Bounded Space On-Line Variable-Sized Bin Packing*

Rainer E. Burkard†  Guochuan Zhang†

Abstract

In this paper we consider the k-bounded space on-line bin packing problem. Some efficient approximation algorithms are described and analyzed. Selecting either the smallest or the largest available bin size to start a new bin as items arrive turns out to yield a worst-case performance bound of 2. By packing large items into appropriate bins, an efficient approximation algorithm is derived from k-bounded space on-line bin packing algorithms and its worst-case performance bounds is 1.7 for $k \geq 3$.

Keywords: On-line, bin packing, approximation algorithm.

1. Introduction

In the one-dimensional classical bin packing problem, a list $L$ of items, i.e. numbers $a_i$ ($i = 1, \ldots, n$) in the range $(0,1]$, are to be packed into bins, each of which has a capacity 1, and the goal is to minimize the number of bins used. Since the problem of finding an optimal packing is NP-hard, research has focused on finding near-optimal approximation algorithms. The classical bin packing problem and many of its variations are of fundamental importance, reflected in the impressive amount of research reported [1].

A bin packing algorithm is on-line if it packs items $a_i$ solely on the basis of the sizes of the items $a_j$, $1 \leq j \leq i$ (i.e. the preceding items) and without any information on subsequent items.

For a list $L$ of items and an on-line algorithm $A$, let $s(A,L)$ and $s(OPT,L)$ denote the total size of bins used by algorithm $A$ and an optimal off-line algorithm, respectively. Then the worst-case performance bound of $A$ is defined as

$$S_A^\infty = \lim_{k \to \infty} \sup_L \{s(A,L)/s(OPT,L)|s(OPT,L) \geq k\}$$

In classical bin packing $s(A,L)$ is just the number of bins used by algorithm $A$ and $s(OPT,L)$ is the number of bins used by an optimal algorithm.

---

*This work was supported by Spezialforschungsbereich F 003 "Optimierung und Kontrolle", Projektbereich Diskrete Optimierung.
†TU Graz, Institut für Mathematik B, Steyrergasse 30, A-8010 Graz, Austria
In this paper, we pay our attention to the following two restrictions of on-line bin packing (For a rather complete survey on the worst case behaviour of on-line bin packing algorithms, see Galambos and Woeginger [7]).

**Bounded space algorithms**

We say, a bin becomes active (open), when it gets its first item. Once it is declared closed, it can never be active again. A bin packing algorithm uses $k$-bounded space if for each item $a_i$, the choice for the bin to pack it is restricted to a set of $k$ or fewer active (open) bins.

Lee and Lee [10] proved that for every bounded space on-line bin packing algorithm $A$, $S^\infty_A \geq h_\infty = \sum_{i=1}^{\infty} 1/t_i \approx 1.69103$, where

$$t_1 = 1, t_{i+1} = t_i(t_i - 1), \text{ for } i \geq 1.$$  

Galambos and Woeginger [6] even proved that the bound $h_\infty$ could not be beaten by repacking.

Essentially, the following six types of bounded space on-line bin packing approximation algorithms have been studied.

(i) The first fit first close algorithm $NkF$ ($k \geq 2$) is a simple extension of the Next Fit algorithm (Johnson [8]). Csirik and Imreh [3] constructed lists of items for which $NkF$ is a factor $17/10 + 3/(10k - 10)$ away from the optimum. Mao [11] proved that this indeed is the worst that can happen. Hence $S^\infty_{NkF} = 17/10 + 3/(10k - 10)$ holds.

(ii) Mao [12] showed for the best fit first close algorithm $ABF_k$ ($k \geq 2$) with bounded space $k$ the performance bound $S^\infty_{ABF_k} = 17/10 + 3/(10k)$.  

(iii) The best fit best close algorithm $BBF_k$ ($k \geq 2$) was introduced by Csirik and Johnson [4]. They showed in a very sophisticated proof that "best is better than first", since independently of the value of $k$, always $S^\infty_{BBF_k} = 17/10$ holds.

(iv) Zhang [15] showed that for the first fit best close algorithm $AFB_k$ ($k \geq 2$) which was also introduced by Csirik and Johnson [4], $S^\infty_{AFB_k} = 17/10 + 3/(10k - 10)$ holds.

(v) The HARMONIC algorithm $HARM_k$ by Lee and Lee [10]. They showed that as $k$ tends to infinity, $S^\infty_{HARM_k}$ tends to the number $h_\infty$.

