in the input ($N = |L|$). If the number of large items is relatively small, that is $N < \frac{1}{\delta^4}$, we simply enumerate all possible solutions for these large items (these are partial packings of the large items where the unpacked items are rejected) into at most N bins. Since a packing contains at most N bins, and each item can be either placed into one of these bins or rejected, there are at most $(N+1)^N \le (\frac{1}{\delta^4})^{\frac{1}{\delta^4}}$ possible packings. Note that in this process a set of bins is opened, where some of them possibly remain empty. At this time, the set of empty bins is removed from the packing. We later allow the usage of empty bins and test all possible numbers of empty bins. These additional bins are created in order to accommodate small items.

For the case where $N \ge \frac{1}{\delta^4}$, we perform a rounding of the rejection costs of all items in L. This done with the goal of reducing the number of different types of large items into a constant number. We will first reduce the number of possible rejection penalties, and later we reduce the number of possible sizes. We define intervals $(\delta + i\delta^2, \delta + (i+1)\delta^2]$ for $i = 0, \ldots, \Delta = \frac{1}{\delta^2} - \frac{1}{\delta} - 1$. For every item $j \in L$, we define r'_j to be the left endpoint of the interval to which r_j belongs (i.e., it is the value r_j, rounded down to the closest value $\delta + i\delta^2$). Let I' be the adapted input. Let $A(I')$ be the cost of a solution of an algorithm A for the rounded input, and let $A'(I)$ be the cost of the same solution on the original items. The only change applied on the input is rounding of the rejection penalties of large items. Therefore, we can show the following.

Lemma 1 $A'(I) \le (1+\delta)A(I')$ *and* $\mathrm{OPT}(I') \le \mathrm{OPT}(I)$.

Proof To show $\mathrm{OPT}(I') \le \mathrm{OPT}(I)$ we note that given a solution to I, we convert it into a solution to I' by replacing the rejection costs by the rounded ones, and the cost can only decrease.

To show $A'(I) \le (1+\delta)A(I')$, note that each rejection cost decreases by at most an additive factor of δ^2 in the rounding procedure. However, since all rejection costs that are modified are larger than δ (which holds for both the rounded and the original costs), each rejection cost increases by a factor of at most $1+\delta$ when the rounded rejection costs are replaced by the original ones. □

For $0 \le i \le \Delta$, let N_i be the number of items with rounded rejection cost $\delta + i\delta^2$, and let $a_{i,1} \ge \cdots \ge a_{i,N_i}$ be (the sizes of) these items. Note that $N = \sum_{i=0}^{\Delta} N_i$.

We can consider only the sizes of items for each i, since they all have the same rejection cost $\delta + i\delta^2$. Therefore, in this case we can identify between items and their sizes. For a given $0 \le i \le \Delta$, denote the multiset of item sizes by B_i. The next step would be to reduce the number of different sizes of items with rejection cost $\delta + i\delta^2$. We use linear grouping on each such set separately. This rounding allows to reduce the number of sizes of large items into a constant. The rounding is performed in a way that the change in the cost of an optimal solution is small enough, and for every rounded size, there exists an item in the original input that has this size.

Formally, we perform a linear grouping on each one of the multisets of large items $B_i = \{a_{i,1}, \dots, a_{i,N_i}\}$. Let $m = \frac{1}{\delta^2}$. We partition the sorted set of large items into m consecutive sequences $S_{i,j}$ ($j = 1, \dots, m$) of $k_i = \lceil \frac{N_i}{m} \rceil = \lceil N_i \delta^2 \rceil$ items each (to make the last sequence be of the same cardinality, we define $a_{i,t} = 0$ for $t > N_i$). I.e., $S_{i,j} = \{a_{i,(j-1)k_i+1}, \dots, a_{i,(j-1)k_i+k_i}\}$ for $j = 1, 2, \dots, m$. For $j \ge 2$, we define a modified sequence $\hat{S}_{i,j}$ which is based on the sequence $S_{i,j}$ as follows. $\hat{S}_{i,j}$ is a multiset which contains exactly k_i items of size $a_{i,(j-1)k_i+1}$, i.e., all items are rounded up to the size of the largest element of $S_{i,j}$. The set $S_{i,1}$ is not rounded and therefore $\hat{S}_{i,1} = S_{i,1}$. Let L'_i be the union of all multisets $\hat{S}_{i,j}$ ($L'_i = \bigcup_{j=1}^{m} \hat{S}_{i,j}$) and $L' = \bigcup_{i=0}^{\Delta} L'_i$ and let $L''_i = \bigcup_{j=2}^{m} \hat{S}_{i,j}$, $L'' = \bigcup_{i=0}^{\Delta} L''_i$.

We find solutions for the two sets $L_1 = \bigcup_{i=0}^{\Delta} S_{i,1} = L' \setminus L''$ and L'' separately. The items of L_1 are packed each in a separate bin. The input L'' is treated as follows. This input contains at most $T = (m-1)(\Delta+1) < \frac{1}{\delta^4}$ different type of items (where two items are of the same type if they are of the same rounded size and have the same rounded rejection cost).

We enumerate all possible packings of the L'' items into i bins, where $0 \le i \le N$. The input L'' contains at most T distinct sizes of elements. We are interested in computing all solutions of a bin packing instance with a constant number of distinct large types. Let $(b_1, \rho_1), \dots, (b_T, \rho_T)$ be the set of types, where $\delta < b_j \le 1$ is the size of items of type (b_j, ρ_j) and $\delta \le \rho_j \le 1$ is its (rounded) rejection cost. We represent a multiset of items by a vector $J = (u_1, \dots, u_T)$, where u_j is the number of items of type (b_j, ρ_j). Let $\hat{N} = (n_1, \dots, n_T)$ denote an input. A *pattern* is a vector of non-negative integers such that the multiset of items represented by it can fit in a single bin, i.e. q is a pattern if $\sum_{j=1}^{T} q_j b_j \le 1$. Let Q be the set of all patterns. A packing can be described by specifying for every $q \in Q$, the number of bins y_q that are packed using pattern q.

As noted above, we remove empty bins from the packing, therefore an empty pattern (for which $q_i = 0$ for $1 \le i \le T$), may be considered to be a legal pattern, but is useless. The difference between n_j and the number of items of type (b_j, ρ_j) that are packed in the packing are rejected items.

We now argue that $|Q| \le (T+1)^{\frac{1}{\delta}}$. A bin can contain at most $\frac{1}{\delta}$ items. To show the bound, we can represent each bin by a list of length $\frac{1}{\delta}$. In this list we first provide a complete enumeration of all items of this bin, if any slots remain empty, we fill them with "null". There are $T+1$ options for each item in the list, since an item can be absent as well as of any size among the T possible sizes. This gives an upper bound of $(T+1)^{\frac{1}{\delta}}$ on the number of patterns $|Q|$.

A vector $y \in \mathcal{N}_0{}^Q$ specifies a valid packing of an input $\hat{N}$ into ℓ bins if and only if the following constraints hold.

$$\sum_{q \in Q} y_q = \ell, \quad \text{and} \quad \text{for all } 1 \le j \le T, \quad \sum_{q \in Q} q_j y_q \le n_j. \tag{1}$$

Since for each $1 \le j \le T$, there are $n_j - \sum_{q \in Q} q_j y_q$ items of this type which remain unpacked. The rejection cost of each of them is ρ_j and thus the cost of the entire packing including rejection costs of rejected items is $\ell + \sum_{j=1}^{T} \rho_j (n_j - \sum_{q \in Q} q_j y_q)$.

Since $\ell \le N$, we are only interested in vectors y where each component is in the set $\{0, \dots, N\}$. Thus, the number of vectors y to be enumerated is polynomially bounded.

For every packing, constructed for large items, we do the following. Consider all non-empty bins packed with large items. If the packing was created for the original items (in the case where N is small), the packing is not changed.

Otherwise, keep the bins of L_1 items unchanged. Note that a vector y defines a packing of the L'' items completely, these are linearly grouped items, and not the input items. After the process of packing is completed, including the packing of small items that are packed in the next step, we can replace the items of $\hat{S}_{i,j}$ in the packing by items of $S_{i,j}$. Clearly, the items of $S_{i,j}$ are never larger than the items of $\hat{S}_{i,j}$, and so the resulting packing is feasible.

