**Noname manuscript No.**
(will be inserted by the editor)

# Tight Bounds for NF-based Bounded-Space Online Bin Packing Algorithms

**József Békési · Gábor Galambos**

the date of receipt and acceptance should be inserted later

**Abstract** In [14] Zheng et al. modelled a surgery problem by the one-dimensional bin packing, and developed a semi-online algorithm to give an efficient feasible solution. In their algorithm they used a buffer to temporarily store items, having a possibility to lookahead in the list. Because of the considered practical problem they investigated the 2-parametric case, when the size of the items is at most 1/2. Using an NF-based online algorithm they proved an ACR of $13/9 = 1.44\dots$ for any given buffer size not less than 1. They also gave a lower bound of $4/3 = 1.33\dots$ for the bounded-space algorithms that use NF-based rules.

Later, in [13] an algorithm was given with an ACR of 1.4243, and they improved the lower bound to 1.4230. In this paper we present a tight lower bound of $h_\infty(r)$ for the $r$-parametric problem when the buffer capacity is 3. Since $h_\infty(2) = 1.42312$, our result – as a special case – gives a tight bound for the algorithm-class given in [13]. To prove that the lower bound is tight, we present an NF-based online algorithm that considers the $r$-parametric problem, and uses a buffer with capacity of 3. We prove that this algorithm has an ACR that is equal to the lower bounds for arbitrary $r$.

## 1 Introduction

Bin packing is one of the most deeply studied problem in the field of combinatorial optimization. The one-dimensional version can be defined as follows. Let $L = \{a_1, a_2, \dots, a_n\}$ be a list of $n$ items, with sizes $s(a_i) \in (0, 1]$, $i = 1, \dots, n$. The task is to assign the items to the *minimal* number of unit capacity bins, subject to the constraint that the total size of the items assigned to any bin is *at most* 1. If the sizes of the items for some integer $r \geq 2$ are chosen from the interval $(0, \frac{1}{r}]$, then we speak about *r-parametric* (or simply, *parametric*) bin packing problem.

It is well-known that the problem is NP-hard [9]. Therefore a couple of approximation algorithms have been developed in the last 40 years. The quality of the

Department of Applied Informatics,
Gyula Juhász Faculty of Education, University of Szeged,
H-6701 Szeged, POB 396, Hungary
E-mail: {bekesi,galambos}@jgypk.szte.hu

approximation algorithms is a central question in the algorithm theory. There are several methods to measure the quality of an algorithm: experimental examination, worst case competitive analysis and probabilistic analysis.

In this paper we apply the asymptotic competitive analysis. Using this method we are looking for performance guarantees which are valid for any – even for the extreme, so-called "pathological" – instances. Let $A$ be an arbitrary approximation algorithm, and let $I$ be an instance of the given problem. Let $A(I)$ and $\mathrm{OPT}(I)$ denote the solution of $A$ and the optimal solution, respectively. The *asymptotic competitive ratio* (ACR) for minimum problems is defined as follows.

$$R_A^\infty = \limsup_{k\to\infty} \left\{ \max_I \left\{ \frac{A(I)}{k} \,\middle|\, \mathrm{OPT}(I) = k \right\} \right\}.$$

The efficiency of an algorithm strongly depends on the information available in each step of the algorithm. If we have complete information about the list we need to put into bins, then we speak about *off-line algorithms*.

An online algorithm packs the items one by one. Packing the actual item the algorithm does not know anything about the remaining part of the list: neither the number of unpacked items nor their sizes are known. It is clear that the online algorithms do not have enough information to produce as good packing as the offline algorithm. The best online algorithm was given by Heydrich and van Stee (see [8]) with 1.5816 of ACR, which is an improvement of the algorithm by Seiden (see [10]) with 1.58889 of ACR. The best known lower-bound ($\frac{248}{161} \approx 1.5403$) for online algorithms was given by Balogh et al. [1].

It is a question whether we can improve the efficiency of online algorithms by getting more – but not complete – information during the packing? If it is allowed for an online algorithm to apply at least one of the following operations: repacking some items, lookahead into several next elements, or some kind of preprocessing, then we speak about *semi-online algorithms.* Among others, semi-online algorithms were investigated in the papers [2,3,5,6,7].

In [14] Zheng et al. considered the following surgery problem. In each day there is a uniform time interval available for an operating room to process surgical operations. Each request with a planned operation time is temporarily stored in a waiting pool. In each day, a surgery scheduler selects a subset of requests to be executed the next day. The total planned operation time of the selected requests cannot exceed the time available for the day. They modelled this problem by the one-dimensional bin packing, and developed a semi-online algorithm to give an efficient feasible solution. In their algorithm they used a buffer to temporarily store items, having a possibility to lookahead in the list. Because of the considered practical problem they investigated the 2-parametric problem, i.e. $\max_{a\in L} s(a) \leq 1/2$. In each iteration step their algorithm puts the largest items of the buffer into a new opened bin using a Next Fit-based rule (NF): packing the actual preprocessed contents of the buffer the algorithm opens a new, empty bin, puts iteratively the largest items of the buffer into this bin until they fit and closes the bin. This means that at most 1 bin can be open during the packing. The algorithms that use constant number of open bins are called bounded-space algorithms. If there is at most 1 open bin, the algorithm is called NF-based online algorithm. Using this NF-based online algorithm they proved an ACR of $\frac{13}{9}$ for any given buffer size not less than 1. They also gave a lower bound of $\frac{4}{3}$ for those bounded-space algorithms that use NF-based rules.

