Acta Cybernetica **16** (2003) 47–56.

On variable sized vector packing

Leah Epstein*

Abstract

One of the open problems in on-line packing is the gap between the lower bound $\Omega(1)$ and the upper bound $O(d)$ for vector packing of d-dimensional items into d-dimensional bins. We address a more general packing problem with variable sized bins. In this problem, the set of allowed bins contains the traditional "all-1" vector, but also a finite number of other d-dimensional vectors. The study of this problem can be seen as a first step towards solving the classical problem. It is not hard to see that a simple greedy algorithm achieves competitive ratio $O(d)$ for every set of bins. We show that for all small $\varepsilon > 0$ there exists a set of bins for which the competitive ratio is $1 + \varepsilon$. On the other hand we show that there exists a set of bins for which every deterministic or randomized algorithm has competitive ratio $\Omega(d)$. We also study one special case for $d = 2$.

1 Introduction

The problem. We consider the following problem. We are given a finite set B of d-dimensional vectors in $[0,1]^d$. This is the set of bin sizes. The "all-1" vector $(1,1,\ldots,1)$ belongs to B. Items of sizes also in $[0,1]^d$ arrive on-line, to be assigned to bins of sizes in B. The packing needs to be valid, i.e. the vector sum of all items assigned to one bin cannot exceed the capacity of the bin (in any component). The "all-1" bin needs to be in B so that every item can fit into some bin. Each item, has to be assigned to a bin upon arrival, and cannot be moved after that. It can be assigned either to an open bin, or to a new bin of some size in B. The cost of a bin b is the sum of its components and the weight of an item is the sum of its components. The goal is to minimize the total cost of the bins that the algorithm uses.

Applications. The problem can be seen as a scheduling problem with limited resources. There are a few types of machines (the bins) with known and limited capacities of several resources as memory, running time, access to other computers etc. The items is this case are jobs that need to be run, each job requires a certain amount of each resource. Another application arises from viewing the problem as

*School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. Email: lea@idc.ac.il

Research supported in part by the Israel Science Foundation, (grant No. 250/01-1).

a storage allocation problem. Each bin has several qualities as volume, weight etc. Each item requires a certain amount of each quality.

In both applications it is likely for the items to arrive one by one, forcing the algorithm to make decisions without any knowledge of the future.

The quality measure. The competitive ratio of an on-line algorithm for this minimization problem is the worst case ratio, over all possible input sequences, of the cost of bins used by the on-line algorithm to the cost of bins used by an optimal off-line algorithm (which is familiar with the complete sequence in advance). Often an additive constant is allowed, yielding the following definition of the competitive ratio.

Definition 1.1. *For an on-line algorithm and a sequence I of items, let $C_{ONL}(I)$ be the cost of the bins used by the on-line algorithm and let $C_{OPT}(I)$ be the cost of the bins used by an optimal off-line algorithm. ($C_{ONL}(I)$ can be abbreviated by C_{ONL} and $C_{OPT}(I)$ can be abbreviated by C_{OPT}.) Let $R \geq 1$.*

An on-line algorithm is R-competitive *if there exists a constant b such that*
$$C_{ONL}(I) \leq R \cdot C_{OPT}(I) + b, \text{ for any sequence } I \text{ of items.}$$
The competitive ratio of an on-line algorithm is
$$r = \inf\{R \mid \text{the on-line algorithm is } R\text{-competitive}\}.$$

If the additive constant b is zero or negative, the algorithm is called *strictly R-competitive*. The negative results given in this paper are valid for the strict competitive ratio as well as for the competitive ratio in general. The positive results are valid for the general competitive ratio, as it is common for bin packing type problems.

For randomized algorithms, the competitive ratio is defined similarly, but $C_{ONL}(I)$ is replaced by $E(C_{ONL}(I))$.

A simple algorithm. The following algorithm achieves competitive ratio at most $2d$ for every set B. Hence the best competitive ratio for any set is $O(d)$. The algorithm uses only "all-1" bins, and packs the items in a "next-fit" fashion. It keeps one open active bin where all arriving items are packed, whenever an arriving item does not fit, this bin is closed and a new active bin is opened. To show the competitive ratio, partition all bins used by the algorithm into pairs, according to the order they were opened. (If the number of bins is odd, the last one is ignored). Now combine the contents of each pair. The items in the two bins could not fit into one bin, since the second is opened when an item does not fit into the first. Hence, at least one component of the combined contents is at least 1. Let W be the total weight of items. Let X be the number of bins used by the algorithm. Then $W \geq (X-1)/2$. The optimal off-line also need to pack the items hence $C_{OPT} \geq W$. Since $C_{ONL} = Xd$, we conclude that the algorithm is $2d$-competitive.