Let ℓ be the number of bins in the packing. Since the final packing cannot contain more than n non-empty bins, we perform the following for all the following values of d, $d = \ell, \dots, n$. Thus, d will be the number of used bins in the resulting packing. For each bin, which is already packed with some large items, compute the empty space in it (that is 1 minus the sum of sizes of all items assigned to it). Denote the empty spaces in bins $z = 1, \dots, d$ by x_z. We define $x_z = 1$ for $\ell < z \le d$. The next goal is to fill the gaps in the bins that are packed with large items using small items, possibly using additional bins. We use a linear program to assign the small items fractionally into the gaps. A basic solution to the linear program will allow to pack most of the items integrally. The small items that remain packed fractionally will be removed and packed into separate bins (if their size is small) or rejected (if their rejection penalty is small).

To assign the small items (all items of M), construct the following integer program. Let $n' = n - N$ be the number of small items, and $\{(c_1, r_1), \ldots, (c_{n'}, r_{n'})\}$ be pairs of sizes and rejection costs of these items. For $1 \le z \le d + 1$ and $1 \le j \le n'$, let $X_{j,z}$ be an indicator variable. If $z \le d$, the value of $X_{j,z}$ is 1 if item j is assigned to bin z and 0 otherwise. If $z = d + 1$ the value $X_{j,z}$ is 1 if item j is rejected and 0 otherwise.

We apply the upper bounds on sum of sizes of items in the bins as follows. For each $1 \le z \le d$, $\sum_{j=1}^{n'} c_j \cdot X_{j,z} \le x_z$. We clearly have $\sum_{z=1}^{d+1} X_{j,z} \ge 1$ for all $1 \le j \le n'$, since each item must be either assigned to at least one bin or rejected. If it is assigned to more than one bin, one of its occurences can be removed without violating the other constraints. If it is both assigned and rejected, it is again removed from any bin it is assigned to.

The linear goal function is to minimize the expression $\sum_{j=1}^{n'} r_j \cdot X_{j,d+1}$. This is the sum of rejected items, and since the number of used bins is d, the cost of an algorithm is d plus the sum of rejection costs.

We relax the integrality constraint, and replace it with $X_{j,z} \ge 0$. We are left with a linear program which clearly has a solution if the original integer program does. Solving the linear program we can find a basic solution. This basic solution has at most $d + n'$ non-zero variables (as the number of constraints). Clearly, each item j has at least one non-zero variable $X_{j,z}$ and thus we get that the number of items that are not assigned completely to a bin or completely rejected (i.e., that have more than one non-zero variable associated with them) is at most d. These items are not assigned according to the solution found by the linear program. Since these items are small, for each item, either the rejection cost is at most δ, or the size it at most δ (or both). Therefore, out of the (at most) d items we still need to assign, we reject all items with rejection cost of at most δ, and pack the other items into bins, so that each bin packed in this way (possibly except for the last one) contains exactly $\frac{1}{\delta}$ items. Out of the d small items that participate in this process, let d_1 be the number of rejected small items and $d - d_1$ the number of small items which are packed into bins.

Therefore, the additional cost for these items is at most $\delta d_1 + \lceil \delta(d - d_1) \rceil \le \delta d + 1$. As an output, it is possible to choose the solution with smallest cost out of all resulting solutions.

We next analyze the performance guarantee of the above algorithm. We make use of the following definitions and lemma.

Consider two multisets A, B, with elements which are pairs of sizes and rejection costs of items. We say that A is *dominated* by B and denote $A \le B$ if there exists an injection $f : A \to B$ with the following properties. Let $a = (p_a, r_a) \in A$, and let $f(a) = b = (p_b, r_b) \in B$, then $p_b \ge p_a$ and $r_b \ge r_a$.

Lemma 2 *If A and B are multisets such that $A \le B$, then* $\mathrm{OPT}(A) \le \mathrm{OPT}(B)$.

Proof Any packing for B can be converted into a packing for A using two steps. First, all items $b' \in B$ for which there is no element $a' \in A$ such that $f(a') = b'$ are removed from the instance. This can only decrease the cost because some bins may become empty, and for some items it is no longer necessary to pay the rejection cost. A second step replaces each other element $\hat{b} \in B$ by the element $\hat{a} \in A$ such that

$f(\hat{a}) = \hat{b}$. By replacing we mean that if $\hat{b}$ is packed in a bin, then $\hat{a}$ is inserted into its location, and if $\hat{b}$ is rejected then $\hat{a}$ is rejected. Since in this situation $p_{\hat{a}} \le p_{\hat{b}}$, the resulting packing is feasible. Since $r_{\hat{a}} \le r_{\hat{b}}$, the rejection cost cannot increase. □

We analyze the cost, and prove the following theorem. It can be seen that the running time is polynomial in the size of the input. The dependence on ε is exponential and relatively high.

Theorem 3 *Algorithm FL is an APTAS.*

Proof The algorithm returns a feasible solution (as all items are either placed in bins or rejected) in polynomial time. It remains to show the asymptotic performance guarantee of the algorithm. Let $J = L'' \cup M$. Compare this set with the set I' of the original items with rounded rejection costs. We can show $J \le I'$ as follows. Each small item of j is mapped to its occurence in I'. Each item of L'' belongs to some set $\hat{S}_{i,j}$ for $j \ge 2$. We map it to one of the items of $S_{i,j-1}$, which are never smaller than $a_{i,(j-1)k_i+1}$. Since the cardinalities of all sets $S_{i,j}$ for a given value of i are equal, such a mapping is possible. Using Lemmas 2 and 1, we can see that $\mathrm{OPT}(J) \le \mathrm{OPT}(I') \le \mathrm{OPT}(I)$. In the case where no linear grouping was performed ($N \le \frac{1}{\delta^4}$), we simply define $J = I'$.

Consider now an optimal solution for the input J and its restriction to large items only. Let d' be the number of bins used for this solution. Next, remove empty bins from the solution. Since we enumerate all possible solutions for L'' (or for L if no linear grouping is done), the resulting solution is one of these constructed by the algorithm, and thus the best solution which is given as output is no worse than the best solution computed for the given assignment of large items which is based on $\mathrm{OPT}(J)$. Among such solutions, consider the one for which d' bins are used in total (i.e., the solution where the number of bins to be used by the linear program is d').

Since the value of a fractional solution for packing the small items in no larger than the value of an integral solution, namely, no larger than $\mathrm{OPT}(J)$, we can get an upper bound on the cost of the output as follows.

As written above, replacing the fractional solution by an integral one results in an additional cost of at most $\delta d + 1$. In $\mathrm{OPT}(J)$, there are d' used bins and thus $\mathrm{OPT}(J) \ge d'$. Thus the additional cost is at most $\delta\mathrm{OPT}(J) + 1 \le \delta\mathrm{OPT}(I) + 1$.

If linear grouping is done, then the cost of packing L_1 is at most $\sum_{i=0}^{\Delta} k_i = \sum_{i=0}^{\Delta} \lceil \delta^2 N_i \rceil \le N\delta^2 + (\Delta + 1) \le N\delta^2 + \frac{1}{\delta^2}$. Since $N \ge \frac{1}{\delta^4}$ in this case, and there are N items whose size and rejection cost are at least δ, we have $\mathrm{OPT}(I) \ge \delta N \ge \frac{1}{\delta^3}$. We get that the additional cost here is at most $2\delta\mathrm{OPT}(I)$.

