Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 390 (2008) 110–119

Theoretical Computer Science

www.elsevier.com/locate/tcs

Online bin packing with arbitrary release times[☆]

Yongqiang Shi[a], Deshi Ye[b,*]

[a] College of Economics, Zhejiang University, Hangzhou 310027, China
[b] College of Computer Science, Zhejiang University, Hangzhou 310027, China

Received 20 May 2007; received in revised form 18 October 2007; accepted 18 October 2007

Communicated by X. Deng

Abstract

We study a new variant of the online bin-packing problem, in which each item a_i is associated with a size a_i and also a release time r_i so that it must be placed at least r_i above the bottom of a bin. Items arrive in turn and must be assigned without any knowledge of subsequent items. The goal is to pack all items into unit-size bins using the minimum number of bins. We study the problem with all items have equal size. First, we show that the ANY FIT algorithm cannot be approximated within any constant. Then we present a best possible online algorithm with asymptotic competitive ratio of two.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Bin packing; Online algorithm; Performance evaluation

1. Introduction

Bin packing is one of the basic problems since the early 1970s in theoretical computer science and combinatorial optimization literature and different variants of the problem continue to attract researchers' attentions. In this problem, we are given a list L of items $a_1, a_2, \ldots, a_n$, each item $a_i \in (0, 1]$, and the goal is to pack this sequence of items into unit-size bins using the minimum number of bins. A packing algorithm is called *on-line* if it packs items one by one without any information on subsequent items. We call a packing algorithm *offline*, if the algorithm knows all the information of items.

In this paper we study a new variant of the online bin packing problem, called *online bin packing with arbitrary release time*. In this variant, each bin has a time axis [0, 1] from the bottom to top, jobs arrive in turn and must be assigned without information of future items. Each item a_i is associated with a size a_i and a release time r_i so that it must be placed at least r_i above from the bottom of a bin. The goal is to pack all items into unit-size bins and minimize the number of bins used. Without loss of generality, we assume that $a_i + r_i \leq 1$, otherwise, this item cannot

[☆] An extended abstract of this paper appeared in Proceedings of the International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental methodologies (ESCAPE 2007).

[*] Corresponding author. Tel.: +86 571 88916744.
E-mail address: yedeshi@zju.edu.cn (D. Ye).

0304-3975/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2007.10.023

be assigned to any bin. In this paper, we focus on the problem of *unit size items*. All items have the same size of $1/K$ for some integer $K \geq 1$.

Unlike traditional online models, where items arrive one by one without release times, or items arrive over time but the items appear only after their release times, the online model studied in this paper is that items arrive one by one with arbitrary release times. Upon the arrival of items, their sizes and release times are known to the scheduler. Items must immediately and irrevocably be assigned without any information on subsequence items. Furthermore, items should be fitted into a position after their release times.

To the best of our knowledge, this kind of online model was first proposed by Li and Huang [8]. In their paper, each job has a release time and a processing time; jobs cannot be scheduled before their release time. Jobs arrive in turn and must be scheduled without any information about subsequent jobs while their release times are independent of the order. The problem is to find a schedule of these jobs while minimizing the overall completion time, i.e. the *makespan*. They gave the tight competitive ratio $3 - 1/m$ of List Scheduling (LS) and also proposed an improved algorithm for $m \geq 2$, where m is the number of machines.

The online bin packing problem with arbitrary release time can be regarded as a variant of the online scheduling problem for jobs with arbitrary release time. Each job is also associated with processing time and release time; however, all jobs have common due date. The goal of this scheduling is to schedule all the jobs before their due date while minimizing the number of machines used. This scheduling problem is motivated by an air cargo import terminal problem. Cargo agents will make requests daily either by fax or through the web or by phone to book for truck docks at the terminal for cargo collection. Cargo agents will also indicate their preference time per day to come to the terminal. Since the truck docks are a scarce resource, it is significant to efficiently schedule all the jobs in a day to reduce using the number of docks. Comparing to the bin-packing problem, the processing time of a job can be regarded as the size of an item and the common due date can be regarded as the size of a bin. This bin-packing problem arises also in many real world applications: e.g. in the real-world packing of bins, some items are fragile and cannot be assigned to a bin with too many items above.

The standard measure of an algorithm's quality for online bin packing is the *asymptotic competitive ratio*. Given a list L of items and an online algorithm A, we denote by $\mathrm{OPT}(L)$ and $A(L)$, respectively, the minimum possible number of bins used to pack items in L and the number of bins used by algorithm A on L. The *asymptotic competitive ratio* R_A^∞ of algorithm A is defined to be

$$R_A^\infty = \limsup_{n\to\infty} \max_L \{A(L)/\mathrm{OPT}(L) | \mathrm{OPT}(L) = n\}.$$

Throughout of this paper, we will often simply write *competitive ratio* instead of “asymptotic competitive ratio”.

Related results: Our studied problem becomes the classical online bin-packing problem when all the jobs are released at time zero. There is a long history of results for the classical bin packing problem (see the survey [3–5]). The bin packing is well known to be NP-hard [6]. The classical online bin packing problem was first investigated by Ullman [10]. The FIRST FIT algorithm [7] was shown to have the competitive ratio of 17/10. A straightforward class of online algorithm is the ANY FIT (AF) algorithm: which never puts an item a_i into an empty bin unless the item does not fit into any partially filled bins. NEXT FIT, FIRST FIT and BEST FIT algorithms belong to the class of ANY FIT. The current best-known lower and upper bounds for the classical online bin packing problem are 1.54 due to van Vliet [11] and 1.588 by Seiden [9], respectively.