(vi) The SIMPLIFIED HARMONIC algorithm $SH_k$ by Woeginger [14], works similar to the HARMONIC algorithm but uses another (more complicated) partition of the interval $(0,1)$. Moreover, $S^\infty_{SH_k} \leq S^\infty_{HARM_k}$ for each $k \geq 2$.

**Variable-sized on-line bin packing**

Only few results are known concerning the more general problem in which bins need not be of a single given size [2,5,9,13,16]. The variable-sized bin packing problem is a variant of the classical bin packing, in which bin capacities may vary. We are given a list $L$ of items, and several different types $B^1, \ldots, B^l$ of bins with sizes $1 = s(B^1) > s(B^2) > \cdots > s(B^l) > 0$. There is an inexhaustible supply of bins of each size. The goal is to pack the given items into the bins so that the sum of the sizes of the bins used is minimum. Observe that for the case that all bins
are of size one, this is just the classical one dimensional bin packing problem. This model is considerably more realistic than that of the classical problem.

In the on-line version of variable-sized bin packing, we cannot preview and rearrange the items of \( L \) before packing is started, but must instead accept and immediately pack each item as it arrives.

Friesen and Langston [5] gave three approximation algorithms with worst-case performance bounds of 2, 3/2, and 4/3. Only the first of these algorithms is on-line. Essentially, it is a simple modification of Next Fit and also has the same worst-case performance bound 2 as Next Fit. An off-line fully polynomial time approximation scheme has been devised by Murgolo [13] using a linear programming formulation of the problem. Kinnersley and Langston [9] presented fast on-line algorithms \( FFf \) for the variable-sized bin packing. They devised a scheme based on a user specific factor \( f \geq \frac{1}{2} \) and proved that their strategy guarantees a worst-case performance bound not exceeding \( 1.5 + f/2 \geq 1.75 \). By choosing \( f = 1/2 \), \( FFH \), the best among \( FFf \) algorithms, is obtained. Zhang [16] proved that the tight bound of \( FFH \) is 1.7, the same bound as the First Fit algorithm in the classical bin packing. Csirik [2] derived an algorithm with worst-case performance bound of \( < 1.7 \) from the Harmonic Fit algorithm. To our knowledge, Csirik's algorithm is still the best up to now, for a short discussion see Section 4.

In this paper, we consider algorithms for on-line variable-sized bin packing problem with the added constraint that the algorithms can assign items only to one of \( k \) bins at a time. Two simple algorithms with bounds 2 are presented in Section 2. Section 3 analyses an algorithm with worst-case performance bound of 1.7 (for \( k \geq 3 \)), which derived from bounded space on-line bin packing.

# 2. Some Simple Algorithms

When we design an algorithm for \( k \)-bounded space on-line variable-sized bin packing, we must answer the following questions.

- How to select the bin size when a new bin is required?
- Which bin among the \( k \) active bins is chosen for packing \( a_i \)?
- Which bin among the \( k \) active bins is closed when a new bin has to be created for \( a_i \)?

For \( k \)-bounded space on-line bin packing, Csirik and Johnson [4] presented two packing rules and two closing rules. They are listed as follows.

**P-FF** Pack the current item \( a_i \) into the lowest indexed active bin that has enough space for it. Otherwise, open a new bin and place \( a_i \) in it.

**P-BF** Pack the current item \( a_i \) into the fullest active bin that has enough space for it. Otherwise open a new bin and place \( a_i \) in it.

**C-FF** Close the lowest indexed active bin.

**C-BF** Close the fullest active bin (with ties broken in favor of the lowest indexed bin).

We use \( c(B) \) to denote the sum of the items in \( B \). Given a list of \( L = (a_1, \ldots, a_n) \), let \( B_1, \ldots, B_m \) denote the list of bins ordered in a \( k \)-bounded space
on-line variable-sized algorithm. Let $ALk$ (always largest) denote the algorithm which always uses only bins of size 1, i.e., bins of largest size, packs items using the P-FF or the P-BF rule, and closes bins using the C-FF or the C-BF rule.

**Theorem 2.1.** $s(ALk, L) < 2s(OPT, L) + 1$, $k \geq 1$, for any list $L$.

**Proof.** For $1 \leq i \leq m - 1$, due to our packing rule $c(B_i) + c(B_{i+1}) > 1$. So,

$$s(ALk, L) = (m - 1) + 1 < 2 \sum_{i=1}^{m} c(B_i) + 1$$

$$= 2 \sum_{i=1}^{n} a_i + 1 \leq 2s(OPT, L) + 1. \square$$

Any list consisting of items of size $\frac{1}{2} + \varepsilon$ and bins of size 1 and $\frac{1}{2} + \varepsilon$ for some arbitrarily small $\varepsilon > 0$, demonstrates that the bound of 2 is asymptotically tight for $ALk$.