We can now complete the analysis for both cases (large or small N). From Lemma 1 we know that the cost of changing the rejection cost of rejected items to the real rejection costs may increase the cost of a solution by a factor of at most $1 + \delta$. Thus the total cost of our solution is at most (using $\delta \le 1$) $(1 + \delta)((1 + 3\delta)\mathrm{OPT}(I) + 1) \le \mathrm{OPT}(I)(1 + 7\delta) + 1 + \delta \le \mathrm{OPT}(I)(1 + 8\delta) + 1$, since $\mathrm{OPT}(I) \ge 1$. Taking $\delta \le \frac{\varepsilon}{8}$ gives us the desired approximation. We get a solution of cost at most $(1 + \varepsilon)\mathrm{OPT} + 1$, using an algorithm whose running time is polynomial in n. □

2.2 An Algorithm with an Absolute Approximation Ratio of $\frac{3}{2}$

As mentioned earlier, the bin packing problem with rejection, if analyzed by the absolute approximation ratio, cannot have an approximation algorithm with approximation ratio smaller than $\frac{3}{2}$ (unless $\mathbf{P} = \mathbf{NP}$). In the sequel, we design an algorithm with this (probably best possible) absolute approximation ratio. The method we use is this section is choosing the best solution out of solutions created by several methods. One of these methods would be an application of the APTAS with a constant value of ε, and the other are simple heuristics that perform well in the case where OPT is small.

Consider first the two cases $\mathrm{OPT} < 1$ and $\mathrm{OPT} \geq 2.25$. We design algorithms that perform well for these cases and include the outputs of these algorithms in the set of solutions out of which we choose one with smallest cost.

In the first case we showed that we can get an optimal algorithm, if one of the possible solutions we check is the one which rejects all items. For the second case, if we apply the APTAS with $\varepsilon = \frac{1}{20}$ we get a solution of cost at most $\frac{21}{20}\mathrm{OPT} + 1 \leq \mathrm{OPT}(\frac{21}{20} + \frac{4}{9}) < \frac{3}{2}\mathrm{OPT}$.

Therefore, we need to design algorithms which perform better in the case $1 \leq \mathrm{OPT} < 2.25$. In this case, OPT uses at most two bins. The solution where OPT uses zero bins in already obtained by the simple solution which rejects all items. We are thus left with the case of one or two bins. The sum of rejection costs is therefore less than 1.25 if one bin is used, and less than 0.25 if two bins are used.

Consider first the case where OPT has a single bin. Consider the set Y of all items whose rejection costs are larger than $\frac{1}{2}$. Clearly, in a solution we are interested in, at most two jobs of Y can be rejected. Therefore, we can enumerate all possible subsets of at most two jobs from Y in polynomial time ($O(n^2)$). For each such subset, we create an optimal fractional solution as is done above. Given such a subset X such that $|X| \leq 2$, in order to create a feasible solution as we require, the complement set $Y \setminus X$ must be packed completely into a single bin. We check whether so far the solution is feasible (i.e., the items in $Y \setminus X$ indeed fit into one bin), if so, and given this partial solution, we use a linear program as described above to assign all other jobs (either into the single bin, or to be rejected). We get a solution where at most one item is split between rejection and packing into the bin. We reject this item getting an additional cost of at most $\frac{1}{2}$.

If $\mathrm{OPT} < 2.25$, and OPT uses a single bin, then there exists a choice of X for which the cost of the fractional packing is in the interval $[1, 2.25)$. Thus the cost of the solution we get is at most $\mathrm{OPT} + \frac{1}{2} \leq \frac{3}{2}\mathrm{OPT}$.

Next, consider the case where OPT has two bins. In this case $\mathrm{OPT} \geq 2$. An optimal solution contains at most two items of size in $(\frac{1}{2}, 1]$ which are not rejected. We enumerate all possible subsets of at most two such items, again in polynomial time. For every choice, the remaining items of size in this interval are rejected. The items of size in $(\frac{1}{2}, 1]$ that are chosen to be packed are assigned to bins. The packing of all other items are done by a linear program similar to the above, while all remaining items are either added to one of the two bins, or rejected. In our case, $d = 2$. We use the linear program, and get a solution where at most two items are split in some way. Each one of these (at most two) items is of size at most $\frac{1}{2}$. Thus one new bin is sufficient to accommodate these items. If $\mathrm{OPT} < 2.25$ then the cost of the fractional packing is in the interval $[2, 2.25)$. Thus the cost of the solution we get is again at most $\mathrm{OPT} + 1 \leq \frac{3}{2}\mathrm{OPT}$.

We proved the following theorem.

Theorem 4 *There exists a polynomial offline approximation algorithm, whose absolute approximation ratio is* $\frac{3}{2}$.

3 Online Bin Packing with Rejection

3.1 A Method for Analyzing Online Bin Packing Algorithms with Rejection

In this section we develop a framework which is useful for analyzing bin packing algorithms with rejection. It is possible to apply the method both to offline and online algorithms, however, in this paper we only use it for online algorithms. The method is based on weighting, and is similar to the method used already by Ullman [23] (see also [19, 21]). We describe the basic method briefly as our method generalizes it.

The essence of this method is to assign weights to items. The weights must be assigned so that the cost of the algorithm, i.e., the number of used bins, is roughly the sum of weights. A small deviation is allowed when dealing with the asymptotic competitive ratio (or the asymptotic approximation ratio), since an additive constant does not degrade the performance of an algorithm. As the next step, the problem of upper bounding the asymptotic competitive ratio is reduced into that of finding the supremum sum of weights of items which can fit into a single bin. In some cases, a constant number k of distinct weighting functions is defined to handle several major behaviors of the algorithm (resulting from specific types of inputs). For each outcome of the algorithm, at least one of the k weighting functions needs to have the above property regarding the cost of the algorithm. In this case, an upper bound on the competitive ratio is the maximum between the k supremum sums of weights in a single bin (for the k weight functions). The method in [21] is more complex and generalizes the above method.

Surprisingly, the method can be generalized to deal with weights which are not related only to the cost of packing items (i.e., to numbers of bins) but to rejection costs as well. We assign weights to items so that the weight of a rejected item is related to its rejection penalty. An accepted item receives a weight that is related to the fraction of a bin that it occupies.

Let $\mathcal{A}$ be an online algorithm and let $\mathcal{C}$ be a desired competitive ratio. Let $w^1, \dots, w^k$ be a set of functions $w^i : (0,1] \to \mathbb{R}_0^+$ (where $\mathbb{R}_0^+$ denotes the set of non-negative real numbers). For an item j, we denote its weight with respect to the weight function w^i by w^i_j.

Theorem 5 *A value $\mathcal{C}$ is an upper bound on the asymptotic competitive ratio of algorithm $\mathcal{A}$ if the following conditions hold.*

- *For every item j, and for every weight function w^i, we have that $w^i_j \leq \mathcal{C} r_j$, that is, for every weight function, the weight assigned to each item is no larger than $\mathcal{C}$ times its rejection cost.*

- *There exists a constant μ, such that for every input, there exists a value $1 \le i \le k$ such that $\mathcal{A} \le \sum_{j=1}^{n} w_j^i + \mu$.*
- *For every set of items J such that $\sum_{j \in J} p_j \le 1$, and every $1 \le i \le k$, we have $\sum_{j \in J} w_j^i \le \mathcal{C}$.*

In order to prove the theorem, we consider the cost of the algorithm that immediately translates into the sum of weights according to one of the weight functions. Then we consider an optimal solution OPT, and partition the items into two sets according to the behavior of OPT. Then, we can use the relation between the weights and the cost of OPT.

Proof Given an input I of n items, let i and μ be such that $\mathcal{A} \le \sum_{j=1}^{n} w_j^i + \mu$.

