Computer Science Journal of Moldova, vol.**11**, no.2(32), 2003

Lower Bounds and Semi On-line Multiprocessor Scheduling

T.C. Edwin Cheng, Hans Kellerer, Vladimir Kotov

Abstract

We are given a set of identical machines and a sequence of jobs from which we know the sum of the job weights in advance. The jobs have to be assigned on-line to one of the machines and the objective is to minimize the makespan. An algorithm with performance ratio 1.6 and a lower bound of 1.5 is presented. This improves recent results by Azar and Regev who published an algorithm with performance ratio 1.625 for the less general problem that the optimal makespan is known in advance.

1 Introduction

The on-line version of the classical multiprocessor scheduling problem is one of the well-investigated problems of the last years. A set of independent jobs has to be processed on m parallel, identical machines in order to minimize the makespan. The jobs arrive on-line, i.e. each job must be immediately and irrevocably assigned to one of the machines without any knowledge on future jobs. This problem was first investigated by Graham who showed that the greedy algorithm has a performance ratio of exactly $2 - 1/m$ [4, 5]. A long list of improved algorithms has since been published. The best heuristic is due to Albers [1]. She designed an algorithm with performance ratio 1.923 and

lower bound 1.852. For a survey on recent results in bin packing problems we refer to [3].

We investigate a semi on-line version of this on-line multiprocessor scheduling problem where we assume that the total sum of processing times is given in advance. In a former paper [6] an algorithm with performance ratio 4/3 for the problem with known processing times and two machines was given. Moreover, this bound was best possible. A less general semi on-line version has been introduced by Azar and Regev in [2] named as the *on-line bin stretching problem*. A sequence of items is given which *can* be packed into m bins of unit size. The items have to be assigned on-line to the bins minimizing the *stretching factor* of the bins, i.e. to stretch the sizes of the bins as least as possible such that the items fit into the bins. Thus, the bin stretching problem can be interpreted as a semi on-line scheduling problem where instead of the total processing time even the value of the optimal makespan is known in advance. The motivation for investigating this problem comes from a file allocation problem as illustrated in [2]. In analogy to Azar and Regev we call our problem the *generalized on-line bin stretching problem* (GOBSP).

For the bin stretching problem a sophisticated and lengthy proof for an algorithm with stretching factor 1.625 was given in [2]. Moreover, the authors extended the lower bound of 4/3 on the stretching factor of any algorithm for two machines to any number of machines m.

In this paper we will present an elementary algorithm with performance ratio 1.6 for the more general problem (GOBSP). Moreover, we will show an improved lower bound of 1.5 to $m \geq 6$ machines.

2 Exact Problem Definition and Notation

In the (GOSBP) we are given a set M of m identical machines (bins) of unit size and a sequence I of jobs (items) which have to be assigned on-line to one of the machines. (For the rest of the paper we will use only the expressions bins and items.) Each item j has an associated weight $w_j > 0$ which is often identified with the corresponding item. The *weight* of a bin B is defined as the total sum of the weights of all items assigned to B and denoted by $w(B)$. More exactly, $w_j(B)$ denotes the weight of bin B just before item j is assigned, but most of the time we will just write $w(B)$ if it is clear from the context. When we will speak of *time* j, we mean the state of the system just before item j is assigned. The total sum of the item weights $w(I)$ shall be given in advance. W.l.o.g. $w(I) = m$.

The objective of an algorithm for (GOBSP) is to minimize the stretching factor of the bins, i.e. the maximal weight of the bins after assigning the items. For a given sequence of items I let α denote the stretching factor of an on-line algorithm for (GOBSP), and α^* denote the stretching factor of an optimal off-line algorithm, respectively. Of course, $\alpha^* \geq 1$. An algorithm is defined to have a *stretching ratio* ρ if for any sequence of items I with total weight m the ratio α/α^* is less than or equal to ρ.

If the bins $B_1, \ldots, B_m$ are enumerated in the order when the first time an item is assigned to a bin, we call bin B_i the *i-th opened bin*. Especially, the set containing the bins $B_1, \ldots, B_{\lceil \frac{m}{2} \rceil}$ is denoted by $B(1, m/2)$, and the set with the bins $B_{\lceil \frac{m}{2} \rceil + 1}, \ldots, B_m$ is denoted by $B(m/2, m)$, respectively.