Previous work. To the best of our knowledge, no results exist on variable-sized vector packing. We mention the results for the classical on-line vector packing problem, where B consists of a single, all-1 bin. There is only a small number of results on on-line vector packing, and the problem seems to be difficult. Kou and Markowsky [10] considered a class of algorithms for which there never exists a pair of bins whose contents can be combined legally into a single bin. They showed that any algorithm in this class is $d+1$ competitive. Later [7] improved the analysis for a d-dimensional version of first-fit to $d+0.7$. The lower bounds [6, 1] are all below 2, tending to 2 as d goes to infinity. The best lower bound for $d=2$ is 1.6712 given by Blitz, Van Vliet, Woeginger in [1]. This large gap between the negative and positive results (for large d as well as for $d=2$) encouraged the current study of the model with variable sized bins. The main questions were as follows: Are there any examples for B where the competitive ratio is linear? Are there any examples where the competitive ratio is constant?

The results. We answer both questions positively. Specifically, we show a set B for which we design a constant competitive algorithm, moreover, for any small $\varepsilon > 0$, we give an algorithm of competitive ratio $1+\varepsilon$ (section 2). We further design a set for which we show a lower bound of $\Omega(d)$ on the competitive ratio of deterministic and randomized algorithms (section 3). In section 4, we design an algorithm for a special case $d=2$ and $|B|=3$. This algorithm demonstrates the simplicity in which it is possible to reduce the competitive ratio just by adding a small number of bins to B.

Other related work. A survey on on-line bin packing problems is given in [3]. The one dimensional variable-sized bin packing problem was studied in several papers [5, 9, 13, 2, 11, 4]. Those papers present and analyze various algorithms. Csirik [2] showed that there exists a choice of a set of bins (containing two bins of sizes 1 and 0.7) so that the competitive ratio (1.4) is much lower than the best known lower bound (1.5401 [12]) for the basic bin packing problem (a single type of bin which has size 1). In [4] improved upper bounds and new lower bounds for sets of two bins are given. The overall upper bound on the competitive ratio presented in that paper is $373/228 \approx 1.63596$.

2 A set with $1+\varepsilon$ approximation

In this section we introduce a bin set B for which an asymptotic competitive ratio, arbitrarily close to 1 is achieved by deterministic algorithms. Even though the algorithm is on-line, the methods are somewhat similar to those used in design of polynomial approximation schemes for off-line scheduling problems (see e.g. [8]).

Let $0 < \varepsilon < 1/3$ be a small positive constant. Let $\Delta = \lceil \frac{d}{\varepsilon} \rceil$. Let $\delta = 1/\Delta$. We define the set B_ε of allowed bins, as the set of the vectors $(a_1\delta^2, a_2\delta^2, \dots, a_d\delta^2)$ such that all a_i are integer, and $0 \le a_i \le \Delta^2$ for $1 \le i \le d$. The number of different bins is at most $(\Delta^2+1)^d = \Theta((\frac{d}{\varepsilon})^{2d})$. Note that taking $a_i = \Delta^2$ for all $1 \le i \le d$ gives the "all-1" bin.

We define an algorithm which has asymptotic competitive ratio $1+\varepsilon$ for the set B_ε of bins. An item is called *senior*, if it has at least one component of size at least δ, and *junior* otherwise. Senior items and junior items are packed by different methods. In both cases there is very little room left in the bins (apart from a constant number of bins) and the competitive ratio is proved by area considerations.

Senior items: Each senior item is packed into a separate bin. Given an item $x = (x_1, \dots, x_d)$, x is packed into a bin b such that if $b = (b_1, \dots, b_d)$ then $b_i \ge x_i$ but $b_i - \delta^2 < x_i$. In other words, $b_i = \lceil \frac{x_i}{\delta^2} \rceil \delta^2$.