A generalization of our problem is scheduling with release times and deadlines on a minimum of machines (SRDM) [2]. Each job has a release time, a processing time and a deadline, the goal is to find a nonpreemptive schedule such that all jobs meet their deadlines and the number of machines needed to process all jobs is minimum. They concerned on the off-line case and showed that this problem is at least $\Theta(\log n)$-approximation, where n is the total number of jobs. For the special case with equal processing times, a nine-approximation algorithm was presented. They also presented an asymptotic 4.62-approximation algorithm when all jobs have equal release times. Our problem becomes a special case of the SRDM problem if all jobs have equal deadlines. Another similar problem called *strip packing* with release time was studied by Augustine et al. [1]. They provided an asymptotic fully polynomial time approximation scheme for the offline strip packing with release time.

Our contributions: To the best of our knowledge, we are the first one to study the online bin packing problem with release times. We assume that all items have equal size. The detail of our results are given as follows:

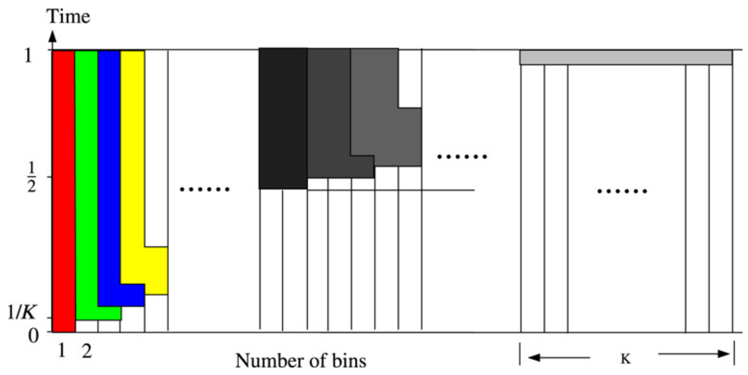


Fig. 1. ANY FIT packing of instance I_0.

(1) There is no constant competitive ratio for the ANY FIT algorithm.
(2) We show that no online algorithm can achieve the competitive ratio less than two.
(3) We present an on-line algorithm with competitive ratio two.

The rest of this paper is organized as follows. Section 2 analyses the lower bound of ANY FIT algorithms. Section 3 shows the general lower bound two and presents an online algorithm with competitive ratio two. The conclusions will be given in Section 4.

2. Lower bound of the ANY FIT algorithm

In this section we show an unbounded lower bound for the ANY FIT algorithm. Note that for the AF algorithm, a new bin is opened only if the item does not fit into any nonempty bin.

Theorem 1. *There is no constant competitive ratio for the ANY FIT algorithm.*

Proof. We construct an instance I_0 to show the theorem. First, we assume that all items have the same size $\frac{1}{K}$, where K is a sufficiently large integer. K^2 items arrive in turn : K items with release time 0, K items with release time $\frac{1}{K}$, K items with release time $\frac{2}{K}, \dots,$ K items with release time $\frac{K-1}{K}$. This instance I_0 will also be used in the rest of this paper.

Clearly, the optimal solution uses K bins, where each item is located exactly at its release time. Now we observe the assignments by the ANY FIT algorithm, see the Fig. 1 for an illustration, each colour represents a set of items with the same release time.

The first K items are packed into the first bin, i.e. only one bin is used. For the second K items, $K-1$ items are in the second bin and one item in the third bin, i.e. use $\frac{K}{K-1}$ bins. Because of the ANY FIT algorithm, when a new bin is opened, no subsequent items can be fitted into a bin with index smaller than this opened new bin. For the K items with release time x/K, at least $\lfloor\frac{1}{1-x/K}\rfloor$ bins are needed. For the last K items with release time $\frac{K-1}{K}$, each piece shall occupy a bin, i.e. total K new bins are needed to pack the last K items. Consequently, the number of bins used by the algorithm AF is at least $\frac{K}{K} + \lfloor\frac{K}{K-1}\rfloor + \cdots + \lfloor\frac{K}{2}\rfloor + K \geq (\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{K})K = K(\log K - 1)$. Then the competitive ratio of the ANY FIT algorithm is at least

$$R_{\mathrm{AF}} \geq \frac{\mathrm{AF}(I)}{\mathrm{OPT}(I)} = \frac{K(\log K - 1)}{K} = \log K - 1.$$

Hence there is no constant competition for the ANY FIT algorithm when K approaches infinity. ■

Note that the above instance I_0 is for the model only with unit item size. Thus one question arises: does not there exist an online algorithm with constant competitive ratio for this case of unit item size? We will give a positive answer in the following section.

3. Optimal online algorithm

3.1. General lower bound

In this section we show that one cannot expect to find an online algorithm with asymptotic competitive ratio less than two. Note that the classical bin-packing problem is solved in polynomial time if all items have equal size, and the ANY FIT algorithm produces an optimal solution for that model. Thus the additional release time of each item brings much more trouble to our investigated problem.