While the worst-case behavior of Best Fit is superior to that of First Fit for the $k$-bounded space on-line bin packing problem, this is not the case for $ALk$ where always the largest possible bins in variable-sized bin packing algorithm are used.

Now we consider the algorithm $ASk$ (always smallest) which uses smallest possible bins, packs items using the P-FF or the P-BF rule, and closes bins using the C-FF or the C-BF rule.

**Theorem 2.2.** $s(ASk, L) < 2s(OPT, L) + 1$, $k \geq 1$, for any list $L$.

**Proof.** For $1 \leq i \leq m - 1$, $c(B_i) + c(B_{i+1}) > s(B_i)$. Therefore, we have

$$s(ASk, L) = \sum_{i=1}^{m} s(B_i) < 2 \sum_{i=1}^{m} c(B_i) - c(B_1) - c(B_m) + s(B_m)$$

$$< 2 \sum_{i=1}^{n} a_i + 1 \leq 2s(OPT, L) + 1. \square$$

Any list consisting of items of size $\frac{1}{2}$ and bins of size 1 and $1 - \varepsilon$ for some arbitrarily small $\varepsilon > 0$, demonstrates that the bound of 2 is asymptotically tight for $ASk$.

3. Algorithms Derived from $k$-Bounded Space On-Line Bin Packing

We start from the open, the packing and the closing rule.

Suppose that $a_i$ is a large item (with size greater than $1/2$). If it can be contained in a bin with size less than 1, it is called a B-item, else it is called an L-item. The smallest bin which can contain a large item $a_i$ is called an $a_i$-home-bin. Obviously, if $a_i$ is an L-item, then the size of the $a_i$-home-bin is 1.

**Open rule:** Suppose that the current item to be packed is $a_i$. If $a_i$ is a B-item, then open an $a_i$-home-bin and pack $a_i$ into it. Otherwise, start a new bin of size 1 for $a_i$. 
Packing rules: P-FF and P-BF.

Closing rules:
C-VF: Close one active bin with size less than 1 if such a bin exists, otherwise use C-FF.
C-VB: Close one active bin with size less than 1 if such a bin exists, otherwise use C-BF.

Since we only have one open rule, we always use it to start a new bin. For any combination of a packing rule with a closing rule, we have four algorithms. The combination of P-FF with C-VF yields $VFF_k$ and the combination of P-BF with C-VB yields $VBB_k$. Let $VBF_k$ denote the (P-BF, C-VF) combination and $VFB_k$ denote the (P-FF, C-VB) combination. In the following, we only analyze $VBB_k$ algorithm. For the others, some remarks are given in the next section.

**Theorem 3.1.** For any list $L$, we have

$$S_{VBB_k} \leq 1.7, \text{ for } k \geq 3.$$

Obviously, if we only have one type of bins, $VBB_k$ is just $BBF_k$. From [4], we have $S_{VBB_k} \geq 1.7$, for $k \geq 3$.

We prove that the lower bound is tight with the help of the weighting function defined as follows.

We can divide all items in list $L$ into 5 parts.

$A_1 = \{a \in L \mid 0 < a \leq 1/6 \}$, $A_2 = \{a \in L \mid 1/6 < a \leq 1/3 \}$, 
$A_3 = \{a \in L \mid 1/3 < a \leq 1/2 \}$, $A_4 = \{a \in L \mid a \text{ is a B-item} \}$, 
$A_5 = \{a \in L \mid a \text{ is an L-item} \}$.

An item is called an $A_i$-item if it belongs to $A_i$, $i = 1, 2, 3, 4, 5$. $A_4$-items and $A_5$-items are large items.

Let us define a weighting function as follows.

$$W(a) = \begin{cases} 
(6/5)a, & \text{if } a \in A_1, \\
(9/5)a - 1/10, & \text{if } a \in A_2, \\
(6/5)a + 1/10, & \text{if } a \in A_3, \\
\max\{1.7a, s(\hat{B})\}, & \text{if } a \in A_4, \\
(6/5)a + 4/10, & \text{if } a \in A_5,
\end{cases}$$

where $\hat{B}$ is the $a$-home-bin. $W(B)$, the weight of the bin $B$, is defined to be the sum of the weight of all items in bin $B$, i.e., $W(B) = \sum_{a_i \in B} W(a_i)$. And $W(L)$, the weight of the list $L$, is defined to be the sum of the weight of all items in $L$, i.e., $W(L) = \sum_{i=1}^{n} W(a_i)$. We are going to show that

$$s(VBB_k, L) - 4/5 \leq W(L) \leq 1.7s(OPT, L).$$

**Lemma 3.1.** For any list $L$, we have
Proof. Similar as in [16].

Lemma 3.2. For any list $L$, we have

$$W(L) \geq s(VBB_k, L) - 4/5, \text{ for } k \geq 3.$$  

Proof. It is obvious that at any time the size of at most one current active bin is not greater than $1/2$, when we use $VBB_k$. Note that except the last $k$ bins, all of the bins used in a $VBB_k$ packing are declared closed one by one. These bins are more than 1/2 full when they declared closed. In our analysis we first investigate the closed bins, thereafter we turn our attention to the last $k$ bins.