Junior items: Those items are packed only into a subset of B_ε. Let $B_\varepsilon(i)$ be the subset of B_ε containing all vectors whose component i equals 1. Throughout the algorithm, for each $1 \le i \le d$ there is one open bin of each size in $B_\varepsilon(i)$, which is used for junior items whose largest component has index i. Given a junior item $x = (x_1, \dots, x_d)$, let $m = \max_{1 \le i \le d} x_i$, let j be the minimum index of a component which achieves the maximum (i.e. $x_j = m$ and $x_k < m$ for $k < j$). Let $y = x/m$. The item is assigned into an open bin of size $b' = (b'_1, \dots, b'_d)$ in $B_\varepsilon(j)$ such that $b'_i = \lceil \frac{y_i}{\delta^2} \rceil \delta^2$. Note that component j of b' is 1. If the item does not fit into the open bin, this bin is closed, a new bin of size b' associated with $B_\varepsilon(j)$ is opened and the item is assigned there. We are going to show that for every bin used by our algorithm (apart from the last bin of every size in $B_\varepsilon(i)$ for every $1 \le i \le d$, that is used for junior items), the cost of a bin is at most $1 + \varepsilon$ times the weight of items assigned to this bin. This would give

$$C_{ONL} \le (1+\varepsilon)W + d|B_\varepsilon| \tag{1}$$

where W is the total weight of items in the sequence. Since $C_{OPT} \ge W$ and $|B_\varepsilon|$ is constant (which depends on d and ε), this gives an asymptotic competitive ratio $1 + \varepsilon$ for a constant ε. We analyze senior and junior items separately.

Bins with a senior item: According to the definition of the algorithm, each bin contains a single item x. Let w_b the weight of the bin b where x is assigned and w_x be the weight of x. By the algorithm $w_b \le w_x + d\delta^2$. Since x is a senior item, $w_x \ge \delta$, hence $w_b \le w_x(1 + d\delta) \le w_x(1+\varepsilon)$.

Bins with junior items: Consider a bin $\beta = (\beta_1, \dots, \beta_d)$ that was used for junior items with maximal component of index k, and was closed during the algorithm. Let $y' = (y'_1, \dots, y'_d)$ be the sum vector of items assigned to this bin..

We prove the following two claims.

Claim 2.1. *For all $1 \le i \le d$, $\frac{y'_i}{\beta_i} \le y'_k$, and $y'_k \ge 1 - \delta$.*

Claim 2.2. *For all $1 \le i \le d$, $y'_i \ge y'_k \beta_i - \delta^2$.*

Before we prove the claims, we show this is sufficient to achieve the required competitive ratio. We need to compare w_β which is the cost of the bin β, to the weight of y', $w_{y'}$, which is the weight of the items in the bin. By Claim 2.2.

$$w_{y'} = \sum_{i=1}^{d} y'_i \ge y'_k \sum_{i=1}^{d} \beta_i - d\delta^2 \ .$$

Using this and the second part of Claim 2.1, we get $w_{y'} \ge w_\beta(1 - \delta - d\delta^2)$ (since $w_\beta \ge 1$). It is left to show that $1/(1 - \delta - d\delta^2) \le 1 + \varepsilon$. Since $\delta \le \varepsilon/d$ it is enough to show $(\varepsilon + 1)^2 \le d$ which is true for $\varepsilon \le \sqrt{2} - 1$ (since $d \ge 2$).

To complete the proof, we need to prove the claims. We start by proving Claim 2.1. Consider an item $\alpha = (\alpha_1, \ldots, \alpha_d)$ assigned to a bin β. Then let k be the maximal component of α which has the minimum index among the maximal components. Then α was assigned according to component k. Recall that β is calculated in the following way:

$$\beta_i = \lceil \frac{\alpha_i/\alpha_k}{\delta^2} \rceil \delta^2$$

and hence $\beta_i \geq \alpha_i/\alpha_k$. Since $\alpha_i/\beta_i \leq \alpha_k$ is true for all items in the bin, then also $y'_i/\beta_i \leq y'_k$. Now let γ be the item that caused this bin to be closed. γ did not fit into the bin, but since it was inserted to a bin of the same size also for $1 \leq i \leq d$, $\gamma_i/\gamma_k \leq \beta_i$. Since γ did not fit, for some component j, $y'_j + \gamma_j > \beta_j$. Hence $y'_k + \gamma_k \geq y'_j/\beta_j + \gamma_j/\beta_j > 1$. Since $\gamma_k < \delta$ (junior item) we get $y'_k > 1 - \gamma_k > 1 - \delta$. This proves Claim 2.1.

