Bin packing with rejection revisited

Leah Epstein

Department of Mathematics, University of Haifa, 31905 Haifa, Israel.
Email: lea@math.haifa.ac.il

Abstract. We consider the following generalization of bin packing. Each item is associated with a size bounded by 1, 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 absolute approximation ratio $\frac{3}{2}$, 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 an algorithms whose competitive ratios tend to the best possible asymptotic competitive ratio. We show that our algorithms have the same asymptotic competitive ratios as these known for the standard problem, whose ratios tend 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.

1 Introduction

In the classical bin packing problem [20, 5, 4], 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 existing supply of such bins is unbounded. Each item must be packed into exactly one bin, minimizing the number of bins used. 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" 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*. 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 used to pack the accepted items, i.e., items which are not rejected. 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. For an item $i \in I$, we denote its size by p_i and its rejection cost by r_i. In this paper we study both offline and online algorithms for bin packing with rejection. In online environments of the bin packing problem, we receive the items as a sequence σ. Every element in the sequence is a pair, giving the size and rejection cost of this element. Thus, we get a sequence $(p_1, r_1), (p_2, r_2) \ldots (p_n, r_n)$, and the set I contains the same elements as σ. The elements arrive one by one. Upon arrival, an item must be either assigned or rejected. Such a decision is irrevocable.

The bin packing problem with rejection was introduced and studied by Dósa and He [8]. 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 is associated with the communication cost of downloading the file from an external server. An algorithm would need to have a cost as low as possible with respect to 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 vertices is denoted by OPT. In this paper we mostly consider the asymptotic competitive ratio and the asymptotic approximation ratio criteria. When we discuss the 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 $\mathcal{R}_{\mathcal{A}} = \limsup_{n\to\infty} \sup_{\sigma} \left\{ \frac{\mathcal{A}(\sigma)}{\mathrm{OPT}(\sigma)} | \mathrm{OPT}(\sigma) = n \right\}$. 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, $\mathcal{A} \leq \mathcal{R} \cdot \mathrm{OPT}$. If the approximation (competitive) ratio of a polynomial time offline (online) algorithm is at most $\mathcal{R}$, we say that it is a $\mathcal{R}$-approximation ($\mathcal{R}$-competitive), this applies to both types of approximation and competitive ratios.

Previous work In [8], Dósa and He study four variants of bin packing with rejection. These are offline and online bin packing with respect to the absolute and the asymptotic measures. For the offline problem, the approximation ratios

of the algorithms shown in the paper 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 [10]). Note that this holds already for standard bin packing.

For the online problem, they design an algorithm of absolute competitive ratio 2.618 and an algorithm of asymptotic competitive ratio $1.75 + \varepsilon$. They show a lower bound of 2.343 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 [21] 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 [7]. This scheme returns for every given value $\varepsilon > 0$ an algorithm with asymptotic approximation ratio $1+\varepsilon$. The algorithm has polynomial running time if ε is seen as a constant. Karmarkar and Karp [15] 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.

As stated above, the absolute approximation ratio of any algorithm cannot be expected to be better 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 [23] and also proved directly using a simple proof in [19]. Several other algorithms with the same approximation ratio are known, (see e.g. [25]).

As for the online problem, the currently best known upper bound on the asymptotic competitive ratio is 1.58889 due to Seiden [18], this problem has been extensively studied. Previous results include a sequence of improvements as follows.

The online bin packing problem was first investigated by Ullman [20]. He showed that the FIRST FIT algorithm has performance ratio $\frac{17}{10}$. This result was then published in [13]. Johnson [14] showed that the NEXT FIT algorithm has performance ratio 2. Yao [22] designed an algorithm called REVISED FIRST FIT and showed that it has performance ratio $\frac{5}{3}$.

Lee and Lee developed the REFINED HARMONIC algorithm, which they showed to have a performance 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 competitive ratio of at most $\frac{538}{333} < 1.61562$ and claimed that the second algorithm has competitive ratio of at most $\frac{239091}{148304} < 1.61217$ [17].

There is much less study of the absolute competitive ratio, and the existent study focuses on the performance of First Fit. Simchi-Levi [19] proved an upper

bound of 1.75 on the absolute competitive ratio. A lower bound of $\frac{5}{3}$ is given by Zhang [24].