The items are divided into several classes. Items with weight in]0; 0.6] are called *small*, items in]0.6; 0.8] are called *medium* and items greater than 0.8 are called *large*. A more detailed partition is given for the small and the large items. Altogether, we have the classes]0; 0.3],]0.3; 0.6],

$]0.6; 0.8]$, $]0.8; 0.9]$ and $]0.9; \infty[$. The corresponding items are called *tiny*, *little*, *medium*, *big*, and *very big*, respectively.

Also some bin classes are introduced. A bin B with no items in it is called *empty*. For $w(B) \in]0; 0.3]$ it is called *tiny*, for $w(B) \in]0.3; 0.6]$ it is called *little* and for $w(B) \in]0; 0.6]$ it is called *small*. If B consists only of a medium item, B is called *medium*. If $w(B) > 0.8$, it is called *large*. If B consists only of a big item, it is called *big*. A bin consisting only of a very big item is called *very big* If a bin contains a large item *and* small items but has weight not exceeding 1.1, it is called *nearly full*. Finally, bins which contain a large item and have weight greater than 1.1, are called *full*.

The number of tiny items is denoted by tI, the number of tiny bins is denoted by tB. The abbreviations for cardinalities of the other classes of bins are depicted in Table 1.

Types	empty	tiny	little	small	medium
Bins	$\emptyset B$	tB	ℓiB	sB	mB
Types	large	nf	big	very big	full
Bins	ℓaB	nfB	bB	vbB	fB

Table 1 Abbreviations for bin classes

3 Phase 1 of the Algorithm

Our algorithm with stretching ratio 1.6 is split into two parts. The first part (called Phase 1) runs (in three of four cases) until there are no more empty bins. At the end of Phase 1 it is decided, depending on the structure of the bins, how the algorithm continues with Phase 2. We will distinguish four different structures, leading to Stages 1, 2, 3 and 4.

During the algorithm we call LB the current lower bound for the stretching factor of an optimal off-line assignment, starting with $LB = 1$. In Phase 1 medium items are put alone into bins, big items are put alone into bins as long as more than $m/2$ bins are empty. When the number of empty bins does not exceed $m/2$, a big item (like all other large items) is put into the largest small bin in which it *fits*, i.e. in which the total weight will not exceed $1.6LB$, or otherwise into an empty bin. Finally, small items are assigned to small bins if the total weight will not exceed 0.6 or to empty bins. Depending on the four conditions in the end, it will be decided with which stage we continue. A formal description of Phase 1 of the algorithm is depicted in Figure 1.

Some simple properties of the bins after Phase 1 are described in the following lemma:

Lemma 1 *During any time of Phase 1 of the algorithm the following properties hold for any bin B:*

(a) If $w(B) \in]0.6; 0.8]$, then B is a medium bin. Moreover, all medium items are alone in bins.

(b) If $w(B) > 0.8$, then B contains a large item.

(c) $tB + nfB \leq 1$.

(d) $sB = 0$ or $vbB = 0$.

Proof: The proofs are straightforward. Assertions (a) and (b) are true by definition of the algorithm. Assertions (c) and (d) follow by induction. We show (c), assertion (d) is straightforward:

Assume first that there is one tiny bin B, thus no nearly full bin. Denote the next arriving item by a. If a is tiny, it will be assigned to

Phase 1 of the Algorithm

$LB := 1$ *initialization of the lower bound*
Let a denote the current item to be assigned and set $LB := \max\{LB, a\}$.

1. If $a > 0.9$ or if a is big and $\emptyset B \leq m/2$, assign a to the largest bin B with $w(B) \leq 0.6$ and $w(B) + a \leq 1.6LB$, otherwise to an empty bin.
2. Assign all medium items and for $\emptyset B > m/2$ also big items to empty bins.
3. If a is small and there is a tiny bin B_1 or a nearly full bin B_2, assign a to B_1 if $w(B_1) + a \leq 0.6$ (or to B_2 if $w(B_2) + a \leq 1.6LB$). Otherwise, assign small item a to the largest bin B with $w(B) > 0.9$ (for $\emptyset B \leq m/2$ even to the largest bin B with $w(B) > 0.8$) and $w(B) + a \leq 1.6LB$, or else to the largest bin B with $w(B) + a \leq 0.6$.

