Acta Cybernetica, Vol. 11, No. 1-2, Szeged, 1993

A Lower Bound for On-Line Vector-Packing Algorithms*

G. Galambos† H. Kellerer ‡ G. Wöginger§

Abstract

In this paper we deal with the vector-packing problem which is a generalization of the well known one-dimensional bin-packing problem to higher dimensions. We give the first, non-trivial lower bounds on the asymptotic worst case ratio of any on-line d-dimensional vector packing algorithm.

Keywords. vector-packing, worst-case analysis, on-line algorithms, lower bounds, competitive algorithms.

1 Introduction

We consider the following problem, called *vector-packing*: Given a list $L_n = \langle a_1, \ldots a_n \rangle$ of n elements where each element is a d dimensional *vector* ($d \geq 1$). The i-th vector in the liste is denoted by $v(a_i) = (v_1(a_i), \ldots, v_d(a_i))$, where $0 \leq v_j(a_i) \leq 1$ for $j = 1, 2, \ldots, d$. The goal is to pack all elements into the minimal number of bins in such a way that for any non-empty B bin of the packing and for any index $1 \leq j \leq d$

$$\sum_{a_i \in B} v_j(a_i) \leq 1.$$

For $d = 1$, this problem is the famous "classical" bin-packing problem, which is known to be NP-hard. Hence, we are mainly interested in 'good' approximation algorithms.

The quality of an approximation algorithm is usually measured by its *asymptotic worst-case ratio* that is defined as follows. For an arbitrary vector-packing algorithm A and an arbitrary list of d-dimensional vectors L, we denote by L^* the minimal number of bins needed to pack the list L and by $A(L)$ the number of bins which algorithm A uses to pack the elements of L. Let $R_A(k)$ denote the supremum of the ratios $A(L)/L^*$ over all lists L with $L^* = k$. The asymptotic worst case ratio R_A is defined by the equation

$$R_A = \limsup_{k \longrightarrow \infty} R_A(k).$$

*This research was supported by a grant from the Hungarian Academy of Sciences (OTKA Nr. 2037) and by the Christian Doppler Laboratorium für Diskrete Optimierung.

†Department of Computer Sciences, Teacher Trainer College, Szeged, Hungary.

‡Institute of Mathematik, University Graz, A-8010 Graz, Austria

§Institute of Mathematik, Technical University Graz, A-8010 Graz, Austria

The first approximation algorithms for vector-packing were designed by Kou and Markowsky [3] . They defined so-called *irreducible* algorithms as-follows. During the packing of an irreducible algorithm, for any two non-empty bins B_p and B_q there exists an index j, $1 \leq j \leq d$ with

$$\sum_{a \in B_p} v_j(a) + \sum_{a \in B_q} v_j(a) > 1.$$

(This means that the algorithm only opens a new bin if a newly arrived item can not be packed into any old bin.) Kou and Markowsky proved the following proposition.

Proposition 1.1 (Kou and Markowsky, [3]) *The asymptotic worst case ratio of any irreducible algorithm fulfills*

$$R_A \leq d + 1.$$

Garey, Graham, Johnson and Yao [1] generalized the First-Fit (FF) and the First-Fit Decreasing (FFD) algorithms to the d-dimensional case. They proved that

$$R_{FF} = d + \frac{7}{10},$$

$$d \leq R_{FFD} \leq d + \frac{3}{10}$$

Note that both of these algorithms are irreducible and hence fulfill the statement of Proposition 1.1.

Now let us turn to lower bounds on the worst case ratios of heuristics. Yao in [6] studied the following class of the "decision-tree" algorithms. Let A be an algorithm for the vector-packing problem. For each $n > 0$, the action of A on a list L can be represented by a ternary tree $T_n(A)$. Each internal node of $T_n(A)$ contains a test. For any input L, the algorithm moves down the tree, testing and branching according to the result of the test, until it reaches some leaf. At the leaf, a packing valid for all lists that lead to this leaf is produced. The cost of A for input size n, $C_n(A)$, is defined to be the number of tests made in the worst-case. (In fact, this is the height of $T_n(A)$). Yao proved that if A is such an algorithm for which $C_n(A) = o(n \log n)$ then $R_A \geq d$.

