Discrete Optimization 21 (2016) 14–24

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Lower bound for 3-batched bin packing☆

János Balogh[a,*], József Békési[a], Gábor Galambos[a], György Dósa[b], Zhiyi Tan[c]

[a] *Department of Applied Informatics, Gyula Juhász Faculty of Education, University of Szeged, H-6701 Szeged, POB 396, Hungary*

[b] *Department of Mathematics, University of Pannonia, H-8200 Veszprém, Egyetem u. 10, Hungary*

[c] *Department of Mathematics, College of Science, Zhejiang University, 310027 Hangzhou, Zhejiang, PR China*

ARTICLE INFO

Article history:
Received 4 September 2015
Received in revised form 21 April 2016
Accepted 25 April 2016
Available online 30 May 2016

Keywords:
Batched bin packing
Worst-case behavior
Lower bound

ABSTRACT

In this paper we will consider a special relaxation of the well-known *online bin packing problem.* In a *batched bin packing problem* (BBPP) – defined by Gutin et al. (2005) – the elements come in batches and one batch is available for packing in a given time. If we have $K \geq 2$ batches then we denote the problem by K-BBPP. In Gutin et al. (2005) the authors gave a 1.3871... lower bound for the asymptotic competitive ratio (ACR) of any on-line 2-BBBP algorithm. In this paper we investigate the 3-BBPP, and we give 1.51211... lower bound for its ACR.

1. Introduction

In the classical one-dimensional bin packing problem a list L of $a_1, a_2, \ldots, a_n$ elements from the interval $(0, 1]$ is given, and we want to assign each item to a unit capacity bin. The objective is to minimize the number of bins. In case of *online problems* the input is not known completely in advance: items come one by one, and an *online algorithm* assigns them to a bin immediately, irrevocably. These online algorithms have been studied widely in last decades. Their performance may be measured by the asymptotic performance ratio, which is defined as follows.

For a list L, let $\mathrm{OPT}(L)$ denote the number of bins in an optimal packing and let $A(L)$ denote the number of bins that algorithm A uses for packing L. If $R_A(N)$ denotes the supremum of the ratios $A(L)/\mathrm{OPT}(L)$ for all lists L with $\mathrm{OPT}(L) = N$, then the asymptotic competitive ratio (ACR) is defined as

$$R_A^{\infty} := \limsup_{N \to \infty} R_A(N).$$

☆ This research was supported by the Chinese-Hungarian bilateral project TÉT_12_CN-1-2012-0028.

* Corresponding author.
E-mail addresses: balogh@jgypk.u-szeged.hu (J. Balogh), bekesi@jgypk.u-szeged.hu (J. Békési), galambos@jgypk.u-szeged.hu (G. Galambos), dosagy@almos.vein.hu (G. Dósa), tanzy@zju.edu.cn (Z. Tan).

http://dx.doi.org/10.1016/j.disopt.2016.04.007
1572-5286/

From asymptotic point of view the best known on-line algorithm is due to Heydrich and van Stee [1] with asymptotic performance ratio at most 1.5815, while the best lower bound is 1.54037 given by Balogh et al. in [2].

The algorithms can be distinguished from each other according to the number of bins can be used while packing the actual item. If the algorithm can use only one open bin then we get the *Next Fit* algorithm for which $R_{NF}^{\infty} = 2$. If we keep open every bin while packing the actual item we can define several algorithms. Among these the most known are the *First Fit* and the *Best Fit*. These algorithms put the element into the first bin into it fits, or into that bin where the item fits the best, respectively. For these algorithms $R_{FF}^{\infty} = R_{BF}^{\infty} = 17/10$, (see [3]). The so-called bounded space algorithms were introduced by Lee and Lee [4]. They defined the *Harmonic Fit* algorithm, which gets the elements online, but only limited number of bins are available to put the item. They proved that $R_{HF}^{\infty} = 1.69103\ldots$.

To relax the strict online condition several relaxations have been investigated. The so-called *lookahead algorithms* were considered by Grove [5]. A k-bounded lookahead algorithm delays to pack an element until the next $k-1$ items arrive. Grove proved that the $1.69103\ldots$ ACR is achievable by these algorithms.

An updated overview on the state of art of the bin packing algorithms can be found in [6].

In this paper we are dealing with another relaxation of the online problem. The *batched bin packing problem* (BBPP) was defined by Gutin et al. (see [7]): the elements come in batches and one batch is available for packing in a given time. Each batch may contain different sizes of items, and any batch can be empty. If we have $K \geq 2$ batches then we speak about K-BBPP.

A *batched algorithm* packs the batch completely before the next batch arrives. It is clear that if each batch contains one element, then we have the classical online problem, and if only one batch is coming then the problem is the general (offline) one-dimensional BBPP.