Claim 3.1. If a bin contains one $A_5$-item or two $A_3$-items, then its weight is not less than 1. If $B_i$ is used in a $VBB_k$ packing and $s(B_i) < 1$, then $W(B_i) \geq s(B_i)$.

Proof. It is trivial. □

We shall analyze the packing as one active bin is to be closed. This active bin is called the currently closed bin. The case that the currently closed bin is not the lowest indexed active bin (the first active bin) will be considered in Claim 3.2. Further cases are treated in Claims 3.3 and 3.4.

Claim 3.2. If the currently closed bin $B_i$ is not the first active bin for a $VBB_k$ packing, then $W(B_i) \geq s(B_i)$.

Proof. Without loss of generality, let the current active bins be $B_1, \ldots, B_i, \ldots, B_k$, $i \neq 1$. By Claim 3.1 and inspection, we can assume that $s(B_i) = 1$ and $B_i$ contains two items at least, and no $A_5$-item, one $A_2$-item at most. From the algorithm, $B_i$ is the fullest bin at this time. If $c(B_1) \geq 5/6$, it is easy to see that $c(B_i) \geq c(B_1)$ and $W(B_i) \geq 1$. In the following, we only consider the case $B_1 < 5/6$. If $B_i$ contains one $B$-item, $B_i$ must contain another item $\alpha$ which can not be placed into $B_1$, i.e., $\alpha > 1/6$. Therefore we have $W(B_i) > 1.7(1/2) + (6/5)(1/6) > 1$. If $B_i$ contains no $B$-item, we can also assume that $c(B_1) > 2/3$, otherwise $B_i$ will belong to the special cases in Claim 3.1.

We only need to consider the two bottommost items of $B_i$, $\alpha$ and $\beta$.

Case 1. $\alpha \in A_2$, $\beta \in A_3$,

$$W(B_i) \geq \frac{6}{5}c(B_i) + \frac{3}{5}\alpha \geq \frac{6}{5}c(B_i) + \frac{3}{5}(1 - c(B_1)) \geq \frac{6}{5}c(B_1) + \frac{3}{5} - \frac{3}{5}c(B_1) > \frac{3}{5} \cdot 2 + \frac{3}{5} = 1.$$  

Case 2. $\alpha \in A_2$, $\beta \in A_2$,

$$W(B_i) > \frac{6}{5}c(B_i) + \frac{3}{5}(1 - c(B_1)) \cdot 2 - \frac{1}{10} \cdot 2 \geq 1. \square$$

In the following, we assume that the currently closed bin is the first active bin as it is declared closed.
Claim 3.3. For a VBB\(_k\) packing, the currently closed bin \(B_i\) is just the first active bin and the next active bin is \(B_{i+1}\). If \(s(B_i) = s(B_{i+1}) = 1\) and \(c(B_i) \geq c(B_{i+1}) > \frac{1}{2}\), then we have, when \(B_{i+1}\) contains no \(B\)-item,

\[
(6/5)c(B_i) + W(B_{i+1}) \geq 1 + (6/5)c(B_{i+1})
\]  

and when \(B_{i+1}\) contains one \(B\)-item,

\[
(6/5)c(B_i) + W(B_{i+1}) > 2.
\]

**Proof.** When \(B_{i+1}\) contains one \(B\)-item, \(B_{i+1}\) must also contain at least one small item, say \(\beta\), which has been accepted before the \(B\)-item. This follows from the fact that only bins with size < 1 can accept \(B\)-items as first items, but \(s(B_{i+1}) = 1\). Clearly, \(c(B_i) + \beta > 1\), otherwise \(\beta\) should be placed in bin \(B_i\). Moreover, \(w(B_{i+1}) > \frac{6}{5}\beta + 1.7 \cdot \frac{1}{2}\). The last two inequalities imply

\[
\frac{6}{5}c(B_i) + W(B_{i+1}) > \frac{6}{5} + 1.7(1/2) > 2.
\]

When \(B_{i+1}\) contains no \(B\)-item, we can assume that \(\frac{1}{2} < c(B_i) < \frac{5}{6}\) and \(B_{i+1}\) contains no \(A_5\)-item and one \(A_3\)-item at most. Otherwise (1) is clear. Thus,

**Case 1.** \(\frac{5}{6} > c(B_i) \geq \frac{2}{3}\)

In this case, every item in \(B_{i+1}\) must be greater than \(\frac{1}{6}\).