In this paper we deal with the class of the *on-line* algorithms: If an algorithm A is in this class then it packs the elements one by one in the order given by the list L. After having packed an element into some bin, the element will be never moved again. E.g. algorithm FF mentioned above is an on-line algorithm. For $d \geq 2$ FF has the best worst case ratio among all known on-line heuristics for d-dimensional vector-packing.

As a consequence of the classical result of Liang [5] for one-dimensional on-line bin-packing algorithms, the inequality $R_A \geq 1.5364\ldots$ holds for all $d \geq 1$. Till today there is no better results were known. In this paper we will prove a d-dependent lower bound for on-line vector-packing algorithms. A formula for our lower bounds is given in Theorem 2.1. Table 1 depicts the numerical values for some small dimensions.

The rest of the paper is organized as follows. Section 2 contains some preliminaries and describes the construction of a bad item list for on-line heuristics. Section 3 gives a rigorous proof for the lower bound. Section 4 finishes with the conclusions.

d	Lower Bound	d	Lower Bound
2	1.67072	7	1.87504
3	1.75098	8	1.88891
4	1.80035	9	1.90002
5	1.83348	10	1.90910
6	1.85722	∞	2.00000

Table 1: Our lower bounds, rounded to five decimal places.

2 The construction

We start with defining the following sequence for any fixed $d \geq 1$. (Note that for every d, the reciprocal values $1/t_i(d)$ sum up to $1/2d$).

$$\begin{aligned} t_0(d) &= 2d+1 \\ t_i(d) &= t_{i-1}(d)(t_{i-1}(d)-1)+1, \qquad i \geq 1. \end{aligned}$$

A similar sequence introduced by Golomb [2] became one of the main tools in on-line bin-packing. Lee and Lee [4] used it to design a good one-dimensional bin-packing heuristic, and Liang [5] based his lower bound proof on the Golomb sequence.

With this definition, our main result may be stated as follows.

Theorem 2.1 *For any on-line d-dimensional vector-packing algorithm A, its asymptotic worst case ratio is at least*

$$R_A(d) \geq \frac{2d + \sum_{j=1}^{\infty} \frac{2d+j}{t_j(d)-1}}{\sum_{j=1}^{\infty} \frac{1}{t_j(d)-1} + d + \frac{1}{2}}.$$

Remark. If we set $d = 1$ in Theorem 2.1, we exactly arrive at the well-known lower bound of Liang [5].

The exact values for $2 \leq d \leq 10$ are depicted in Table 1. As d tends to infinity, the lower bound tends to 2. The remaining part of this paper is devoted to the proof of Theorem 2.1.

Intuitively speaking, the underlying idea of our paper is as follows. We construct an adverse strategy that forces *every* on-line algorithm A to behave poorly on a special item list L or on some prefix of L. In the first step, we give A a list of very small items to pack. In case A spreads these items on many bins, it does not receive any further item and looses the game. In case A produces a 'reasonable' packing for the small items, it receives another list of items. Again, A has the choice between either producing a bad packing and loosing the game immediately, or producing a (currently) good packing and receiving another list. Then in the final step, A gets a list of big items. Now it turns out that everything it did before was wrong. It had better packed the smaller items in such a way that remained enough space to pack the big items. A looses the game against the adversary.

Now we start with the definition of the item lists. Let $d \geq 1$ and $r \geq 1$ be arbitrarily fixed integers. We consider the following lists each consisting of n elements.