Let us consider an input sequence L, which is a *batched sequence*, i.e. $L = \{B_1, B_2, \ldots, B_K\}$, where B_j is a set of elements, $1 \leq j \leq K$. The set of all batched sequences exactly with K batches is denoted by $\mathcal{B}(K)$. Let A be a batched algorithm, then for BBPP the ACR is defined as follows.

$$R_{A,K}^{\infty} := \limsup_{N\to\infty} \left\{ \frac{A(L)}{\mathrm{OPT}(L)} : L \in \mathcal{B}(j),\ j \leq K,\ \mathrm{OPT}(L) = N \right\}.$$

In [7] the authors gave a $1.3871\ldots$ lower bound for the ACR of any on-line 2-BBPP algorithm. It is clear that

$$R_{A,i}^{\infty} \leq R_{A,j}^{\infty} \leq \cdots, \quad \text{if } 1 \leq i < j < \infty,$$

and such an A algorithm is interesting for which $R_{A,i}^{\infty} < 1.5815$ holds, where 1.5815 is the best known upper bound for the one-dimensional online bin packing algorithms [1]. In a recent paper [8] Dósa published an upper bound of $\frac{19}{12} = 1.58333\ldots$ for 2-BBPP.

In this paper we investigate the case 3-BBPP, and we give a $1.51211\ldots$ lower bound for its ACR. The structure of the paper is the following. First we define a concatenated list of batches which we will use to prove a lower bound and we give tight bounds for the optimal packing of the batches of given instances. In Section 3 we filter those algorithms which are working bad on our instance. Section 4 investigates the possible strategies of the 3-BBPP algorithms. To simplify our later discussions we make some reductions in Section 5 on packing patterns of the algorithms under investigation. In Section 6 we introduce an LP model to get the desired lower bound for $R_{A,3}^{\infty}$. Based on our LP model, in Section 7 we give our lower bound for the ACR of any 3-BBPP algorithm. We determine this bound as a solution of a linear optimization problem, and we use theoretical analysis. We close the paper with some conclusions and suggestions for the future research.

2. Construction for $K = 3$ batches

The construction is the following:

- The first batch B_1 contains $n_1 = 6jn$ pieces of *small items* – denoted by a_1 – with equal sizes. Let $j \geq 4$ be a fixed integer, then the size of each element in the batch is $s(a_1) = 1/6j = \varepsilon$.
- In the second batch one of the lists $B_{2,k}$ will be given to be packed. List $B_{2,k}$ contains $n_{2,k} = \frac{6j}{j-k}n$ pieces of $a_{2,k}$ *medium items* with size $s(a_{2,k}) = \frac{1}{3} + k\varepsilon - \frac{\varepsilon}{3} = \frac{1}{3} + \varepsilon\frac{3k-1}{3}$, and $1 \leq k \leq j-1$.
- The third batch $B_{3,k}$ follows the batch $B_{2,k}$. The number of elements in $B_{3,k}$ depends on the second batch. If we pack in the second step the list $B_{2,k}$ then $B_{3,k}$ contains $n_{3,k} = \frac{6j}{j-k}n$ pieces of a_3 *large items* with sizes $s(a_3) = \frac{1}{2} + \frac{\varepsilon}{3}$.

To understand this structure it is important to see: the kth list among the second batches and the third batch with $n_{3,k}$ elements belonging to $B_{2,k}$ form an inseparable "couple". If the list $B_{2,k}$ has been chosen then $B_{3,k}$ contains $n_{3,k}$ items. To get a lower bound we investigate those three batches which consist of the concatenated lists $(B_1, B_{2,1}, B_{3,1})$, $(B_1, B_{2,2}, B_{3,2}), \ldots, (B_1, B_{2,j-1}, B_{3,j-1})$. In fact, we will prove later, that we do not need to take into account all of these batches. It is enough to consider only a part of them. It is obvious that $\mathrm{OPT}(B_1) = n$, and the following lemmas are also true.

Lemma 2.1. *For any k, $1 \leq k \leq j-1$, $OPT(B_1, B_{2,k}) = \frac{3j}{j-k}n$.*

Proof. Since $\frac{1}{3} < s(a_{2,k}) \leq \frac{1}{2}$, so packing only the items of the second batch we need at least $\frac{n_{2,k}}{2} = \frac{3j}{j-k}n$ bins. Therefore

$$\mathrm{OPT}(B_1, B_{2,k}) \geq \frac{3j}{j-k}n. \tag{1}$$

Let us pack the medium items of $B_{2,k}$ into bins, 2 in each bin. Then every bin will have

$$1 - 2s(a_{2,k}) = \frac{1}{3} - 2\varepsilon\frac{3k-1}{3}$$

empty space. Since $2(j-k)\varepsilon = \frac{1}{3} - \frac{k}{3j}$ therefore

$$2\varepsilon\frac{3k-1}{3} = \frac{1}{3j}\frac{3k-1}{3} < \frac{k}{3j}.$$

So, we can pack $2(j-k)$ elements from B_1 into each bin which has 2 pieces from the batch $B_{2,k}$. Thus, into the $\frac{3j}{j-k}n$ bins we can put all the $2(j-k)\frac{3j}{j-k}n = 6jn$ items from the batch B_1. Therefore

$$\mathrm{OPT}(B_1, B_{2,k}) \leq \frac{3j}{j-k}n. \tag{2}$$

So, (1) and (2) give together the statement of the lemma. □

Lemma 2.2. *For any k, $1 \leq k \leq j-1$, $OPT(B_1, B_{2,k}, B_{3,k}) = \frac{6j}{(j-k)}n$.*