Later, Zhang et al. also investigated the problem (see [13]). They presented two algorithms. The first one used a buffer with capacity of 2, and they proved that the ACR of the algorithm is 1.4375. Using a buffer size 3 they gave further improvement on the upper bound. Their algorithm has an ACR of 1.4243. Finally, they gave a lower bound 1.4230, which is also better than the one previously proved in [14].

**Our Contribution** We revisited those NF-based semi-online algorithms that use buffer with constant size. Our results can be summarized as follows. Firstly, instead of the 2-parametric problem, we investigate the parametric problem in general. It means that we consider the lists $L$ where $\max_{a\in L} s(a) \leq \frac{1}{r}$ for a given integer $r \geq 1$. We prove lower bounds for any NF-based online algorithm with constant buffer size for the $r$-parametric case. Our lower bounds for the first few values of $r$ are the following: $h_\infty(1) = 1.69103$, $h_\infty(2) = 1.42312$, $h_\infty(3) = 1.30238$. The special case $r = 2$ gives an improvement for the earlier lower bounds.

On the positive side, we present an NF-based online algorithm that considers the $r$-parametric problem, and uses a buffer with capacity of 3. We prove that this algorithm has ACRs that are equal to the lower bounds, so we also improve the upper bound for the case $r = 2$ to 1.42312.

The rest of the paper is organized as follows. In Section 2 we show the Sylvester sequence, and we define the exact values of $h_\infty(r)$ for $r = 1, 2, \ldots$. In Section 3 we prove new lower bounds of the considered problem for the $r$-parametric case. In Section 4 we define a new, NF-based algorithm with buffer-size 3. Finally, Section 5 concludes this paper.

## 2 Preliminaries

We will use a sequence that was first introduced by Sylvester in [11] (1880) for the case $r = 1$, therefore, we refer to this sequence as *generalized Sylvester sequence.* This sequence was commonly used in the bin packing area, see eg. [1,4,12].
For integers $k > 1$ and $r \geq 1$, the generalized Sylvester sequence $m_1^r, \ldots, m_k^r$ can be given by the following recursion.

$$m_1^r = r + 1, \qquad m_2^r = r + 2, \qquad m_j^r = m_{j-1}^r(m_{j-1}^r - 1) + 1, \text{ for } j = 3, \ldots, k, .$$

| $m_j^r$ | $r = 1$ | $r = 2$ | $r = 3$ | $r = 4$ | $r = 5$ |
|---|---|---|---|---|---|
| $j = 1$ | 2 | 3 | 4 | 5 | 6 |
| $j = 2$ | 3 | 4 | 5 | 6 | 7 |
| $j = 3$ | 7 | 13 | 21 | 31 | 43 |
| $j = 4$ | 43 | 157 | 421 | 931 | 1807 |
| $j = 5$ | 1807 | 24493 | 176821 | 865831 | 3263443 |

Table 2.1. The first few items of the generalized Sylvester sequences if $k \leq 5$.

These sequences have the following properties.

$$\sum_{i=j}^{k} \frac{1}{m_i^r} = \frac{1}{m_j^r - 1} - \frac{1}{m_{k+1}^r - 1}, \quad \text{if } j \geq 2,$$

and

$$\frac{r}{m_1^r} + \sum_{i=2}^{k} \frac{1}{m_i^r} = 1 - \frac{1}{m_{k+1}^r - 1} \quad \text{if } r \geq 2.$$

Similarly, we will use the following notations.

$$h_\infty(r) = 1 + \sum_{i=2}^{\infty} \frac{1}{m_i^r - 1}.$$

The first few values of $h_\infty(r)$ : $h_\infty(1) \approx 1.69103$, $h_\infty(2) \approx 1.42312$, $h_\infty(3) \approx 1.30238$. To avoid the plenty of indices – where it is not confusing – we will denote $m_j^r$ by $m_j$.

## 3 An improved lower bound

First, we give an improvement of the lower bound of any NF-based online algorithm with given buffer size $S \geq 1$. The buffer can store arbitrary items with total size less than the size of the buffer.

**Theorem 1** *Let us consider the $r$-parametric case. If a buffer is given with size of $|B| = S$, then for any $A$ – NF-based online – algorithm $R_\infty(A) \geq h_\infty(r)$.*

*Proof* We will construct the following instance. Let $n > 0$ be a large integer and let $k > 4$ be an integer. Then we will consider the concatenated list $L = (L_1, L_2, \ldots, L_k)$, where

- $L_1$ contains $n(m_k - 1)(m_1 - 1)$ items with size $\frac{1}{m_1} + \varepsilon$.
- $L_i$ contains $n(m_k - 1)$ items with size $\frac{1}{m_i} + \varepsilon$, for $2 \leq i \leq k - 1$,
- $L_k$ contains $n(m_k - 1)$ items with size $\frac{1}{m_k - 1} - k\varepsilon$,

where $\varepsilon$ is arbitrary small, ie. $\frac{1}{m_k - 1} - (m_1 + k - 3)\varepsilon > \frac{1}{m_k}$.

After packing the items of the list $L_1$ there are at most $\lfloor \frac{S}{1/m_1 + \varepsilon} \rfloor < Sm_1$ items in the buffer. Therefore, the algorithm has to pack $n(m_k - 1)(m_1 - 1) - Sm_1$ items from the list $L_1$ into bins. Since $m_1 \geq 3$, so $A$ needs at least

$$\frac{n(m_k - 1)(m_1 - 1) - Sm_1}{m_1 - 1} = n(m_k - 1) - \frac{Sm_1}{m_1 - 1} > n(m_k - 1) - 2S$$

bins to pack the elements of $L_1$.

