Performance Estimations of First Fit Algorithm for Online Bin Packing with Variable Bin Sizes and LIB constraints

J.Y. Lin* P. Manyem† and R.L. Sheu‡

Abstract

We consider the NP Hard problem of online Bin Packing while requiring that larger (or longer) items be placed below smaller (or shorter) items — we call such a version the LIB version of problems. Bin sizes can be uniform or variable. We provide analytical upper bounds as well as computational results on the asymptotic approximation ratio for the first fit algorithm.

Keywords. Online approximation algorithm, asymptotic worst case ratio, bin packing problem, longest item, uniform sized bins, variable sized bins.

1 Background

In the classical one-dimensional Bin Packing problem, we are given a list $L = (i : 1 \leq i \leq n)$ of items. The *size* of item i is a_i, where each $a_i \in (0, 1]$. The problem is to pack these n items into bins such that the number of bins used is minimized. A bin is said to be *used* if it contains at least one item (of non-zero length). A feasible solution is one where the sum of the sizes of the items in each used bin is at most equal to the bin size.

VSBP (Variable Sized Bin Packing Problem) is similar to the classical problem stated above, except that the bin sizes can be different — we are given a collection $\mathcal{B}$ of distinct bin sizes s_1 through s_K, and s_K is the largest (or just *longest*, in the one-dimensional case) bin size with $s_K = 1$. Size s_1 is the smallest. The objective is to minimize the sum of the sizes of the bins used. The dual of Bin Packing is *Bin Covering*, where the item sizes in a bin should total up to *at least* the bin size.

Bin Packing can be offline or online. If the sizes of all items are known in advance, this is referred to as *offline* bin packing. In the *online* version of Bin Packing, items in L arrive one by one. When an item i of length a_i arrives, it must immediately be assigned to a bin (and this assignment cannot be changed later), and the length a_{i+1} of the next item becomes known only after item i has been assigned to its bin. In all versions of Bin Packing, it is assumed that there is an infinite supply of bins of any size. Hence, running out of bins to place items is never an issue.

LIB version of Bin Packing. The bin packing problem considered in this paper is an online version with variable bin sizes and imposes this additional requirement: In any bin,

*National Cheng-Kung University, Tainan, Taiwan. Email:jylin@math.ncku.edu.tw

†Centre for Informatics and Applied Optimisation, University of Ballarat. Email:p.manyem@ballarat.edu.au

‡National Cheng-Kung University, Tainan, Taiwan. Email:rsheu@mail.ncku.edu.tw

Basic Problem	LIB ?	Bin Sizes	Upper Bound	Lower Bound
Bin Packing	No	Uniform	1.59 [9]	1.53 [10]
	No	Variable	1.7 [3]	1.0 (trivial)
	Yes	Uniform	3.0 [6]	unknown
	Yes	Variable	B_{FF}	unknown

Table 1: Bounds on Approximation Ratios in online Bin Packing and Covering with LIB

for any pair of items i and j, if size$(j) = a_j >$ size$(i) = a_i$, then j should be placed in the bin *below* i. In other words, *longer* items should be placed lower in any bin than *shorter* items. We call this the LIB version, for *Longest Item at the Bottom*. Moreover, we assume that the length of each item in L can not be arbitrarily small. Namely, there is a $0 < \gamma \leq s_1$ such that $a_i \in [\gamma, 1]$ for all $i \in L$.

Table 1 summarizes the results known so far in online Bin Packing. The numbers in square brackets refer to the bibliography. The upper bound B_{FF} for Variable Sized Bin Packing with LIB is derived in Theorem 10 of this paper.

Organisation of this Paper. We provide a version of First Fit (FF) heuristic in Section 2 and then prove an upper bound on the guaranteed AAR (Asymptotic Approximation Ratio) in Section 3. The computational results are in Section 4. The results here are more general than the ones mentioned in [7], where the bin sizes are multiples of the smallest bin size.

1.1 Applications

Bin Packing and Covering theory does help to solve practical industry based problems such as assigning semiconductor wafer lots to customer orders [1]. Another interesting application arises during assigning tasks to computer processors based on a task priority. Each bin is analogous to a processor. The size of a bin corresponds to the processor's capabilities (such as speed), and the position of a task in a bin corresponds to its priority.

