On Variable-Sized Bin Packing*

GUOCHUAN ZHANG
Kiel Universität, Germany

Abstract

In variable sized bin packing, a number of bin types are given and the goal is to pack a list of items by choosing appropriate bins such that the total size of bins used is minimized. In this paper, we present absolute worst-case analysis for on-line algorithms. Then we consider an related problem: how to design k different bin sizes such that for any item list the waste space of bins used is minimized. A best possible off-line algorithm is showed for arbitrary k and a best possible on-line algorithm is derived for any fixed number k.

Keywords

Bin Packing, Analysis of Algorithms.

1 Introduction

Variable-sized bin packing problem is a variant of the classical bin packing, in which bin capacities may vary. We are given a list $L = (a_1, \ldots, a_n)$ of items, each with item size $s(a_i) \in (0,1]$, and several different types $B^1, \ldots, B^l$ of bins with sizes $1 = s(B^1) > s(B^2) > \cdots > s(B^l) > 0$, and 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 bin packing problem.

Bin packing is one of the fundamental problems in Computer Science and discrete Optimization. In communication networks, bin packing arises as a scheduling problem with bandwidth allocation. Assume that we have a set of unit-time requests (tasks), which are asked to transmit through a network. Each request has a requirement of the network capacity. At any time the network can transmit a number of the requests provided that the total requirements of capacity does not exceed the capacity of the network. However, the network can provide several different capacities with different costs. Assume that the costs are proportional to capacities. The scheduling problem to minimize the total cost is just the variable-sized bin packing problem, where the network at each unit time is a bin and its capacity is the bin size.

*This work has been supported by Alexander von Humboldt Foundation.

To evaluate the performance of an approximation algorithm, different measures have been proposed. For a list L of items and an approximation algorithm A, let $A(L)$ denote the total size of bins used by algorithm A and let $OPT(L)$ denote the total size of bins used in an optimal packing. Then the *absolute* worst-case ratio of algorithm A is defined to be

$$S_A = \sup_L \{A(L)/OPT(L)\}$$

and the *asymptotic* worst-case ratio of algorithm A is defined as

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

In the on-line version of variable-sized bin packing, we cannot preview and rearrange the items of L before starting to construct a packing, but must accept and immediately pack each item as it arrives.

Known results. Since the variable-sized bin packing problem is NP-hard, efficient approximation algorithms which ensure near-optimal packings are required. To our best known, almost all the approximation results are with respect to the asymptotic worst-case ratio. Friesen and Langston [7] gave three approximation algorithms with asymptotic worst-case ratios of 2, $3/2$, and $4/3$. The first of these algorithms is on-line. Essentially, it is a simple modification of *Next Fit* and also has the same worst-case ratio. The next two are based on *First-Fit Decreasing* strategy. Murgolo [10] proposed an asymptotic fully polynomial time approximation scheme. Kinnersley and Langston [8] presented fast online 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 an asymptotic worst-case ratio not exceeding $1.5 + f/2 \geq 1.75$. Later, Zhang [13] showed that the asymptotic worst-case ratio of algorithm FFH (it is FFf as $f = 1/2$) is exactly 1.7. Csirik [4] derived an algorithm VH with asymptotic worst-case ratio of < 1.7 from the Harmonic Fit algorithm [9]. Actually, Csirik's algorithm uses *bounded space* , which means that at any time the number of open bins is bounded by a constant. Most recently, Seiden [11] proved that algorithm VH is an optimal bounded-space on-line algorithm. However, the algorithm reaches the bound $1.69103\ldots$ only for a very large number of open bins. Burkard and Zhang [1] investigated simple bounded-space algorithms for on-line variable-sized bin packing by designing *open* rules, *packing* rules and *closing* rules. It was proved that algorithm VBB_k has an asymptotic worst-case ratio 1.7 for any $k \geq 3$. More details on bin packing problems can be found in the excellent surveys by Coffman et al. [3] and by Csirik and Woeginger [5].

For special collections of bin types, there are few results. Csirik [4] proved that the proposed on-line algorithm has an asymptotic worst-case ratio 1.4 for two bin sizes 1 and 0.7. Zhang [13] proved that FFH algorithm has an asymptotic worst-case ratio 1.5 for two bin sizes 1 and $x \in [2/3, 3/4]$. Very recently, Epstein et al.