	L_6	L_5	L_4	L_3	L_2	L_1	L^1	L^2
$v_1(\cdot)$	$\frac{1}{2}+\delta$	$\frac{1}{3}+\delta$	0	0	0	0	0	0
$v_2(\cdot)$	$\frac{1}{4}+\delta$	$\frac{1}{4}+\delta$	$\frac{1}{5}+\delta$	0	0	0	0	0
$v_3(\cdot)$	$\frac{1}{6}+\delta$	$\frac{1}{6}+\delta$	$\frac{1}{6}+\delta$	$\frac{1}{6}+\delta$	$\frac{1}{6}+\delta$	$\frac{1}{7}+\delta$	$\frac{1}{43}+\varepsilon_1$	$\frac{1}{1807}+\varepsilon_2$

Table 2: The elements used in the lists for $d = 3$ and $r = 2$

1. For any $j \in \{1, \ldots, r\}$ and $a \in L^j$,

$$v_i(a) = \begin{cases} 0 & \text{if } i < d \\ \frac{1}{t_j(d)} + \varepsilon_j(r) & \text{if } i = d. \end{cases}$$

2. For any $k \in \{1, \ldots, d\}$ and $a \in L_{2k-1}$,

$$v_i(a) = \begin{cases} 0 & \text{if} \quad i \le d-k \\ \frac{1}{2i+1} + \delta & \text{if} \quad i = d-k+1 \\ \frac{1}{2i} + \delta & \text{if} \quad i = d-k+p, p = 2, \ldots, k. \end{cases}$$

3. For any $k \in \{1, \ldots, d\}$ and $a \in L_{2k}$,

$$v_i(a) = \begin{cases} 0 & \text{if} \quad i \le d-k \\ \frac{1}{2i} + \delta & \text{if} \quad i = d-k+p, p = 1, \ldots, k. \end{cases}$$

where

$$\delta < \frac{1}{4d(t_{r+1}(d)-1)},$$

$$\varepsilon_1(r) < \frac{1}{2r(t_{r+1}(d)-1)},$$

and

$$\varepsilon_{j+1}(r) < \frac{1}{t_j(d)-1}\varepsilon_j(r), \quad 1 \le j \le r-1.$$

The lists are presented to the on-line heuristic in the following order: First there come the lists L^j with j going down from r to 1, and afterwards there come the lists L_j with j going up from 1 to $2d$. The lists L^j with superscript contain the very small items (all components of the corresponding vectors are zero with the exception of the component with index d). The lists L_j with subscript, $1 \le j \le d$ contain the larger items; list L_{2d} is the list with the big items that arrive in the final step. An illustration for $d = 3$ and $r = 2$ is given in Table 2.

Convention. Next we shall work under a fixed dimension d and a fixed r. To simplify our notations, we shall use t_j and ε_j instead of $t_j(d)$ and $\varepsilon_j(r)$.

3 The Proof

In this section we prove that any on-line heuristic must perform poorly on the list $L = L^r \dots L^1 L_1 \dots L_{2d}$ (as defined in the preceding section) or on some prefix of L.

Observation 3.1 *For any integer* $1 \le j \le r$,

$$\sum_{i=j}^{r} \left(\frac{1}{t_i} + \varepsilon_i\right) < \frac{1}{t_j - 1} - 2d\delta.$$

Proof. It can be proved by induction from definitions of t_i, ε_i and δ. □

Lemma 3.2 *For any integer* $n > 0$, *if* $(t_{r+1} - 1) | n$ *then*

$$(L^r \dots L^j)^* \le \frac{n}{t_j - 1} \quad 1 \le j \le r.$$

Proof. In this case $\frac{n}{t_j - 1}$ $(j = 1, 2, \dots, r)$ are positive integers. On the other hand, by Observation 3.1, we can pack $t_j - 1$ items of each of the lists $L^r, \dots, L^j$ together into one bin. □

Now for any integer $1 \le j \le 2d$, let us define the set N_j in the following way:

$$N_1 = N_2 = \{k(t_{r+1} - 1) : k = 1, 2, \dots\},$$

$$N_j = \{n(2d + 1 - j) : n \in N_{j-1}\} \quad 3 \le j \le 2d.$$

It is clear that $N_1 \supseteq N_2 \supseteq \dots \supseteq N_{2d}$.