Theorem 2. *For any online algorithm A, $R_A^\infty \geq 2$.*

Proof. Suppose that R is the competitive ratio of online algorithm A. For an integer $k \geq 3$, the sequence of items generated by the adversary is given as follows. First, n items with release time 0 and all have item size $1/k$. Then for each i, n items with release time i/k and all have item size $1/k$, are presented one by one, where $i = 1, \dots, k-1$.

After the first n items, the cost of the optimal solution is n/k. So, algorithm A uses at most Rn/k bins, which implies that there are at most Rn/k items packed at positions 0 in these bins. So on for other items with release times i/k, the cost of the optimal solution is in/k. Thus algorithm A uses at most iRn/k bins, which implies that there are at most iRn/k items packed at positions $\frac{i-1}{k}$ in these bins. So, summing the arithmetic sequence, the total number of packed items at the end is at most $\sum_{i=0}^{k-1}(i+1)Rn/k = R(k+1)n/2$. But this must be at least kn, i.e. $R(k+1)n/2 \geq kn$ yielding $R \geq 2k/(k+1)$.

Thus, the asymptotic competitive of any on-line algorithm is at least two by choosing a sufficiently large integer k. ∎

3.2. Upper bound

In this section, we present an optimal on-line algorithm for the problem with unit size, i.e. all the items have the same size $1/K$ for some integer K, therefore we can assume that the release times of all items are at $\{i/K \mid i = 0, 1, \dots, K-1\}$.

Main idea: From the analysis of the lower bound of the algorithm AF, we observe that it is not good only packing items into a nonempty bin. Thus the key point in design of an efficient algorithm is to find a suitable rule for when a new bin will be opened.

1. One is to balance the height of all open bins, but not average the load (capacity);
2. The second is to locate the item just at the position of its release time as far as possible;
3. The third is to bound the number of new opened bins to control the ratio.

Before designing our algorithm, some notations are given as follows. When packing a new item a_j, suppose that the optimal solution for the first j items is $\mathrm{OPT}_j = M$, then denote by l_i^M the number of items with release time $i/K (i = 0, 1, \dots, K-1)$. Let $L_i^M = \sum_{j=i}^{K-1} l_j^M \leq (K-i)M$. Clearly, we have the following lemma due to the capacity constraint of any optimal solution:

Lemma 3.

$$M \geq \max_{0 \leq i \leq K-1} \left\{ \frac{L_i^M}{K-i} \right\}.$$

Algorithm $\mathrm{AH}_C\{\frac{1}{K}\}$**(Average Height):** $(C \geq 2)$

For any new item a_j with release time r_j, we pack it as follows:

Step 1 Calculate the optimal solution $\mathrm{OPT}_j = M = \lceil \max_{0 \leq i \leq K-1}\{\frac{L_i^M}{K-i}\} \rceil$.
Step 2 If the optimal solution increases, i.e. $\mathrm{OPT}_j > \mathrm{OPT}_{j-1}$, open C bins.
Step 3 Find the lowest indexed bin so that the item a_j is located as low as possible in all open bins.

From our algorithm, we always first put an item a_j into a bin with lowest index so that it can be located at r_j or above r_j, then it is easy to obtain the following fact.

Fact 4. *If there is a vacancy at j/K in the bin B', there must be a vacancy at j/K in the nonempty bins with index larger than B'. Furthermore, if there is no vacancy at j/K in the bin B', there must be no vacancy at j/K in the bins with index smaller than B'.*

Here we want to find the minimum constant C so that all items can be packed into $C \cdot \mathrm{OPT}_n$ bins. Then the competitive ratio of the algorithm $\mathrm{AH}_C\{\frac{1}{K}\}$ is at most C.

Lemma 5. *For any instance I, if there are two adjacent items a_j and a_{j+1} with release times $r_j > r_{j+1}$, then the cost of $\mathrm{AH}_C\{\frac{1}{K}\}$ will not be reduced. Moreover, the instance I_0 generated in the proof of Theorem 1 is one of the worst instances by $\mathrm{AH}_C\{\frac{1}{K}\}$.*

Proof. Without loss of generality, we assume that all items of the instance I can be packed into $C \cdot \mathrm{OPT}_n$ bins by $\mathrm{AH}_C\{\frac{1}{K}\}$. Denote I' to be the instance I after interchanging the items a_j and a_{j+1}. For instance, I', after a_{j+1} has been assigned, the optimal solution is denoted by OPT'_j and the optimal solution after a_j is OPT'_{j+1}. According to the possibilities of the change of the optimal solution when a_j and a_{j+1} arrive, we just need to consider three cases.

• Case 1: $\mathrm{OPT}_{j-1} = \mathrm{OPT}_j = \mathrm{OPT}_{j+1}$.

∘ Case 1.1: The two items are both located exactly at their release times in the instance I. It implies that when a_j comes there are vacancies at r_j and r_{j+1}. For instance, I', their locations will not be changed by $\mathrm{AH}_C\{\frac{1}{K}\}$.

∘ Case 1.2: Only a_j is located at its release time in the instance I.

-Subcase 1.2.1: a_{j+1} is located below r_j. After interchanging of the two items a_j and a_{j+1}, their locations will not change.

-Subcase 1.2.2: a_{j+1} is located at or above r_j. Consider the instance I', the locations of a_j and a_{j+1} will interchange comparing to their locations in the instance I. This will not affect the locations of subsequential items.

∘ Case 1.3: a_j is located above r_j in the instance I. It means there is no vacancy between r_j and the location of a_j in any open bin. Similarly with *case 1.2*.