The LIB version of Bin Packing has applications in the Transportation industry, especially with loading of pallets in a truck. If long items are placed at the bottom of a pallet inside a truck, transportation is easier. In terms of weight, if heavier items are placed at the bottom, better stability of the truck can be achieved, and smaller items will not get crushed by larger items.

The dual of Bin Packing is *Bin Covering*, where the item sizes in a bin should total up to *at least* the bin size. Bin Covering has been applied in the industry, from packing peaches into cans in an "online" manner (so that the weight of each can is at least equal to its advertised weight) to breaking up a large company into smaller companies such that each new company is viable [11].

2 Problem and Algorithm

Problem Statement: Online LIB Variable-Sized Bin Packing (OLIBP). Given an infinite supply of variable sized bins, and n items, each item with size in $[\gamma, 1]$ and $0 < \gamma < 1$. Each item should be placed in a bin assigned to it (on top of items previously placed in that bin) as soon as it arrives. This placement cannot be changed later. In addition, the following LIB

a_i	Size of item i
b_j	Bin number j
B	Set of used bins
$\mathcal{B}$	Set of available bin sizes, s_1 through s_K
i	Index for an item (usually)
K	Number of available bin sizes (cardinality of $\mathcal{B}$)
L	Input list of items, in a given sequence
N (or n)	Cardinality of L (usually)
p	percentage of ones (used in computational studies)
R_{ALG}	Worst case asymptotic approximation ratio for algorithm ALG
$s_1(s_K)$	Size of the smallest (largest) available bin size
$topSize(b_j)$	Size of the item at the top of bin b_j
$totalSize(b_j)$	Sum of the sizes of the items in bin b_j

Table 2: Notation (in alphabetical order)

LIB	Largest (Longest, in the one-dimensional case) Item at the Bottom
FF	First Fit heuristic
AAR	asymptotic approximation ratio
SU	Space Utilization factor (in a bin, or set of bins)
VSBP	Variable Sized Bin Packing
OLIBP	Online and LIB version of VSBP

Table 3: Acronyms

constraint should be obeyed for any used bin:

$$[i \text{ is below } j \text{ in a used bin}] \implies [a_i \geq a_j]. \tag{1}$$

A feasible solution is one where the sum of the item sizes in each used bin is at equal to the bin size. The available bin sizes consist of a finite set $\mathcal{B} = \{s_j : 1 \leq j \leq K\}$, $s_j < s_{j+1}$, $1 \leq j \leq K-1$. The bin sizes are normalized, that is, s_K (the largest bin size) is equal to one. The smallest bin size s_1 is greater than zero. The goal is to find a feasible solution that minimizes the sum of the size of used bins.

The online condition essentially reduces to the following *Online Constraint*: In a used bin, if item i is below item j, then i should have arrived prior to j in the input list L, that is,

$$[i \text{ is below } j \text{ in a used bin}] \implies [i < j]. \tag{2}$$

The First Fit algorithm can be modified to accommodate OLIBP. (See [6] or [5] or [2] for descriptions of First Fit.) The behaviour of FF is summarized as follows: When an item i arrives, assume that bins b_1 through b_m have already been *used*, in that order. Each such bin b_j, $1 \leq j \leq m$, has two parameters, $topSize(b_j)$ and $totalSize(b_j)$, representing the size of the topmost item in b_j and the sum of the item sizes in b_j respectively. FF scans b_1 through b_m in that order. For each such bin b_j, it checks if (1) $a_i \leq topSize(b_j)$, and (2) $a_i \leq size(b_j) - totalSize(b_j)$. FF places item i in the first such bin b_j that satisfies both these conditions and updates $topSize(b_j)$ as well as $totalSize(b_j)$. If no such bin among b_1 through b_m satisfies these conditions, FF opens a new bin b_{m+1} of size

$$\lceil a_i \rceil_{\mathcal{B}} = min\{s_j | s_j \geq a_j, s_j \in \mathcal{B}\} \tag{3}$$

to place i. For instance, if $\mathcal{B} = \{0.2, 0.4, 0.6, 0.8, 1.0\}$, an arriving item of size 0.64 will be placed in a bin of size $\lceil .64 \rceil_{\mathcal{B}} = 0.8$, not in a bin with a size of one.