We let OPT_{rej} be the set of items rejected by OPT and let OPT_{acc} be the set of items accepted by OPT. Furthermore, let OPT_{nacc} be the *number of bins* used by OPT for the items it accepted. We denote by $\mathrm{OPT}_{acc}(i)$, the set of items assigned by OPT to its i-th bin, for $1 \le i \le \mathrm{OPT}_{nacc}$.

$$\begin{aligned}
\mathcal{A} &\le \sum_{j=1}^{n} w_j^i + \mu \le \Bigg(\sum_{j \in \mathrm{OPT}_{rej}} w_j^k \Bigg) + \Bigg(\sum_{j \in \mathrm{OPT}_{acc}} w_j^k \Bigg) + \mu \\
&\le \Bigg(\sum_{j \in \mathrm{OPT}_{rej}} \mathcal{C} \cdot r_j \Bigg) + \Bigg(\sum_{j \in \mathrm{OPT}_{acc}} w_j^k \Bigg) + \mu \\
&= \mathcal{C} \cdot \sum_{j \in \mathrm{OPT}_{rej}} r_j + \sum_{i=1}^{\mathrm{OPT}_{nacc}} \Bigg(\sum_{j \in \mathrm{OPT}_{acc(i)}} w_j^k \Bigg) + \mu \\
&\le \mathcal{C} \cdot \sum_{j \in \mathrm{OPT}_{rej}} r_j + \sum_{i=1}^{\mathrm{OPT}_{nacc}} \mathcal{C} + \mu \\
&= \mathcal{C} \cdot \sum_{j \in \mathrm{OPT}_{rej}} r_j + \mathcal{C}\mathrm{OPT}_{nacc} + \mu.
\end{aligned}$$

Consider now the solution of OPT. We have $\mathrm{OPT} = \sum_{j \in \mathrm{OPT}_{rej}} r_j + \mathrm{OPT}_{nacc}$. Therefore, $\mathcal{C}$ is an upper bound on the asymptotic competitive ratio of the algorithm. □

To summarize, the technique used here is therefore as follows. We assign weights to items, so that the cost of an algorithm is roughly the sum of weights. The weights of items cannot be much larger than their rejection costs. Then, we reduce the problem into that of finding the maximum sum of weights of items in a single bin.

3.2 Algorithm Rejective Harmonic

We now define our adaptation of the Harmonic$_k$ algorithm of Lee and Lee [19]. The algorithm is called Rejective Harmonic$_k$ (RejH$_k$). The fundamental idea of "harmonic-based" algorithms is to first classify items by size, and then pack an item according to its class (as opposed to letting the exact size influence packing decisions). We use a similar classification, but after classification is applied, we further use a decision rule (based on a threshold) to identify whether the item should be packed or rejected.

For the classification of items, we partition the interval (0, 1] into sub-intervals. We use $k-1$ sub-intervals of the form $I_i = (\frac{1}{i+1}, \frac{1}{i}]$ for $i = 1, \dots, k-1$ and one final sub-interval $I_k = (0, \frac{1}{k}]$. An interval I_i is also called interval i. Each packed bin will contain only items from one sub-interval. Items in sub-interval i that are not rejected, are packed i to a bin for $i = 1, \dots, k-1$ (except for possibly the very last bin dedicated to this interval). The items in interval k that are not rejected are packed using the greedy algorithm Next Fit. This algorithm keeps a single open bin and packs items of interval k that are not rejected to this bin until some item does not fit. Then a new bin is opened for interval k, and the previous bin is never used again. For $1 \le i \le k-1$, a bin which received the full amount of items (according to its type) is closed, therefore a total of at most $k-1$ bins are open or active simultaneously (one per interval, except for $(\frac{1}{2}, 1]$ which does not need an active bin).

We next define the thresholds for acceptance or rejection of a new item. Given an item $a \in I$, such that $p_a \in I_i$, let $s_a = i$. If $s_a < k$, item a is rejected if $r_a \le \frac{1}{s_a}$, and otherwise a is accepted and packed according to the algorithm above. If $s_a = k$, item a is rejected if $r_a \le \frac{k}{k-1} p_a$, and otherwise a is accepted and packed according to the algorithm above.

As a first step of analyzing the algorithm, we assign weights to items. We will use the method introduced in the previous section for the analysis. The assignment is similar to the proof of [19], however, unlike the proof in [19], the weights we define here depend both on the sizes of items and on their rejection costs. We use a single weight function w, and the weight of item $a \in I$ is denoted by w_a.

In order to use the method, we need to assign the weights so that the three conditions in Theorem 5 hold. We do the assignment so that the cost of the algorithm satisfies $\mathrm{RejH}_k \le \sum_{a \in I} w_a + k - 1$.

An item a which is rejected by the algorithm gets weight r_a. An item a which is accepted gets weight $\frac{1}{s_a}$, if $s_a < k$ and $\frac{k}{k-1} p_a$, if $s_a = k$. Thus each item of sub-intervals $1, \dots, k-1$ gets weight $\min\{r_a, \frac{1}{s_a}\}$ and each item of sub-interval k gets weight $\min\{r_a, \frac{k}{k-1} p_a\}$.

For the analysis, we use the following well known sequence π_i, $i \ge 1$, which often occurs in bin packing. Let $\pi_1 = 2, \pi_{i+1} = \pi_i(\pi_i - 1) + 1$ and let $\Pi_\infty = \sum_{i=1}^{\infty} \frac{1}{\pi_i - 1} \approx 1.69103$. This sequence is presented in [19]. It is not difficult to show that $1 - \sum_{i=1}^{t} \frac{1}{\pi_i} = \frac{1}{\pi_{i+1} - 1}$. It is shown in [19] that the sequence of asymptotic competitive ratios of the algorithms Harmonic_k tends to Π_∞ as k grows, and that no bounded space algorithm can have an asymptotic competitive ratio smaller than Π_∞. We show that the generalization RejH_k has the same properties. Clearly, the lower bound for the problem with rejection follows from the lower bound on the special case without rejection.

Theorem 6 *The asymptotic competitive ratio of* RejH_k *tends to* Π_∞ *as* k *grows. No online bounded space algorithm can have a smaller asymptotic competitive ratio.*

Proof As mentioned above, the lower bound follows directly from the lower bound in [19] as standard online bin packing is a special case of online bin packing with rejection (where all rejection costs are infinite). Therefore, we focus on the proof of the upper bound.

For every k, we need to show that all conditions of Theorem 5 hold for the value $\Pi_\infty + \varepsilon_k$, where $\varepsilon_k \to 0$ as k grows. The first condition of the theorem trivially holds since $w_j \le r_j$ for every item.

We analyze an algorithm $\mathcal{A} = \mathrm{RejH}_k$. Let $\mathcal{A}_{cacc}(i)$ be the set of accepted items of sub-interval i for $1 \le i \le k$. Each bin for interval i $(1 \le i \le k-1)$ can contain exactly i items, and since we have a single open bin for this interval at any time, each such bin except for possibly the last one, contains exactly this amount. Each bin for interval k is occupied by items of total size of at least $\frac{k-1}{k}$ (except for possibly the last one), since a new bin is opened when an item (which has size at most $\frac{1}{k}$) does not fit into it. Let $\mathcal{A}_{rej}$ be the set of items rejected by $\mathcal{A}$. We get,

$$\begin{aligned}
\mathcal{A} &\le \sum_{a\in\mathcal{A}_{rej}} r_a + \sum_{i=1}^{k-1}\left\lceil \frac{|\mathcal{A}_{cacc}(i)|}{i}\right\rceil + \left\lceil \sum_{a\in\mathcal{A}_{cacc}(k)} p_a \frac{k}{k-1}\right\rceil \\
&\le \sum_{a\in\mathcal{A}_{rej}} r_a + |\mathcal{A}_{cacc}(1)| + \sum_{i=2}^{k-1}\left(\frac{|\mathcal{A}_{cacc}(i)|}{i} + 1\right) + \sum_{a\in A_{cacc}(k)} p_a \frac{k}{k-1} + 1 \\
&\le \sum_{a\in\mathcal{A}_{rej}} w_a + \sum_{i=1}^{k}\left(\sum_{a\in\mathcal{A}_{cacc}(i)} w_a\right) + (k-1) = \sum_{a\in I} w_a + k - 1.
\end{aligned}$$

Next, we need to upper bound the sum of weights of a set of items which can fit into a single bin. Since the weight of an item a never exceeds $\frac{1}{s_a}$, for an item of size $p_a = x$, $x \in (\frac{1}{s_a+1}, \frac{1}{s_a}]$ $(s_a < k)$ and does not exceed $\frac{k}{k-1}x$, if $x \le \frac{1}{k}$, we can use the result of [19], which states an upper bound which tends to Π_∞ on the sum of weights in this case. A proof of this property also appears in [7]. For completeness, we include a proof.