Stop, if one of the following four conditions holds:

(a) If $\emptyset B = 0$ and $sB = 0$, goto Stage 1.

(b) If $\emptyset B = 0$, $sB > 0$ and $bB = 0$, goto Stage 2.

(c) If $\emptyset B = 0$, $sB > 0$ and $bB > 0$, goto Stage 3.

(d) If $\emptyset B > 0$ and $2(mB + \emptyset B) \leq \ell iB$, goto Stage 4.

Figure 1 Algorithmic description of Phase 1

B since the total weight will not exceed 0.6. If a is little, it can be assigned to B or to an empty bin, forming a little bin, or to a large bin or full bin, forming a full bin. If a is large and is assigned to B, we get either a full bin or B is changed into a nearly full bin. So, in any case $tB + nfB \leq 1$ holds.

Now assume that there is one nearly full bin B, thus no tiny bin. If a fits into B, it is assigned to B and B remains a nearly full bin or becomes full. If a does not fit into B, we have $a > 0.5LB$, and the bin to which a will be assigned, will become either a little bin or a full bin.

■

The next lemma describes the structure of the bins at the end of Phase 1 depending on which Stage is entered in Phase 2.

Lemma 2 *The structure of the bins after Phase 1 at the beginning of Stages 1 to 4 can be described as follows:*

(a) Stage 1: There are full bins, very big bins, big bins, medium bins plus at most one nearly full bin.

(b) Stage 2: There are full bins, medium bins, little bins and at most one tiny bin or nearly full bin. Moreover we have

$$2mB \geq \ell iB - 2. \tag{1}$$

(c) Stage 3: There are full bins, big bins, medium bins, little bins and at most one tiny or nearly full bin. The bins of $B(m/2, m)$ are all medium bins and thus

$$mB \geq \left\lfloor \frac{m}{2} \right\rfloor. \tag{2}$$

(d) Stage 4: There are full bins, medium bins, little bins, at most one tiny bin and empty bins. Moreover we have

$$\ell iB \geq 2(mB + \emptyset B) \geq \ell iB - 2 \tag{3}$$

Proof: ad (a): The claim follows directly from Lemma 1(a), (b) and (c).

ad (b): The structure of the bins is again a consequence of Lemma 1. Since we do not enter Stage 4, for $\emptyset B > 0$ always $2(mB + \emptyset B) > \ell iB$ holds. Especially for $\emptyset B = 1$ we have $2(mB + 1) > \ell iB$. By assigning an item to the last empty bin, the left hand side of the inequality decreases by at most two and the right hand side can increase by at most one. Thus, we get inequality (1).

ad (c): The different types of bins which can exist are a direct consequence of Lemma 1. It remains to show that the bins in $B(m/2, m)$ are all medium bins. Remember that big items are put alone into bins as long as $\emptyset B > m/2$. After the first bin in $B(m/2, m)$ becomes nonempty, big items can be combined with small items and vice versa.

It is easy to show by induction for $\emptyset B \leq m/2$ a statement similar to Lemma 1(d) for the big and small items: If $\emptyset B \leq m/2$ and $bB = 0$ or $sB = 0$, then also $bB = 0$ or $sB = 0$ must hold for the rest of Phase 1.

It follows, that if $bB > 0$ and $sB > 0$ hold at the end of Phase 1, $bB > 0$ and $sB > 0$ must hold already after the last bin in $B(1, m/2)$ becomes nonempty. After this time large items can be assigned to small bins and small items to small bins or bins with load greater than 0.8, none of them to empty bins. Consequently, only medium items can be elements of $B(m/2, m)$.

ad (d): First note that we do not enter Stage 4 (or any other stage) before the number of empty bins is less than or equal to $m/2$. This means that there is some time when small items are allowed to be packed with big items and vice versa.

After this time small items are assigned to bins with a single large item in it or to a nearly full bin, before they enter an empty bin. On the other side, large items are assigned to small bins and assigning a medium item to an empty bin does not change the inequality $2(mB + \emptyset B) >$

ℓiB. Thus, it is only possible that the left hand side of inequality (3) becomes true if there are no bins with a single large item in it and no nearly full bins. The right hand side of inequality (3) follows with an argumentation analogous to (b). ∎

4 Phase 2 of the Algorithm

Phase 2 of the algorithm is split into four stages depending on the structure of the bins after Phase 1. For Stages 1 to 3 we apply a best fit approach. First, we try to put item a into the largest bin in which it fits, and if this is not possible, we assign it to the bin with smallest weight. The formal algorithm for Stages 1 to 3 is depicted in Figure 2.