Algorithm (ALG). First Fit (online variable-sized LIB Bin Packing).
Given: Items $1 \cdots N$ with sizes $a_1 \cdots a_N$, $\gamma \leq a_i \leq 1$ for $1 \leq i \leq N$,
bin sizes $s_1 \cdots s_K$, $0 < s_1 < s_2 < \cdots < s_{K-1} < s_K = 1$.
Running Time: $O(KN^2)$.

```
1  nBin (number of bins used) = 0;
2  for (item = 1 to N) do
3      placed[item] = NO;
4      bin = 1;
5      While (bin ≤ nBin AND placed[item] == NO) do
6          X = (topSize[bin] ≥ size[item]);
7          Y = (size[bin]-totalSize[bin] ≥ size[item]);
8          if (X == true AND Y == true) then
9              place item in bin;
10             update topSize[bin] and totalSize[bin];
11             placed[item] = YES;
12         end if
13         bin = bin + 1;
14     end While
15     if (placed[item] == NO) then (item not placed in any previous bin)
```



```
16         nBin = nBin + 1; (new, fresh, unused bin)
17         size[nBin] = s_1;
18         While (size[nBin] < size[item])
19             increase size[nBin] to next higher bin size available;
20         place item in nBin;
21         topSize[nBin] = size[item];
22         totalSize[nBin] = size[item];
23         placed[item] = YES;
24     end if
25 end for
```

The key difference between original FF algorithm (where only bins of unit-size are used) and ALG is that, in case an unused bin is needed, ALG searches (in lines 18-19) the set of bin sizes $\{s_1, s_2, \cdots, s_K\}$ for the *best fitting bin* for the item to be placed.

3 Proof of A Bound of AAR

In this subsection, we give estimations of the asymptotic approximation ratio (AAR) for ALG. Let $ALG(L)$ denote the sum of the sizes of bins generated online by ALG to pack L. Let $OPT(L)$ be the optimal value of bin sizes necessary for packing items in L. The AAR is defined by

$$R_{ALG} = \lim_{s \to \infty} \sup_L \{ \frac{ALG(L)}{OPT(L)} | OPT(L) > s \}.$$

Define the *SU (Space Utilization)* factor for a set of used bins $B = \{b_1, b_2, \cdots b_m\}$ as follows:

$$SU(B) = \frac{\sum_{b_j \in B} \sum_{i \in b_j} a_i}{\sum_{b_j \in B} \text{ size}(b_j)} = \frac{\sum_{b_j \in B} totalSize(b_j)}{\sum_{b_j \in B} \text{ size}(b_j).} \tag{4}$$

In other words, $SU(B)$ is the ratio of the space occupied by items in the bins B to the sum of the sizes of the bins in B. If B consists of just one bin b_j, we will simply write $SU(b_j)$ as a shorthand for $SU(\{b_j\})$. The following observation follows immediately from the definition of SU.

Lemma 1. *If $SU(b_i)$ of each used bin b_i, $i = 1, 2, \cdots, m$ is greater than or equal to δ, then R_{ALG} has an upper bound of $\frac{1}{\delta}$.*

Proof. Since $SU(b_j) \geq \delta$, the total item size in b_j should be larger than $\delta \times size(b_j)$. Hence

$$\sum_{b_j \in B} totalSize(b_j) \geq \delta \sum_{b_j \in B} size(b_j). \tag{5}$$

Any feasible packing, including the optimal one, must use bins whose total size is at least $\sum_{b_j \in B} totalSize(b_j)$. Therefore,

$$\lim_{s \to \infty} \sup \{ \frac{R_{ALG}(L)}{OPT(L)} | OPT(L) > s \} \leq \lim_{s \to \infty} \sup \{ \frac{\sum_{b_j \in B} size(b_j)}{\delta \sum_{b_j \in B} size(b_j)} | OPT(L) > s \} = \frac{1}{\delta}.$$

□

In contrast, let us now consider instances where the above condition is false for solutions returned by ALG.