Lemma 3.3 *For any* $1 \le j \le 2d$ *and* $n \in N_j$,

$$(L^r \dots L^1 L_1 \dots L_j)^* \le \frac{j}{2d} \cdot n.$$

Proof. The statement is proved by induction on j. First, the simple cases $j = 1$ and $j = 2$ are considered; the induction step is structured into two subcases. All we have to do that is to give a feasible packing. Note that Observation 3.1 yields

$$\sum_{i=1}^{r} \left(\frac{1}{t_i} + \varepsilon_i\right) < \frac{1}{t_1 - 1} - 2d\delta.$$

$(j = 1)$. Let $n \in N_1$ be arbitrary. By the definition of N_j, $2d | n$. So we always pack $2d$ elements from each list of $(L^r \dots L^1 L_1)$ together into one bin B. If $i < d$ then for any $a \in B$ $v_i(a) = 0$ holds, and if $i = d$ we have

$$\sum_{a \in B} v_d(a) < 2d\left(\frac{1}{2d+1} + \delta + \frac{1}{2d(2d+1)} - 2d\delta\right) < 1.$$

Hence we have a legal packing, using $\frac{n}{2d}$ bins.

($j = 2$). Let $n \in N_2$ be arbitrary. Then $d|n$. Let us pack together d elements from every list. For $i < d$ $v_i(a) = 0$ holds for each $a \in (L^r \dots L^1 L_1 L_2)$, and for $i = d$ we have

$$\sum_{a \in B} v_d(a) < d\left(\frac{1}{2d+1} + \delta + \frac{1}{2d} + \delta + \frac{1}{2d(2d+1)} - 2d\delta\right) \leq 1.$$

Therefore we obtain a feasible packing, using $\frac{n}{d}$ bins.

(Induction step) Now let $3 \leq j \leq 2d$ and assume that for any positive integer $j' < j$, the statement is valid. Let $n \in N_j$ be arbitrary. We shall distinguish two cases depending on whether j is odd or even.

A. $j = 2l - 1$ for some $2 \leq l \leq d$. In the sequel we say that a non-empty bin has type $\tau = (\tau^r, \dots \tau^1, \tau_1, \dots, \tau_{2d})$ if it contains exactly τ^i resp. τ_i elements from the list L^i resp. L_i. Let us pack the elements of the concatenated list $L^r \dots L^1 L_1 \dots L_j$ together into a bin B with type

$$\tau = (\underbrace{1, \dots, 1}_{2l+r-2}, 2d - 2l + 2, \underbrace{0, \dots, 0}_{2d-2l+1}).$$

First, we will prove that this gives a legal packing, i.e. the following claim holds for the bin B.

Claim 3.4

$$\sum_{a \in B} v_i(a) \leq 1 \qquad 1 \leq i \leq d.$$

Proof. The proof of this claim is divided into cases (i) thru (iv).

(i) If $i \leq d - l$, then $\sum_{a \in B} v_i(a) = 0$.

(ii) If $i = d - l + 1$ then only the elements of L_{2l-1} have non-zero coordinates and therefore

$$\sum_{a \in B} v_i(a) = (2d - 2l + 2)\left(\frac{1}{2d - 2l + 3} + \delta\right) < 1.$$

(iii) If $d - l + 1 < i < d$ then

$$\begin{aligned} \sum_{a \in B} v_i(a) &= (2d - 2l + 2)\left(\frac{1}{2i} + \delta\right) + (2i - 2 - 2d + j)\left(\frac{1}{2i} + \delta\right) \\ &+ \left(\frac{1}{2i+1} + \delta\right) \\ &= (2i - 1)\left(\frac{1}{2i} + \delta\right) + \left(\frac{1}{2i+1} + \delta\right) < 1. \end{aligned}$$