An important version of online bin packing (which is not studied in [8]) 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. For the classical bin packing problem, Lee and Lee [16] presented an algorithm called HARMONIC, which partitions items into $m > 1$ 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 performance ratio of at most $(1 + \varepsilon)\Pi_\infty$ [16], where $\Pi_\infty \approx 1.69103$ is the sum of series (see Section 3.2). They also showed there is no bounded space algorithm with a performance ratio below Π_∞. The algorithms mentioned above REFINED HARMONIC, MODIFIED HARMONIC and MODIFIED HARMONIC 2 are all unbounded space adaptations of HARMONIC. Note that the 1.75 upper bound of Dósa and He [8] 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 competitive ratios tends to 1.75 from above.

There has been a fair amount of research on variants of well known problem, where a notion of rejection is introduced. Such studies include research on variants of various important scheduling problems (see [2, 12, 1, 9]). 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 problem which uses techniques from [7] but also from [11] and [3]. For a given value of ε, the APTAS has cost of at most $(1+\varepsilon)\mathrm{OPT}+1$.

Next, we design an algorithm with absolute approximation ratio $\frac{3}{2}$. To do that, we use the APTAS using a constant value of ε, combined with adaptations of the APTAS and and additional arguments for cases where the value OPT is small. 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 [8] 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 [20, 18] 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 [16] 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 re-

jections. Both our algorithms, the rejective variants of HARMONIC and MODIFIED HARMONIC achieve better asymptotic competitive ratios than the algorithm of [8]. Their ratios are approximately 1.69103 and 1.61562, whereas the algorithm of [8] has a competitive ratio $1.75 + \varepsilon$.

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 [7]. The adaptation we design here has some similarities with [3], 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 methods similar to ones used for scheduling, as in [11].

We assume that without loss of generality, each rejection cost r_i satisfies $r_i < 1$. We can make this assumption since an item of rejection cost at least 1, that is rejected in some solution, can be placed in a bin of its own instead, and the solution cost does not increase. We also assume OPT ≥ 1. In order to be able to assume this, note that if 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 we output this solution and otherwise, we run the APTAS. We can always check the solution which rejects all jobs and output it if it turns out to be better than the result of the APTAS. This will be useful to get a better approximation for small values of OPT which is done later.

As in [7], 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 $r_j \geq \delta$ and $p_j \geq \delta$. 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 a 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 $(N+1)^N \leq (\frac{1}{\delta^4})^{\frac{1}{\delta^4}}$ possible packings. Note that in this process with opened bins but possibly some of them remained empty. The set of bins remaining empty after this process are removed from the packing. We would like to add empty bins later and to test all possible amounts of empty bins, such bins are added to the packing to accommodate small items.

For the case where $N \geq \frac{1}{\delta^4}$, we perform a rounding of the rejection costs of all items in L. 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 of 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. Then we can show the following.

Lemma 1. $A'(I') \leq (1+\delta)A(I') \quad and \quad OPT(I') \leq OPT(I)$

Proof. To show $OPT(I') \leq OPT(I)$ 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 $A'(I') \leq (1+\delta)A(I')$, note that each rejection cost decreases by at most an additive factor of δ^2 in the rounding procedure. However, since all (rounded and original) rejection costs are at least δ, each rejection cost increases by a factor of at most $1+\delta$ when the rounded rejection costs are replaced by the original ones.

For $0 \leq i \leq \Delta$, let N_i be the number of items with rounded rejection cost $\delta + i\delta^2$, and let $a_{i,1} \geq \ldots \geq a_{i,N_i}$ 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 $1 \leq i \leq \Delta$, denote the multi-set of item sizes by B_i.

We perform a linear grouping on each one of the multi-sets of large items $B_i = \{a_{i,1}, \ldots, a_{i,N_i}\}$. Let $m = \frac{1}{\delta^2}$. We partition the sorted set of large items into m consecutive sequences $S_{i,j}$ $(j = 1, \ldots, m)$ 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., $S_{i,j} = \{a_{i,(j-1)k_i+1}, \ldots, a_{i,(j-1)k_i+k_i}\}$ for $j = 1, 2, \ldots, m$. For $j \geq 2$, we define a modified sequence $\hat{S}_{i,j}$ which is based on the sequence S_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 $L_1 = \bigcup_{i=0}^{\Delta} S_{i,1} = L' - L''$ 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 size and have the same rounded rejection cost).