Lemma 2. *Let* $0 < \delta \leq \min_{1 \leq i \leq K-1} \frac{s_i}{s_{i+1}}$ *and* $SU(b_J) < \delta$ *where* b_J *is the last bin in* B *for which* $SU(b_j) < \delta$ *is true. In other words,* $J = \max_{1 \leq j \leq m}\{j | SU(b_j) < \delta\}$. *If* I *is the bottom item of bin* b_J, *then we have (1)* $a_I \in (\gamma, s_1]$; *(2)* $size(b_J) = s_1$; *and (3)* $a_I < \delta s_1$.

Proof. Clearly, a new bin b_J is opened when I arrives, since I has been placed at the bottom of b_J. (1) By ALG, if $a_I \in (s_i, s_{i+1}]$, $1 \leq i \leq K-1$, then $size(b_J) = s_{i+1}$ and $totalSize(b_J) \geq a_I > s_i$. Hence

$$SU(b_J) = \frac{totalSize(b_J)}{size(b_J)} > \frac{s_i}{s_{i+1}} \geq \delta,$$

which contradicts to $SU(b_J) < \delta$. Therefore, $a_I \in (\gamma, s_1]$.

(2) Since $a_I \in (\gamma, s_1]$, $size(b_J) = s_1$.

(3) Since

$$\frac{a_I}{size(b_J)} \leq SU(b_J) < \delta,$$

we have $a_I < \delta s_1$. □

Remark. If $\gamma \geq \delta s_1$, then the SU factor of every used bin is at least δ. By Lemma 1, R_{ALG} is bounded above by $1/\delta$.

Let us continue with the assumptions made in the first sentence of Lemma 2. Upon I's arrival, there are two reasons why I was placed in a new bin b_J and not in any of the bins b_j $(1 \leq j \leq J-1)$ used earlier: Either (i) $totalSize^I(b_j) + a_I > size(b_j)$, or (ii) $topSize^I(b_j) < a_I$, where $totalSize^I(b_j)$ is the sum of the sizes of items in b_j and $topSize^I(b_j)$ is the top item in b_j when item I arrived. Since there could be some items that arrived after I and were placed in b_j, the following inequalities hold

$$totalSize^I(b_j) \leq totalSize(b_j); \tag{6}$$

$$topSize^I(b_j) \geq topSize(b_j). \tag{7}$$

Now, partition $\{b_j | 1 \leq j \leq J-1\}$ into two disjoint sets $\mathcal{C}$ and $\mathcal{D}$ with the following definition:

Type-c bins: First, consider the set $\mathcal{C}$ of bins, with $|\mathcal{C}| = c$, and

$$\mathcal{C} = \{b_j | totalSize^I(b_j) + a_I > size(b_j), 1 \leq j \leq J-1\}.$$

Refer to $\mathcal{C}$ as *type-c* bins. Since $a_I < \delta s_1$, it follows that

$$totalSize^I(b_j) > size(b_j) - \delta s_1, \quad \forall b_j \in \mathcal{C}.$$

By the inequality (6),

$$totalSize(b_j) > size(b_j) - \delta s_1, \quad \forall b_j \in \mathcal{C}. \tag{8}$$

Define $p = \sum_{b_j \in \mathcal{C}} size(b_j)$. Then,

$$OPT(L) \geq \text{sum of item sizes} \geq \sum_{b_j \in \mathcal{C}} totalSize(b_j) \geq \sum_{b_j \in \mathcal{C}} [size(b_j) - \delta s_1] = p - \delta c s_1. \tag{9}$$

On the other hand, for type-c bins, the solution returned by ALG has a value of $\sum_{b_j \in \mathcal{C}} size(b_j) = p$. The upper and lower bounds for p are:

$$cs_1 \leq p \leq c. \tag{10}$$

Lemma 3. *For* $\delta < \frac{1}{2}$, $\mathcal{C} \subseteq \{b_j | SU(b_j) \geq \delta, 1 \leq j \leq J-1\}$.