-Subcase 1.3.1: a_{j+1} is located below r_j. After interchanging of the two items a_j and a_{j+1}, their locations will not change.

-Subcase 1.3.2: a_{j+1} is located above r_j. After interchanging of the two items a_j and a_{j+1}, their locations will interchange comparing to their locations in the instance I. This will not affect the locations of subsequential items, too.

• Case 2: $\mathrm{OPT}_{j-1} = \mathrm{OPT}_j < \mathrm{OPT}_{j+1}$. In this case, C bins will be opened before assigning a_{j+1}.

∘ Case 2.1: a_{j+1} isn't packed into these C new bins. So, it must be located at its release time. Consider the instance I', their locations will not change.

∘ Case 2.2: a_{j+1} is packed into the first of these C new bins. There is no vacancy just at r_{j+1} in the open bins before a_{j+1} comes. For instance I', we have $\mathrm{OPT}'_{j-1} = \mathrm{OPT}'_j = \mathrm{OPT}_j < \mathrm{OPT}'_{j+1}$.

-Subcase 2.2.1: a_j is located at its release time in the instance I. The detail proof will be given in the Appendix A. For instance I', the solution by the algorithm $\mathrm{AH}_C\{\frac{1}{K}\}$ either will not change, or will increase by one.

-Subcase 2.2.2: a_j is located above r_j in the instance I. similar to the analysis in *Subcase 2.2.1*. For instance I', the solution by $\mathrm{AH}_C\{\frac{1}{K}\}$ either will not change, or will increase by one.

• Case 3: $\mathrm{OPT}_{j-1} < \mathrm{OPT}_j$. Details are given in Appendix B.

From the above all, by interchanging such items a_j and a_{j+1} with $r_j > r_{j+1}$ in the stance I, we obtain that an instance with release times in nondecreasing order will achieve the worst solution by the algorithm $\mathrm{AH}_C\{\frac{1}{K}\}$, i.e. for the instances with the same items and but with different arrival sequences, the instance with release times in nondecreasing order will use the largest number of bins by $\mathrm{AH}_C\{\frac{1}{K}\}$. However, their optimal solutions are the same. Hence, the instance I_0 generated in the proof of Theorem 1 is one of the worst instances. ■

Theorem 6. *The competitive ratio of algorithm $\mathrm{AH}_C\{\frac{1}{K}\}$ is 2 if we let $C = 2$.*

Proof. By Lemma 5, the instance I_0 is one of the worst instances, and the optimal solution of I_0 is K. First, we construct the configuration of the solution for I_0 by $\mathrm{AH}_C\{\frac{1}{K}\}$. Afterwards we analyse the other worst instance with release times in nondecreasing order.

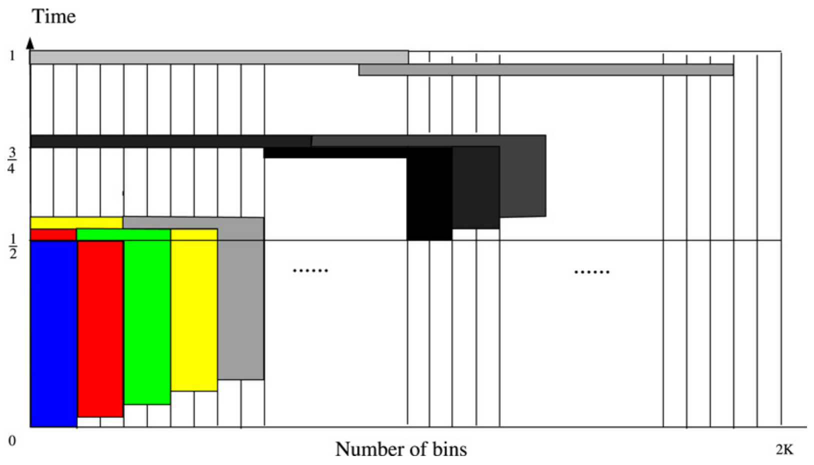


Fig. 2. $AH_2\{\frac{1}{K}\}$ packing for even K.

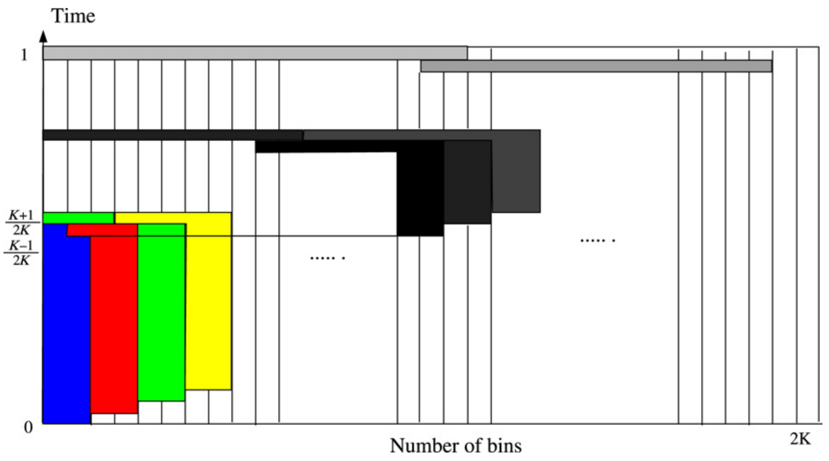


Fig. 3. $AH_2\{\frac{1}{K}\}$ packing for odd K.