[6] studied the on-line version with two bin sizes. For any combinations of two bin types, they presented two algorithms based on *Harmonic* and *Refined Harmonic* and combine them with algorithm VH. They proved an asymptotic worst-case ratio at most $373/228 \approx 1.636$. They also gave a lower bound (≈ 1.3356) for the case with two bin sizes.

Recently, Chu and La [2] considered first the absolute worst-case performance of approximation algorithms for variable-sized bin packing. They proposed and analyzed four approximation algorithms, namely *LFLAW* (Largest item First with Least Absolute Waste), *LFLRW* (Largest item First with Least Relative Waste), *LAW* (Least Absolute Waste) and *LRW* (Least Relative Waste), all of which are off-line. The absolute worst-case ratios of these four algorithms are 2,2,3 and $2+\ln 2$, respectively.

Our results. We study on-line algorithms in terms of the absolute worst-case ratio. We prove that a lower bound of 2 holds for any on-line algorithms with only two bin sizes. For the case with more than two bin sizes, a lower bound 2.245 is given. A simple on-line algorithm is shown to have an absolute worst-case ratio of 3. For the special case with two bin sizes, we provide an on-line algorithm with an absolute worst-case ratio of at most 2.75.

Note that all the known results are concerning about how to pack a list of items by choosing appropriate bins from given bin types such that the total size of bins used is minimized. We want to consider variable sized bin packing from a different point of view. If we are allowed to design k bin sizes, what sizes we should choose such that for any instances, by using these bins, the waste space of a packing is smallest possible. To evaluate an algorithm in this issue, we use the asymptotic waste ratio defined as follows. For any list L of items, let $S(L)$ be the total size of items in L and let $A(L)$ be the total size of bins used by an algorithm A. Then the *asymptotic waste ratio* of algorithm A is

$$W_A^\infty = \lim_{k\to\infty} \sup_L \{A(L)/S(L) | S(L) \geq k\}.$$

Note that the definition of asymptotic waste ratio is different from that of worst-case ratio. In terms of asymptotic waste ratio, we give the best collection for bin sizes and provide best possible algorithms.

The remainder of this paper is organized as follows. Section 2 provides the absolute worst-case analysis for on-line algorithms, while Section 3 considers how to design bin sizes. Concluding remarks are given in Section 4.

2 Absolute worst-case analysis

We first investigate a special case where only two bin sizes are available. Assume that A is any on-line algorithm. The two bin sizes are K and $2-1/K$, respectively, where K is an arbitrarily large positive integer. The items are all have size 1.

The first item must be assigned to a bin with a smaller size, otherwise no items come any longer and the corresponding absolute worst-case ratio is close to $K/2$. Assume that the first l items have been assigned each to a bin with a smaller size. If $l < K/2$, then the $(l+1)$-st item will be assigned to a smaller bin too; otherwise the objective value by algorithm A is $(2-1/K)l+K$, while the optimal value is only $(l+1)(2-1/K)$. It implies that the ratio is arbitrarily close to 2 as K is large enough. Then at least $K/2$ items are assigned each into a smaller bin. In this case the total number of items is K. The optimal packing is just using one bigger bin with a size K while the packing produced by algorithm A uses at least $2K-1$ bin sizes. Hence for two bin sizes case, the lower bound (in terms of absolute worst-case ratio) for any on-line algorithms is 2.

Let a and b be the two bin sizes, where $b > a$. A bin of size b is called larger while a bin of size a is called smaller. An item is called *large* if its size is greater than a; otherwise it is called *small*. We provide an on-line algorithm below. Let α be the absolute worst-case ratio of on-line algorithm *First-Fit* for classical bin packing. Simchi-Levi [12] Proved $1.7 \leq \alpha \leq 1.75$.

Algorithm $A2$**.**

1. Put the first item into a smaller bin if it is a small item; otherwise put is into a larger bin.
2. Put the current item a_i into the lowest indexed bin which it can fit in. If such a bin does not exist, a new bin is needed.
   - If a_i is a large item, open a larger bin for it.
   - If a_i is a small item, check if there exists an opened larger bin.
     - If yes, open a smaller bin for a_i.
     - If no, check how many bins have been opened. If the number of opened bin is $\lfloor \alpha b/a \rfloor$, open a larger bin for a_i; otherwise open a smaller bin.

Theorem 1 $S_{A2} \leq 1+\alpha$.