Proof. By inequality (8),

$$\frac{totalSize(b_j)}{size(b_j)} > 1 - \frac{\delta s_1}{size(b_j)}.$$

Since $size(b_j) \geq s_1$,

$$SU(b_j) = \frac{totalSize(b_j)}{size(b_j)} \geq 1 - \frac{\delta s_1}{s_1} = 1 - \delta > \delta.$$

□

Type-d bins: Secondly, let $\mathcal{D}$ be the subsets of bins, with $|\mathcal{D}| = d$ and

$$\mathcal{D} = \{b_j | topSize^I(b_j) < a_I, 1 \leq j \leq J-1\} \cap \bar{\mathcal{C}}.$$

Name these bins as *type-d* bins.

Among type-d bins, consider any two, say b_j and b_k with $j < k$, meaning that bin b_j was opened before bin b_k. Let $e^I(j)$ $[e^I(k)]$ be the topmost item of b_j $[b_k]$ when I arrived.

Lemma 4. *If* $b_j, b_k \in \mathcal{D}$ *with* $j < k$, *then (1)* $totalSize^I(b_j) + a_{e^I(k)} \leq size(b_j)$; *(2)* $a_{e^I(j)} < a_{e^I(k)} < a_I$.

Proof. By the definition of $\mathcal{D}$, $a_{e^I(k)} = topSize^I(b_k) < a_I$ for all $b_k \in \mathcal{D}$, and

$$totalSize^I(b_j) + a_I \leq size(b_j).$$

It follows that

$$totalSize^I(b_j) + a_{e^I(k)} < size(b_j).$$

In other words, there was enough space in b_j for item $e^I(k)$. If $e^I(k)$ had arrived after $e^I(j)$, then a placement of $e^I(k)$ over $e^I(j)$ would have been attempted and failed due to $a_{e^I(j)} < a_{e^I(k)}$. On the other hand, if $e^I(j)$ had arrived after $e^I(k)$, then, a placement of $e^I(k)$ over an earlier item $x < e^I(j)$ in b_j would have been attempted and failed, implying that $a_x < a_{e^I(k)}$. Since x is below $e^I(j)$ in b_j, $a_{e^I(j)} \leq a_x$. It follows that $a_{e^I(j)} \leq a_x < a_{e^I(k)}$. □

Lemma 5. *Let* $\mathcal{D} = \{b_{t_1}, b_{t_2}, \cdots, b_{t_d}\}$, *with* $t_1 < t_2 < \cdots < t_d$ — *thus among the* $\mathcal{D}$ *bins,* b_{t_1} *was opened the earliest and* b_{t_d} *the last. For any item* $l \in b_{t_k}$, $k = 2, 3, \cdots, d$, *if* $l < I$ *and* $a_l < a_I$, *then there is another item* $j \in b_{t_{k-1}}$ *such that* $j < l$ *and* $a_j < a_l$.

Proof. Since $b_{t_{k-1}} \in \mathcal{D}$, we have the following inequality:

$$totalSize^I(b_{t_{k-1}}) + a_I \leq size(b_{t_{k-1}}).$$

By assumption, l arrived before I. Therefore,

$$totalSize^{l}(b_{t_{k-1}}) \leq totalSize^{I}(b_{t_{k-1}}).$$

Moreover, $a_l < a_I$ implies that

$$totalSize^{I}(b_{t_{k-1}}) + a_l < size(b_{t_{k-1}}).$$

Then

$$toialSize^{l}(b_{t_{k-1}}) + a_l \leq size(b_{t_{k-1}}). \tag{11}$$

This implies that, when l arrives, there is at least one item j already in $b_{t_{k-1}}$ so that $a_j < a_l$. Otherwise, by (11), l would have been placed in $b_{t_{k-1}}$. □

Let the topmost item of b_{t_d}, as I arrived, be α_d. Then $\alpha_d < I$ and $a_{\alpha_d} < a_I$. Apply Lemma 5 backward repeatedly, we obtain a sub-list of L such that $\alpha_1 < \alpha_2 < \cdots < \alpha_d < I$, $a_{\alpha_1} < a_{\alpha_1} < \cdots < a_{\alpha_d} < a_I$ with $\alpha_k \in b_{t_k}$, $k = 1, 2, \cdots, d$. Let $\Lambda = \{\alpha_1, \alpha_2, \cdots, \alpha_d\}$. According to the online and LIB constraints, every item in Λ must be placed in distinct bins and each item has a length at least γ. As a result, we have