Let us pack the items of $L_2$. We have $n(m_k - 1)$ pieces. Having packed the elements of $L_2$ the buffer contains $\lfloor \frac{S}{1/m_2 + \varepsilon} \rfloor < Sm_2$ elements. So, the algorithm has to pack $n(m_k - 1) - Sm_2$ items, $m_2 - 1$ pieces in each bin. So, the algorithm uses at least

$$\frac{n(m_k - 1) - Sm_2}{m_2 - 1} = \frac{n(m_k - 1)}{m_2 - 1} - \frac{Sm_2}{m_2 - 1} > \frac{n(m_k - 1)}{m_2 - 1} - 2S$$

bins. Since at most one active bin exits, therefore at least $\frac{n(m_k-1)}{m_2-1} - 2S - 1$ new bins were opened while the algorithm packed the elements of $L_2$.

Following this train of thought while the algorithm packs the items of $L_i$, $3 \leq i \leq k-1$, it will open $\frac{n(m_k-1)}{m_2-1} - 2S - 1$ new bin for each list.

In the end the algorithm packs the items of $L_k$. Each bin contains $(m_k - 1)$ items from this list, so the algorithm opens $(n-1)$ new bins. Therefore, the number of bins used by the algorithm $A$ is at least

$$A(L) \geq n(m_k - 1) - 2S + \sum_{i=2}^{k} \frac{m_k - 1}{m_i - 1} n - (2k-1)S - (k-1).$$

It is easy to check that $(m_1 - 1)$ pieces of $L_1$, and one item from each $L_i$, $1 \leq i \leq k$ can be packed into one bin, therefore $\mathrm{OPT}(L) \leq n(m_k - 1)$. So,

$$\frac{A(L)}{\mathrm{OPT}(L)} \geq 1 + \sum_{i=2}^{k} \frac{1}{m_i - 1} - \frac{(2k-1)S - (k-1)}{(m_k - 1)n}$$

and so,

$$R_\infty(A) \geq \lim_{k\to\infty} \lim_{n\to\infty} \frac{A(L)}{\mathrm{OPT}(L)} = 1 + \sum_{i=2}^{\infty} \frac{1}{m_i - 1} = h_\infty(r).$$

□

For the problem considered in [14] and [13] the given best lower bound is 1.4230, and since $h_\infty(2) = 1.423117\ldots$ our lower bound gives an improvement for the case $r = 2$.

Investigating online bounded-space algorithms in [5] a weighting function was defined for the items. Generalizing the idea we define the following weighting function.

$$W(x) = \begin{cases} x + \dfrac{1}{m_i(m_i - 1)}, & \text{if } \frac{1}{m_i} < x \leq \frac{1}{m_i - 1} \\ \\ \dfrac{m_i + 1}{m_i} x, & \text{if } \frac{1}{m_{i+1} - 1} < x \leq \frac{1}{m_i} \end{cases}$$

The weight of a bin is defined as the weight of all elements in it, and generally, the weight of a set is the weight of all items in the set. It is easy to see that the following statements are true.

**Fact 2**
*(i)* $W(x)$ *is nondecreasing in* $(0, 1]$.
*(ii) For* $i \geq 1$, $\frac{W(x)}{x} \leq \frac{m_i+1}{m_i}$ *if* $x \leq \frac{1}{m_i}$,
*(iii) For* $i \geq 1$, $\frac{W(x)}{x} \geq \frac{m_i+1}{m_i}$ *if* $x \geq \frac{1}{m_{i+1}-1}$.

**Lemma 1** *Let us consider the r-parametric problem. Then any packing of a list* $L$, *the weight of any bin is at most* $h_\infty(r)$.

*Proof* Let us suppose that a bin contains the items $x_1, x_2, \dots, x_t$, where $x_1 \geq x_2 \geq \dots \geq x_t$.

**Case A.** First we suppose that there are exactly $r$ items from the interval $(\frac{1}{m_1}, \frac{1}{m_1-1}]$ in an arbitrary bin $\mathcal{B}$. We denote the remaining items by $p_1, \dots, p_{t-r}$.

**Case A.1.** Now, we suppose that each $p_i$ is in the interval $(\frac{1}{m_{i+1}}, \frac{1}{m_{i+1}-1}]$ for $i = 1, \dots, (t-r)$. Taking into account that $r = m_1 - 1$, $m_1 = m_2 - 1$, and $\frac{1}{m_i(m_i-1)} = \frac{1}{m_{i+1}-1}$ we get

$$\begin{aligned} W(\mathcal{B}) &= \textstyle\sum_{i=1}^{r} W(x_i) + \sum_{i=1}^{t-r} W(p_i) \\ &= \textstyle\sum_{i=1}^{r} x_i + \frac{r}{m_1(m_1-1)} + \sum_{i=1}^{t-r} p_i + \sum_{i=1}^{t-r} \frac{1}{m_{i+1}(m_{i+1}-1)} \\ &\leq 1 + \textstyle\frac{1}{m_2-1} + \sum_{i=3}^{t-r+2} \frac{1}{m_i-1} \\ &= 1 + \textstyle\sum_{i=2}^{t-r+2} \frac{1}{m_i-1} < h_\infty(r). \end{aligned}$$

**Case A.2.** Let us suppose that $\mathcal{B}$ contains $s < t-r$ pieces of $p_i$ items, each of them in the interval $(\frac{1}{m_i}, \frac{1}{m_i-1}]$, $i = 2, 3, \dots, s+1$.