Proof. Assume that algorithm $A2$ uses p smaller bins and q larger bins. Let $m = p+q$. If $p \geq 2$ and $q \geq 2$, it can be easily verified that the sum of items in smaller bins is more than $pa/2$ and the sum of items in larger bins is more than $qb/2$. Hence the ratio is at most two. If $p = 0$ or $q = 0$, the proof is also simple. Now we assume that either $p = 1$ or $q = 1$. Denote by B_m the last bin.

Case 1. $p = 1$. Assume that the smaller bin is B_i. The other bins B_j, $j = 1, \ldots, m$, $j \neq i$, are larger bins. Note that $c(B_j) + c(B_{j+1}) > s(B_j)$, for $j = 1, \ldots, m-1$, and $c(B_m) + c(B_1) > s(B_1)$. If $i \neq 1$, then $2OPT(L) \geq 2 \sum_{j=1}^{m} c(B_j) > (m-1)b + a =$

$A2(L)$. It implies that the absolute worst-case ratio is no more than 2. Suppose $i = 1$, i.e., all but the first bin are larger. If $m = 2$, there are just two bins, one of which is smaller and another of which is larger. Clearly the optimal packing must have a larger bin and thus the total size of bins used is at least b, while $A2(L) = a + b$. It produces a good bound. Now let $m \geq 3$. Note that $A2(L) = a + (m-1)b$. $OPT(L) \geq \sum_{j=1}^{m} c(B_j) > (m-2)b/2 + a$ and $OPT(L) \geq \sum_{j=2}^{m} c(B_j) > (m-1)b/2$. If $a > b/2$,

$$\begin{aligned} A2(L)/OPT(L) &< (a + (m-1)b)/((m-2)b/2 + a) \\ &< (2m-1)/(m-1) \leq 5/2. \end{aligned}$$

If $a \leq b/2$,

$$\begin{aligned} A2(L)/OPT(L) &< (a + (m-1)b)/((m-1)b/2) \\ &< (2m-1)/(m-1) \leq 5/2. \end{aligned}$$

Case 2. $q = 1$. Assume that the larger bin is B_i. The other bins B_j, $j = 1, \ldots, m$, $j \neq i$, are smaller bins. If $i \neq m$, $2OPT(L) \geq 2 \sum_{j=1}^{m} c(B_j) > (m-1)a + b = A2(L)$. Assume that $i = m$. The same as the analysis in Case 1, we can assume that $m \geq 3$. If B_m contains a large item, then $OPT(L) \geq b$. By the algorithm the number of smaller bins is no more than $\lfloor \alpha b/a \rfloor$. $A2(L) \leq \lfloor \alpha b/a \rfloor a + b$ and

$$A2(L)/OPT(L) \leq \frac{\lfloor \alpha b/a \rfloor a + b}{b} \leq 1 + \alpha.$$

We consider the last case where B_m contains no large items. In this case $m = \lfloor \alpha b/a \rfloor + 1 > \alpha b/a$. If $OPT(L) \geq b$, we can prove the bound as the above. Assume that $OPT(L) < b$. Then any optimal packing uses only smaller bins. By the definition of α, $ma \leq \alpha OPT(L)$. $OPT(L) \geq \lceil m/\alpha \rceil a$ and

$$A2(L)/OPT(L) \leq \frac{(m-1)a + b}{\lceil m/\alpha \rceil a} < \alpha + \frac{\alpha b}{ma} < 1 + \alpha.$$

■

Remark 1 *The absolute worst-case ratio of algorithm First-Fit for bin packing is still open. Simchi-Levi [12] proved that it is at most 7/4. In the above algorithm we can take* $\alpha = 7/4$. *Then the ratio of algorithm A2 is at most 11/4.*

Now we consider the general case that the number of bin types is at least 3.

Theorem 2 *There does not exist any on-line algorithms with an absolute worst-case ratio smaller than* 2.245.