(iv) If $i = d$ then

$$\begin{aligned} \sum_{a \in B} v_i(a) &\leq (2d - 2l + 2)\left(\frac{1}{2d} + \delta\right) + (j - 2)\left(\frac{1}{2d} + \delta\right) \\ &+ \left(\frac{1}{2d+1} + \delta\right) + \left(\frac{1}{2d(2d+1)} - 2d\delta\right) \\ &= \frac{2d-1}{2d} + \frac{1}{2d+1} + \frac{1}{2d(2d+1)} = 1. \end{aligned}$$

This completes the proof of Claim 3.4 □

To get a feasible packing for $(L^r \dots L^1 L_1 \dots L_j)$, we first take $\frac{n}{2d-2l+2} = \frac{n}{2d-j+1}$ pieces of r type bins. By the definition of N_j, we know that $2d+1-j \mid n$, and so, we can pack all the elements of L_j into $\frac{n}{2d+1-j}$ bins. From the other lists, there remain $\bar{n} = n - \frac{n}{2d+1-j} = \frac{n}{2d+1-j}(2d-j)$ items. By the definition of N_j, $\bar{n} \in N_{j-1}$. But then, by our induction hypothesis, these remaining items can be packed into $\bar{n}\frac{j-1}{2d}$ bins. Therefore, we can pack all elements of $(L^r \dots L^1 L_1 \dots L_j)$ into

$$\frac{n}{2d-j+1} + \bar{n}\frac{j-1}{2d} = n\frac{2d+(j-1)(2d-j)}{(2d+1-j)2d} = n\frac{j}{2d}$$

bins, and case **A** is settled.

B. $j = 2l$, $2 \le l \le d$. In this case we are going to pack $d-l+1$ items using the bin type below:

$$r = (\underbrace{1, \dots, 1}_{2l+r-2}, d-l+1, d-l+1, \underbrace{0, \dots, 0}_{2d-2l}).$$

Claim 3.5

$$\sum_{a \in B} v_i(a) \le 1 \qquad 1 \le i \le d.$$

Proof. The proof is done in a similar way as the proof of Claim 3.4:
(i) if $i \le d-l$ holds then the above sum is equal to 0,

(ii) if $i = d-l+1$ then only the lists L_{2l-1} and L_{2l} have positive coordinates on the position i

$$\sum_{a \in B} v_i(a) = (d-l+1)(\frac{1}{2i} + \delta) + (d-l+1)(\frac{1}{2i+1} + \delta) < 1,$$

(iii) if $d-l+1 < i < d$ then

$$\begin{aligned} \sum_{a \in B} v_i(a) &= (2d-2l+2)(\frac{1}{2i} + \delta) + (2i-3-2d+j)(\frac{1}{2i} + \delta) + (\frac{1}{2i+1} + \delta) \\ &= (2i-1)(\frac{1}{2i} + \delta) + (\frac{1}{2i+1} + \delta) < 1, \end{aligned}$$

(iv) if $i = d$ then

$$\begin{aligned} \sum_{a \in B} v_i(a) &\le (2d-2l+2)(\frac{1}{2d} + \delta) + (j-3)(\frac{1}{2d} + \delta) + (\frac{1}{2d+1} + \delta) \\ &+ (\frac{1}{2d(2d+1)} - 2d\delta) = 1. \end{aligned}$$

Thus, Claim 3.5 is true.

To obtain a feasible packing for $(L^r \dots L^1 L_1 \dots L_j)$, we first take $\frac{n}{d-l+1}$ pieces of r type bins. By the definition of N_j, from $n \in N_j$ it follows that $n = (2d+1-j)(2d+2-j)n'$ with $n' \in N_{j-2}$, provided that $j \geq 4$. But then $n = 2(2d+1-j)(d-l+1)n'$. Therefore, $d-l+1|n$, and so, we can pack all the elements of L_{j-1} and L_j into $\frac{n}{d-l+1}$ bins. After this packing each list from $(L^r, \dots, L^1, L_1, \dots, L_{j-2})$ contains $\bar{n}$ unpacked elements where $\bar{n} = n - \frac{n}{d-l+1} = \frac{n}{d-l+1}(d-l)$.