If \(B_{i+1}\) contains one \(A_3\)-item,

\[
\frac{6}{5}c(B_i) + W(B_{i+1}) > \frac{6}{5}c(B_i) + \frac{6}{5}c(B_{i+1}) + \frac{3}{5}(1 - c(B_i))
\]

\[
= \frac{3}{5}c(B_i) + \frac{6}{5}c(B_{i+1}) + \frac{3}{5}
\]

\[
\geq \frac{6}{5}c(B_{i+1}) + \frac{3}{5} \cdot \frac{2}{3} + \frac{3}{5}
\]

\[
= 1 + \frac{6}{5}c(B_{i+1}).
\]

If \(B_{i+1}\) contains no \(A_3\)-item then it is easy to see that \(B_{i+1}\) contains two \(A_2\)-items at least. Therefore

\[
\frac{6}{5}c(B_i) + W(B_{i+1}) > \frac{6}{5}c(B_i) + \frac{6}{5}c(B_{i+1}) + \frac{3}{5}(1 - c(B_i)) \cdot 2 - \frac{1}{10} \cdot 2
\]

\[
= \frac{6}{5}c(B_{i+1}) + \frac{6}{5} - \frac{1}{5}
\]

\[
= 1 + \frac{6}{5}c(B_{i+1}).
\]

**Case 2.** \(\frac{5}{6} > c(B_i) > \frac{1}{2}\)

In this case, every item in \(B_{i+1}\) is greater than \(\frac{1}{3}\), i.e., belongs to \(A_3\) or \(A_5\). Therefore, \(B_{i+1}\) must contain one \(A_5\)-item or two \(A_3\)-items. From Claim 3.1, \(W(B_{i+1}) \geq 1\) but this is the easy case mentioned above. So, Claim 3.3 holds. □
Claim 3.4. For a $VBB_k$ packing $(k \geq 3)$, the currently closed bin $B_i$ is just the first active bin and the next active bin is $B_{i+1}$. Assume that $s(B_i) = 1$ and $c(B_i) < \frac{5}{6}$. Assume, moreover, that the sum of items in $B_{i+1}$ is currently $\leq \frac{1}{2}$, but when the $VBB_k$ packing is finished, $c(B_{i+1}) > 1/2$. Suppose that $B_j$ with $s(B_j) = 1$ is the active bin next to $B_{i+1}$, when $B_{i+1}$ accepts a new item $\gamma$ after $B_i$ has been closed.

(i) If $B_j$ is closed before $B_{i+1}$, then

$$\frac{6}{5}c(B_i) + W(B_{i+1}) + W(B_j) \geq 2 + \frac{6}{5}c(B_{i+1}).$$

(ii) If $B_{i+1}$ is closed before $B_j$, then

$$\frac{6}{5}c(B_i) + W(B_{i+1}) + W(B_j) \geq 2 + \frac{6}{5}c(B_j).$$

Proof. If $B_{i+1}$ contains at least two items when $B_i$ is closed, then it is easy to prove that $\frac{6}{5}c(B_i) + W(B_{i+1}) \geq 1 + \frac{6}{5}c(B_{i+1})$ and (3) or (4) hold. Hence we only have to consider the case that $B_{i+1}$ contains just one item $\alpha$, when $B_i$ is closed. We suppose that the bottommost item of $B_j$ is $\beta$. Then $\alpha$ is greater than $\frac{1}{6}$ (if not, $\alpha$ can be placed into $B_i$) and $\beta > 1 - \alpha \geq \frac{1}{2}$. Afterwards, an item $\gamma$ is placed into $B_{i+1}$ and $c(B_j) + \gamma > 1$.

(i) $B_j$ is closed before $B_{i+1}$.