Proof. Consider any on-line algorithm A. For any list L of items. Let $k > 0$ be a sufficiently large integer and let $\varepsilon > 0$ be a sufficiently small number. We consider the case where three bin types are available. They are denoted by B^1, B^2 and B^3, with sizes $2-\varepsilon$, $5k/2+2$ and βk, respectively, where $\beta = (1+\sqrt{181})/4$. The items are coming in the following way. Small items with size 1 come successively until either the algorithm A opens a bin of type B^2 or B^3, or the number of arriving items is k. If a bin of type B^2 or B^3 is opened, stop. Denote by k' be the number of items, where $k' \leq k$. Then $A(L) \geq (k'-1)(2-\varepsilon)+5k/2+2$, while $OPT(L) = k'(2-\varepsilon)$. The ratio S_A is arbitrarily close to $9/4 = 2.25$ as ε is sufficiently small. If there are already k items and each of them is packed into a B^1 bin, an item with size 2 comes. If a B^3 bin is opened for the item, stop. Then $A(L) \geq \beta k + (2-\varepsilon)k$, while $OPT(L) = 5k/2+2$. The ratio S_A is arbitrarily close to 2.245. If a B^2 bin is created for the item, the last item with size $5k/2+1$ comes. In this case, then $A(L) \geq \beta k + 5k/2 + (2-\varepsilon)k$, while $OPT(L) = \beta k$. The ratio S_A is arbitrarily close to 2.245. ■

In the following, we will consider an on-line algorithm and prove that the absolute worst-case ratio is 3. For any item a_i, a bin is called its *home-bin* if the bin size is the smallest among those bin sizes not smaller than $s(a_i)$.

Algorithm FFS (First Fit using Smallest possible bin sizes)

1. Assign the first item into its home-bin.
2. If the coming item can fit in one of the opened bin, apply *First Fit* to it. Otherwise open a home-bin for the new item. Repeat the process until all items have been assigned.

Theorem 3 $S_{FFS} = 3$.

Proof. Let m be the total number of bins used by algorithm FFS. For $i = 1, \ldots, m$, let $s(B_i)$ and $c(B_i)$ be the size of bin B_i and the sum of items in bin B_i, respectively.

- Repeatedly find the first pair of bins B_i and B_{i+1} such that $s(B_{i+1}) \leq s(B_i)$. Remove the pair from the bin list. Note that $c(B_{i+1}) + c(B_i) \geq s(B_i) \geq (s(B_i) + s(B_{i+1}))/2$.
- The remaining bins fulfill that the bin sizes are in strictly increasing order. Without loss of generality, let the remaining bins be $B_1, \ldots, B_l$. Clearly, $c(B_{i+1}) > s(B_i)$. Then $\sum_{i=1}^{l} c(B_i) > s(B_1) + \cdots + s(B_{l-1})$.
- $OPT(L) \geq s(B_l)$.

Combining the above three arguments and noting that $OPT(L) \geq \sum_{i=1}^{m} c(B_i)$ and $FFS(L) = \sum_{i=1}^{m} s(B_i)$, we have proved that the worst-case ratio is at most 3.

Consider an instance below. There are two bin sizes: $2-1/k$ and $k+2$, where $k>0$ is a sufficiently large integer; the list consists of k identical items with size 1 and one item with size 2. Obviously the optimal packing needs just one larger bin size while algorithm FFS uses k additional bins with the smaller size, i.e., $OPT(L)=k+2$ and $FFS(L)=3k+1$. We thus get this theorem. ■

At the end of this section, we present a very simple off-line algorithm by applying algorithm FFS to a sorted list of items. It is proved that the absolute worst-case ratio is 2. Note that Chu and La [2] proposed four off-line algorithms. At each step, the algorithms try every bin size to pack a subset of unpacked items and chose the one with the least absolute (relative) waste. Among these algorithms they proved the best two have absolute worst-case ratio of 2. Comparing with their algorithms the following algorithm is much simpler.

Algorithm $FFDS$ (First Fit Decreasing with Smallest possible bin sizes)

1. Sort the items in nonincreasing order of sizes.
2. Apply FFS to the sorted list of items.

Theorem 4 $S_{FFDS}=2$.