Let t be a maximal integer such that $\pi_t \le k$. We claim that the total weight packed in a single bin is upper bounded by $\sum_{i=1}^{t}\frac{1}{\pi_i-1} + \frac{1}{\pi_{t+1}-1}\cdot\frac{k}{k-1} = \sum_{i=1}^{t+1}\frac{1}{\pi_i-1} + \frac{1}{(\pi_{t+1}-1)\cdot(k-1)}$. Since t is maximal, we have $\pi_{t+1} > k$ and thus $\pi_{t+1}-1 \ge k$. Therefore the upper bound is at most $\Pi_\infty + \frac{1}{(k-1)^2}$. This will imply that the limit of asymptotic competitive ratios of RejH_k tends to Π_∞ or to a smaller value. We can exclude the possibility of a smaller value due to the lower bound of Π_∞ on the limit of asymptotic competitive ratios of any bounded space algorithm, and thus of RejH_k. Therefore proving the upper bound above is a sufficient condition.

Consider a sequence χ and assume by contradiction that the total weight of this sequence is larger than $\sum_{i=1}^{t}\frac{1}{\pi_i-1} + \frac{1}{\pi_{t+1}-1}\cdot\frac{k}{k-1}$. We next show that a sequence, that contains items of sizes slightly above the values $\frac{1}{\pi_i} + \varepsilon$, is a worst case sequence.

Claim 7 *For $i = 1, \ldots, t$, χ must contain an item of size in the interval $(\frac{1}{\pi_i}, \frac{1}{\pi_i - 1}]$.*

Proof We prove the claim by induction, showing at every step that an additional item a^i such that $r_{a^i} \in (\frac{1}{\pi_i}, \frac{1}{\pi_i - 1}]$ belongs to χ.

Assume that we already proved that χ contains items of sizes from intervals $(\frac{1}{\pi_v}, \frac{1}{\pi_v - 1}]$ for $v = 1, \dots, i-1$. The weights of these items are at most $\frac{1}{\pi_v - 1}$, and thus the sum of weights of the other items in the bin is strictly larger than $\sum_{v=i}^{t} \frac{1}{\pi_v - 1} + \frac{1}{\pi_{t+1} - 1} \cdot \frac{k}{k-1}$. The sum of sizes of items which are already proved to exist is strictly larger than $\sum_{v=1}^{i-1} \frac{1}{\pi_v} = 1 - \frac{1}{\pi_i - 1}$. Thus, the sum of sizes of other items is strictly smaller than $\frac{1}{\pi_i - 1}$. If an item of size in $(\frac{1}{\pi_i}, \frac{1}{\pi_i - 1}]$ exists as well, we are done with the inductive step. Otherwise, all other items are no larger than $\frac{1}{\pi_i}$. The ratio of weight to size of such items never exceeds the factor $\frac{\pi_i}{\pi_i - 1}$. Thus, the weight of all additional items is strictly smaller than $\frac{\pi_i + 1}{\pi_i} \cdot \frac{1}{\pi_i - 1} = \frac{1}{\pi_i - 1} + \frac{1}{\pi_i(\pi_i - 1)} = \frac{1}{\pi_i - 1} + \frac{1}{\pi_{i+1} - 1}$. We get that the total weight of all items is no larger than $\sum_{v=1}^{i+1} \frac{1}{\pi_v - 1}$. For every value of i, this is smaller than the sum of weights we assumed. This leads to a contradiction. □

Given the set of t items which must occur, their sum of weights is $\sum_{i=1}^{t} \frac{1}{\pi_i - 1}$. The sum of all other items is strictly less than $1 - \sum_{i=1}^{t} \frac{1}{\pi_i} = \frac{1}{\pi_{t+1} - 1}$. All these items are smaller than $\frac{1}{k}$, thus their total weight is smaller than $\frac{k}{k-1} \cdot \frac{1}{\pi_{t+1} - 1}$. We get that the total is smaller than assumed which is a contradiction.

We proved all properties and therefore by Theorem 5, we establish the competitive ratio. □

3.3 Algorithm REJECTIVE MODIFIED HARMONIC

In this section we show how to design improved algorithms which use unbounded space. As an example, we adapt one of the best algorithms known for online bin packing to allow rejection. This algorithm MODIFIED HARMONIC was introduced by Ramanan et al. [20]. We give a short description of this algorithm.

As HARMONIC, MODIFIED HARMONIC also classifies items by size, and packs items according to classes. A disadvantage of HARMONIC is in the packing of items of the sub-interval $I_1 = (\frac{1}{2}, 1]$. These items are packed one per bin, possibly wasting much space in each such bin. To avoid this large waste of space, MODIFIED HARMONIC and other later algorithms (see [21]) use two extra interval endpoints, of the form $\frac{1}{2} < \Delta < \frac{2}{3}$ and $1 - \Delta$. Then, some small items can be combined in one bin together with an item of size in $(\frac{1}{2}, \Delta]$. Items larger than Δ (i.e., in the interval later called I_1^1) are still packed one per bin as in HARMONIC. Furthermore, These algorithms use parameters α^i ($i = 2, \dots, k-1$) which represent the fraction of items of intervals $I_i = (\frac{1}{i+1}, \frac{1}{i}]$ which are supposed to be combined with an item of size in $I_1^2 = (\frac{1}{2}, \Delta]$. For $i = 2$, α^2 is the fraction of items in the interval $I_2^2 = (\frac{1}{3}, 1 - \Delta]$. This fraction of items, when they arrive, is either immediately combined with such a large item (if this large item was not combined with items of different intervals yet), or else space is reserved for the larger item. Once such a large item arrives, it is inserted into a space reserved for it. The remaining bins with items of interval I_i (or I_2^2, for $i = 2$) still contain i items per bin. Moreover, items of the interval $I_2^1 = (1-\Delta, \frac{1}{2}]$ are not combined with larger items and are packed in pairs. The items of the last interval $I_k = (0, \frac{1}{k}]$ are not combined with larger items and are packed using NEXT FIT. Note that keeping an exact fraction α^i of the items the interval I_i (or I_2^2, for $i = 2$) packed in one way, and an exact $1 - \alpha^i$ fraction packed in the other way, is impossible. Therefore, if the number of items of an interval I_i (or I_2^2, for $i = 2$) is X_i, the number of items that are to be combined with an item of size in I_1^2 is kept as at least $\lfloor \alpha^i X_i \rfloor > \alpha^i X_i - 1$, and the number of remaining items of the same interval is at least $(1-\alpha^i)X_i$ and at most $(1-\alpha^i)X_i + 1$.

MODIFIED HARMONIC (MH) is defined using four intervals of items in $(\frac{1}{3}, 1]$ as above, 35 intervals I_i for $i = 3, \ldots, 37$ and one last interval $I_{38} = (0, \frac{1}{38}]$. It uses $\Delta = \frac{419}{684} \approx 0.612573$.

$$\alpha^2 = \frac{1}{9}; \qquad \alpha^3 = \frac{1}{12}; \qquad \alpha^4 = \alpha^5 = 0;$$

$$\alpha^i = \frac{37-i}{37(i+1)}, \quad \text{for } 6 \le i \le 36, \quad \text{and} \quad \alpha^{37} = 0.$$

The results of [20] imply that the asymptotic competitive ratio of MODIFIED HARMONIC (for standard bin packing) is at most $\frac{538}{333} < 1.61562$. (In the original definition, Δ was used to denote $1 - \Delta$.) Note that for every interval I_i (or I_2^2, for $i = 2$), for which smaller items that are possible to be combined with a larger item in a bin, we compute the maximum number m_i of such items that can fit into the bin, leaving an empty space of size at least Δ. In this calculation, the maximum size of any item is taken into account. Thus we get $m_2 = m_3 = 1$, $m_6 = m_7 = 2$, $m_8 = m_9 = m_{10} = 3$, $m_{11} = m_{12} = 4$, $m_{13} = m_{14} = m_{15} = 5$, $m_{16} = m_{17} = m_{18} = 6$, $m_{19} = m_{20} = 7$, $m_{21} = m_{22} = m_{23} = 8$, $m_{24} = m_{25} = 9$, $m_{26} = m_{27} = m_{28} = 10$, $m_{29} = m_{30} = 11$, $m_{31} = m_{32} = m_{33} = 12$, $m_{34} = m_{35} = m_{36} = 13$.