Let us denote the remaining $(t-r-s)$ items by $q_l$, $l = 1, 2, \dots, t-r-s$. Then Q $= \sum_{i=1}^{t-r-s} q(i) \leq \frac{1}{m_{s+2}-1}$, and $\sum_{i=1}^{r} x_i + \sum_{i=2}^{s+1} p_i \leq 1 - Q$. Because of the Fact 2 (ii), we get

$$\sum_{i=1}^{t-r-s} W(q(i)) \leq Q \frac{m_{s+2}+1}{m_{s+2}}.$$

Therefore

$$\begin{aligned} W(\mathcal{B}) &= \textstyle\sum_{i=1}^{r} W(x_i) + \sum_{i=2}^{s+1} W(p_i) + \sum_{i=1}^{t-r-s} W(q_i) \\ &= \textstyle\sum_{i=1}^{r} x_i + \frac{r}{m_1(m_1-1)} + \sum_{i=2}^{s+1} p_i + \sum_{i=2}^{s+1} \frac{1}{m_i(m_i-1)} + Q\frac{m_{s+2}+1}{m_{s+2}} \\ &\leq 1 - Q + Q\textstyle\frac{m_{s+2}+1}{m_{s+2}} + \sum_{i=2}^{s+1} \frac{1}{m_i-1} \\ &= 1 + \textstyle\sum_{i=2}^{s+3} \frac{1}{m_i-1} < h_\infty(r). \end{aligned}$$

**Case B.** Let us suppose that the bin $\mathcal{B}$ contains $q \leq r-1$ items belonging to the interval $(\frac{1}{m_1}, \frac{1}{m_1-1}]$. Since $W(x)$ is a monotone increasing function, so the weighting function is maximal if these items have maximal sizes i.e.

$$\sum_{i=1}^{q} x_i = \frac{q}{m_1 - 1}.$$

Then the remaining place in the bin is $1 - \frac{q}{m_1-1}$. We know that for any item $x$ for which $x \leq \frac{1}{m_1}$ the weight is $W(x) \leq x\frac{m_1+1}{m_1}$.

$$\begin{aligned} W(\mathcal{B}) &= \textstyle\sum_{i=1}^{q} W(x_i) + \sum_{i=q+1}^{t} W(x_i) \\ &\leq \textstyle\frac{q}{m_1-1} + \frac{q}{m_1(m_1-1)} + \left(1 - \frac{q}{m_1-1}\right)\frac{m_1+1}{m_1} \\ &= 1 + \textstyle\frac{1}{m_1} = 1 + \frac{1}{m_2-1} \\ &< h_\infty(r). \end{aligned}$$

□

**Corollary 1** *For any list $L$, $W(L) \leq h_\infty(r)\mathrm{OPT}(L)$.*

## 4 The Algorithm

In the sequel we will call a bin *good bin* if the sum of the weights of the items in the bin is at least one, and a set of items is *good subset* if the sum of the weights of the items is greater than or equal to one and the sum of the sizes is at most one. We consider a buffer with capacity 3 and we will apply three virtual bins – with capacity one – for preprocessing the items in the buffer before we pack them into bin. *Next Fit with First Fit Decreasing in Buffer-length 3*, *NFFD-B3* – is as follows.

```
(1) Fill up the buffer with the subsequent elements of the list until the next item
    cannot fit into the buffer.
(2) Order the items in the buffer in nonincreasing order, and put the items in
    three virtual bins – denoted by $VBIN_i$, $i = 1, 2, 3$ – each of them with capacity
    1 using the FFD rule. The items that do not fit in any of the virtual bins,
    remain in the buffer.
(3) Check the contents of the virtual bins. For all those virtual bins that are
    good bins, open a new empty bin, put the items from the good bin into this
    new-opened bin, and close the bin. Go to step (5).
(4) Find a good subset in the contents of $VBIN_i$, $i = 1, 2, 3$, open a new empty
    bin, put the items from the virtual bins into this new-opened bin, and close
    the bin.
(5) If there is unplaced item then go to (1),
(6) Empty the contents of the virtual bins into new-opened bins. Close the bins,
    and quit.
```

**Table 4.1.** The steps of the *NFFD-B3* algorithm

We remark that we speak about *virtual bins* since after ordering the items we do not move them from the buffer into bins, but they get two indices, where the first one denotes which of the virtual bin belongs to the item, and the second signs its position within the virtual bin. (Items could not be assigned to any virtual bins that have index values 0.) The position within the virtual bin depends on the size of the item: the larger an item the smaller its position.

Let us divide the interval $(0, \frac{1}{r}]$ into subintervals as follows.

$$\begin{aligned}
A &= (1/m_1, 1/(m_1-1)] & \\
B_i &= (1/m_i, 1/(m_i-1)] & \text{for } i \geq 2. \\
C_i &= (1/(m_i+1), 1/m_i] & \text{for } i \geq 2. \\
D_i &= (1/(m_{i+1}-1), 1/(m_i+1)] & \text{for } i \geq 2.
\end{aligned}$$

| Type | Interval | $W(x)$ | Type | Interval | $W(x)$ |
|---|---|---|---|---|---|
| $A$ | $(1/2, 1]$ | $x + 1/2$ | | | |
| $B_2$ | $(1/3, 1/2]$ | $x + 1/6$ | $B_3$ | $(1/7, 1/6]$ | $x + 1/42$ |
| $C_2$ | $(1/4, 1/3]$ | $4x/3$ | $C_3$ | $(1/8, 1/7]$ | $8x/7$ |
| $D_2$ | $(1/6, 1/4]$ | $4x/3$ | $D_3$ | $(1/42, 1/8]$ | $8x/7$ |