Lemma 6. *The sum of the bin sizes optimal algorithm to pack type-d bins should be at least γd, whereas the value of the solution returned by ALG is $q = \sum_{b_j \in \mathcal{D}} size(b_j) \leq d$.*

Type-f bins: Beyond bin b_J, the last bin with space utility $SU(b_J) < \delta$, there could be several used bins all of which have $SU \geq \delta$. Name these bins as *type-f* and denote them by $\mathcal{F} = \{b_j \in B | j \geq J+1\}$. Let the sum of their sizes be

$$f = \sum_{b_j \in \mathcal{F}} size(b_j), \tag{12}$$

which is the value returned by ALG. Again, by equation (5) in Lemma 1, the sum of item sizes in type-f bins is at least

$$\sum_{b_j \in \mathcal{F}} totalSize(b_j) \geq f\delta, \tag{13}$$

which will be used as a lower bound for packing items in type-f bins the optimal way.

Thus the entire set of bins used by ALG is made up of, in this order: (i) a mixture of type-c bins and type-d bins, (ii) bin b_J containing item I, and (iii) type-f bins. The lower bound estimations for the sum of item sizes in each category are: (i) $p - \delta c s_1$ for type-c bins (by (9)); (ii) γd for type-d bins (by Lemma 6); (iii) γ for bin b_J; (iv) $f\delta$ for type-f bins (by (13)). Therefore, the lower bound for the optimal bin sizes is $p - \delta c s_1 + d\gamma + \gamma + f\delta$, whereas the solution returned by the ALG is $p + q + s_1 + f$ (q is defined in the statement of this lemma.) The asymptotic ratio AAR requires us to consider the ratio

$$\frac{ALG(L)}{OPT(L)} \leq \frac{p + q + s_1 + f}{p - \delta c s_1 + d\gamma + \gamma + f\delta} \tag{14}$$

for all large inputs L for which $OPT(L) > s$ and $s \to \infty$. Equivalently, one of the numbers p, q, f must tend to infinity in the limit. Observe that, by (10), $p \to \infty$ implies $c \to \infty$ and also by Lemma 6, $q \to \infty$ implies $d \to \infty$. In what follows, we shall write $(\cdot, \cdot, \cdots, \cdot) \to \infty$

to indicate at least one of the components tend to infinity. Taking the limit on both sides of (14) gives

$$R_{ALG} \leq \lim_{(p,c,q,d,f)\to\infty} \frac{p + q + s_1 + f}{p - \delta c s_1 + d\gamma + \gamma + f\delta}$$

$$= \lim_{(p,c,q,d,f)\to\infty} \frac{p + q + f}{p - \delta c s_1 + d\gamma + f\delta}$$

where s_1 in the numerator and γ in the denominator do not affect the limit in any case.

Lemma 7. *If $c > 0, \frac{s_1(1-\delta)}{\delta} < 1$ and $\gamma < \delta$, then*

$$\frac{p + q + f}{p - \delta c s_1 + d\gamma + f\delta} \leq \frac{1 + \frac{d}{c}(1 - \frac{\gamma}{\delta})}{s_1(1-\delta)}.$$

Proof. Since $cs_1 \leq p \leq c$ and $q \leq d$,

$$\frac{p + q + f}{p - \delta c s_1 + d\gamma + f\delta}$$

$$\leq \frac{c + d + f}{cs_1 - \delta c s_1 + d\gamma + f\delta}$$

$$= \frac{1}{\delta} \frac{c\delta + d(\delta - \gamma) + d\gamma + f\delta}{cs_1 - \delta c s_1 + d\gamma + f\delta}.$$

Since $\delta > \gamma$ and $cs_1 > \delta c s_1$,

$$\frac{c\delta + d(\delta - \gamma) + d\gamma + f\delta}{cs_1 - \delta c s_1 + d\gamma + f\delta} \leq \max\{1, \frac{c\delta + d(\delta - \gamma)}{cs_1 - \delta c s_1}\}.$$