Proof. The proof is similar to the one in the previous lemma. Any item from the third batch needs its own bin. Therefore

$$\mathrm{OPT}(B_1, B_{2,k}, B_{3,k}) \geq \frac{6j}{j-k}n. \tag{3}$$

Into each bin we can pack a further item from the second batch. As $n_{2,k} = n_{3,k}$, all items from the second batch can be packed into these bins. Now in each bin $1 - a_{2,k} - a_{3,k} = 1/6 - k\varepsilon$ empty space remained. Since

$(j-k)\varepsilon = \frac{1}{6} - \frac{k}{6j}$, $j-k$ items will fit here from the batch B_1. So, all $(j-k)\frac{6j}{j-k}n = 6jn$ items from the batch B_1 can be packed into the existing bins. Therefore

$$\mathrm{OPT}(B_1, B_{2,k}, B_{3,k}) \leq \frac{6j}{j-k}n. \tag{4}$$

From (3) and (4) the statement of the lemma follows. □

Let us denote by $(i_1, i_2, i_3)_1$, $(i_1, i_2, i_3)_{2,k}$, and $(i_1, i_2, i_3)_{3,k}$ the type of a bin after packing the batch B_1, $B_{2,k}$, and $B_{3,k}$, respectively. If a bin is of the type of $(i_1, i_2, i_3)_{2,k}$ then it is obvious that $i_3 = 0$. We call a triplet (i_1, i_2, i_3) as *valid packing pattern* (or *feasible packing-pattern*) if

$$i_1 s(a_1) + i_2 s(a_{2,k}) + i_3 s(a_{3,k}) \leq 1.$$

The set of all feasible packing-patterns will be denoted by V. We define the subsets

$$V_t = \{v \in V \mid i_t > 0 \text{ and } i_r = 0, \text{ for } r < t\}, \quad t = 1, 2, 3.$$

Clearly, $V_t \cap V_r = \emptyset$ if $t \neq r$.

Let us denote by $V_{2,k}$ the set of all opened bins after having packed the medium items of $B_{2,k}$. There will be bins which belong to V_1, and some of them will belong to V_2. Then

$$V_{2,k} = V_1 \cup V_2.$$

We can define similarly $V_{3,k}$, and

$$V_{3,k} = V_1 \cup V_2 \cup V_3.$$

It is important to emphasize that if we open a bin while we pack the small items of the batch B_1, then its type is $(i_1, i_2, i_3)_1$ and it belongs to V_1 i.e. $(i_1, i_2, i_3)_1 \in V_1$. If we put medium items from $B_{2,k}$ into this bin, then $i_2 > 0$, therefore its type will be changed to $(i_1, i_2, i_3)_{2,k}$, and it will belong to $V_{2,k} \supset V_1$ i.e. its type will be $(i_1, i_2, i_3)_{2,k}$, but this bin henceforward will belong to V_1.

Let $x^k_{i_1,i_2,i_3}$ denote the number of bins which contain exactly i_1, i_2, i_3 pieces from the lists $B_1, B_{2,k}, B_{3,k}$, respectively.

To understand the following proofs we remind the reader that among the batches any batch may be empty, so it is possible that after the first (second) batch the 3-batched algorithm will not receive any element.

3. Filtering the less efficient algorithms

Lemma 3.1. *Let A be a batched algorithm. If A packs the elements of B_1 and $B_{2,k}$ batches, and it does not open a new bin while packing the elements of $B_{2,k}$ then $R^\infty_{A,3} \geq 3$.*

Proof. If the condition holds then we know that

$$A(B_1) = A(B_1, B_{2,k}) \geq \mathrm{OPT}(B_1, B_{2,k}) = \frac{3jn}{j-k}.$$

Then

$$R^\infty_{A,3} \geq \frac{A(B_1)}{\mathrm{OPT}(B_1)} = \frac{A(B_1)}{n} \geq \frac{3j}{j-k} \geq 3. \quad \square$$

Lemma 3.2. *Let A be a batched algorithm. If A packs the elements of B_1, $B_{2,k}$ and $B_{3,k}$ batches, and it does not open a new bin while packing the elements of $B_{3,k}$ then $R^\infty_{A,3} \geq 2$.*

Proof. If the condition holds then we know that

$$A(B_1, B_{2,k}) = A(B_1, B_{2,k}, B_{3,k}) \geq \mathrm{OPT}(B_1, B_{2,k}, B_{3,k}) = \frac{6jn}{j-k}.$$

Then

$$R^{\infty}_{A,3} \geq \frac{A(B_1, B_{2,k})}{\mathrm{OPT}(B_1, B_{2,k})} \geq \frac{6jn}{j-k} \cdot \frac{j-k}{3jn} = 2. \quad \square$$

In the remaining part of the paper it is enough to investigate only those batched algorithms for which $R^{\infty}_{A,3} < 2$. On the other hand, from theoretical point of view, we are interested in those algorithms, for which $R^{\infty}_{A,3} \leq 1.6$.