**Table 4.2.** The weighting function $W(x)$ for $r = 1$.

| Type | Interval | $W(x)$ | Type | Interval | $W(x)$ |
|---|---|---|---|---|---|
| $A$ | $(1/3, 1/2]$ | $x + 1/3$ | | | |
| $B_2$ | $(1/4, 1/3]$ | $x + 1/12$ | $B_3$ | $(1/13, 1/12]$ | $x + 1/156$ |
| $C_2$ | $(1/5, 1/4]$ | $5x/4$ | $C_3$ | $(1/14, 1/13]$ | $13x/12$ |
| $D_2$ | $(1/12, 1/5]$ | $5x/4$ | $D_3$ | $(1/156, 1/14]$ | $13x/12$ |

**Table 4.3.** The weighting function $W(x)$ for $r = 2$.

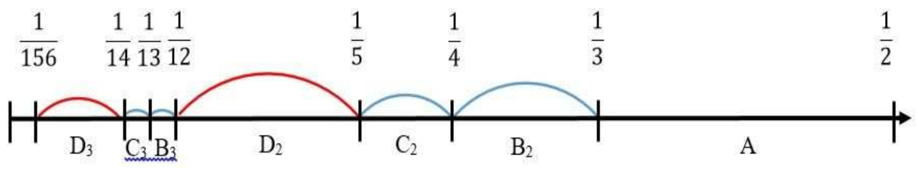

**Figure 4.1.** Spliting the interval $(0, \frac{1}{r}]$ into subintervals for $r = 2$.

We will call an item *X-item* if it is in the interval $X$, $X \in \{A, B_i, C_i, D_i\}$, $i \geq 2$. A bin is *X-homogeneous*, if it only contains $X$-items, and there is no space for further $X$-items in the bin.

**Lemma 2** *Any X-homogeneous bin is a good bin.*

*Proof* **Case $B_i$.** We remind the reader that the interval $A$ is a $B_1$ interval. So, if $\mathcal{B}$ is a $B_i$-homegeneous bin $(i \geq 1)$ then we can put $m_i - 1$ pieces of items into this bin, so the total size of the items is larger than $\frac{m_i-1}{m_i}$. Therefore

$$W(\mathcal{B}) = \sum_{a \in \mathcal{B}} s(a) + (m_i - 1)\frac{1}{m_i(m_i - 1)} > \frac{m_i - 1}{m_i} + \frac{1}{m_i} = 1.$$

**Case $C_i$.** If $\mathcal{B}$ is a $C_i$-homogeneous bin $(i \geq 2)$ then we can put $m_i$ pieces into the bin, so the total size of the items is larger than $\frac{m_i}{m_i+1}$. Therefore

$$W(\mathcal{B}) = \frac{m_i + 1}{m_i} \sum_{a \in \mathcal{B}} s(a) > \frac{m_i + 1}{m_i} \frac{m_i}{m_i + 1} = 1.$$

**Case $D_i$.** If $\mathcal{B}$ is a $D_i$-homogeneous bin $(i \geq 2)$ then we can put at least $m_i + 1$ pieces into the bin. Since we can not put further $D_i$-item into the bin, so the total size of the items is larger than $1 - \frac{1}{m_i+1} = \frac{m_i}{m_i+1}$. Therefore

$$W(\mathcal{B}) \geq \frac{m_i + 1}{m_i} \sum_{a \in \mathcal{B}} s(a) > \frac{m_i + 1}{m_i} \frac{m_i}{m_i + 1} = 1.$$

□

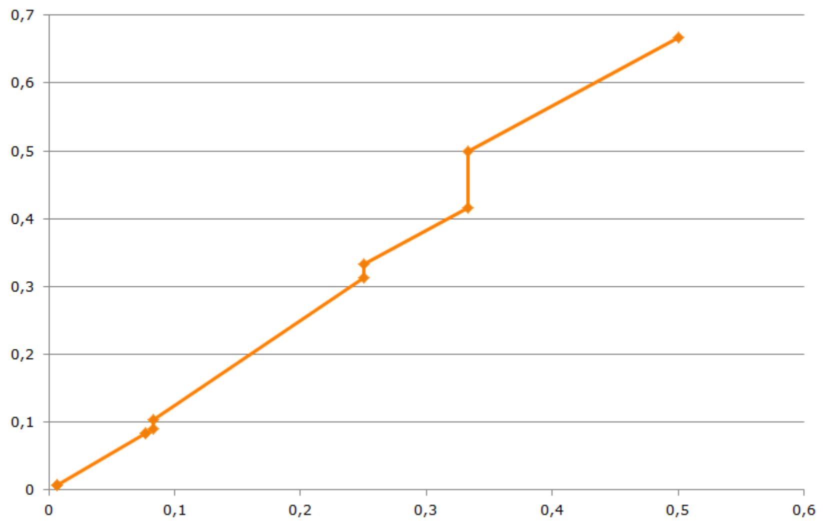

**Figure 4.2.** The weighting function $W(x)$ for $r = 2$.The breaking points of the line are at $\frac{1}{12}$, $\frac{1}{4}$ and $\frac{1}{3}$.

Our main theorem is the following.

**Theorem 3** *If we pack the items of any list by the algorithm NFFD-B3 then in Step (3) we either find at least one good bin, or we find a good subset in the contents of the three virtual bins.*

*Proof* We will assume the contrapositive. We suppose that a list $L_0$ exists for which the algorithm *NFFD-B3* after the Step (2) neither produces at least one virtual bin nor good subsets can be found. We can suppose that the list $L_0$ has the following properties.