Now let us observe that $\bar{n} \in N_{j-2}$. Then, by our induction hypothesis, the unpacked items can be packed into $\bar{n}\frac{j-2}{2d}$ bins. Therefore, we can pack all elements of $(L^r \dots L^1 L_1 \dots L_j)$, into

$$\frac{n}{d-l+1} + \bar{n}\frac{j-2}{2d} = n\frac{2d+(j-2)(d-l)}{(d-l+1)2d} = n\frac{j}{2d}$$

bins, which completes the considered case and the proof of Lemma 3.3 too. □

Lemmas 3.2 and 3.3 give us upper bounds for the number of bins in the optimal packings. Next, we will investigate the potential behaviour of arbitrary on-line algorithms on the constructed list L. We introduce the following notations:

- $\beta = \{B_1, \dots, B_{A(L^r \dots L^1 L_1 \dots L_{2d})}\}$ denotes the final packing of the concatenated list $(L^r \dots L^1 L_1 \dots L_{2d})$ produced by the on-line heuristic A. For any type $\tau = (\tau^r \dots \tau^1 \tau_1 \dots \tau_{2d})$, the number $a(\tau)$ equals the number of bins of type τ in the packing β.
- The subset β^i resp. β_j, contain only those bins which were used for the first time by the on-line heuristic A during the packing of the list L^i resp. L_j (i.e. their first item comes from L^i resp. L_j). Moreover, define for every $1 \leq i \leq r$ and $1 \leq j \leq 2d$ the sets:

  $T^i = \{\tau$: there exists a bin of type τ in $\beta^i\}$,

  $T_j = \{\tau$: there exists a bin of type τ in $\beta_j\}$,

  and

  $T = \{\tau$: there exists a bin of type τ in $\beta\} = \bigcup_{1 \leq i \leq r} T^i \cup \bigcup_{1 \leq j \leq 2d} T_j$.

Now we investigate the number of bins used by an arbitrary on-line algorithm A while A is packing the elements of the concatenated list $(L^r \dots L^1 L_1 \dots L_j)$.

$$A(L^r \dots L^i) = \sum_{l=i}^{r} \sum_{\tau \in T^l} a(\tau), \qquad 1 \leq i \leq r, \tag{1}$$

$$A(L^r \dots L^1 L_1 \dots L_j) = \sum_{l=1}^{r} \sum_{\tau \in T^l} a(\tau) + \sum_{l=1}^{j} \sum_{\tau \in T_l} a(\tau) \qquad 1 \leq j \leq 2d \tag{2}$$

and the number of the packed elements for each i resp. j, $1 \leq i \leq r$, $1 \leq j \leq 2d$:

$$n = \sum_{\tau \in T} \tau^i a(\tau), \qquad 1 \le i \le r. \tag{3}$$

$$n = \sum_{\tau \in T} \tau_j a(\tau), \qquad 1 \le j \le 2d. \tag{4}$$

Let us multiply the equations of (3) by $\frac{2d+i}{t_i-1}$. Summarizing the equations of (1) – (2) and subtracting the multiplied equations of (3) and (4) we get:

$$\sum_{i=1}^{r} A(L^r \dots L^i) + \sum_{j=1}^{2d} A(L^r \dots L^1 L_1 \dots L_j) - 2dn - n \sum_{i=1}^{r} \frac{2d+i}{t_i - 1} =$$

$$= \sum_{i=1}^{r} (2d+i) \sum_{\tau \in T^i} a(\tau) + \sum_{j=1}^{2d} (2d-j+1) \sum_{\tau \in T_j} a(\tau) - \tag{5}$$

$$\sum_{\tau \in T} a(\tau) \left(\sum_{i=1}^{r} \frac{2d+i}{t_i - 1} \tau^i + \sum_{j=1}^{2d} \tau_j \right).$$

Lemma 3.6 *The right hand side of (5) is non-negative.*

Proof. The proof is constructed into three parts.