4. Reduction on packing patterns

Lemma 4.1. *Let A be a batched algorithm for which $R^{\infty}_{A,3} \leq 1.6$. After packing the elements of $B_{2,k}$ by A, at least one $(0,2,0)_{2,k}$ type bin must be created.*

Proof. Because of Lemma 3.1, we know that new bins have been opened while the elements of $B_{2,k}$ have been packed. Their type is either $(0,1,0)_{2,k}$ or $(0,2,0)_{2,k}$. If the statement of the lemma is false then all new-opened bins are $(0,1,0)_{2,k}$ type. Let us suppose that after the first batch and after the second batch the algorithm uses y_1 and y_2 bins, respectively. Then for a fixed constant C, $0 < C < \infty$, the following inequalities are true.

$$\mathrm{OPT}(B_1) \leq y_1 \leq R^{\infty}_{A,3} \cdot \mathrm{OPT}(B_1) + C, \tag{5}$$

$$\mathrm{OPT}(B_1, B_{2,k}) \leq y_2 \leq R^{\infty}_{A,3} \cdot \mathrm{OPT}(B_1, B_{2,k}) + C \tag{6}$$

and the total sum of the sizes in the first two batches is

$$n + \frac{6j}{j-k} n \cdot \left(\frac{1}{3} + \frac{3k-1}{3} \varepsilon \right).$$

On the other hand – since we have only $(0,1,0)_{2,k}$ type new-opened bins – the maximal size of the items which can be put into the used bins is

$$y_1 + \left(\frac{1}{3} + k\varepsilon \right) (y_2 - y_1).$$

Therefore

$$n + \frac{6j}{j-k} n \cdot \left(\frac{1}{3} + \frac{3k-1}{3} \varepsilon \right) \leq y_1 + \left(\frac{3k-1}{3} \varepsilon \right) (y_2 - y_1).$$

The right hand side takes its maximum if y_1 and y_2 are maximal. Therefore

$$\begin{aligned} n + \frac{6j}{j-k} n \left(\frac{1}{3} + \frac{3k-1}{3} \varepsilon \right) \leq \quad & R^{\infty}_{A,3}(\mathrm{OPT}(B_1) + C) + \left(\frac{1}{3} + \frac{3k-1}{3} \varepsilon \right) \\ & \times \left(R^{\infty}_{A,3} \cdot \mathrm{OPT}(B_1, B_{2,k}) - R^{\infty}_{A,3} \cdot \mathrm{OPT}(B_1) \right) \end{aligned} \tag{7}$$

is a valid inequality. Substituting the values of the optimal packing of the batches we get

$$n + \frac{6j}{j-k} n \left(\frac{1}{3} + \frac{3k-1}{3} \varepsilon \right) \leq (R^{\infty}_{A,3} n + C) + \left(\frac{1}{3} + \frac{3k-1}{3} \varepsilon \right) \left(R^{\infty}_{A,3} n \frac{3j}{j-k} - R^{\infty}_{A,3} n \right).$$

Let us divide by n both sides and take into account that $R^{\infty}_{A,3} \leq 1.6$. If $n \to \infty$, the $C/n \to 0$, therefore we get

$$1 + \frac{6j}{j-k} \cdot \left(\frac{1}{3} + \frac{3k-1}{3} \varepsilon \right) \leq \frac{16}{10} + \frac{16}{10} \left(\frac{1}{3} + \frac{3k-1}{3} \varepsilon \right) \left(\frac{3j}{j-k} - 1 \right).$$

So

$$\left(\frac{1}{3} + \frac{3k-1}{3} \varepsilon \right) \left[\frac{6j}{j-k} - \frac{16}{10} \left(\frac{3j}{j-k} - 1 \right) \right] \leq \frac{6}{10}.$$

Since $\frac{3k-1}{3}\varepsilon > 0$, we get

$$\left[\frac{6j}{j-k} - \frac{16}{10}\Big(\frac{3j}{j-k} - 1\Big)\right] \leq \frac{18}{10},$$

and so we get that $10j + 2k \leq 0$ which is a contradiction since $1 \leq k \leq j-1$. □