We enumerate all possible packings of the L'' items into i bins, where $0 \leq i \leq N$. 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 $(b_1, \rho_1), \ldots, (b_T, \rho_T)$ be the set of types, where $\delta \leq b_j \leq 1$ is the size of items of type (b_j, ρ_j) and $\delta \leq \rho_j \leq 1$ is its (rounded) rejection cost. We represent a multiset of items by a vector $J = (u_1, \ldots, u_T)$, where u_j is the number of items of type (b_j, ρ_j). Let $\hat{N} = (n_1, \ldots, n_T)$ 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 $\sum_{i=j}^{T} q_j b_j \leq 1$. 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 $1 \leq i \leq T$), 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 $|Q| \leq (T+1)^{\frac{1}{\delta}}$. 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 an 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.

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

Since for each $1 \leq j \leq T$, 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 $\ell \leq N$, we are only interested in vectors y where each component is in the set $\{0, \ldots, N\}$. 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, $d = \ell, \ldots, n$. 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 $z = 1, \ldots, d$ by x_z. We define $x_z = 1$ for $\ell < z \leq d$. 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 $1 \leq z \leq d+1$ and $1 \leq j \leq n'$, let $X_{j,z}$ be an indicator variable. If $z \leq d$, 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 $1 \leq z \leq d$, $\sum_{j=1}^{n'} c_j \cdot X_{j,z} \leq x_z$. We clearly have $\sum_{z=1}^{d+1} X_{j,z} \geq 1$ for all $1 \leq j \leq n'$, 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 $X_{j,z} \geq 0$. 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 $\delta d_1 + \lceil \delta(d - d_1) \rceil \leq \delta d + 1$. 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.

For two multisets A, B, whose elements are pairs of sizes and rejection costs of items. We say that A is *dominated* by B and denote $A \leq B$ 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 $p_b \geq p_a$ and $r_b \geq r_a$.

Lemma 2. *If A and B are multisets such that $A \leq B$, then $OPT(A) \leq OPT(B)$.*

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 $p_{\hat{a}} \leq p_{\hat{b}}$, the resulting packing is feasible. Since $r_{\hat{a}} \leq r_{\hat{b}}$, the rejection cost cannot increase.

We would like to analyze the minimum cost of any solution we get. To upper bound the cost of this minimal solution, we actually upper bound the cost of one specific solution, defined later. We prove the following theorem.

We 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 1. *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 $J \leq I'$ 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 $j \geq 2$. 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 $OPT(J) \leq OPT(I') \leq OPT(I)$. In the case where no linear grouping was performed ($N \leq \frac{1}{\delta^4}$), 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 $OPT(J)$. 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 $OPT(J)$, 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 $OPT(J)$, there are d' used bins and thus $OPT(J) \geq d'$. Thus the additional cost is at most $\delta \mathrm{OPT}(J)+1 \leq \delta \mathrm{OPT}(I)+1$.

If linear grouping is done, then the cost of packing L_1 is at most $\sum_{i=0}^{\Delta} k_i = \sum_{i=1}^{\Delta} \lceil \delta^2 N_i \rceil \leq N\delta^2 + (\Delta+1) \leq N\delta^2 + \frac{1}{\delta^2}$. Since $N \geq \frac{1}{\delta^4}$ in this case, and there are N items whose size and rejection cost are at least δ, we have $OPT(I) \geq \delta N \geq \frac{1}{\delta^3}$. 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 $\delta \leq 1$) $(1+\delta)((1+3\delta)OPT(I)+1) \leq OPT(I)(1+7\delta)+1+\delta \leq OPT(I)(1+8\delta)+1$, since $OPT(I) \geq 1$. Taking $\delta \leq \frac{\varepsilon}{8}$ 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.

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.

Consider first the two cases $\mathrm{OPT} < 1$ and $\mathrm{OPT} \geq 2.25$. We can 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.