Moreover, by assumption,

$$\frac{c\delta + d(\delta - \gamma)}{cs_1 - \delta c s_1} = \frac{c + d(1 - \frac{\gamma}{\delta})}{c} \frac{\delta}{s_1(1-\delta)} > 1.$$

This implies that

$$\frac{p + q + f}{p - \delta c s_1 + d\gamma + f\delta} \leq \frac{1}{\delta} \frac{c\delta + d\delta(1 - \frac{\gamma}{\delta})}{cs_1 - \delta c s_1} = \frac{1 + \frac{d}{c}(1 - \frac{\gamma}{\delta})}{s_1(1-\delta)}.$$

□

Define $\lim_{(c,d)\to\infty} \frac{d}{c} = M$, which might be infinite. Then, under the assumption of Lemma 7,

$$R_{ALG} \leq \frac{1 + M(1 - \frac{\gamma}{\delta})}{s_1(1-\delta)}.$$

Lemma 8. *If $c = 0, d > 0$ and $\gamma < \delta$, then*

$$\frac{p + q + f}{p - \delta c s_1 + d\gamma + f\delta} \leq \frac{1}{\gamma}.$$

Proof. Since $c = 0$ implies $p = 0$ and also $q \leq d$ and $\gamma < \delta$,

$$\frac{p+q+f}{p-\delta cs_1+d\gamma+f\delta} = \frac{q+f}{d\gamma+f\delta} \leq \frac{1}{\delta}\frac{d\delta+f\delta}{d\gamma+f\delta} \leq \frac{1}{\delta}\frac{d\delta}{d\gamma} = \frac{1}{\gamma}.$$

□

Lemma 9. *If $c > 0$, $d = 0$ and $\delta \leq \min\{\frac{1}{2}, \min_{1\leq i\leq K-1}\{\frac{s_i}{s_{i+1}}\}\}$, then $R_{ALG} \leq \frac{1}{\delta}$.*

Proof. Since $d = 0$, there are no type-d bins and $\mathcal{C} = \{b_j \mid 1 \leq j \leq J-1\}$. By Lemma 3, $SU(\mathcal{C}) \geq \delta$. This makes type-c and type-f indistinguishable and the lower bound estimations (of the optimal solution value) for the item sizes should follow the same rationale: (i) $p\delta$ for type-c bins and (ii) $f\delta$ for type-f bins. Therefore,

$$R_{ALG} \leq \lim_{(p,f)\to\infty} \frac{p+s_1+f}{p\delta+\gamma+f\delta} = \lim_{(p,f)\to\infty} \frac{p+f}{p\delta+f\delta} = \frac{1}{\delta}.$$

□

Remark. When $d = 0$, $\frac{1}{\delta}$ is a better bound than that of Lemma 7, which reduces to $\frac{1}{s_1(1-\delta)}$. By assumption in Lemma 7, $s_1(1-\delta) < \delta$.

Remark. If $c = d = 0$, by Lemma 1, $R_{ALG} \leq \frac{1}{\delta}$.

Theorem 10. *The asymptotic approximation ratio obtained by ALG (the FF heuristic) for the online VSBP Problem with the LIB constraint is guaranteed to be at most*

$$\max\{\frac{1+M(1-\frac{\gamma}{\delta})}{s_1(1-\delta)}, \frac{1}{\delta}, \frac{1}{\gamma}\}.$$

If there are only type-c and type-f bins, then R_{ALG} is less than $\frac{1}{\delta}$. If there are only type-d and type-f bins, then R_{ALG} is less than $\frac{1}{\gamma}$. If there are only type-f bins, then R_{ALG} is less than $\frac{1}{\delta}$.

4 Computational Studies in OLIBP

We carried out simulations to study the numerical performance of ALG for OLIBP. Here we use a branch and bound (B&B) algorithm to compute the exact optimal bin sizes. A node t of the B&B tree represents a partial solution that packs items, in this order, from 1 to some $i \in L$. The children of t represent different ways to pack the item $i+1$ either to used bins at node t or to a new bin. The lower bound was computed at each node and tested against the ALG solution. If the lower-bound was larger, all its sub-trees were pruned.