A. First we prove that for any $1 \le i \le r$ and $\tau \in T^i$

$$\sum_{s=1}^{r} \frac{2d+s}{t_s - 1} \tau^s + \sum_{v=1}^{2d} \tau_v \le 2d + i.$$

Since $\tau \in T^i$, $\tau^r = \dots = \tau^{i+1} = 0$ and $\tau^i > 0$. Now if we have some component $\tau_v > 0$ for some v (i.e. some item from L_v is contained in the corresponding bin), then we replace this item by $2d$ elements of L^1. After the replacement we obtain a feasible packing of the considered bin and a new bin type $\bar{\tau}$ which is not neccessarily contained in T^i, but its first nonzero component is $(\bar{\tau})^i$. On the other hand, it is easy to check that the weighted sums on the left hand side do not decrease. Therefore, it is enough to prove that for any bin type τ of the items from the lists $L^r, \dots, L^1, L_1, \dots, L_{2d}$, if $\tau^r = \dots = \tau^{i+1} = 0$, then

$$\sum_{s=1}^{r} \frac{2d+s}{t_s - 1} \tau^s \le 2d + i.$$

Now we replace each element of L^u by $t_u - 1$ elements of L^{u+1}. This replacement results a feasible packing, since

$$(t_u - 1)\left(\frac{1}{t_{u+1}} + \varepsilon_{u+1}\right) \le \frac{1}{t_u} + \varepsilon_u.$$

On the other hand, the weighted sum in the newly constructed packing increases:

$$(t_u - 1)\frac{2d+u+1}{t_{u+1}-1} = \frac{2d+u+1}{t_u} > \frac{2d+u}{t_u - 1}.$$

Repeating this procedure for every $u < i$, we finally obtain a feasible packing with only items from L^i and with an increased weighted sum. Since for every feasible packing in a bin, $\tau^i \le t_i - 1$ holds, we obtain the desired result.

B. Secondly, we prove that for any $1 \le j \le 2d$ and $\tau \in T_j$

$$\sum_{v=1}^{2d} \tau_v \left(= \sum_{v=j}^{2d} \tau_v\right) \le 2d - j + 1.$$

B1. Let us consider the subcase $j = 2k$, $1 \le k \le d$. We examine the $(d-k+1)$-th coordinate of the list $L_j, \dots, L_{2d}$. Because of the definitions, it follows for each list that for any $a \in (L_j \dots L_{2d})$, $v_{d-k+1}(a) = \frac{1}{2d-j+2} + \delta$, and so the statement is true.

B2. If $j = 2k - 1$ then we again consider the $(d-k+1)$-th coordinate. Now the smallest elements in this coordinate are those ones which belong to the list L_j: if $a \in L_j$ then $v_{d-k+1}(a) = \frac{1}{2d-j+2} + \delta$, and so the desired inequality holds.

C. Finally, we prove that the right hand side of (5) is nonnegative. Indeed, by case **A**, we obtain

$$\begin{aligned}
\sum_{i=1}^{r}(2d+i)\sum_{\tau\in T^i} a(\tau) &= \sum_{i=1}^{r}\sum_{\tau\in T^i} a(\tau)(2d+i) \\
&\ge \sum_{i=1}^{r}\sum_{\tau\in T^i} a(\tau)\Big(\sum_{s=1}^{r}\frac{2d+s}{t_s-1}\tau^s + \sum_{v=1}^{2d}\tau_v\Big) \\
&= \sum_{\tau\in \bigcup_{1\le i\le r} T^i} a(\tau)\Big(\sum_{s=1}^{r}\frac{2d+s}{t_s-1}\tau^s + \sum_{v=1}^{2d}\tau_v\Big).
\end{aligned}$$

On the other hand by the case **B**,

$$\begin{aligned}
\sum_{j=1}^{2d}(2d-j+1)\sum_{\tau\in T_j} &= \sum_{j=1}^{2d}\sum_{\tau\in T_j} a(\tau)(2d-j+1) \\
&\ge \sum_{j=1}^{2d}\sum_{\tau\in T_j} a(\tau)\Big(\sum_{v=1}^{2d}\tau_v\Big) \\
&= \sum_{\tau\in \bigcup_{1\le j\le 2d} T_j} a(\tau)\Big(\sum_{v=1}^{2d}\tau_v\Big).
\end{aligned}$$