To prove Claim 2.2 recall that $\beta_i \leq \alpha_i/\alpha_k + \delta^2$. Hence $\alpha_i \geq \alpha_k\beta_i - \alpha_k\delta^2$. Let I be the set of all items assigned to β.

$$\begin{aligned}
y'_i = \sum_{\alpha \in I} \alpha_i &\geq \sum_{\alpha \in I} \alpha_k\beta_i - \sum_{\alpha \in I} \alpha_k\delta^2 = \\
(\beta_i - \delta^2)\sum_{\alpha \in I} \alpha_k &= y'_k(\beta_i - \delta^2) = \\
y'_k\beta_i - y'_k\delta^2 &\geq y'_k\beta_i - \delta^2.
\end{aligned}$$

The last inequality holds since $y'_k \leq 1$. This completes the proof of Claim 2.2.

Theorem 2.1. *The above algorithm has asymptotic competitive ratio of at most* $1 + \varepsilon$.

Proof. Follows from (1). □

3 A set with only $\Omega(d)$ approximations

In the introduction we showed that for every set, there exists an algorithm with competitive ratio $O(d)$. However, in the previous section we showed a set where it is possible to get an $1 + \varepsilon$ approximation. In this section we show a set for which we give a lower bound of $\Omega(d)$ on the competitive ratio.

We give a deterministic lower bound, and later show how to extend it to a randomized lower bound.

We start by a description of B. The set B contains apart from the vector $(1, \ldots, 1)$ also $2^{d/2}$ vectors which are called small bins. (We assume that d is even, for odd values of d it is possible to use the construction for the even dimension $d - 1$, setting the last component to zero in all items, and in all the bins apart from the "all-1" bin.)

For $k = 1, \dots, d/2$, the $(2k-1)^{th}$ and the $(2k)^{th}$ components of the bins are either $\frac{1}{d^{2k}}$ and $\frac{1}{d^{2k-1}}$ or $\frac{1}{d^{2k-1}}$ and $\frac{1}{d^{2k}}$ (respectively). Since every pair has two options, there are $2^{d/2}$ such possible bins. Throughout the sequence, the optimal off-line cost is going to be $\Theta(n/d)$. There are $d/2$ phases of items, with n items in each. (We pick n to be a large constant so that the lower bound is valid also for general competitive ratio and not only strict competitive ratio.) All items have weight $\Theta(1/d^2)$. Note that all bin costs are $\Theta(1/d)$ (apart from the "all-1" bin whose cost is d).

We now define the items of phase i. In phase i, for all items and for all $j > 2i$, the j^{th} component is zero. The components $2i-1$ and $2i$ are both $\frac{1}{d^{2i}}$. For all $j < i$, the components $2j-1$ and $2j$ are either 0 and $\frac{2}{d^{2j}}$ or $\frac{2}{d^{2j}}$ and 0 (respectively). Note that there can fit only at most one item of each phase in a small bin. The choice of the $(2j-1)^{th}$ and the $(2j)^{th}$ coordinates is done according to the behavior of the algorithm until the completion of phase j.

We say that the algorithm "may use" a bin in phase j if the bin is opened in phase j or if it was opened during phases $1, \dots, j-1$ and can still accommodate an item of phase j.

For $1 \le j \le \frac{d}{2}$, let N_j be the number of small bins, where the $(2j)^{th}$ component is $\frac{1}{d^{2j}}$, that the algorithm may use for items in phase j. (Not including bins opened after the arrival of the items of phase $j+1$.) If $j > 1$, in the beginning of phase j, some old bins cannot accommodate any more items due to a wrong structure of the $(2j-3)^{th}$ and the $(2j-2)^{th}$ components, those bins will never be used again. For $1 \le j \le \frac{d}{2}$, let M_j be the number of other small bins that may be used in phase j, that have the opposite structure of components $2j$ and $2j-1$ than bins counted in N_j. Note that the numbers M_j and N_j include new small bins opened during phase j, and previously opened bins that can still be used (the latter is true only for $j > 1$). If $N_j \le M_j$ then all future items have a zero in the $(2j)^{th}$ component and $\frac{2}{d^{2j}}$ in the $(2j-1)^{th}$ component. Otherwise, the structure is opposite. Hence all the M_j bins will never be used in the first case, and all N_j bins will never be used again in the second case. We use the following notations for $0 \le j \le d/2 - 1$. For $0 \le j \le \frac{d}{2} - 1$, let L_j be the number of small bins opened in phase $j+1$ and let S_j be the number of items assigned to an "all-1" bin in phase $j+1$. For $0 \le j \le \frac{d}{2} - 1$, let K_j be the total number of small bins that the algorithm may use in phase $j+1$. According to the above definitions, for $0 < j \le \frac{d}{2}$, $K_{j-1} = M_j + N_j$ and