- The total size of the items in $L_0$ is $\sum_{a \in L_0} s(a) > 3 - \frac{1}{r}$.
- If $\min_{a \in L_0} s(a)$ is an $X$-item, then in Step (2) no further $X$-items can be put into any of the virtual bins.
- After Step (2) none of the virtual bins are empty.

During our proof we will distinguish different cases according to which interval the smallest item belongs to.

**Lemma 3** *If* $\min_{a \in L_0} s(a)$ *is an $A$-item, then $VBIN_1$ (and $VBIN_2$) is a good bin.*

*Proof* Let us suppose that we have at least one $A$-item in $VBIN_2$ or $VBIN_3$. Then $VBIN_1$ is an $A$-homogeneous bin. □

**Corollary 2** *If* $r = 1$ *then* $\max_{a \in L_0} s(a) \leq \frac{1}{m_1}$.

**Corollary 3** *After Step (2) neither $VBIN_2$ nor $VBIN_3$ contains $A$-items, and $VBIN_1$ contains at most $r - 1 = m_1 - 2$ pieces of $A$-item.*

**Lemma 4** *If* $\min_{a \in L_0} s(a)$ *is a $B_2$-item, then $VBIN_1$ is a good bin.*

*Proof* First we consider the case $r = 1$. Since $VBIN_1$ does not contain $A$-item, therefore, $B_2$ is the largest item in $L_0$. If $VBIN_2$ or $VBIN_3$ contains $B_2$ item then $VBIN_1$ must be a $B_2$-homogeneous bin. Therefore, if $r = 1$ then neither $VBIN_2$ nor $VBIN_3$ contains $B_2$ item.

Since $VBIN_2$ does not contain $A$-item, $VBIN_2$ is a $B_2$-homogeneous bin, and therefore it is a good bin. □

**Corollary 4** *If $r = 1$ then neither $VBIN_2$ nor $VBIN_3$ contains $B_2$-item.*

**Lemma 5** *If $\min_{a\in L_0} s(a)$ is a $C_2$-item, then either at least one of $VBIN_1$ and $VBIN_2$ is a good bin, or the algorithm can collect a good subset from the items in the virtual bins.*

*Proof* First we consider the case $r = 1$. From the Corollary 4 it follows that neither $VBIN_2$ nor $VBIN_3$ can contain $A$- and $B_2$-items. Therefore, if there is a $C_2$-item that does not fit in $VBIN_1$ then $VBIN_2$ must be a $C_2$-homogeneous bin.

Now, we can suppose that $r \geq 2$. Because of the Lemma 3 and Lemma 4, $VBIN_3$ contains only $C_2$-items. Since $\sum_{a\in VBIN_1} s(a) + \sum_{a\in VBIN_2} s(a) \leq 2$, and the buffer was at least to the level $\frac{3r-1}{r}$ full after Step (1), in the $VBIN_3$ there is at least $\frac{3r-1}{r} - 2 = \frac{r-1}{r}$ place which contains $C_2$-items only. Therefore, there are at least $r$ pieces of $C_2$-items in the $VBIN_3$.

**Case A.** Let us suppose that we have $r-1$ $A$-items in the $VBIN_1$, and we denote the total sum of the sizes of $A$-items and the $B_2$-items in $VBIN_1$ by $x_A$ and $x_{B_2}$, respectively. Then

$$W(VBIN_1) = x_A + \frac{r-1}{r(r+1)} + x_B + \frac{1}{(r+1)(r+2)}.$$

Since $x_A + x_{B_2} > \frac{r+1}{r+2}$, therefore

$$W(VBIN_1) > (x_A + x_{B_2}) + \frac{r-1}{r(r+1)}\frac{1}{(r+1)(r+2)} = 1 + \frac{r-2}{r^3+3r^2+2r}.$$

Since $r \geq 2$, the right hand side is greater than 1, so $VBIN_1$ is a good bin.

**Case B.** Now, we suppose that there are at most $(r-2)$ $A$-items in the bin $VBIN_1$. In this case at least 2 pieces of $B_2$-items are in the $VBIN_1$. From the Corollary 3 there is no $A$-item in $VBIN_2$.

**Case B.1.** If $VBIN_2$ contains $r-1$ pieces of $B_2$-items then these items together with the 2 pieces of $B_2$-items in $VBIN_1$ give a good subset.

**Case B.2.** Since the $VBIN_2$ is at least to the level $\frac{r+1}{r+2}$ level full, if it contains at most $(r-2)$ pieces of $B_2$-items then the bin must contain at least 2 pieces of $C_2$-items. So, there are at least $r+2$ pieces of $C_2$-items together in the $VBIN_3$ and $VBIN_3$, and so, they can form a good subset. □

**Corollary 5** *If $r = 1$ then neither $VBIN_2$ nor $VBIN_3$ contains $C_2$-item.*

**Lemma 6** *If $\min_{a\in L_0} s(a)$ is a $D_i$-item, $i \geq 2$, then $VBIN_1$ is a good bin.*

*Proof* When the algorithm *NFFD-B3* puts the first $D_i$ item into $VBIN_j$, $j = 2, 3$, then $VBIN_1$ is at least $\frac{m_i}{m_i+1}$ full. By the fact 2 (iii) $W(x) \geq x(m_i + 1)/m_i$, so the total weight in the $VBIN_1$ is at least 1, so $VBIN_1$ is a good bin. □

**Corollary 6** *If $r = 1$ then neither $VBIN_2$ nor $VBIN_3$ contains $D_i$-item, $i \geq 2$.*

We introduce the following notations. Let $S(X^+, j)$ denote the sum of the sizes of all items in the virtual bin $VBIN_j$, $j = 1, 2, 3$, that are larger than the $X$-elements, where $X \in \{B_i, C_i\}$, $i \geq 3$. Furthermore, let $N(X, j)$ and $S(X, j)$ denote the number and the overall sizes of $X$-elements in $VBIN_j$, respectively.