Lemma 4.2. *If A is a batched algorithm with $R^{\infty}_{A,3} \leq 1.6$, there always exists such an algorithm A' which does not use more bins than A does, and it does not create bins with types $(i_1, 1, 1)_{3,k} \in V_1$ nor $(i_1, 0, 1)_{3,k} \in V_1$, where $i_1 > 0$.*

Proof. Let us suppose that we packed all items. Now, we replace bins of some types by other bins step by step. By Lemma 4.1, a bin of type $(0, 2, 0)$ must exist (note that such a bin does not receive a large item, so such a bin still has type $(0, 2, 0)$ after all items have been packed). We execute each *replacement step* as follows. A pair of $(i_1, x, 1)$ and $(0, 2, 0)$ types of bins will be replaced by a pair of $(i_1, x+1, 0)$ and $(0, 1, 1)$ types of bins, where $x \in \{0, 1\}$, and $i_1 > 0$.

After applying such step once, Lemma 4.1 holds again, because the original algorithm A can be transformed in a straightforward way to repack the items considered here. Note that this transformation can be performed in an online manner, leading to a new online algorithm that uses the same number of bins. We can repeat the replacement step until there are no more bins of type $(i_1, x, 1)$ for any $i_1 > 0$ and $x \in 0, 1$. □

Lemma 4.3. *Let A be a batched algorithm with $R^{\infty}_{A,3} < 2$. If A packs the items of the batches $B_1, B_{2,k}, B_{3,k}$ in such a way that at least one $(0, 1, 0)_{2,k}$ type bin will be produced after finishing the packing of all the batches, then always exists an A' online batched algorithm which does not create $(0, 1, 0)_{2,k}$ type bins and uses less bins than A.*

Proof. Since $R^{\infty}_{A,3} < 2$, therefore – if n is large enough – A opens at least one $(0, 1, 0)_{2,k}$ type new bin. Suppose that while A packs the items of $B_{2,k}$ more than one $(0, 1, 0)_{2,k}$ type bin will be produced.

Let us change the strategy by packing two items into each new opened bin. If the number of $(0, 1, 0)_{2,k}$ type bins was even, then the contents of two such bins always can be packed into one bin, and we are ready. Otherwise, one $(0, 1, 0)_{2,k}$ type bin remains. Since at least one $(0, 0, 1)_{3,k}$ type bin were created, therefore the contents of the bin $(0, 1, 0)_{2,k}$ and a $(0, 0, 1)_{3,k}$ bin can be put into one bin, and so the new A' algorithm will use less bins than A. □

Corollary 4.4. *It is enough to investigate those algorithms which pack the batches $B_1, B_{2,k}, B_{3,k}$ in such a way that they create only bins of types*

$$(i_1, 0, 0)_1, \quad (i_1, 1, 0)_{2,k}, \quad (i_1, 2, 0)_{2,k}, \quad (0, 1, 1)_{3,k}, \quad (0, 2, 0)_{2,k}, \quad (0, 0, 1)_{3,k}.$$

5. An LP model

Now we will construct a linear program problem to minimize the ACR of any online 3-BBPP algorithm. We will give conditions for the number of elements in the batches and we also give lower bounds for the possible values of the ACR of the algorithms. To simplify the notation, instead of $R^{\infty}_{A,3}$ we will use R. The first condition concerns the number of elements in the first batch B_1.

$$\sum_{(i_1,i_2,i_3)\in V_1} i_1 x^k_{i_1,i_2,i_3} = \sum_{i_1=1}^{6j} i_1 x^k_{i_1,i_2,i_3} = n_1 \tag{8}$$

where $x^k_{i_1,i_2,i_3}$ denote the number of (i_1, i_2, i_3) type bins while we pack the batches $B_1, B_{2,k}, B_{3,k}$. Let us remind the reader: $x^r_{i_1,i_2,i_3}$ and $x^s_{i_1,i_2,i_3}$ $(1 \le r, s \le j-1)$ *are the same variables*, their upper indices sign only the type of batches we used. Since the number of items in the first batch is independent of k, so we have here one equality.

We can give conditions for the number of items in the batch $B_{2,k}$. If we put only one element instead of two items from the second batch in a bin, then this bin must contain at least $2j-2k+1$ items from the first batch and it may not contain more items than $4j-k$. Since in this case $i_1 > 0$ then – because of the Lemma 4.2 – we get that in these type of bins $i_3 = 0$.