In the analysis we ignore incomplete bins which did not receive the full number of items they are supposed to get. These are bins with items of size in $(0, \frac{1}{2}]$, that were not supposed to be combined with larger items, and bins with items of these sizes that are supposed to be combined with a larger item, but did not get m_i items. The number of such incomplete bins is bounded by a constant since we do not open a new bin until the previous one receives the full amount of items. However, a bin which received an item of size in I_1^2 but did not receive smaller items, or a bin which has space reserved for am item of size in I_1^2, that never arrived, cannot be ignored since the number of such bins can be arbitrary. We note however, that after removing incomplete bins, there cannot be both types of bins mentioned above, and we either need to deal with "waiting" bins with an items of size in I_1^2, or "waiting" bins with space reserved for such an item.

We define a version of MODIFIED HARMONIC which allows rejection, and call it REJECTIVE MODIFIED HARMONIC (MHR). This algorithm has a decision rule for every interval. Upon arrival of an item, it is either rejected, or assigned by MH, where items that were rejected are simply ignored by this sub-routine that runs MH.

We therefore only need to define a rejection rule for every interval. Let x be the size of an item, we consider all possible cases for the value of x, which are as follows.

- If $x \in I_1^1 = (\Delta, 1]$, x is rejected if $r_j \le 1$ and otherwise accepted.
- If $x \in I_1^2 = (\frac{1}{2}, \Delta]$, x is rejected if $r_j \le \frac{2}{3}$ and otherwise accepted.
- If $x \in I_2^1 = (1-\Delta, \frac{1}{2}]$, x is rejected if $r_j \le \frac{1}{2}$ and otherwise accepted.
- If $x \in I_2^2 = (\frac{1}{3}, 1-\Delta]$, x is rejected if $r_j \le \frac{4}{9}$ and otherwise accepted.
- If $x \in I_3 = (\frac{1}{4}, \frac{1}{3}]$, x is rejected if $r_j \le \frac{11}{36}$ and otherwise accepted.
- If $x \in I_i = (\frac{1}{i+1}, \frac{1}{i}]$, for the following values of i; $i = 4, 5, 37$, x is rejected if $r_j \le \frac{1}{i}$ and otherwise accepted.
- If $x \in I_i = (\frac{1}{i+1}, \frac{1}{i}]$ for $6 \le i \le 36$, x is rejected if $r_j \le \frac{38}{37(i+1)}$ and otherwise accepted.
- If $x \in I_{38} = (0, \frac{1}{38}]$, x is rejected if $r_j \le \frac{38 \cdot p_j}{37}$ and otherwise accepted.

We assign two sets of weights w^1 and w^2 to items as follows. The proof is similar to the proof in [20], with differences resulting from rejections.

- A rejected item has $w_j^1 = w_j^2 = r_j$.
- An accepted item j of an interval I_i for $i = 4, 5, 37$ is assigned weight $w_j^1 = w_j^2 = \frac{1}{i}$.
- An accepted item of interval I_1^1 gets weight $w_j^1 = w_j^2 = 1$.
- An accepted item of interval I_1^2 gets weight $w_j^1 = 1$, $w_j^2 = \frac{2}{3}$.
- An accepted item of interval I_2^1 gets weight $w_j^1 = w_j^2 = \frac{1}{2}$.
- An accepted item of interval I_2^2 gets weight $w_j^1 = \frac{4}{9}$, $w_j^2 = \frac{5}{9}$.
- An accepted item of interval I_3 gets weight $w_j^1 = \frac{11}{36}$, $w_j^2 = \frac{7}{18}$.
- An accepted item of interval I_i for $6 \le i \le 36$ gets weight $w_j^1 = \frac{38}{37(i+1)}$, $w_j^2 = w_j^1 + \frac{37-i}{37m_i(i+1)} = \frac{38m_i+37-i}{37m_i(i+1)}$.
- An accepted item of interval I_{38} gets weight $w_j^1 = w_j^2 = \frac{38 \cdot p_j}{37}$.

These weights are defined as in [20] except for rejected items and items in the interval I_1^2 for which we defined $w_j^2 = \frac{2}{3}$. Given a rejected item i with size p_i and rejection penalty r_i, consider an item j with $p_j = p_i$ and $r_j = \infty$. Note that $w_i^1 = w_i^2 \le \min\{w_j^1, w_j^2\}$, that is, weights and rejection rules are defined so that for a rejected item i, the weight assigned to it never exceeds the weight that an accepted item j of the same size would have received.

In order to analyze the competitive ratio and show that it is at most $\mathcal{C}_1 = \frac{538}{333} < 1.61562$ (as for the original algorithm), we show that all conditions of Theorem 5 hold. We start with the second condition. We proof this condition with $w_j^2 = 0$ in I_1^2, which may only reduce the weight. The cost of rejected items is exactly their weights, and therefore we need to prove it for packed items only.

The proof of [20] shows that the condition holds in the case where no items are rejected, and for an item j in the interval I_1^2, the second weight function is defined by $w_j^2 = 0$. We give the proof here for completeness.

For the analysis, we assume that the fraction of items for interval I_i that are to be combined with items of size in I_1^2 is exactly α^i (for $i=2$ this is the fraction of items of size in I_2^2). This may change the total weight by a constant factor.

To prove the condition, note that items of size in I_4, I_5 and I_{37} are packed i to a bin, and thus their total weight is equal to the number of bins used for them (up to a constant factor, that is caused by at most one bin for each such interval, that we neglect in the sequel). Items of size in I_{38} are packed by NEXT FIT. Each such bin (except for possibly the last one) is occupied by at least a total of $\frac{37}{38}$, and therefore, the total weight of items in it is at least 1.

If all bins that were to be combined with an item of size in I_1^2, were indeed combined, we use w^1. The cost of such bins is covered by these larger items, and we need to consider the cost of bins with other items in intervals I_2^2 and I_i for $i=3$ and $6 \le i \le 36$. An $1-\alpha^i$ fraction of the items are packed i to a bin. Thus the average cost per item (including all items of such an interval) is $\frac{1-\alpha^i}{i}$. This is exactly the definition of w^1.

If all bins that received an item of size in I_1^2 received additional items, we use w^2. The cost of such bins are covered by smaller items. An $1-\alpha^i$ fraction of the items are packed i to a bin, and an α^i fraction, m_i to a bin. Thus the average cost per item (including all items of such an interval) is $\frac{1-\alpha^i}{i} + \frac{\alpha^i}{m_i}$. This is exactly the definition of w^2.

To prove the first condition, note that for each item j, either its weight is equal to its rejection cost, or its rejection cost is at least $\min\{w_j^1, w_j^2\}$. Thus we need to show for every item that $w_j^2 \le \mathcal{C}_1 w_j^1$ and $w_j^1 \le \mathcal{C}_1 w_j^2$. We only need to consider cases in which the two weights are not the same. For an item j in the interval I_1^2 we have $\frac{w_j^2}{w_j^1} = \frac{2}{3}$. For an item j in the interval I_2^2 we have $\frac{w_j^2}{w_j^1} = 1.25$. For an item j in the interval I_3 we have $\frac{w_j^2}{w_j^1} = \frac{14}{11}$. For an item j in the interval I_i for $6 \le i \le 36$ we have $\frac{w_j^2}{w_j^1} = 1 + \frac{37-i}{38 m_i}$. This value is maximized for $i=6$, since m_i is monotonically increasing. For $i=6$ we have $m_i = 2$ and thus $\frac{w_j^2}{w_j^1} \le 1 + \frac{31}{76} < \frac{3}{2}$, and $\frac{w_j^1}{w_j^2} \le \frac{3}{2}$.

To prove the last condition of Theorem 5, we need to consider the possible contents of a bin, and the total weight of items in this bin. For each of the two weight functions w^1 and w^2, and for a given size of an item, we consider its maximum weight, which is, as explained above, the weight of an accepted item of this size. In all cases except for one case, we define the weights of accepted items exactly as the definition of weights of items in [20]. Therefore, it is possible to use the results of that paper, and analyze only cases where an item of size in I_1^2 is present, and only according to w^2. Nevertheless, we give a full proof here for completeness.