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 done above. Given such a subset X such that $|X| \leq 2$, the complement set $Y - X$ is clearly packed completely into the single bin. We check whether so far the solution is feasible (i.e., the items in $Y - X$ 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 $\mathrm{OPT} < 2.25$ 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 an optimal solution there are no rejected items whose rejection cost is larger than $\frac{1}{4}$. Thus, all these items should be assigned to bins. This gives a linear program similar to the above, however items which may not be rejected, do not have a variable $X_{j,d+1}$ (i.e., define $X_{j,d+1} = 0$ and omit this variable), where is our case, $d = 3$. We use the linear program, and get a solution where at most two items are split in some way. We reject these two items getting an additional cost of at most $\frac{1}{2}$.

If OPT < 2.25 then the cost of the fractional packing is in the interval $[1, 2.25)$. Thus the cost of the solution we get is again at most OPT$+\frac{1}{2} \leq \frac{3}{2}$OPT.

We proved the following theorem.

Theorem 2. *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 scheme 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 [20] (see also [16,18]). 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, thus 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 maximum sum of weights of items which can fit into a single bin. In some cases, a constant number k of distinct weighting functions are defined to handle several major behaviors of the algorithm (resulting from specific 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 maximum sums of weights in a single bin for the k weight functions. The method in [18] 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., numbers of bins) but to rejection costs as well.

Let $\mathcal{A}$ be an online algorithm and let $\mathcal{C}$ be a desired competitive ratio. Let $w^1, \ldots, w^k$ be a set of functions $w^i : (0,1] \rightarrow R_0^+$ (where R_0^+ denotes the set of non-negative real numbers). For an item j, we denote its weight with respect to weight function w^i by w_j^i.

Theorem 3. *A value $\mathcal{C}$ is an upper bound on the asymptotic competitive ratio of algorithm $\mathcal{A}$ if the following conditions hold.*
1. For every item j, and for every weight function w^i, we have that $w_j^i \leq \mathcal{C} r_j$, that is, for every weight function, the weight assigned to each item is no larger than $\mathcal{C}$ times its rejection cost.
2. There exists a constant μ, such that for every input, there exists a value $1 \leq i \leq k$ such that $\mathcal{A} \leq \sum_{j=1}^{n} w_j^i + \mu$.
3. For every set of items J such that $\sum_{j \in J} p_j \leq 1$, and every $1 \leq i \leq k$, we have $\sum_{j \in J} w_j^i \leq \mathcal{C}$.

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

For any algorithm $\mathcal{B}$ (including an optimal offline algorithm OPT) Let $\mathcal{B}_{rej}$ be the set of items rejected by $\mathcal{B}$ and let $\mathcal{B}_{acc}$ be the set of items accepted by $\mathcal{B}$. Furthermore, let $\mathcal{B}_{nacc}$ be the number of bins opened by $\mathcal{B}$. We denote by $\mathcal{B}_{acc}(i)$, the set of items assigned by $\mathcal{B}$ to its ith bin, for $1 \leq i \leq \mathcal{B}_{nacc}$.

$$\mathcal{A} \leq \sum_{j=1}^{n} w_j^i + \mu \leq \sum_{j \in \mathrm{OPT}_{rej}} w_j^k + \sum_{j \in \mathrm{OPT}_{acc}} w_j^k + \mu \leq \sum_{j \in \mathrm{OPT}_{rej}} \mathcal{C} \cdot r_j$$
$$+ \sum_{j \in \mathrm{OPT}_{acc}} w_j^k + \mu = \mathcal{C} \cdot \sum_{j \in \mathrm{OPT}_{rej}} r_j + \sum_{i=1}^{\mathcal{B}_{nacc}} \sum_{j \in \mathcal{B}_{acc}(i)} w_j^k + \mu \leq \mathcal{C} \cdot \sum_{j \in \mathrm{OPT}_{rej}} r_j$$
$$+ \sum_{i=1}^{\mathcal{B}_{nacc}} \mathcal{C} + \mu = \mathcal{C} \cdot \sum_{j \in \mathrm{OPT}_{rej}} r_j + \mathcal{C}\mathcal{B}_{nacc} + \mu \ .$$

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

3.2 Algorithm Rejective Harmonic

We now define our adaptation of the Harmonic$_k$ algorithm of Lee and Lee [16]. 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 $(\frac{1}{i+1}, \frac{1}{i}]$ for $i = 1, \ldots, k-1$ (intervals