Let us observe that for any $1 \le j \le 2d$ and $\tau \in T_j$, $\tau^r = \dots = \tau^1 = 0$, and so

$$\sum_{\tau\in \bigcup_{1\le j\le 2d} T_j} a(\tau)\Big(\sum_{s=1}^{r}\frac{2d+s}{t_s-1}\tau^s\Big) = 0.$$

Therefore the last three inequalities give us that the considered right hand side is nonnegative which completes the proof of Lemma (3.6). □

Now we are ready to prove of Theorem 2.1. For this reason let $n \in N_{2d}$ be arbitrary. Lemma 3.6 together with equation (5) yields

$$\sum_{i=1}^{r} A(L^r \dots L^i) + \sum_{j=1}^{2d} A(L^r \dots L^1 L_1 \dots L_j) \geq 2dn + n \sum_{i=1}^{r} \frac{2d+i}{t_i - 1} \qquad (6)$$

We define

$$r^i = \frac{A(L^r \dots L^i)}{(L^r \dots L^i)^*} \qquad 1 \leq i \leq r$$

$$r_j = \frac{A(L^r \dots L^1 L_1 \dots L_j)}{(L^r \dots L^1 L_1 \dots L_j)^*} \qquad 1 \leq j \leq 2d$$

and

$$R = \max\left\{ \max_i r^i, \max_j r_j \right\}.$$

Now plugging R into (6) and using the results stated in Lemmas 3.2 and 3.3, we get

$$nR \sum_{i=1}^{r} \frac{1}{t_i - 1} + R\frac{n}{2d} \sum_{j=1}^{2d} j \geq n(2d + \sum_{i=1}^{r} \frac{2d+i}{t_i - 1})$$

Finally, dividing by n and making $r \to \infty$ yields the statement of Theorem 2.1 □

4 Conclusion

In this paper we derived the first non-trivial lower bound for d-dimensional on-line vector packing algorithms. The best on-line algorithm known today, the First-Fit algorithm has asymptotic worst case ratio $d + \frac{7}{10}$. In relation to this result, our lower bound is not too attractive, as it remain beneath 2 for any given d and there is a wide gap to the upper bound.

Of course, the main open (and probably very hard) problem consists in giving a better lower bound for on-line approximation algorithms that tends to infinity as d tends to infinity, e. g. $\Omega(\sqrt{d})$ or $\Omega(\log d)$. Moreover, we invite the researchers to design better on-line algorithms with smaller asymptotic worst-case ratios. A good candidate might be the vector-generalization of the *Harmonic Fit* algorithm analysed by Lee and Lee [4].

Acknowledgment. We thank Günter Rote and Balázs Imreh for constructive criticisms on the earlier version of this paper.

References

[1] M.R.Garey, R.L.Graham, D.S. Johnson and A.C.C. Yao, Resource constrained scheduling as generalized bin packing, *J. Comb. Th. Ser. A.* **21**, (1976), 257-298.

[2] S. Golomb, On certain non-linear sequences, *American Math. Monthly* **70**, (1963), 403-405.

[3] L.T. Kou and G. Markowsky, Multidimensional Bin Packing Algorithms, *IBM Journal of Research and Development*, (1977), 443-448.

[4] C.C. Lee and D.T. Lee, A Simple On-line Bin Packing Algorithm, *J. Assoc. Comp. Mach.* **32**, (1985), 562-572.

[5] F.M. Liang, A Lower Bound for On-line Bin Packing, *Inf. Proc. Letters* **10**, (1980), 76-79.

[6] A.C.C.Yao, New Algorithms for Bin Packing, *J. Assoc. Comp. Mach.* **27**, (1980), 207-227.

(Received May 20, 1991.)

(Revised September 10, 1993 and November 15, 1993.)