• If $\frac{1}{6} < \alpha \leq \frac{1}{3}$, then, $c(B_j) \geq \beta > 1 - \alpha \geq \frac{2}{3}$, $c(B_i) > \frac{2}{3}$.

$$\frac{6}{5}c(B_i) + W(B_j) + W(B_{i+1})$$

$$> \frac{6}{5}(c(B_i) + c(B_j)) + \frac{2}{5} + \frac{6}{5}c(B_{i+1})$$

$$> \frac{6}{5} \cdot \frac{4}{3} + \frac{2}{5} + \frac{6}{5}c(B_{i+1})$$

$$= 2 + \frac{6}{5}c(B_{i+1}).$$

• If $\frac{1}{3} < \alpha \leq \frac{1}{2}$ and $\frac{2}{3} < c(B_j) < \frac{5}{6}$, then, $\gamma > \frac{1}{6}$ and $\gamma + c(B_j) > 1$.

When $\frac{1}{6} < \gamma \leq \frac{1}{3}$,

$$\frac{6}{5}c(B_i) + W(B_j) + W(B_{i+1})$$

$$\geq \frac{6}{5}c(B_i) + \frac{6}{5}c(B_j) + \frac{2}{5} + \frac{6}{5}c(B_{i+1}) + \frac{3}{5} \gamma$$

$$> \frac{6}{5}\cdot \frac{1}{2} + \frac{3}{5} \cdot \frac{2}{3} + \frac{2}{5} + \frac{3}{5} + \frac{6}{5}c(B_{i+1})$$

$$= 2 + \frac{6}{5}c(B_{i+1}).$$
When $\frac{1}{3} < \gamma$,

$$\frac{6}{5} c(B_i) + W(B_j) + W(B_{i+1})$$

$$\geq \frac{6}{5} c(B_i) + \frac{6}{5} c(B_j) + \frac{2}{5} + \frac{6}{5} c(B_{i+1}) + \frac{2}{10}$$

$$> \frac{6}{5} \cdot \frac{1}{2} + \frac{6}{5} \cdot \frac{2}{3} + \frac{2}{5} + \frac{1}{5} + \frac{6}{5} c(B_{i+1})$$

$$= 2 + \frac{6}{5} c(B_{i+1}).$$

- If $\frac{1}{3} < \alpha \leq \frac{1}{2}$ and $\frac{1}{2} < c(B_j) \leq \frac{2}{3}$, then, $\gamma > \frac{1}{3}$, $c(B_{i+1}) \geq \alpha + \gamma > \frac{2}{3}$ and $c(B_j) \geq c(B_{i+1})$. But this is impossible.

- If $\frac{5}{6} \leq c(B_j)$, then

$$\frac{6}{5} c(B_i) + W(B_j) + W(B_{i+1})$$

$$\geq \frac{6}{5} c(B_i) + \frac{6}{5} c(B_j) + \frac{2}{5} + \frac{6}{5} c(B_{i+1})$$

$$> \frac{6}{5} \cdot \frac{1}{2} + \frac{6}{5} \cdot \frac{5}{6} + \frac{2}{5} + \frac{6}{5} c(B_{i+1})$$

$$= 2 + \frac{6}{5} c(B_{i+1}).$$

(ii) $B_{i+1}$ is closed before $B_j$.

It is clear that $c(B_{i+1}) \geq c(B_j)$ when $B_{i+1}$ is closed. If $c(B_{i+1}) < \frac{2}{3}$ at this time, then $\beta \leq c(B_j) < \frac{2}{3}$. This implies that $\alpha > \frac{1}{3}$ and $\gamma > \frac{1}{3}$, i.e., $c(B_{i+1}) > \frac{2}{3}$. This is a contradiction. Therefore, we have $c(B_{i+1}) \geq \frac{2}{3}$.

- If $\frac{1}{6} < \alpha \leq \frac{1}{3}$, then

$$\frac{6}{5} c(B_i) + W(B_{i+1}) + W(B_j)$$

$$> \frac{6}{5} c(B_i) + \frac{6}{5} c(B_{i+1}) + \frac{3}{5} \alpha - \frac{1}{10} + \frac{2}{5} + \frac{6}{5} c(B_j)$$

$$> \frac{6}{5} \cdot \frac{2}{3} + \frac{3}{5} \cdot \frac{1}{2} - \frac{1}{10} + \frac{2}{5} + \frac{6}{5} c(B_j)$$

$$= 2 + \frac{6}{5} c(B_j).$$
• If $\frac{1}{3} < \alpha \leq \frac{1}{2}$ and $\frac{1}{6} < \gamma \leq \frac{1}{3},$ then $\frac{2}{3} < c(B_j).$

$$
\frac{6}{5} c(B_i) + W(B_{i+1}) + W(B_j) \\
\geq \frac{6}{5} c(B_i) + \frac{6}{5} c(B_j) + \frac{2}{5} + \frac{6}{5} c(B_{i+1}) + \frac{3}{5} \gamma \\
> \frac{6}{5} \cdot \frac{1}{2} + \frac{6}{5} c(B_j) + \frac{2}{5} + \frac{3}{5} (1 - c(B_j)) + \frac{6}{5} c(B_j) \\
> \frac{3}{5} + \frac{3}{5} + \frac{2}{5} + \frac{6}{5} c(B_j) \\
= 2 + \frac{6}{5} c(B_j).
$$