Let $C = 2$. See Fig. 2 for a demonstration of the packing instance I_0 when K is even.

The first K items are packed into two bins under $\frac{1}{2}$. $K - 2$ items of the second K ones are packed into the third and fourth bins under $\frac{1}{2}$, and each of the other two items is put into the first two bins at $\frac{1}{2}$. For any K items with release times $i/K (i = 1, 2, \ldots, K - 1)$, consider the following cases:

(a) if i is odd, suppose $i = 2h - 1$, $K - 2h$ items of them are packed into the $(2i + 1)$-th and $(2i + 2)$-th bins under $\frac{1}{2} + \frac{h-1}{K}$, and each of the other $2h$ of them is located at $\frac{1}{2} + \frac{h-1}{K}$ in the former $2h$ bins;

(b) i is even, assume $i = 2h$, $K - 2h - 2$ items of them are packed into the $(2i + 1)$-th and $(2i + 2)$-th bins under $\frac{1}{2} + \frac{h-1}{K}$, and each of the other $2h + 2$ of them is located at $\frac{1}{2} + \frac{h-1}{K}$ in the later $2h + 2$ bins.

For the last K items whose release times are $\frac{K-1}{K}$, $(K - 1)$ is odd. These K items are just located at $\frac{K-1}{K}$ in the former K bins and total $2K$ bins are opened.

If K is odd, see the Fig. 3. For the first K items, $\frac{K+1}{2}$ items are packed into the first bin and $\frac{K-1}{2}$ items are packed into the second bin. Similar to the above analysis, when K items comes with release times $i/K (i = 1, 2, \ldots, K - 1)$, we consider the following cases:

(a) if i is odd, set $i = 2h - 1$, $K - 2h - 1$ items of them are packed into the $(2i + 1)$-th and $(2i + 2)$-th bins under $\frac{K-1}{2K} + \frac{h-1}{K}$, and each of the other $2h + 1$ of them is located at $\frac{K-1}{2K} + \frac{h-1}{K}$ in the later $2h + 1$ bins.

(b) i is even, set $i = 2h$, $K - 2h - 1$ items of them are packed into the $(2i + 1)$-th and $(2i + 2)$-th bins under $\frac{K-1}{2K} + \frac{h}{K}$, and each of the other $2h + 1$ of them is located at $\frac{K-1}{2K} + \frac{h}{K}$ in the former $2h + 1$ bins.

For the last K items with release time $\frac{K-1}{K}$, $(K - 1)$ is even. These K items are just located at $\frac{K-1}{K}$ in the former K bins of all the $2K$ ones.

From the above analysis, the last two bins are empty. But, if there are only first $P(P < K)$ batches of items with P different release times, the optimal solution is P while the solution of $\mathrm{AH}_2\{\frac{1}{K}\}$ is $2P$ and none of the bins is empty. Thereby the ratio is two.

Above all, we have discussed the case that the number of items with the same release time are all equal to K, i.e. $l_i = K, i = 0, 1, \dots, K - 1$. Let $\Omega = \max_{0 \le i \le K-1}\{\frac{\Sigma_{j=i}^{K-1} l_j}{K-i}\}$. We get that $\mathrm{OPT} = \lceil \Omega \rceil$ by Lemma 3. Note that in the above case that $\mathrm{OPT} = \Omega = K$ is an integer. Consequently, for the case that $\mathrm{OPT} = \Omega$ is an integer and $l_i = \theta K, i = 0, 1, \dots, K - 1$(where θ is an integer), the configuration of the algorithm $\mathrm{AH}_2\{\frac{1}{K}\}$ is similar. Then we turn to the case that $l_i = \theta K - \sigma$, where σ is a positive integer less than K, and $i = 0, 1, \dots, K - 1$. By the algorithm, the first l_0 items will be assigned into 2θ bins, and the number of the vacancies is larger than it is in the case of $l_i = \theta K$ after assigning the first l_0 items. Hence all items can be packed into 2OPT bins. This completes the proof of the theorem. ■

In the following, we give an improved algorithm $\mathrm{MAH}_C\{\frac{1}{K}\}$ with tight competitive ratio $2 - \frac{1}{K}$.

Algorithm $\mathrm{MAH}_C\{\frac{1}{K}\}$**:**
For any new item a_j with release time r_j, we pack it as follows:

```
Step 1  Calculate the optimal solution $\mathrm{OPT}_j = M = \lceil \max_{0 \le i \le K-1}\{\frac{L_i^M}{K-i}\} \rceil$.
Step 2  If $\mathrm{OPT}_j = 1$, put the item into the first bin.
Step 3  If $\mathrm{OPT}_j \ge 2$ and $\mathrm{OPT}_j > \mathrm{OPT}_{j-1}$, open new $C$ bins.
Step 4  Find the lowest indexed bin so that the item $a_j$ is located as low as possible in all open bins.
```

Similar as the proof of Theorem 6, we get the following theorem, the detail proof is given in Appendix C.

Theorem 7. *Let $C = 2$, the competitive ratio of algorithm* $\mathrm{MAH}_C\{\frac{1}{K}\}$ *is* $2 - \frac{1}{K}$.