Similarly, if a bin contains 2 pieces from $B_{2,k}$ then the number of the items from the first batch may not be more than $2j-2k$. For these type of bins $i_3 = 0$ is also valid, since we cannot put any item from the batch $B_{3,k}$ if a bin contains 2 items from the batch $B_{2,k}$. Therefore

$$\sum_{i_1=2j-2k+1}^{4j-k} x^k_{i_1,1,0} + 2\sum_{i_1=1}^{2j-2k} x^k_{i_1,2,0} + x^k_{0,1,1} + 2x^k_{0,2,0} = n_{2,k}, \quad k = 1, 2, \ldots, j-1. \tag{9}$$

The last $j-1$ equations concern to the number of elements in the batch $B_{3,k}$ while we pack the third batch:

$$x^k_{0,1,1} + x^k_{0,0,1} = n_{3,k}, \quad k = 1, 2, \ldots, j-1. \tag{10}$$

Now we give three lower bounds for the ACR.

$$\sum_{i_1=1}^{6j} x^k_{i_1,i_2,0} \le R \cdot \mathrm{OPT}(B_1) \tag{11}$$

$$\sum_{i_1=1}^{6j} x^k_{i_1,i_2,0} + x^k_{0,1,1} + x^k_{0,2,0} \le R \cdot \mathrm{OPT}(B_1, B_{2,k}) \tag{12}$$

$$\sum_{i_1=1}^{6j} x^k_{i_1,i_2,0} + x^k_{0,1,1} + x^k_{0,2,0} + x^k_{0,0,1} \le R \cdot \mathrm{OPT}(B_1, B_{2,k}, B_{3,k}). \tag{13}$$

Let us consider the linear programming problem which has as conditions the $2(j+1)$ inequalities of (8)–(13) and its objective function to minimize the value of R.

Now, we will eliminate the variables $x^k_{0,1,1}$, $x^k_{0,2,0}$, and $x^k_{0,0,1}$. Therefore we do the following calculations. For $k = 1, 2, \ldots, j-1$ we add the appropriate pairs of (12) and (13), and substitute the appropriate equalities (9) and (10).

We remind the reader that

$$n_1 = 6jn, \qquad n_{2,k} = \frac{6j}{j-k}n, \quad \text{and} \quad n_{3,k} = \frac{6j}{j-k}n.$$

Let us substitute the values of $\mathrm{OPT}(B_1)$, $\mathrm{OPT}(B_1, B_{2,k})$, $\mathrm{OPT}(B_1, B_{2,k}, B_{3,k})$, $n_1, n_{2,k}$, and $n_{3,k}$ into the right hand sides. So we get the following $j+1$ conditions.

$$\sum_{i_1=1}^{6j} i_1 x^k_{i_1,i_2,0} = 6jn \tag{14}$$

$$\sum_{i_1=1}^{6j} x^k_{i_1,i_2,0} \le R \cdot n \tag{15}$$

$$2\sum_{i_1=1}^{6j} x^k_{i_1,i_2,0} - \sum_{i_1=2j-2k+1}^{4j-k} x^k_{i_1,1,0} - 2\sum_{i_1=1}^{2j-2k} x^k_{i_1,2,0} \le R \cdot \left(\frac{6j}{2(j-k)} + \frac{6j}{j-k}\right) n - \frac{12j}{j-k}n, \quad k = 1, 2, \ldots, j-1. \tag{16}$$

If we divide the inequalities by n, and introduce the following variables for every valid triplets:

$$z^k_{i_1,i_2,i_3} = \frac{x^k_{i_1,i_2,i_3}}{n},$$

then we get the following conditions.

$$\sum_{i_1=1}^{6j} i_1 z^k_{i_1,i_2,0} = 6j \tag{17}$$

$$\sum_{i_1=1}^{6j} z^k_{i_1,i_2,0} \le R \tag{18}$$

$$2\sum_{i_1=1}^{6j} z^k_{i_1,i_2,0} - 2\sum_{i_1=1}^{2j-2k} z^k_{i_1,2,0} - \sum_{i_1=2j-2k+1}^{4j-k} z^k_{i_1,1,0} \le (3R-4)\frac{6j}{2(j-k)}, \tag{19}$$

$$k = 1, 2, \ldots, j-1.$$

We need to take into account that

$$\sum_{i_1=1}^{6j} z^k_{i_1,i_2,0} = \sum_{i_1=1}^{2j-2k} z^k_{i_1,2,0} + \sum_{i_1=2j-2k+1}^{4j-k} z^k_{i_1,1,0} + \sum_{i_1=4j-k+1}^{6j} z^k_{i_1,0,0}.$$

So we get

$$\sum_{i_1=1}^{6j} i_1 z^k_{i_1,i_2,0} = 6j \tag{20}$$

$$\sum_{i_1=1}^{6j} z^k_{i_1,i_2,0} \le R \tag{21}$$

$$\sum_{i_1=2j-2k+1}^{4j-k} z^k_{i_1,1,0} + 2\sum_{i_1=4j-k+1}^{6j} z^k_{i_1,0,0} \le (3R-4)\frac{6j}{2(j-k)}, \tag{22}$$

$$k = 1, 2, \ldots, j-1.$$

6. The lower bound

As we mentioned earlier, we need not consider all conditions from the $k = 1, \ldots, j-1$ possible ones. We assume that d of them is enough, i.e. erasing the conditions for $k = d+1, \ldots, j-1$, the lower bound remains the same. Later we will give the optimal value for d. Moreover, we further make some calculations and simplification. For each value of k, $(k = 1, 2, \ldots, d)$ we multiply Eqs. (20) by (-1), inequalities (21) by $2(j-d)$. Furthermore, for $k = 1$ we multiply (22) by $2j-d+2$ and all the other ones $(k = 2, \ldots, d)$ by 2.