$1, \ldots, k-1$) and one final sub-interval $(0, \frac{1}{k}]$ (interval k). 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 $i = 1, \ldots, k-1$ (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 $1 \leq i \leq k-1$, 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 $a \in I$, let $\frac{1}{s_a}$ be the right endpoint of the sub-interval $1 \leq s_a \leq k$ to which p_a belongs. If $s_a < k$, item a is rejected if $r_a \geq \frac{1}{s_a}$, and otherwise a is accepted and packed according to the algorithm above. If $s_a = k$, item a is rejected if $r_a \geq \frac{k}{k-1} p_a$, 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 [16], however, unlike the proof in [16], our weights are a function of both the sizes and rejection costs. We use a single weight function w, and the weight of item $a \in I$ is denoted w_a.

In order to use the method, we need to assign the weights so that the three conditions in Theorem 3 hold. We do the assignment so that the cost of the algorithm satisfies $RejH_k \leq \sum_{a \in I} w_a + k - 1$.

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 $1, \ldots, k-1$ 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, $i \geq 1$, 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 [16]. 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 [16] 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 4. *The asymptotic competitive ratio of $RejH_k$ tends to Π_∞ as k grows. No algorithm can have a smaller asymptotic competitive ratio.*

Proof. As mentioned above, the lower bound follows directly from the lower bound in [16] 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 3 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 $w_j \leq r_j$ for every item.

We analyze an algorithm $\mathcal{A} = RejH_k$. Let $\mathcal{A}_{cacc}(i)$ be the set of accepted items of sub-interval i for $1 \leq i \leq k$. Each bin for interval i ($1 \leq i \leq k-1$) 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. We use the same notations as in the proof of Theorem 3 and get,

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

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 $x \leq \frac{1}{k}$, we can use the result of [16], 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 [6]. For completeness, we include a proof.

Let t be a maximal integer such that $\pi_t \leq k$. 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 $\pi_{t+1} - 1 \geq k$. 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 $\sum_{i=1}^{t} \frac{1}{\pi_i - 1} + \frac{1}{\pi_{t+1}-1} \cdot \frac{k}{k-1}$.

Claim. 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 $v = 1, \ldots, i-1$. 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 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 sum 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. 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 3, we establish the competitive ratio.

Consider an optimal offline algorithm OPT. For this algorithm, denote by R_{OPT} the set of rejected items and by A_{OPT} the set of accepted items. Let B_{OPT} denote the number of used bins.

Then $\mathrm{OPT} = B_{\mathrm{OPT}} + \sum_{a \in R_{\mathrm{OPT}}} r_a \geq B_{\mathrm{OPT}} + \sum_{a \in R_{\mathrm{OPT}}} w_a$. Therefore, in order to prove an asymptotic competitive ratio $\mathcal{C}$, it is enough to prove $\sum_{a \in A_{\mathrm{OPT}}} w_a \leq \mathcal{C} \cdot B_{\mathrm{OPT}}$. To prove this, it is enough to consider every bin of OPT separately, and to show that the sum of weights of items in this bin is at most $\mathcal{C}$. Finally, to show this, we upper bound the sum of weights of items that can fit in a single bin.

To summarize the technique used here, we assign weights to items, so that the cost of an algorithm is roughly the sum of weights. Then we reduce the problem into that of finding the maximum sum of weights of items in a single bin. This method is often used in bin packing problems, and was already used in [20]. Surprisingly, the method here is applied even though the weights are not related only to packing items but to rejection costs as well.

3.3 Algorithm Rejective Modified Harmonic

In this section we show how to design improved algorithms which are 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. [17]. 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 a lot of space in each single bin. To avoid this large waste of space, MODIFIED HARMONIC and other later algorithms (see [18]) use two extra interval endpoints, of the form $\frac{1}{2} < \Delta < 1$ 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 I_1^1) are still packed one per bin as in HARMONIC. These algorithms furthermore use parameters α^i $(i = 2, \ldots, n-1)$ 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 $I_n = (0, \frac{1}{n}]$ are not combined with larger items and are packed using NEXT FIT.