4. Conclusions

In this paper we have studied the online bin packing problem with release times. For the problem with all items have equal size, we presented a general lower bound two and designed an online algorithm with asymptotic competitive ratio two. We also showed that the ANY FIT algorithm cannot be approximated within any constant asymptotic competitive ratio.

If we turn to the offline case of our problem, all the classical online bin packing algorithms can be applied to our problem by sorting the items in the order of nonincreasing release times and all items are packed to a bin as near the top as possible.

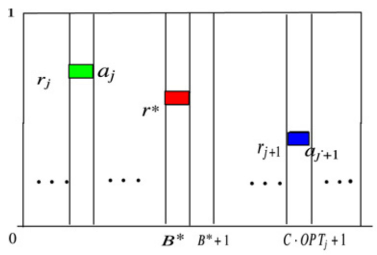
There are many challenging aspects of the bin packing with release time. For example, the online problem with items having different sizes, and the offline problem whether there exists a full polynomial time approximation scheme.

Acknowledgements

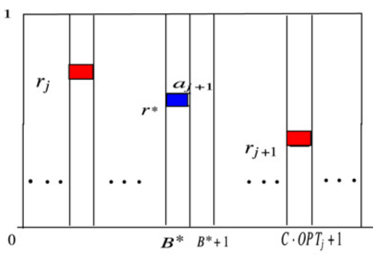
We would like to thank anonymous referees for many helpful suggestions to design a better general lower bound and for many comments and suggestions that improved the presentation of this paper. The second author's research was supported by NSFC (10601048) and 973 (2007CB310906).

Appendix A. The proof of subcase 2.2.1 in Lemma 5

In this subcase , a_{j+1} is located at r^* between r_{j+1} and r_j in the bin B^* in the instance I'. And the position for a_{j+1} before reversing is now empty. See the configuration by $\mathrm{AH}_C\{\frac{1}{K}\}$ for I by and I' just after the item a_{j+1} be assigned in the following figure (where the "red" area is empty) .



The Configuration of I by $AH_C\{\frac{1}{K}\}$ after a_{j+1} arrived



The Configuration of I' by $AH_C\{\frac{1}{K}\}$ after arrived a_{j+1} before a_j

We consider two cases which may lead to the change of the solution after reversing: one is that in the instance I, the position $\frac{K-1}{K}$ in the last bin is filled, which may lead to the increase of the solution after reversing. Denote this situation by S_I; the other is in the instance I, only one item is packed into the last bin and it is not at $\frac{K-1}{K}$, which may lead to the decrease of the solution after reversing. Denote this situation by S_{II}. For other cases, the solution will not change after reversing the orders of any two items.

(1) If r^* is still free after the assignment of the last item in I.

(α) If the position for a_{j+1} in the instance I is always empty in the instance I', this position is in the $\mathrm{C}\cdot\mathrm{OPT}_j+1$ bin and the same position in the bins after this one are empty. So, the configuration of items after a_{j+1} should not change. The solution of $\mathrm{AH}_C\{\frac{1}{K}\}$ holds the line. See the Fig. 4, only the location of a_{j+1} changed.

(β) If the position for a_{j+1} in the instance I is packed by an item a_k in the instance I'. See the Fig. 5. Since they are empty at r^* in the bins with index larger than B^* in the instance I, the configuration above r^* in all bins in the instance I' will not change. Only the packing of the items in the "gray" area may change, an vacancy increase. Then, for the situation S_I, the number of used bins doesn't change. For the situation S_{II}, the release time of the exclusive item in the last bin must larger than r^*, or it should be packed in a certain bin before. The number of used bins by these two instances doesn't change.

(2) If r^* will be used by some subsequential item a^* in I.

(α) If the position for a_{j+1} in the instance I is always empty in the instance I', it means the items come later with release time less than r_{j+1} could not change their locations in the instance I'. And from the algorithm, for the instance I', the item a^* should be packed at r^* in the bin B^*+1 . If there is a vacancy above or at r^* after the assignment of the last item in the instance I, for the situation S_I and S_{II}, the number of used bins will not make a change for the instance I'. If there is no vacancy above or at r^* after the assignment of the last job in the instance I, which happens only in the situation S_I, see the Fig. 6, because the "grey" area is full before reversing, while after reversing the item a^* occupies one vacancy for some item which cannot be assigned in opened bins except the "grey" areas on, there must be an excessive item which was assigned in the "grey" area before reversing and could not be packed into the used bins after reversing. Then a new bin should be opened for it in the instance I'. Hence the solution of algorithm $\mathrm{AH}_C\{\frac{1}{K}\}$ for the instance I' increases 1.

(β) If the position for a_{j+1} in the instance I is packed by an item a_k in the instance I'. Similarly , if there is a vacancy above or at r^* after all items have been assigned in the instance I, for the situation S_I , the number of used bins doesn't change for the instance I'. Suppose this vacancy to be at r'', for the situation S_{II}, the release time of the exclusive item in the last bin must larger than r'', or it should be packed in a certain bin before. The number of used bins doesn't change, either. Otherwise, which shall happen only in the case S_I with no vacancy above or at r^*? See the Fig. 7, the "heavy grey" area is full, only the items packed in the heavy and soft "grey" area may change their locations. In the "soft grey", a vacancy increases, while in the "heavy grey" area, there must be an excessive item which could not be packed into the used bins and a new bin should be opened for the instance I'. Hence the solution of algorithm increases by one.