**Lemma 7** *Let $\min_{a \in L_0} s(a)$ be a $B_i$-item, $i \geq 3$. Then the number of $B_i$-items in the $VBIN_1$ is*

$$N(B_i, 1) > m_i - \frac{2m_i}{m_i - m_{i-1}}.$$

*Proof* By the assumption $L_0$ is a counterexample, so – using the Fact 2 (iii) – we get

$$W(VBIN_1) = \frac{m_{i-1}+1}{m_{i-1}} S(B_i^+, 1) + S(B_i, 1) + \frac{N(B_i, 1)}{m_{i+1} - 1} < 1. \tag{1}$$

Let $x$ be the first $B_i$-item that has not fit into the bin $VBIN_1$. Since $x$ did not fit into $VBIN_1$ and $x \leq \frac{1}{m_i - 1}$ the total sizes of the items in $VBIN_1$ is

$$S(B_i^+, 1) + S(B_i, 1) > \frac{m_i - 2}{m_i - 1} \tag{2}$$

Let us eliminate $S(B_i^+, 1)$ from the inequalities (1)and (2) multiplying (1) by $m_{i-1}$ and (2) by $-(m_{i-1} + 1)$. Adding these two inequalities we get

$$N(B_i, 1)\frac{m_{i-1}}{m_{i+1} - 1} - S(B_i, 1) < m_{i-1} - \frac{m_i - 2}{m_i - 1}(m_{i-1} + 1). \tag{3}$$

Since every $B_i$-item has size at most $\frac{1}{m_i - 1}$ and there are $N(B_i, 1)$ items in $VBIN_1$, therefore $S(B_i, 1) \leq \frac{N(B_i,1)}{m_i - 1}$. Substituting this into the inequality (3) we get

$$N(B_i, 1)(\frac{1}{m_i - 1} - \frac{m_i - 1}{m_{i+1} - 1}) > \frac{m_i - 2}{m_i - 1}(m_{i-1} + 1) - m_{i-1}, \tag{4}$$

and so

$$N(B_i, 1) > \frac{m_i - m_{i-1} - 2}{m_i - 1} \cdot \frac{m_i(m_i - 1)}{m_i - m_{i-1}} = \frac{m_i(m_i - m_{i-1} - 2)}{m_i - m_{i-1}}, \tag{5}$$

which yields the desired result. □

**Lemma 8** *Having packed the list $L_0$ by the algorithm NFFD-B3 there is at most one $B_i$-item, $i \geq 3$ in the $VBIN_2$ and $VBIN_3$ together.*

*Proof* Let $x$ be the first $B_i$-item that has not fit into the bin $VBIN_1$.
**Case A.** Let $r = 1$.
**Case A.1.** If $i = 3$ then the Lemma 7 states that $N(B_3, 1) > \frac{7}{2}$. Therefore $N(B_3, 1) \geq 4$. If there are at least two $B_3$ items in $VBIN_2$ and $VBIN_3$, we have at least $6 = m_3 - 1$ pieces of $B_3$ elements. These items can fit into one bin, so their total weight is at least 1. So, they form a good subset, which is a contradiction.

**Case A.2.** Let $i \geq 4$. Then $\frac{2m_i}{m_i - m_{i-1}} \leq 3$, therefore $N(B_i, 1) \geq m_i - 2$. Together with $x$ these $m_i - 1$ pieces of $B_i$ items form a good subset, which is a contradiction.

**Case B.** Let $r \geq 2$. Now, for every $i \geq 3$ we get $\frac{2m_i}{m_i - m_{i-1}} < 3$, and therefore $N(B_i) \geq m_i - 2$. So, we can use the same argument as in Case A.2. □

**Corollary 7** *After the Step(2) the bin $VBIN_2$ and $VBIN_3$ may contain at most one $B_3$ item if $r = 1$.*

**Lemma 9** *If $r = 1$ then after Step (2) $VBIN_3$ does not contain any $B_3$-items.*

*Proof* If $r = 1$ then from the Corollaries 2, 4, 5, 6 follows that in $VBIN_2$ and $VBIN_3$ a $B_3$-item is the largest item. We also know – see Corollary 7 – that at most one $B_3$-item is in bins $VBIN_2$ and $VBIN_3$. So, the $B_3$-item must be placed in the $VBIN_2$. □

**Lemma 10** *If $\min_{a \in L_0} s(a)$ is a $C_i$-item, $i \geq 3$, then either $VBIN_2$ is a good bin, or the algorithm can collect a good subset from the $C_i$-items in the virtual bins.*

*Proof* Let $x$ be the first $C_i$-item in $VBIN_3$. If there is no larger item in $L_0$ then $x$ then both virtual bins, $VBIN_1$ and $VBIN_2$, are $C_i$-homogeneous bins. So, we can suppose that there are larger items in $VBIN_1$ and $VBIN_2$.