Lemma 6.1. *Making the linear combination with the above coefficients, the left hand side of the inequality is nonnegative.*

Proof. For every k, $k = 1, 2, \ldots, d$, according to the value of i_1, $1 \le i_1 \le 6j$, we will distinguish five different cases.

- Case 1. $1 \le i_1 \le 2(j-d)$. Since $z^k_{i_1,i_2,i_3}$ does not appear in the last $j-d$ inequalities, now the coefficients are $2(j-d) - i_1 \ge 0$.
- Case 2. $2(j-d)+1 \le i_1 \le 2j-2$. In this case for the coefficients we get

$$2(j-d) - i_1 + 2\lceil (i_1 - 2j + 2d)/2 \rceil \ge 2(j-d) - i_1 + 2(i_1 - 2j + 2d)/2 = 0.$$

- Case 3. $2j-1 \le i_1 \le 4j-d$. In this case for the coefficients we get

$$2(j-d)-i_1+(2j-d+2)+2(d-1)=4j-d-i_1 \ge 0.$$

- Case 4. $4j-d+1 \le i_1 \le 4j-1$. In this case the coefficients are

$$2(j-d)-i_1+(2j-d+2)+2\,(d-(i_1-4j+d+1))+4(i_1-4j+d)=-4j+d+i_1 \ge 0.$$

- Case 5. $4j \le i_1 \le 6j$. In this case the coefficients are

$$2(j-d)-i_1+2(2j-d+2)+4(d-1)=6j-i_1 \ge 0. \quad \square$$

As the left hand side of the linear combination with the given coefficients is nonnegative, we get the following inequality:

$$6j \le 2(j-d)R+6j(2j-d+2)\frac{3R-4}{2(j-1)}+(3R-4)\cdot\sum_{k=2}^{d}\frac{6j}{(j-k)}. \tag{23}$$

The following inequality is easy to prove, see [7].

$$\sum_{k=2}^{d}\frac{1}{j-k}<\ln\frac{j-2}{j-d-1}.$$

Since $3R-4>0$, using the inequality (23), and making some calculations we get

$$6j+\frac{12j(2j-d+2)}{j-1}+24j\ln\frac{j-2}{j-d-1} \le \left(2j-2d+\frac{9j(2j-d+2)}{j-1}+18j\ln\frac{j-2}{j-d-1}\right)R \tag{24}$$

and therefore

$$\begin{aligned}&30j^2+18j-12jd+24j(j-1)\ln\frac{j-2}{j-d-1}\\ &\le\left(20j^2+16j-11j\cdot d+2d+18j(j-1)\ln\frac{j-2}{j-d-1}\right)R.\end{aligned} \tag{25}$$

Ordering this inequality we get

$$f(j,d)=\frac{30j^2+18j-12j\cdot d+24j(j-1)\ln\frac{j-2}{j-d-1}}{20j^2+16j-11j\cdot d+2d+18j(j-1)\ln\frac{j-2}{j-d-1}} \le R. \tag{26}$$

Now we have a lower estimation on the asymptotic ratio R for our construction. Formula (26) has 2 parameters j and d, $(d=1,\dots,j-2)$. For a fixed j we want to find such d, which maximizes the left hand side of (26). In order to determine this d, we consider the function $f(j,d)$ as a *1-variable* continuous function $f(d)$ for a fixed j on the interval $[1,j-2]$ (see Fig. 1).
It is easy to see that $f(d)$ has one maximum in the given interval. To decide the maximum value of d first we need to derivate $f(d)$ by d.

$$\begin{aligned}f'(d)=&\frac{-12\,j+24\,\frac{j(j-1)}{j-d-1}}{20\,j^2+16\,j-11\,jd+2\,d+18\,j\,(j-1)\ln\left(\frac{j-2}{j-d-1}\right)}\\ &-\frac{\left(30\,j^2+18\,j-12\,jd+24\,j\,(j-1)\ln\left(\frac{j-2}{j-d-1}\right)\right)\left(-11\,j+2+18\,\frac{j(j-1)}{j-d-1}\right)}{\left(20\,j^2+16\,j-11\,jd+2\,d+18\,j\,(j-1)\ln\left(\frac{j-2}{j-d-1}\right)\right)^2}.\end{aligned}$$