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

$$\alpha^2 = \frac{1}{9};\ \alpha^3 = \frac{1}{12};\ \alpha^4 = \alpha^5 = 0;\ \alpha^i = \frac{37 - i}{37(i+1)},\ \text{for } 6 \leq i \leq 36 \text{ and } \alpha^{37} = 0.$$

The results of [17] imply that the asymptotic performance ratio of MODIFIED HARMONIC 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 amount m_i of such items that can fit into the bin, leaving an empty space of size at least Δ. In this calculation, a maximum size of 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 amount 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 their amount 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 an item, we consider all possible cases. If $x \in I_1^1 = (\Delta, 1]$, x is rejected if $r_j \leq 1$ and otherwise accepted. If $x \in I_1^2 = (\frac{1}{2}, \Delta]$, x is rejected if $r_j \leq \frac{2}{3}$ and otherwise accepted. If $x \in I_2^1 = (1-\Delta, \frac{1}{2}]$, x is rejected if $r_j \leq \frac{1}{2}$ and otherwise accepted. If $x \in I_1^2 = (\frac{1}{3}, \Delta]$, x is rejected if $r_j \leq \frac{4}{9}$ and otherwise accepted. If $x \in I_3 = (\frac{1}{4}, \frac{1}{3}]$, x is rejected if $r_j \leq \frac{11}{36}$ 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 $r_j \leq \frac{1}{i}$ and otherwise accepted. If $x \in I_i = (\frac{1}{i+1}, \frac{1}{i}]$ for $6 \leq i \leq 36$, x is rejected if $r_j \leq \frac{37(i+1)}{38}$ and otherwise accepted. If $x \in I_{38} = (0, \frac{1}{38}]$, x is rejected if $r_j \leq \frac{38 \cdot p_j}{37}$ 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 [17], 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 item of interval I_1^1 gets weight $w_j^1 = w_j^2 = 1$. An item of interval I_1^2 gets weight $w_j^1 = 1$, $w_j^2 = \frac{2}{3}$. An item of interval I_2^1 gets weight $w_j^1 = w_j^2 = \frac{1}{2}$. An item of interval I_2^2 gets weight $w_j^1 = \frac{4}{9}$, $w_j^2 = \frac{5}{9}$. An item of interval I_3 gets weight $w_j^1 = \frac{11}{36}$, $w_j^2 = \frac{7}{18}$. An item of interval I_i for $6 \leq i \leq 36$ gets weight $w_j^1 = \frac{38}{37(i+1)}$, $w_j^2 = w_j^1 + \frac{37-i}{37m_i(i+1)}$. An item of interval I_{38} gets weight $w_j^2 = w_j^1 = \frac{38 \cdot p_j}{37}$.

These weights are defined as in [17] except for rejected items and items in the interval I_1^2 for which we defined $w_j^2 = \frac{2}{3}$. Note that weights and rejection rules are defined in a way that for a rejected item, its weight is never larger than the weight $\min\{w_j^1, w_j^2\}$ that an item j would have received if had a larger rejection cost and were accepted.

In order to analyze the competitive ratio and show it is at most $\mathcal{C}_1 = \frac{538}{333} < 1.61562$ (as for the original algorithm), we show that all conditions of Theorem 3 hold. The second condition holds due to the following. The proof of [17] 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 $w_j^2 = 0$. Since the weight of rejected items is exactly their rejection cost, and the weights we define are never smaller than the weights in [17], the condition follows.

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 its weight w_j^1. Thus we need to show for every item that $w_j^2 \leq \mathcal{C}_1 w_j^1$. 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 $\frac{w_j^2}{w_j^1} = 1.5$. 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 $6 \leq i \leq 36$ we have $\frac{w_j^2}{w_j^1} = 1 + \frac{37-i}{38m_i}$. This value is maximized for $i = 6$, since m_i is monotonically increasing. For $i = 6$ we have $m_i = 2$ and thus $\frac{w_j^2}{w_j^1} \leq 1 + \frac{31}{76} < \frac{3}{2}$.