$$K_j = L_j + \min(N_j, M_j) \le L_j + K_{j-1}/2 \ . \tag{2}$$

Since all items have either $\frac{1}{d^2}$ in both the first and the second components, or $\frac{2}{d^2}$ in one of the first two components, the algorithm can pack only at most d^2 items in one large bin. Hence $C_{ONL} \ge (\sum_{j=0}^{\frac{d}{2}-1} S_j)/d + 1/d \sum_{j=0}^{d/2-1} L_j$. We need to get a bound on those two sums. Note that in order to pack all items, for $j > 0$, $K_j + S_j \ge n$. Using (2) we get that $L_j + S_j \ge n - K_{j-1}/2$. $L_j + S_j$ represents the number of items that need to be either assigned to "all-1" bins, or have new bins opened for them.

On the other hand (2) gives the relations between the number of valid bins in phase $j+1$ to valid bins in phase j $(0 < j < d/2 - 1)$. Summing up the two equations for all $1 \le j \le \frac{d}{2} - 1$, and setting $L = \sum_{j=0}^{\frac{d}{2}-1} L_j$, $K = \sum_{j=0}^{\frac{d}{2}-1} K_j$, and $S = \sum_{j=0}^{\frac{d}{2}-1} S_j$, we get:

$$L - L_0 + S - S_0 \ge (\frac{d}{2} - 1)n - 1/2(K - K_{\frac{d}{2}-1}) .$$

and

$$K - K_0 \le L - L_0 + 1/2(K - K_{\frac{d}{2}-1}) .$$

Hence $L+S+\frac{1}{2}K \ge L_0+S_0+\frac{d}{2}n-n+1/2K_{\frac{d}{2}-1}$, and $L-K/2 \ge -K_0+L_0+1/2K_{\frac{d}{2}-1}$. Since $L_0 + S_0 \ge n$ (need to assign all items of phase 1), and all variables are non-negative, $L+S+K/2 \ge \frac{d}{2}n$. Since $K_0 = L_0$ then $L-K/2 \ge 0$. Hence $2L+S \ge \frac{d}{2}n$. We are interested in $(S+L)/d$. Easy substitution gives $C_{ONL} \ge (S+L)/d \ge (L+S/2)/d \ge n/4$.

On the other hand, the optimal off-line algorithm picks n small bins according to into which bin, an item of the last phase fits. Since $\frac{d}{2}(\frac{2}{d^{2j}}) = \frac{1}{d^{2j-1}}$, it is possible to place one item from each phase in a bin, the bin cost is $\Theta(1/d)$ and hence $C_{OPT} = \Theta(n/d)$. The competitive ratio follows.

To extend the proof for randomized algorithms, each one of the variables should be replaced by the expectation of this variable. By linearity of expectation, we get the same lower bound.

This proves the following Theorem.

Theorem 3.1. *There exists a finite set of bins, for which every deterministic or randomized algorithm for bin packing has competitive ratio $\Omega(d)$.*

4 A special case for $d = 2$

In this section we demonstrate by example, that letting the algorithm choose the set B, even if its size is very limited, allows the algorithm to improve the competitive ratio it achieves. In particular we consider $d = 2$ and $|B| = 3$. In section 1 it was shown that if $|B|$ is large enough (but finite), it is possible to achieve a very small competitive ratio. Here we focus on an example where $|B|$ is small, but a simple algorithm already improves on the best known algorithm for the classical case $B' = \{(1,1)\}$ given in [7] (whose competitive ratio is 2.7).

Let $B = \{(1,1),(1,\mu),(\mu,1)\}$. The constant $0 < \mu < 1/2$ is fixed later. We partition items into two classes:

- Items (β,γ) where $\beta \le \gamma$.
- All other items (i.e. items (β',γ') where $\beta' > \gamma'$).