Since $L_0$ is a counterexample, using the Fact 2 (iii) for the total weight of the items in the $VBIN_2$ we get

$$\frac{m_{i-1}+1}{m_{i-1}} S(C_i^+, 2) + S(C_i, 2) \frac{m_i + 1}{m_i} < 1. \tag{6}$$

Since $x$ did not fit into $VBIN_2$ and $x \leq \frac{1}{m_i}$ the total size of the items in $VBIN_2$

$$S(C_i^+, 2) + S(C_i, 2) > \frac{m_i - 1}{m_i} \tag{7}$$

Let us eliminate $S(C_i^+, 2)$ from the inequality system (7) and (6) by multiplying 7 by $-(m_{i-1} + 1)$ and 6 by $m_{i-1}$. Adding these two inequalities we get

$$S(C_i, 2) > \frac{m_i - m_{i-1} - 1}{m_i - m_{i-1}} = 1 - \frac{1}{m_i - m_{i-1}}. \tag{8}$$

Since every $C_i$-item has size at most $\frac{1}{m_i}$ and there are $N(C_i, 2)$ items in $VBIN_2$, therefore $N(C_i, 2) > m_i - \frac{m_i}{m_i - m_{i-1}} = m_i - 1 - \frac{m_{i-1}}{m_i - m_{i-1}} > m_i - 2$. So, there are at least $m_i - 1$ pieces of $C_i$ items in $VBIN_2$, and at least one $C_i$-item in $VBIN_3$. The sum of the sizes of these items is at most one, and the total weight of these items is at least one. So, we can construct a good subset again. □

With the help of the above Lemmas we have shown that if $L_0$ is a counterexample, ie. packing the items of $L_0$ by the algorithm *NFFD-B3* there is neither a good bin among the virtual bins nor a good subset, then $L_0$ may not contain items from any of the interval $X$, where $X \in \{A, B_i, C_i, D_i\}$, $i \geq 2$. Therefore $L_0$ must be an empty list, which completes the proof our main theorem. □

The algorithm *NFFD-B3* opens at least one bin in each iteration step, fills up the new opened bin(s) with items with total weight of at least one, and closes it. At the end the three virtual bins will be added to the number of used bins. So the following theorem is valid.

**Theorem 4** *Let $r$ be a positive integer, and $L_r$ be an arbitrary list with items $a_i \leq \frac{1}{r}$. Let us pack the items of $L$ by the algorithm NFFD-B3. Then*

$$\textit{NFFD-B3}(L) \leq W(L) + 3.$$

The Theorem 1 and Corollary 1 together yield that

$$R_\infty(\textit{NFFD-B3}) = h_\infty(r).$$

Our algorithm needs maximum $O(n \log n)$ operations to order the contents of the buffer, and $O(n)$ operations to find a good subset among the items in the buffer in each iteration step. Since the number of iterations is maximum $O(n)$, the time-complexity of the algorithm is $O(n^2 \log n)$.

## 5 Conclusions

In two earlier papers a bounded-space semi-online bin packing problem was considered. In papers [14] and [13] lower and upper bounds were given for the online algorithms for this problem. Both of the papers investigated the case $r = 2$. The best lower- and upper-bound was 1.4230 and 1.4243, respectively.

In this paper we defined an algorithm with asymptotic competitive ratio of $h_\infty(r)$ for any $r \geq 1$ integer. We also proved that these upper bounds are sharp for every $r$. Especially, for $r = 2$ this value is $h_\infty(2) = 1.423117\ldots$.

We used a buffer with capacity 3 in our algorithm. It is an open question whether the upper bound given in [14] is improvable if the buffer capacity is 2, since our algorithm does not work in this case. It is also a possibility for the further study, if we ask what kind of other preprocessing results in a better ACR for the NF-based algorithm?

## References

1. J. Balogh, J. Békési, G. Galambos, New Lower Bounds for Certain Classes of Bin Packing Algorithms, Theoretical Comp. Sci., **440-441**, 1–13, 2012.
2. J. Balogh, J. Békési, G. Galambos, G. Reinelt, Online bin packing with restricted repacking, *J. of Comb. Opt.,* **27(1)**, 115–131, 2014.
3. L. Epstein, E. Kleiman, Resource augmented semi-online bounded space bin packing. *Discrete Appl. Math.,* **157**, 2785-2798, 2009.
4. G. Galambos, Parametric Lower Bound for online Bin Packing, *SIAM J. on Alg. and Discr. Meth.,* **7**, 362–367, 1986.

5. G. Galambos, G. J. Woeginger, Repacking helps in bounded space online bin packing, *Computing,* **49**, 329–338, 1993.
6. E.F. Grove: Online bin packing with lookahead. In: Proceedings of the sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 430–436, 1995.
7. G. Gutin, T. Jensen, A. Yeo: Batched bin packing, *Discrete Optimization,* **2(1)**, 71–82, 2005.
8. S. Heydrich, R. van Stee: Beating the Harmonic Lower Bound for Online Bin Packing. In: Y. Rabani, I. Chatzigiannakis, M. Mitzenmacher and D. Sangiorgi, editors, ICALP 2016, vol. 55 of Leibniz International Proceedings in Informatics, pages 41:1-41:14, 2016.
9. D.S. Johnson: Near-optimal bin-packing algorithms. Doctoral Thesis, MIT, Cambridge, 1973.
10. S. Seiden, On the online bin packing problem, *Journal of ACM,* **49**, 640–671, 2002.
11. J. Sylvester, On a Point in the Theory of Vulgar Fractions, *American Journal of Mathematics,* **3**, 332–335, 1880.
12. A. van Vliet, An Improved Lower Bound for online Bin Packing Algorithms, *Inf. Proc. Letters,* **43**, 274–284, 1992.
13. M. Zhang, X. Han, Y. Lan, H-F. Ting, Online bin packing problem with buffer and bounded size revisited, *Journal of Combinatorial Optimization,* **33(2)**,530–542, 2017.
14. F. Zheng, L. Huo, E. Zhang, NF-based algorithms for online bin packing with buffer and bounded item size. *Journal of Combinatorial Optimization,* **30(2)**,360–369, 2015.