In a word, in the subcase 2.2.1, the solution of the algorithm $\mathrm{AH}_C\{\frac{1}{K}\}$ for the instance I' will increase by one or does not decrease. ■

Appendix B. The proof of Case 3 in Lemma 5

In this case, we prove that after interchanging two items, the solution of $\mathrm{AH}_C\{\frac{1}{K}\}$ does not decrease.

(1) $\mathrm{OPT}_j < \mathrm{OPT}_{j+1}$. We have

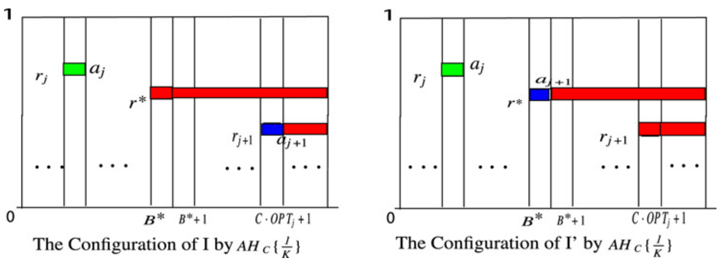


Fig. 4. The configuration transform in (1) (α).

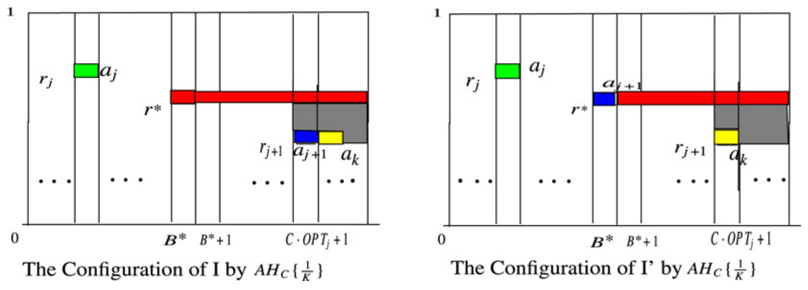


Fig. 5. The configuration transform in (1) (β).

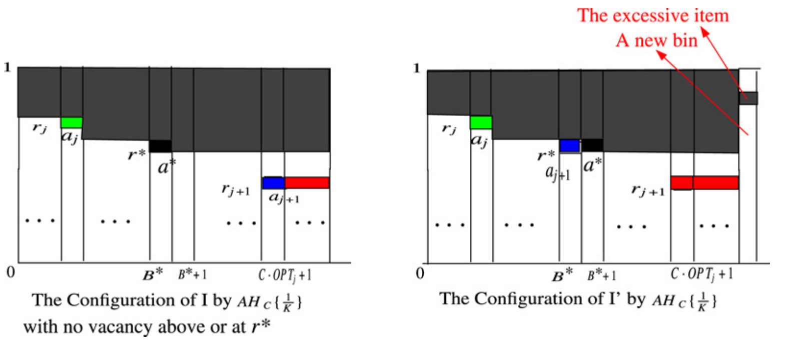


Fig. 6. The configuration transform in (2) (α) with no vacancy above or at r^*.

$$\mathrm{OPT}_{j-1} < \mathrm{OPT}'_j = \mathrm{OPT}_j < \mathrm{OPT}'_{j+1} = \mathrm{OPT}_{j+1}.$$

The two items a_j and a_{j+1} are packed at their release times, respectively, in both instances I and I'. Consequently, the solutions of these two instances are the same.

(2) $\mathrm{OPT}_j = \mathrm{OPT}_{j+1}$.

(α) $\mathrm{OPT}_{j-1} < \mathrm{OPT}'_j = \mathrm{OPT}_j$, again the two items a_j and a_{j+1} are packed at their release times in both instances, respectively. Hence, the solutions don't change.

(β) $\mathrm{OPT}_{j-1} = \mathrm{OPT}'_j < \mathrm{OPT}_j$. For instance I, the two items a_j and a_{j+1} are packed at their release times, respectively. For instance I', assume that the item a_{j+1} is packed at $r^* \geq r_{j+1}$ in the bin B^*. If $r^* = r_{j+1}$, the locations of a_j and a_{j+1} will not change. If $r^* > r_{j+1}$ the analysis is similar as *Subcase 2.2.1*. The solution of algorithm $\mathrm{AH}_C\{\frac{1}{K}\}$, for instance, I' is not smaller than the solution of algorithm $\mathrm{AH}_C\{\frac{1}{K}\}$, for instance, I in the case S_{II} . ■

Appendix C. Proof of Theorem 7

Proof. The conclusion is trivial if $K = 1$. When $K \geq 2$, from the Fact 4 and Lemma 5, I_0 is still one of the worst instances. Similar analysis as Theorem 6, we can get this theorem. We consider the case when K is even. The case that K is odd can be proved by analogy. Let $C = 2,\ K = 2m$. See the Fig. 8 for an illustration of packing I_0 by the algorithm $\mathrm{MAH}_C\{\frac{1}{K}\}$.