• If $\frac{1}{3} < \alpha \leq \frac{1}{2}$ and $\frac{1}{3} < \gamma,$ then

$$
\frac{6}{5} c(B_i) + W(B_{i+1}) + W(B_j) \\
\geq \frac{6}{5} c(B_i) + \frac{6}{5} c(B_j) + \frac{2}{5} + \frac{6}{5} c(B_{i+1}) + \frac{2}{10} \\
> \frac{6}{5} \cdot \frac{1}{2} + \frac{6}{5} c(B_j) + \frac{2}{5} + \frac{1}{5} + \frac{6}{5} c(B_j) \\
= 2 + \frac{6}{5} c(B_j).
$$

• If $\frac{1}{3} < \alpha \leq \frac{1}{2}$ and $\gamma \leq \frac{1}{6},$ then $c(B_{i+1}) \geq c(B_j) > \frac{5}{6}.$

$$
\frac{6}{5} c(B_i) + W(B_{i+1}) + W(B_j) \\
> \frac{6}{5} c(B_i) + \frac{6}{5} c(B_{i+1}) + \frac{2}{5} + \frac{6}{5} c(B_j) \\
> \frac{6}{5} \cdot \frac{1}{2} + \frac{6}{5} \cdot \frac{5}{6} + \frac{6}{5} c(B_j) \\
= 2 + \frac{6}{5} c(B_j).
$$

Thus Claim 3.4 holds. □

**Note 3.1.** If the hypothesis in Claim 3.4 does not hold, i.e., when the VBB$_k$ packing ends, $B_{i+1}$ is not greater than $\frac{1}{2}$ yet, then $B_{i+1}$ belongs to the last $k$ bins.

Now, we consider the last $k$ bins. Without loss of generality, suppose $S_1 = \{B_1, \ldots, B_m\}$ is the set of those closed bins (of size one) which are mentioned in Claim 3.3 and Claim 3.4. And suppose that $B_i$ is closed before $B_{i+1},$ $i = 1, \ldots, m - 1.$ We also denote by $S_2 = \{B_{m+1}, \ldots, B_{m+t}\},$ $t \leq k$ those bins among the last $k$ bins, whose weights are less than their sizes when the packing ends.
Note 3.2. If the size of the first one of the last k bins is less than 1, then all of the closed bins have their sizes less than 1, i.e., $S_1 = \emptyset$, by the closing rule of our algorithm.

To see this, if there are some closed bins with size 1, let $B_p$ denote the last closed bin of size 1. After $B_p$ is closed, the first active bin may not be a bin with size less than 1 by our algorithm. It is a contradiction.

Claim 3.5. Each bin $B$ used in $VBB_k$ packing but not in $S_1 \cup S_2$ has a weight $W(B)$ not less than $s(B)$.

Proof. Follows immediately from Claim 3.1 and Claim 3.2.

Claim 3.6. If the case in Note 3.1 happens, then

$$
\begin{cases}
\sum_{i=1}^{m+1} W(B_i) \geq m + (6/5)c(B_{m+1}), & \text{if } B_{m+1} \text{ contains no B-item,} \\
\sum_{i=1}^{m+1} W(B_i) \geq m + 1, & \text{if } B_{m+1} \text{ contains one B-item.}
\end{cases}
$$

Proof. Claim 3.6 follows directly from Claim 3.3 and Claim 3.4. □

Similarly, we get

Claim 3.7. If the case in Note 3.1 does not happen, then

$$
\sum_{i=1}^{m+1} W(B_i) \geq m - 1 + (6/5)c(B_m) + W(B_{m+1}) \text{ for a } VBB_k \text{ packing.}
$$

In the following, we consider $S_2$.

Claim 3.8. When $B_{m+1}$ contains one B-item,

$$W(B_{m+2}) + \cdots + W(B_{m+t}) \geq t - 1 - 4/5.$$

Proof. We will consider two cases.

Case 1. $c(B_{m+2}), \ldots, c(B_{m+t}) > 1/2$.