Each one of the two classes is packed separately, independently from the other class. We explain how to pack the first class, the algorithm for the second class is symmetric (i.e. bins of size $(1,\mu)$ are used instead of bins of size $(\mu,1)$ and so on). The class is partitioned into six sub-classes. The algorithm also packs each one of the six sub-classes separately, independently from the other sub-classes. Let α be a constant $0 < \alpha \leq 1/3$, whose exact value is fixed later. The sub-class of an item (β, γ) is determined as follows:

- Sub-Class 1: $\gamma > 1/2$ and $\beta > \mu$.
- Sub-Class 2: $\gamma > 1/2$ and $\beta \leq \mu$.
- Sub-Class 3: $\alpha < \gamma \leq 1/2$ and $\beta > \mu/2$.
- Sub-Class 4: $\alpha < \gamma \leq 1/2$ and $\beta \leq \mu/2$.
- Sub-Class 5: $\gamma \leq \alpha$ and $\beta > \mu\gamma$.
- Sub-Class 6: $\gamma \leq \alpha$ and $\beta \leq \mu\gamma$.

Each item of sub-classes 1 and 2, is packed as an only item in a bin. For sub-class 1 the bin is of size $(1,1)$ and for sub-class 2 the bin is of size $(\mu, 1)$. Items of sub-classes 3 and 4 are packed in pairs (from the same sub-class). Pairs of sub-class 3 use bins of size $(1,1)$ whereas pairs of sub-class 4 use bins of size $(\mu, 1)$. By the definition of this class, (i.e. the conditions on γ and β), any two items of each of those sub-classes can fit into a bin together. The algorithm always has at most one bin for sub-class 3 with a single item. The same is true for sub-class 4 as well. The items of sub-class 5 are packed by a next-fit manner into bins of size $(1,1)$. The items of sub-class 6 are similarly packed into bins of size $(\mu, 1)$. In each of those two sub-classes, there is always one active bin. When an item does not fit, the bin is closed, and a new active bin is opened for the sub-class. Consider the sub-class 6. Given a set of items A, let $a = (a_1, a_2)$ be their sum vector. Then if $a_2 \leq 1$, $a_1 \leq \mu a_2 \leq \mu$. Hence $a_2 \leq 1$ is a satisfactory condition for all items in A to fit into one bin of size $(\mu, 1)$. For sub-classes 3 and 5, it is easy to see that the second component determines whether an item fits into a non-empty bin. This is true since for all items $\beta \leq \gamma$, but all the bins are of size $(1,1)$.

Next, we calculate the amount of occupied space in all closed bins. Those are all bins for sub-classes 1 and 2, and all bins but the very last ones opened for sub-classes 3, 4, 5 and 6. Those four last bins add an additive constant which is calculated later.

Sub-Class 1: The minimum weight of an item is $1/2 + \mu$ and the cost of a bin is 2.

Sub-Class 2: The minimum item weight is $1/2$ and the cost of a bin is $1 + \mu$.

Sub-Class 3: The weight of a pair of items is at least $2(\alpha + \mu/2) = 2\alpha + \mu$. The cost of the bin is 2.

Sub-Class 4: The weight of a pair of items is at least 2α, the cost of a bin is $1 + \mu$.

Before we proceed to the other two sub-classes, we discuss the way next-fit runs in those two cases. Consider a case where a new bin is opened, when an items does not fit into the previous active bin. By the above arguments, it means that the second component of an item can determine whether it fits. Since the second component is bounded by α, all closed bins are occupied by at least $1 - \alpha$ in that component. Bins of sub-phase 5 are also occupied by at least $(1 - \alpha)\mu$ in the first component.

Sub-Class 5: The weight of items in a closed bin is at least $(1 - \alpha)(1 + \mu)$, and

the cost of a bin is 2.
Sub-Class 6: The weight of items in a closed bin is at least $1-\alpha$, and the cost of a bin is $1+\mu$.