At any partial solution, let $L' = \{I+1, I+2, \ldots, N\}$ be the set of items not yet placed. V is a subset of L' so that, if $i \in V$, at least one of these three conditions are satisfied:

1. $a_i > 0.5$ (When i was tried to place on top of j with $a_j \geq a_i$, it would find $a_i + a_j > 1$.);
2. a_i is larger than the *topSize* of any of the used bins;
3. a_i is larger than the empty space in any of the used bins.

List Size (N)	Number of Bin Sizes (K)	Maximum Ratio	Average Ratio	Number Of Runs	Percentage of Ones (p)	Running Time(B&B)
10	5	1.222	1.033	5000	37.14	47 secs
15	5	1.179	1.042	5000	13.98	26 mins
20	5	1.152	1.048	2000	4.627	26 hours
10	10	1.222	1.049	5000	18.26	216 secs
15	10	1.226	1.074	2000	1.2	36 hours

Table 4: Testing FF Heuristic for VSBP: Approximation Ratios

In other words, items in V can not be placed in any used bins so far, while items in $L' - V$ are allowed to place in a used bin provided there is sufficient space. Let (i) u be the sum of the sizes of bins used thus far; (ii) x be the space leftover in used bins; (iii) v be the sum of sizes of items in V; and (iv) w be the sum of the sizes of items in $L' - V$. Then we obtain a lower bound as follows: $u + v + \max\{0, w - x\}$.

Three different values for N was considered: 10, 15, and 20. For each N, we performed 2000-5000 runs of the simulation, depending on the time taken for the runs (last column of Table 4). The table 4 summarizes results of (a) the worst R_{ALG} in column 3, (b) the average R_{ALG} in column 4, and (c) the percentage of instances where R_{ALG} was one (the lowest possible) in column 5. We observe that, as the list size (N) grows, the percentage of ones (p) (the proportion of the instances where ALG produces a solution as good as that of the exact algorithm) drop dramatically. The average ratio, however, stays almost the same.

5 Scope for Further Research

All problems referred to here are online LIB versions.

- Bin Covering: The discussion in this paper can be extended to its Bin Covering counterpart.
- Testing of HF: A Harmonic Fit (HF) heuristic could be developed and computationally tested for the online and LIB version of VSBP.
- Higher dimensions: All problems considered here can be extended to their two and three dimensional counterparts.

References

[1] M. Carlyle, K. Knutson, and J. Fowler. Bin covering algorithms in the second stage of the lot to order matching problem. *Journal of the Operational Research Society*, 52:1232–1243, 2001.

[2] E.G. Coffman, M.R. Garey, and D.S. Johnson. Bin Packing Approximation Algorithms: A Survey. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 46–93. PWS Publishing Company, Boston, MA, 1997.

[3] J. Csirik. An On-line algorithms for Variable Sized Bin Packing. *Acta Informatica*, 26:697–709, 1989.

[4] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. Karp, editor, *Complexity of Computations*, pages 43–73. SIAM-AMS Proceedings (no.7), 1974.

[5] P. Manyem. Bin packing and covering with longest items at the bottom: Online version. *The ANZIAM Journal (formerly Journal of the Austral. Math. Soc., Series B)*, 43(E):E186–E231, June 2002.

[6] P. Manyem. Uniform Sized Bin Packing and Covering: Online Version. In J.C. Misra, editor, *Topics in Industrial Mathematics*, pages 447–485. Narosa Publishing House (New Delhi), 2003.

[7] P. Manyem, R.L. Salt, and M.S. Visser. Lower Bounds and Heuristics for Online LIB Bin Packing and Covering. In *Proceedings of the 13th Australasian Workshop on Combinatorial Algorithms (Fraser Island, Queensland, Australia)*, pages 11–42, July 2002.

[8] C.H. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences*, 43(3):425–440, December 1991.

[9] M.B. Richey. Improved Bounds for Harmonic Based Bin Packing Algorithms. *Discrete Applied Mathematics*, 34:203–227, 1991.

[10] A. Van Vliet. Optimal On-Line Algorithms For Variable-Sized Bin Covering. *Information Processing Letters*, 43:277–284, 1992.

[11] G.J. Woeginger and G. Zhang. Optimal On-Line Algorithms For Variable-Sized Bin Covering. *Operations Research Letters*, 25:47–50, 1999.