(i) If both $B_{m+i+1}$ and $B_{m+i+2}$ contain one B-item each, $1 \leq i \leq t - 2$, then

$$W(B_{m+i+1}) + W(B_{m+i+2}) > 2.$$

(ii) If $B_{m+i+1}$ contains no B-item and $B_{m+j+1}$ contains a B-item $\beta$, $1 \leq i < j \leq t - 2$ then

$$(6/5)c(B_{m+i+1}) + W(B_{m+j+1}) > 2.$$ 

(iii) If $B_{m+i+1}$ and $B_{m+j+1}$ contain no B-item, $1 \leq i < j \leq t - 2$ then

$$(6/5)c(B_{m+i+1}) + W(B_{m+j+1}) \geq 1 + (6/5)c(B_{m+j+1}).$$
(i), (ii) and (iii) can be proved in the similar way as in the proof of Claim 3.3. Therefore,

$$W(B_{m+2}) + \cdots + W(B_{m+t}) > t - 3 + \frac{1}{2} \cdot \frac{17}{10} + \frac{1}{2} \cdot \frac{6}{5} = (t - 1) - \frac{11}{20}.$$  

**Case 2.** There exists a bin \( c(B_{m+j}) \leq 1/2, 2 \leq j \leq t. \) In this case, all bins following \( B_{m+j} \) contain one L-item, i.e., have their weights greater than 1. Thus it is easy to show that

$$W(B_{m+2}) + \cdots + W(B_{m+t}) > (t - 1) - 4/5. \quad \Box$$

Similarly, it can be shown

**Claim 3.9.** When \( c(B_{m+1}) > 1/2 \) contains no B-item,

$$(6/5)c(B_{m+1}) + W(B_{m+2}) + \cdots + W(B_{m+t}) \geq t - 4/5. \quad \Box$$

**Claim 3.10.** If \( c(B_{m+1}) \leq 1/2, \) when the packing ends, then

$$\sum_{i=m+1}^{m+t} W(B_i) + (6/5)c(B_m) > t + 1 - 4/5.$$  

**Proof.** If \( c(B_{m+1}) \leq 1/2, \) when the packing ends, then \( B_j \) contains one \( A_5 \)-item for \( j = m + 2, \ldots, m + t. \) Therefore,

$$\sum_{i=m+1}^{m+t} W(B_i) + (6/5)c(B_m) > t - 1 + (6/5)(c(B_m) + c(B_{m+1})) > t - 4/5. \quad \Box$$

By Claims 3.5 to 3.10, we get \( W(L) \geq s(VBB_k, L) - 4/5. \) This completes the proof of Lemma 3.1. \( \quad \Box \)

Combining Lemma 3.1 with Lemma 3.2, we have \( S_{VBB_k}^\infty \leq 1.7, \) for \( k \geq 3. \) Therefore, we conclude that

$$S_{VBB_k}^\infty = 1.7, \quad k \geq 3.$$  

Thus Theorem 3.1 is proven. \( \quad \Box \)

**4. Conclusions and Remarks**

This paper deals with an on-line variable-sized bin packing problem which uses bounded space. Up to now, the best bound of the known on-line variable-sized bin packing algorithms [2] is a bit smaller than 1.7. In fact, the algorithm so-called \( VH_M \) in [2] which is derived from Harmonic Fit is a bounded space algorithm. Let \( M > 1 \) be a positive integer and let \( M_j = \lfloor M \cdot s(B') \rfloor \) \( (j = 1, 2, \ldots, l). \) Then the algorithm uses \( k \)-bounded space where

$$k = M_1 + M_2 + \cdots + M_l - l + 1.$$
When \( M_l \leq 5 \), the worst-case performance bound of \( VH_M \) is greater than 1.7. If and only if \( M_l \geq 7 \), \( VH_M \) can do better than \( VBB_k \) where \( k \geq 6l + 1 \). To see \( VBB_k \) is efficient, we observe that it only needs \( k \geq 3 \) to reach the bound 1.7 which does not depend on the number \( l \) of bin sizes.

We can also analyse the other three algorithms \( VFB_k \), \( VFF_k \) and \( VBF_k \) with the similar technique. For example, with a modified weighting function

\[
W^*(a) = \begin{cases} 
\max\{1.7a, s(\hat{B})\} + 3/(10k - 10), & \text{if } a \in A_4, \\
(6/5)a + 4/10 + 3/(10k - 10), & \text{if } a \in A_5, \\
W(a), & \text{otherwise}
\end{cases}
\]

where \( W(a) \) is defined in Section 3, we can prove that \( S^\infty_{VFB_k} = 1.7 + \frac{3}{10(k-1)} \), for \( k \geq 2 \). It is clear that all the three algorithms can not beat algorithm \( VBB_k \).

**Acknowledgement**

The authors wish to thank the anonymous referee for his helpful comments on an earlier version of this paper.

**References**


Received March, 1996