For every interval $\mathcal{I}$, where $\mathcal{I} \in \{I_1^1, I_1^2, I_2^1, I_2^2\}$, or $\mathcal{I} = I_j$ for some $3 \le j \le 38$, we define four values $\tau^i(\mathcal{I})$ (for $i=1,2$) and $\phi^i(\mathcal{I})$ (for $i=1,2$), where $\tau^i(\mathcal{I})$ denotes the supremum ratio between the weight of an item in interval $\mathcal{I}$, according to weight function w^i (for $i=1$ or $i=2$) and its size, and $\phi^i(\mathcal{I})$ denotes the supremum weight of an item in the interval. The following definition of the functions τ^1,τ^2,ϕ^1 and ϕ^2 results from the definition of weights.

- If $\mathcal{I} = I_i$ for $i \in \{4, 5, 37\}$, then $\tau^1(\mathcal{I}) = \tau^2(\mathcal{I}) = \frac{i+1}{i}$, and $\phi^1(\mathcal{I}) = \phi^2(\mathcal{I}) = \frac{1}{i}$.
- If $\mathcal{I} = I_1^1$, then $\tau^1(\mathcal{I}) = \tau^2(\mathcal{I}) = \frac{1}{\Delta} = \frac{684}{419} \approx 1.63246$, and $\phi^1(\mathcal{I}) = \phi^2(\mathcal{I}) = 1$.
- If $\mathcal{I} = I_1^2$, then $\tau^1(\mathcal{I}) = 2$, $\tau^2(\mathcal{I}) = \frac{4}{3}$, $\phi^1(\mathcal{I}) = 1$ and $\phi^2(\mathcal{I}) = \frac{2}{3}$.
- If $\mathcal{I} = I_2^1$, then $\tau^1(\mathcal{I}) = \tau^2(\mathcal{I}) = \frac{1}{2(1-\Delta)} = \frac{342}{265} \approx 1.29$, and $\phi^1(\mathcal{I}) = \phi^2(\mathcal{I}) = \frac{1}{2}$.
- If $\mathcal{I} = I_2^2$, then $\tau^1(\mathcal{I}) = \frac{4}{3}$, $\tau^2(\mathcal{I}) = \frac{5}{3}$, $\phi^1(\mathcal{I}) = \frac{4}{9}$, and $\phi^2(\mathcal{I}) = \frac{5}{9}$.
- If $\mathcal{I} = I_3$, then $\tau^1(\mathcal{I}) = \frac{11}{9}$, $\tau^2(\mathcal{I}) = \frac{14}{9}$, $\phi^1(\mathcal{I}) = \frac{11}{36}$, and $\phi^2(\mathcal{I}) = \frac{7}{18}$.
- If $\mathcal{I} = I_i$ for $6 \le i \le 36$, then $\tau^1(\mathcal{I}) = \frac{38}{37}$, $\tau^2(\mathcal{I}) = \frac{38m_i+37-i}{37m_i}$, $\phi^1(\mathcal{I}) = \frac{38}{37(i+1)}$, and $\phi^2(\mathcal{I}) = \frac{38m_i+37-i}{37m_i(i+1)}$.

  Note that for $i \ge 19$, $m_i \ge 7$, and $37 - i \le 18$, so $\tau^2(\mathcal{I}) < \frac{41}{37}$.
  For $12 \le i \le 18$, $m_i \ge 4$, and $37 - i \ge 25$, so $\tau^2(\mathcal{I}) \le \frac{177}{148}$.
  Otherwise, $m_i \ge 2$, and $37 - i \le 31$, so $\tau^2(\mathcal{I}) < \frac{54}{37} < \frac{3}{2}$.
- If $\mathcal{I} = I_{38}$, then $\tau^1(\mathcal{I}) = \tau^2(\mathcal{I}) = \frac{38}{37}$, and $\phi^1(\mathcal{I}) = \phi^2(\mathcal{I}) = \frac{1}{37}$.

We next find an upper bound on the total weight that can be packed into one bin according to each one of the two weight functions. Consider therefore a given packed bin.

If the bin contains no items of size in $(\frac{1}{2}, 1]$, since for $\tau^1(\mathcal{I}) \le \frac{3}{2}$ for all intervals $\mathcal{I} \notin \{I_1^1, I_1^2\}$, the total weight of all items in the bin, according to w^1, is no larger than $\frac{3}{2}$. As for w^2, the only intervals in which $\tau^2(\mathcal{I}) > \frac{3}{2}$ (in additional to I_1^1) are I_2^2 and I_3. Let n_1 and n_2 be the numbers of items in a bin of size in these intervals, respectively. Since $\tau^2(\mathcal{I}) \le \frac{3}{2}$ in all smaller intervals, we get a total weight of at most $n_1\frac{5}{9} + n_2\frac{7}{18} + (1 - \frac{n_1}{3} - \frac{n_2}{4})\frac{3}{2} = \frac{3}{2} + \frac{n_1}{18} + \frac{n_2}{72}$. We have $n_1 + n_2 \le 3$ and $n_1 \le 2$. The cases $n_1 = 0$ and $n_1 = 1$ give a total weight of less than 1.6. The case $n_1 = 2$ and $n_2 = 0$ gives $\frac{29}{18} < \frac{538}{333}$. In the case $n_1 = 2$ and $n_2 = 1$, the remaining space in the bin is less than $\frac{1}{12}$, for such items $\tau^2(\mathcal{I}) \le \frac{177}{148}$, so we get a total weight of less than 1.6 again.

Therefore, we need to consider a bin that contains an item of size in $(\frac{1}{2}, 1]$.

Consider the function w^1 first. All items of size in $(\frac{1}{2}, 1]$ have a weight of 1. The remaining space for additional items is therefore smaller than $\frac{1}{2}$. If all additional items are no larger than $\frac{1}{5}$, using the property $\tau^i(\mathcal{I}) \le \frac{6}{5}$ for the relevant intervals, we get a total weight of at most 1.6.

If the bin contains an item of size in I_2^1, its weight is $\frac{1}{2}$, and the remaining space is less than $\frac{77}{684}$. Using the values of τ^1, the total weight of these items is at most $\frac{77}{666}$, and the total weight is no larger than $\frac{538}{333}$.

If the bin contains an item of size in I_2^2, its weight is $\frac{4}{9}$, and the remaining space is less than $\frac{1}{6}$. Using the values of τ^1, the total weight of these items is at most $\frac{19}{111}$, and the total weight is no larger than $\frac{538}{333}$.

Otherwise, the bin may contain one or two items of size in $(\frac{1}{5}, \frac{1}{3}]$ (but at most one item of size in $(\frac{1}{4}, \frac{1}{3}]$). However, if there are no items of size in I_4, the value of τ^1 for all possible intervals is at most $\frac{11}{9}$, which gives a total weight of at most $\frac{29}{18} < \frac{538}{333}$.

Therefore, it is left to consider the three cases: two items of size in I_4, two items, where one item has size in I_4 and the other item has size in I_3, and finally, the case of one item in I_4.

The total weight of these items in the three cases are two items of sizes in I_4 and I_3 are $\frac{1}{2}$, $\frac{5}{9}$ and $\frac{1}{4}$, respectively, where the space left for other items, in these three cases is $\frac{1}{10}$, $\frac{1}{20}$, and $\frac{3}{10}$. In the first two cases, $\tau^1 = \frac{38}{37}$ for all intervals of items in $(0, \frac{1}{10}]$, so the total weights are $\frac{593}{370} < \frac{538}{333}$ and $\frac{5351}{3330} < \frac{538}{333}$. In the last case, $\tau^1 \leq \frac{6}{5}$, and we get a total weight of at most 1.61.

Next, we consider only w^2. Consider first the case that the item of size larger than $\frac{1}{2}$ has size in I_1^2. Such a bin can contain in addition only items smaller than $\frac{1}{2}$. For all such intervals, $\tau^2(\mathcal{I}) \leq \frac{5}{3}$. Thus an upper bound on the total weight in the bin with respect to w^2 is $\frac{2}{3} + \frac{1}{2} \cdot \frac{5}{3} = \frac{3}{2}$.