Let c be the maximum ratio of cost to weight in sub-classes 3, 4, 5 and 6. Specifically

$$c = \max(\frac{2}{2\alpha+\mu}, \frac{1+\mu}{2\alpha}, \frac{2}{(1-\alpha)(1+\mu)}, \frac{1+\mu}{1-\alpha}) .$$

Since $\alpha < 1/3$, we do not need to consider the fourth possibility. Hence

$$c = \max(\frac{2}{2\alpha+\mu}, \frac{1+\mu}{2\alpha}, \frac{2}{(1-\alpha)(1+\mu)}) .$$

Taking α to be a solution of $8x^3 - 8x^2 + 5x - 1 = 0$ and $\mu = (1-3\alpha)/\alpha$, all other values are the same (this gives $\alpha \approx 0.302$, $\mu \approx 0.315$, $c \approx 2.177$). Consider now also bins used for the symmetric case, i.e. the class of items (β', γ') where $\beta' > \gamma'$. There are at most 4 bins of cost $1+\mu$ and 4 bins of size 2 that might be open but not occupied by enough weight, and are ignored in previous calculations. We add those into the calculations to get the value of the additive constant.

Let N_L be the number of bins used for sub-class 1 (in both classes) and N_S the number of bins used for sub-class 2 (in both classes). Let W be the total weight of all items. Clearly, $C_{OPT} \geq W$. Also $C_{OPT} \geq N_L + N_S$. The last inequality is true since those items can be packed either alone (possibly together with items of other sub-classes) or in pairs, in bins of size $(1,1)$. Hence the cost of the optimal off-line algorithm for each such item is at least 1 (if all items of sub-classes 3, 4, 5 and 6 are ignored).

The weight of items packed into closed bins of sub-classes 3, 4, 5 and 6 is at most $W - 1/2N_S - (1/2+\mu)N_L$.

Hence

$$\begin{aligned} C_{ONL} &\leq c(W - 1/2N_S - (1/2+\mu)N_L) + (1+\mu)N_S + 2N_L + 12 + 4\mu \\ &\leq cC_{OPT} + N_S(1+\mu-c/2) + N_L(2-c/2-c\mu) + 12 + 4\mu \end{aligned}$$

For the above choices of α and μ,

$$1+\mu-c/2 = 2-c/2-c\mu \approx 0.226 .$$

Hence $C_{ONL} \leq 2.403C_{OPT} + 13.26$.

This proves the following Theorem:

Theorem 4.1. *The competitive ratio of the above algorithm is 2.403.*

5 Conclusions

We have seen that there is a large difference between possible competitive ratios for different sets, and that the competitive ratio can actually vary between 1 and $\Theta(d)$. The classical case ($B = \{(1,1\ldots,1)\}$) seems to be easier than the most difficult cases, but harder than the easiest cases. We conjecture that the competitive ratio for that problem should be non constant, but sub-linear.

References

[1] D. Blitz, A. Van Vliet, and G. J. Woeginger. Lower bounds on the asymptotic worst-case ratio of online bin packing algorithms. Unpublished manuscript, 1996.

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

[3] J. Csirik and G. Woeginger. On-Line Packing and Covering Problems. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms*, volume 1442 of *LNCS*, chapter 7, pages 147–177. Springer-Verlag, 1998.

[4] L. Epstein, S. S. Seiden, and R. van Stee. New bounds for variable-sized and resource augmented online bin packing. submitted, 2001.

[5] D. K. Friesen and M. A. Langston. A storage-size selection problem. *Inform. Process. Lett.*, 18:295–296, 1984.

[6] G. Galambos, H. Kellerer, and G. J. Woeginger. A lower bound for online vector packing algorithms. *Acta Cybernetica*, 11:23–34, 1994.

[7] 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.*, 21:257–298, 1976.

[8] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *J. of the ACM*, 34(1):144–162, 1987.

[9] N. G. Kinnersley and M. A. Langston. Online variable-sized bin packing. *Discr. Appl. Math.*, 22:143–148, 1988.

[10] L. T. Kou and G. Markowsky. Multidimensional bin packing algorithms. *IBM J. Research and Development*, 21:443–448, 1977.

[11] S. S. Seiden. An optimal online algorithm for bounded space variable-sized bin packing. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, pages 283–295, 2000.

[12] A. Van Vliet. An improved lower bound for online bin packing algorithms. *Inform. Process. Lett.*, 43:277–284, 1992.

[13] G. Zhang. Worst-case analysis of the FFH algorithm for online variable-sized bin packing. *Computing*, 56:165–172, 1996.

Received February, 2002