The first K items are packed into the first bin. When the K items comes with release time $i/K (i = 1, 2, \ldots, K-1)$,

(a) if i is odd, set $i = 2h - 1 (h = 1, 2, \ldots, m)$, $K - 2h$ items of them are packed into the $2i$-th and $(2i + 1)$-th bins under $\frac{1}{2} + \frac{h-1}{K}$, and each of the other $2h$ of them is located at $\frac{1}{2} + \frac{h-1}{K}$ in the later $2h$ bins;

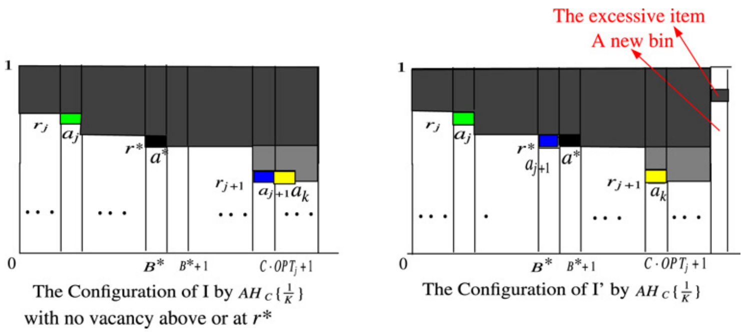

The excessive item
A new bin
1
0
r_j
a_j
r^*
a^*
r_{j+1}
a_{j+1}
a_k
B^*
B^*+1
$C \cdot OPT_j+1$
The Configuration of I by $AH_C\{\frac{1}{K}\}$ with no vacancy above or at r^*
The Configuration of I' by $AH_C\{\frac{1}{K}\}$

Fig. 7. The configuration transform in (2) (β) with no vacancy above or at r^*.

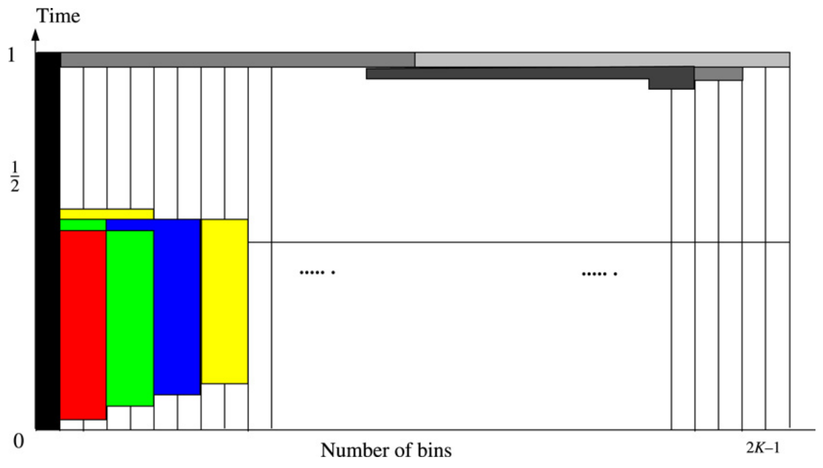


Fig. 8. $\mathrm{MAH}_2\{\frac{1}{K}\}$ packing for even K.

(b) i is even, set $i = 2h(h = 1, \dots, m-1)$, $K - 2h$ items of them are packed into the $2i$-th and $(2i+1)$-th bins under $\frac{1}{2} + \frac{h}{K}$, and each of the other $2h$ of them is located at $\frac{1}{2} + \frac{h}{K}$ in the bins from the $2rd$ to the $(2h+1)$-th.

Since the last K items's release times are $\frac{K-1}{K}$ and $(K-1)$ is odd, these K items are just located at $\frac{K-1}{K}$ in the last K bins. Thus total $2K-1$ bins are used.

The remaining case that there exist many kinds of items with the same release times could be proved analogously as Theorem 6. ■

References

[1] J. Augustine, S. Banerjee, S. Irani, Strip packing with precedence constraints and strip packing with release times, in: Proceedings of the Eighteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, 2006, pp. 180–188.

[2] M. Cieliebak, T. Erlebach, F. Hennecke, B. Weber, P. Widmayer, Scheduling with release times and deadlines on a minimum number of machines, in: IFIP TCS, 2004, pp. 209–222.

[3] E.G. Coffman Jr., G. Galambos, S. Martello, D. Vigo, Bin packing approximation algorithms: Combinatorial analysis, in: D.-Z. Du, P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, Kluwer Academic Publishers, 1998.

[4] E.G. Coffman Jr., M.R. Garey, D.S. Johnson, Bin packing approximation algorithms: A survey, in: D.S. Hochbaum (Ed.), Approximation Algorithm for NP-Hard Problems, PWS, 1996, pp. 46–93.

[5] J. Csirik, G.J. Woeginger, On-line packing and covering problems, in: A. Fiat, G.J. Woeginger (Eds.), On-Line Algorithms — The State of The Art, in: Lecture Notes in Computer Science, vol. 1442, 1996, pp. 147–177.

[6] M.R. Garey, D.S. Johnson, Computers and Intractability (A Guide to The Theory of NP-Completeness), W. H. Freeman and Company, San Francisco, 1979.

[7] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, R.L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM Journal on Computing 3 (1974) 256–278.

[8] R. Li, H.-C. Huang, On-line scheduling for jobs with arbitrary release times, Computing 73 (2004) 79–97.

[9] S. Seiden, On the online bin packing problem, Journal of the ACM 49 (2002) 640–671.

[10] J.D. Ullman, The performance of a memory allocation algorithm, Technial Report 100, Princeton University, Princeton, NJ, 1971.

[11] A. van Vliet, An improved lower bound for on-line bin packing algorithms, Information Processing Letters 42 (1992) 277–284.