Proof. For any list L of items, assume that algorithm $FFDS$ generates a list of bins $B_1,\ldots,B_m$. Clearly the bin sizes fulfill the inequalities: $s(B_1)\geq\cdots\geq s(B_m)$. If $m=1$, i.e., only one bin is used by $FFDS$, the solution is optimal. Assume that $m\geq 2$. We have $c(B_i)+c(B_{i+1})>s(B_i)$ for $i=1,\ldots,m-1$. Moreover $c(B_1)+c(B_m)>s(B_1)$. Then

$$\begin{aligned} 2FFDS(L) &= 2\sum_{i=1}^{m} c(B_i) > \sum_{i=1}^{m-1} s(B_i)+c(B_1)+c(B_m) \\ &> \sum_{i=1}^{m-1} s(B_i)+s(B_1) \geq \sum_{i=1}^{m} s(B_i) \geq OPT(L). \end{aligned}$$

Consider the following instance. Two bin sizes are 1 and $1-\varepsilon$, respectively, and two items have size $1/2$. The optimal value is 1 while the value of the packing by $FFDS$ is $2-\varepsilon$. Thus the bound 2 is tight. ■

3 Designing bin sizes

In this section we design bin sizes in terms of asymptotic waste ratio. We first consider a lower bound of any algorithms for a collection of bin sizes $\mathcal{B}=\{B^1,\ldots,B^l\}$. Assume that there are k bin sizes in $\mathcal{B}$, which are greater than 1/2, denoted by $1=s_1>s_2>\cdots>s_k>1/2$. The remaining bin sizes in $\mathcal{B}$, which are at most $1/2$, will not be taken into account. Let $s_{k+1}=1/2$ be a dummy bin size. Define

$$r(\mathcal{B})=\max_{i=1}^{k}\frac{s_i}{s_{i+1}}.$$

Theorem 5 *For any algorithm A, the asymptotic waste ratio* $W_A^\infty \geq r(\mathcal{B})$.

Proof. Omitted. ■

Corollary 1 *For any collection of bin types in which there are k bin sizes are greater than 1/2, the asymptotic waste ratio of any algorithm is at least* $2^{\frac{1}{k}}$.

By simple calculation, we can get the *best collection of k bin sizes*:

$$\mathcal{B}_k^* = \{1, 2^{-\frac{1}{k}}, \ldots, 2^{-\frac{i-1}{k}}, \ldots, 2^{-\frac{k-1}{k}}\}.$$

In the following we present an off-line algorithm and an on-line algorithm and prove that their asymptotic waste ratios match the lower bound.

Algorithm FFR (First-Fit with Repacking)

1. Open a bin of size $s_1 = 1$. Set it to be *open*.
2. If the current item is a large item (with size greater than 1/2), immediately assign the item to a smallest possible empty bin which it can fit in. Close this bin.
3. If the current item is a small item (with size at most 1/2), pack it into the open bin. If the content of the open bin is over 1/2, empty the open bin by moving its items into a smallest possible empty bin. The latter is immediately closed.

Theorem 6 $W_{FFR}^\infty = r(\mathcal{B})$ *for any bin collection* $\mathcal{B}$.

Proof. For any bin B, let its size be $s(B)$ and its content be $c(B)$. If $s_{i+1} < c(B) \leq s_i$ $(1 \leq i \leq k)$, then $s(B) = s_i$. It implies that $s(B)/c(B) \leq s_i/s_{i+1} \leq r(\mathcal{B})$. If $c(B) \leq 1/2$, then $s(B) = s_k$. It is clear that at most one bin has a content not greater than 1/2. Thus

$$FFR(L) = \sum s(B) \leq \sum r(\mathcal{B})c(B) + s_k = r(\mathcal{B})S(L) + s_k.$$

The theorem holds. ■

Corollary 2 *Algorithm FFR is the best possible for any combination of bin sizes. In particular, for the collection of bin sizes* $\mathcal{B}_k^*$ *the asymptotic waste ratio is* $2^{\frac{1}{k}}$, *which is the best possible over all combinations of k bin sizes.* ■

Algorithm FFR is off-line, since *repacking* is required. We provide an on-line algorithm as follows. Consider the collection $\mathcal{B}_k^*$ of bin sizes. Let $\Delta = \max_{i=1}^{k}(s_i - s_{i+1})$. Note that $s_i = 2^{-\frac{i-1}{k}} = s_2^{i-1}$ for $i = 2, \ldots, k$, and $s_k = 1/(2s_2)$. Then $s_i -$

$s_{i+1} = s_i(1-s_2) = s_i(s_1-s_2) < s_1 - s_2$, for $i = 2, \ldots, k-1$. And $s_k - s_{k+1} = s_k - 1/2 = (1-s_2)/(2s_2) < s_1 - s_2$. Thus $\Delta = s_1 - s_2 = 1 - 2^{-1/k}$. Let m be the smallest integer such that $2^{-(m+1)} < \Delta$.