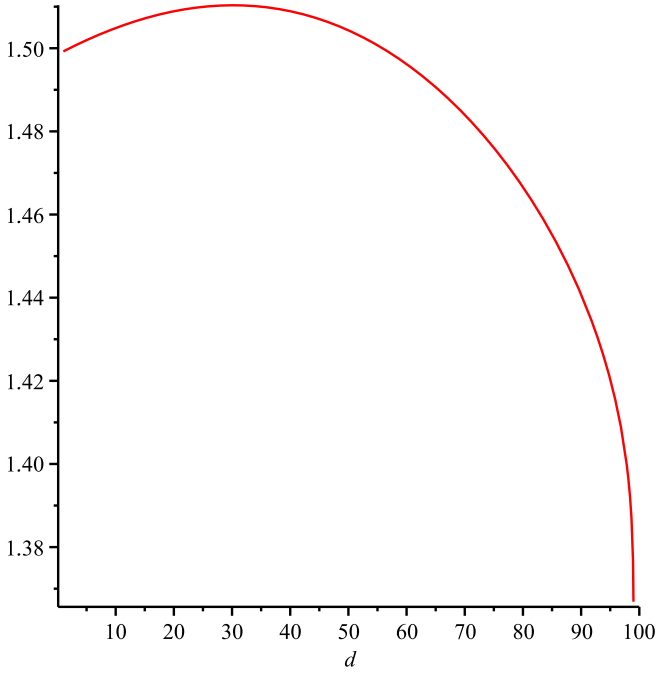

Fig. 1. Graph of the function $f(d)$ for $j = 100$.

Table 1
The values of $f(j, d)$.

j	d	$f(j, d)$
5	1	1.480075901
10	3	1.494928787
20	6	1.503357743
50	15	1.508573181
100	30	1.510335641
200	60	1.511221192
500	152	1.511757013
1000	305	1.511935384

Now we can solve the equation $f'(d) = 0$, so we get the value of d_0 where the function takes its maximum.

$$d_0 = \frac{1}{4}\frac{9j-4}{W\left(-1, -\frac{1}{4}\,\frac{e^{\frac{-1}{8}\,\frac{23j-2}{j-1}}(9j-4)}{j-2}\right)} + j - 1$$

where $W(-1, x)$ is the negative branch of the Lambert function. Substituting d_0 into $f(d)$ and taking its limit while $j \to \infty$ we get the required formula for R.

$$R \geq \frac{32W\left(-1, -\frac{9}{4}e^{-\frac{23}{8}}\right) + 36}{24W\left(-1, -\frac{9}{4}e^{-\frac{23}{8}}\right) + 33} \approx 1.51211383.$$

Table 1 shows the lower bounds for different values of j and d. So, we have the following theorem.

Theorem 6.2. *If A is a batched algorithm for 3-BBPP, then*

$$R^{\infty}_{A,3} \geq 1.51211383.$$

7. Conclusion

In [7] Gutin et al. defined the batched bin packing problem, and they gave a $1.3871\ldots$ lower bound for the ACR of any online 2-BBPP algorithm. In this paper we take a step ahead, and we investigated the 3-BBPP, and we give a lower bound for any on-line 3-BBPP algorithm. It would be interesting to construct a good algorithm for this problem giving a narrow gap between the upper and lower bounds. The second possibility is to investigate the K-BBPP for $K \geq 4$. We note that we have promising efforts for $K = 4$, but the analysis is much more complicated than any of the earlier cases.

Acknowledgment

The authors are grateful to the anonymous reviewer for giving an elegant improvement to the proof of Lemma 4.2, which resulted in significant shortening of the original analysis.

References

[1] S. Heydrich, R. van Stee, Beating the Harmonic lower bound for online bin packing, 2016. http://arxiv.org/pdf/1511.00876.pdf (accessed 18.02.16).
[2] J. Balogh, J. Békési, G. Galambos, New lower bounds for certain classes of bin packing algorithms, Theoret. Comput. Sci. 440–441 (2012) 1–13.
[3] D.S. Johnson, Near-optimal bin-packing algorithms (Doctoral Thesis), MIT, Cambridge, 1973.
[4] C.C. Lee, D.T. Lee, A simple on-line packing algorithm, J. ACM 32 (1985) 562–572.
[5] E.F. Grove, Online bin packing with lookahead, in: Proceedings of the Sixth Annual ACM–SIAM Symposium on Discrete Algorithms, 1995, pp. 430–436.
[6] E. Coffman, J. Csirik, G. Galambos, S. Martello, D. De Vigo, Bin packing approximation algorithms: Survey and classification, in: D.-Z. Du, P.M. Pardalos, R.L. Graham (Eds.), Handbook of Combinatorial Optimization, Springer, New York, 2013, pp. 455–531.
[7] G. Gutin, T. Jensen, A. Yeo, Batched bin packing, Discrete Optim. 2 (1) (2005) 71–82.
[8] Gy. Dósa, Batched bin packing revisited, J. Sched. (2015). http://dx.doi.org/10.1007/s10951-015-0431-3. in press.