To prove the last condition of Theorem 3, we note again that weights of rejected items are never larger than their weights according to each weight function, and thus we need to consider the weight functions as they are defined. Items for which the weights are defined as in [17], the proof follows from the result in that paper. Thus we need to consider only w^2 and only sets of items that can fit in a bin and which contain an item of size in I_1^2. Denote this large item by x. Such a bin can contain in addition only items smaller than $\frac{1}{2}$. In order to give an upper bound on the total weight of items in the bin, we find an upper bound on the ratio $\frac{w_j^2}{p_j}$ for items no larger than $\frac{1}{2}$. We can see that this ratio is no larger than $\frac{5}{3}$ for items in $(\frac{1}{6}, \frac{1}{2}]$, no larger than $\frac{38}{37}$ for items in $(0, \frac{1}{37}]$, and no larger than $\frac{38}{37} + \frac{37-i}{37m_i}$. Again, this ratio is maximized for $i = 6$ and is smaller than $\frac{3}{2}$. Thus an upper bound on the total weight in the bin with respect to w^2 is $\frac{2}{3} + \frac{1}{2} \cdot \frac{5}{3} = \frac{3}{2}$.

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

Theorem 5. *The competitive ratio of* Rejective Modified Harmonic *is at most* $\mathcal{C}_1 = \frac{538}{333}$.

References

1. N. Bansal, A. Blum, S. Chawla, and K. Dhamdhere. Scheduling for flow-time with admission control. In *Proc. of the 11th Annual European Symposium on Algorithms (ESA2003)*, pages 43–54, 2003.
2. Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. Sgall, and L. Stougie. Multiprocessor scheduling with rejection. *SIAM Journal on Discrete Mathematics*, 13(1):64–78, 2000.
3. A. Caprara, H. Kellerer, and U. Pferschy. Approximation schemes for ordered vector packing problems. *Naval Research Logistics*, 92:58–69, 2003.
4. E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum, editor, *Approximation algorithms*. PWS Publishing Company, 1997.
5. J. Csirik and G. J. Woeginger. On-line packing and covering problems. In *A. Fiat and G. J. Woeginger, editors,* Online Algorithms: The State of the Art, pages 147–177, 1998.
6. J. Csirik and G. J. Woeginger. Resource augmentation for online bounded space bin packing. *Journal of Algorithms*, 44(2):308–320, 2002.
7. W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1+\varepsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.
8. G. Dósa and Y. He. Bin packing problems with rejection penalties and their dual problems. *Information and Computation*, 204(5):795–815, 2006.

9. D. W. Engels, D. R. Karger, S. G. Kolliopoulos, S. Sengupta, R. N. Uma, and J. Wein. Techniques for scheduling with rejection. *Journal of Algorithms*, 49(1):175–191, 2003.
10. M. R. Garey and D. S. Johnson. *Computers and intractability.* W. H. Freeman and Company, New York, 1979.
11. D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987.
12. H. Hoogeveen, M. Skutella, and G. J. Woeginger. Preemptive scheduling with rejection. In *Proc. of the 8th Annual European Symposium on Algorithms (ESA2000)*, pages 268–277, 2000.
13. D. S. Johnson, A. Demers, J. D. Ullman, Michael R. Garey, and Ronald L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3:256–278, 1974.
14. David S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8:272–314, 1974.
15. N. Karmarkar and R. M. Karp. 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*, pages 312–320, 1982.
16. C. C. Lee and D. T. Lee. A simple online bin packing algorithm. *Journal of the ACM*, 32(3):562–572, 1985.
17. P. Ramanan, D. J. Brown, C. C. Lee, and D. T. Lee. Online bin packing in linear time. *Journal of Algorithms*, 10:305–326, 1989.
18. S. S. Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, 2002.
19. D. Simchi-Levi. New worst-case results for the bin-packing problem. *Naval Res. Logist.*, 41(4):579–585, 1994.
20. J. D. Ullman. The performance of a memory allocation algorithm. Technical Report 100, Princeton University, Princeton, NJ, 1971.
21. A. van Vliet. An improved lower bound for online bin packing algorithms. *Information Processing Letters*, 43(5):277–284, 1992.
22. A. C. C. Yao. New algorithms for bin packing. *Journal of the ACM*, 27:207–227, 1980.
23. M. Yue. A simple proof of the inequality $FFD(L) \leq (11/9)OPT(L) + 1$, $\forall$ L, for the FFD bin-packing algorithm. *Acta. Math. Appl. Sinica*, 7:321–331, 1991.
24. G. Zhang. Private communication.
25. G. Zhang, X. Cai, and C.K. Wong. Linear time approximation algorithms for bin packing. *Operations Research Letters*, 26:217–222, 2000.