Next, consider the case where an item that the item of size larger than $\frac{1}{2}$ has size in I_1^1. The remaining space for other items is smaller than $1 - \Delta = \frac{265}{684} \approx 0.3874$. If all these items are no larger than $\frac{1}{4}$, then for all such intervals, $\tau^2(\mathcal{I}) \leq \frac{3}{2}$. Thus an upper bound on the total weight in the bin with respect to w^2 is $1 + \frac{3}{2} \cdot \frac{265}{684} < 1.6$.

If an item of size in I_3 is present in the bin, the remaining space for smaller items is smaller than $\frac{94}{684}$. For these items $\tau^2(\mathcal{I}) \leq \frac{3}{2}$, so we get a total weight of at most $1 + \frac{7}{18} + \frac{141}{684} = \frac{1091}{684} < 1.6$.

Otherwise, an item of size in I_2^2 and weight $\frac{5}{9}$ is present in the bin. The remaining space for other items is smaller than $\frac{37}{684}$. If all additional items are no larger than $\frac{1}{19}$, then $\tau^2(\mathcal{I}) < \frac{41}{37}$ for their intervals, and the total weight is at most $1 + \frac{5}{9} + \frac{37 \cdot 41}{684} < \frac{538}{333}$. Otherwise, an item of size in I_{18} and weight $\frac{13}{222}$ is present, and the remaining space for additional items is smaller than $\frac{1}{684}$. For such items $\tau^2 = \frac{38}{37}$, so we get a total weight of $1 + \frac{5}{9} + \frac{13}{222} + \frac{1}{666} = \frac{538}{333}$.

Since all conditions hold for $\mathcal{C}_1 = \frac{538}{333}$ we establish the following theorem.

Theorem 8 *The asymptotic competitive ratio of* REJECTIVE MODIFIED HARMONIC *is at most* $\mathcal{C}_1 = \frac{538}{333}$.

4 Concluding Remarks

In this paper, we studied offline and online bin packing with rejection.

Offline bin packing with rejection turned out to be similar enough to standard bin packing in the sense that it is possible to provide an algorithm with an absolute approximation ratio of $\frac{3}{2}$ and an APTAS. The running times of these algorithms are unfortunately relatively high. Some subsequent work on the subject can be found in [3, 12]. Specifically, an approximation scheme, which uses a different approach, and has a reduced running time, is given in [3]. Our APTAS creates at most $n^{(O(\frac{1}{\varepsilon})^{O(\frac{1}{\varepsilon})})}$ packings of large items, and a linear programming instance with $O(n^2)$ variables and $O(n)$ constraints is solved for each one, whereas the APTAS of [3] has a running time of $n^{O(\frac{1}{\varepsilon^2})}$. Recently, the complexity of the problem was resolved and an AFPTAS is given in [12], using some methods of the current work, together with new techniques for dealing with small items. The design of a simple and efficient heuristic with an absolute approximation ratio of $\frac{3}{2}$ is left as an open problem. Note that the heuristics of Dósa and He [10] are very efficient and have a worst case running time of $O(n \log n)$, while our heuristic is based on the APTAS, and though its running time is polynomial in n, the exponent of n is extremely large.

As for online bin packing with rejection, this case also turned out to be similar enough to standard bin packing. It seems possible to adapt most online algorithms for the standard problem into algorithms for the problem with rejection, without any loss in the asymptotic competitive ratio. This paper did not deal with the absolute competitive ratio. For that model, there is clear evidence that the problem with rejection is very different from the standard problem. An upper bound of 1.75 on the absolute competitive ratio of FIRST FIT for standard bin packing was proved in [22]. However, Dósa and He [10] proved a lower bound of 2.343 on the absolute competitive ratio of any online algorithm for bin packing with rejection.

References

1. Bansal, N., Blum, A., Chawla, S., Dhamdhere, K.: Scheduling for flow-time with admission control. In: Proc. of the 11th Annual European Symposium on Algorithms (ESA2003), pp. 43–54 (2003)
2. Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., Stougie, L.: Multiprocessor scheduling with rejection. SIAM J. Discrete Math. **13**(1), 64–78 (2000)
3. Bein, W.W., Correa, J.R., Han, X.: A fast asymptotic approximation scheme for bin packing with rejection. In: Proc. of the 1st International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies (ESCAPE2007), pp. 209–218 (2007)
4. Caprara, A., Kellerer, H., Pferschy, U.: Approximation schemes for ordered vector packing problems. Naval Res. Logist. **92**, 58–69 (2003)
5. Coffman, E.G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing: a survey. In: Hochbaum, D. (ed.) Approximation Algorithms. PWS Publishing Company, Boston (1997)
6. Csirik, J., Woeginger, G.J.: On-line packing and covering problems. In: Fiat, A., Woeginger, G.J. (eds.) Online Algorithms: The State of the Art, pp. 147–177 (1998)
7. Csirik, J., Woeginger, G.J.: Resource augmentation for online bounded space bin packing. J. Algorithms **44**(2), 308–320 (2002)
8. Fernandez de la Vega, W., Lueker, G.S.: Bin packing can be solved within $1 + \varepsilon$ in linear time. Combinatorica **1**, 349–355 (1981)
9. Dósa, G.: The tight bound of first fit decreasing bin-packing algorithm is $\mathrm{FFD}(I) \leq 11/9\mathrm{OPT}(I) + 6/9$. In: Proc. of the 1st International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies (ESCAPE2007), pp. 1–11 (2007)
10. Dósa, G., He, Y.: Bin packing problems with rejection penalties and their dual problems. Inf. Comput. **204**(5), 795–815 (2006)
11. Engels, D.W., Karger, D.R., Kolliopoulos, S.G., Sengupta, S., Uma, R.N., Wein, J.: Techniques for scheduling with rejection. J. Algorithms **49**(1), 175–191 (2003)
12. Epstein, L., Levin, A.: AFPTAS results for common variants of bin packing: a new method to handle the small items. Manuscript (2007)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman, New York (1979)
14. Hochbaum, D.S., Shmoys, D.B.: Using dual approximation algorithms for scheduling problems: theoretical and practical results. J. ACM **34**(1), 144–162 (1987)
15. Hoogeveen, H., Skutella, M., Woeginger, G.J.: Preemptive scheduling with rejection. In: Proc. of the 8th Annual European Symposium on Algorithms (ESA2000), pp. 268–277 (2000)
16. Johnson, D.S.: Fast algorithms for bin packing. J. Comput. Syst. Sci. **8**, 272–314 (1974)
17. Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L.: Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J. Comput. **3**, 256–278 (1974)

18. Karmarkar, N., Karp, R.M.: An efficient approximation scheme for the one-dimensional bin-packing problem. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS'82), pp. 312–320 (1982)
19. Lee, C.C., Lee, D.T.: A simple online bin packing algorithm. J. ACM **32**(3), 562–572 (1985)
20. Ramanan, P., Brown, D.J., Lee, C.C., Lee, D.T.: Online bin packing in linear time. J. Algorithms **10**, 305–326 (1989)
21. Seiden, S.S.: On the online bin packing problem. J. ACM **49**(5), 640–671 (2002)
22. Simchi-Levi, D.: New worst-case results for the bin-packing problem. Naval Res. Logist. **41**(4), 579–585 (1994)
23. Ullman, J.D.: The performance of a memory allocation algorithm. Technical Report 100, Princeton University, Princeton, NJ (1971)
24. van Vliet, A.: An improved lower bound for online bin packing algorithms. Inf. Process. Lett. **43**(5), 277–284 (1992)
25. Yao, A.C.C.: New algorithms for bin packing. J. ACM **27**, 207–227 (1980)
26. Yue, M.: A simple proof of the inequality $\mathrm{FFD}(L) \leq (11/9)\mathrm{OPT}(L) + 1, \forall L$, for the FFD bin-packing algorithm. Acta Math. Appl. Sin. **7**, 321–331 (1991)
27. Zhang, G.: Private communication
28. Zhang, G., Cai, X., Wong, C.K.: Linear time approximation algorithms for bin packing. Oper. Res. Lett. **26**, 217–222 (2000)