Classify the small items a_j (with size $s(a_j) \leq 1/2$) as follows:

$$I_{i,l} = \{a_j | s_{i+1}/2^l < s(a_j) \leq s_i/2^l\},$$
$$i = 1, \ldots, k,\ l = 1, \ldots m$$
$$I_0 = \{a_j | s(a_j) \leq s_{k+1}/2^m = 2^{-(m+1)}\}.$$

Algorithm *RV*

1. If the current item a_j is a large item, pack it into a smallest possible empty bin and close this bin.

2. If the current item a_j is a small item and belongs to I_0, put it into the open bin of class I_0 if such a bin exists and has enough space for a_j. Otherwise close the open bin for class I_0 and open a new bin of size 1.

3. If the current item a_j is a small item and belongs to the $I_{i,l}$, put it into the open bin of class $I_{i,l}$ if such a bin exists and has enough space for a_j. Otherwise close the open bin for class $I_{i,l}$ and open a new bin of size s_i.

Note that the number of open bins at a time may be $mk+1$ where the value of m depends on the value of k.

Theorem 7 *$W_{RV}^{\infty} = r(\mathcal{B}_k^*) = 2^{1/k}$ for the bin collection $\mathcal{B}_k^*$, where k is a fixed integer.*

Proof. Omitted. ■

4 Concluding remarks

In this paper we have considered on-line algorithms in terms of absolute worst-case ratio. To design bin sizes we have presented best possible algorithms in terms of asymptotic waste ratio. The best collection of bin sizes has been also given.

For variable-sized bin packing there are many open questions. In terms of asymptotic worst-case ratio, for only two different bin types, which combination of sizes produces the smallest worst-case ratio? What can be said about the problem with at least three bin sizes? In terms of absolute worst-case ratio, what is a lower bound for off-line algorithms? How to design an optimal on-line algorithm?

References

[1] R. E. Burkard and G. Zhang. Bounded space on-line variable-sized bin packing. *Acta Cybernetica*, 13: 63–76, 1997.

[2] C. Chu, R. La. Variable-sized bin packing: tight absolute worst-case performance ratios for four approximation algorithms. *SIAM J. Computing*, 30: 2069–2083, 2001.

[3] E.G. Coffman, Jr., M.R. Garey, and D.S. Johnson. Approximation Algorithms for Bin Packing: A Survey. In *Approximation Algorithms for NP-Hard Problems*, D. Hochbaum (editor), PWS Publishing, Boston, 46-93, 1996.

[4] J. Csirik. An on-line algorithm for variable-sized bin packing. *Acta Informatica* 26: 697–709, 1989.

[5] J. Csirik, G.J. Woeginger. On-line packing and covering problem. In *On-Line Algorithms - The State of the Art*, A. Fiat and G.J. Woeginger, Eds., Lecture Notes in Computer Science 1442: 147-177, 1998.

[6] L. Epstein, S. Seiden, and R. van Stee. New bounds for variable-sized and resource augmented online bin packing. In *Proc. of 29th International Colloquium on Automata, Languages and Programming (ICALP)*, 306-317, 2002.

[7] D.K. Friesen, M.A. Langston. Variable sized bin packing. *SIAM J. Computing* 15: 222–229, 1986.

[8] N.G. Kinnersley, M.A. Langston. Online variable-sized bin packing. *Discrete Applied Mathematics* 22: 143–148, 1988/89.

[9] C.C. Lee, D.T. Lee. A simple on-line bin-packing algorithm. *J. ACM* 2: 562–572, 1985.

[10] F.D Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM J. Computing* 16: 149–161, 1987.

[11] S. Seiden. An optimal online algorithm for bounded space variable-sized bin packing. *SIAM J. Discrete Mathematics* 14: 458–470, 2001.

[12] D. Simchi-Levi. New worst-case results for the bin packing problem. *Naval Research Logistics* 41: 579–585, 1994.

[13] G. Zhang. Worst-case analysis of the FFH algorithm for online variable-sized bin packing. *Computing* 56: 65–172, 1996.

Guochuan Zhang is with Institut für Informatik und Praktische Mathematik , Kiel Universität, Olshausenstr. 40, 24098 Kiel, Germany. E-mail: gzh@informatik.uni-kiel.de. On leave from the Department of Mathematics, Zhejiang University, Hangzhou, China.