Phase 2 for Stages 1 to 3 of the Algorithm

Let a denote the current item to be assigned and set $LB := \max\{LB, a\}$. If a is the $(m+1)$-st item not smaller than β, then set $LB := \max\{LB, 2\beta\}$.
Assign item a to the largest bin B, for which $w(B) + a \leq 1.6LB$, else assign a to the bin with smallest weight.

Figure 2 Algorithmic description of Phase 2 for Stages 1 to 3

If our algorithm for (GOBSP) has stretching ratio greater than 1.6, there is a *failure item* z_f which shall be the first item being assigned to a bin B ($w(B) < 1$) with $w(B) + z_f > 1.6\,\alpha^*$. Then the following lemma is easy to verify:

Lemma 3 *(a) If z_f is assigned to bin B, we have $z_f \leq \alpha^* < \frac{5}{3} w_{z_f}(B) < \frac{5}{3}$. While there are bins with weight not greater than 0.6, the stretching ratio does not exceed 1.6.*

(b) Let arriving item a be assigned to bin B with $w(B) \leq 0.9$ and $\alpha^* \geq w(B) + 0.6$. Then, $w(B) + a \leq 1.6\,\alpha^*$.

Proof: ad (a): The assertion follows directly from $w(B) + \alpha^* \geq w(B) + z_f > 1.6\,\alpha^*$.

ad (b): Asume the assertion does not hold, i.e. a is identical to the failure item z_f. Then we get from $w(B)+z_f > 1.6\,\alpha^* \geq 1.6\,(w(B)+0.6)$ that $z_f > 0.96 + 0.6\,w(B)$. Inserting $z_f < \frac{5}{3} w(B)$ from Lemma 3(a) into the preceding inequality we get $w(B) > 0.9$, a contradiction. ∎

Before we show that the algorithm works well in Phase 1, we introduce some further notation. Consider the bins at the end of Phase 1. Then the very big bins are denoted by $VB_1, \ldots, VB_r$ and the set $\{VB_1, \ldots, VB_r\}$ is called the *V-group*. Analogously, the big bins are denoted by $BB_1, \ldots, BB_s$, the medium bins by $MB_1, \ldots, MB_t$, and the small bins by $SB_1, \ldots, SB_u$, respectively. The corresponding sets are called *B-group*, *M-group* and *S-group*, respectively. Recall that the M-group consists of *all* medium items assigned to separate bins in Phase 1. The nearly full bin N shall be contained in the V-group or B-group, depending on whether $w(N)$ is greater than 0.9 or not.

All bin groups shall be sorted in non-increasing order of weight at the end of Phase 1, i.e.

$$w(VB_1) \geq \ldots \geq w(VB_r) \geq w(BB_1) \geq \ldots \geq w(BB_s) \geq w(MB_1) \geq \ldots$$

$$\ldots \geq w(MB_t).$$

W.lo.g. we may assume that these inequalities are strict (by changing the item weights slightly).

Proposition 1 *For Stage 1, the algorithm has stretching ratio 1.6.*

Proof: W.l.o.g. assume there is a failure item z_f. We distinguish several cases with respect to the item group to which z_f is assigned:

a) z_f is assigned to bin $MB_{t'}$ ($t' \leq t$) of the M-group: Let $m_{t'}$ denote the medium item from Phase 1 in $MB_{t'}$ and a the first item assigned to $MB_{t'}$ after $m_{t'}$.

If $a \neq z_f$, then we get $w_a(MB_{t'}) + a < 1$ and by $w(m_{t'}) > 0.6$ also $a < 0.4$. Thus, all bins besides $MB_{t'+1}, \ldots, MB_t$ have weight at least 1.2. Since z_f is not assigned to $MB_{t''}$ ($t'' > t'$), we have $w_{z_f}(MB_{t''}) \geq w_{z_f}(MB_{t'})$ for $t'' > t'$ and the last item assigned to $MB_{t''}$ before z_f must be greater than 0.6 (otherwise it would be assigned to $MB_{t'}$). Thus, also the bins $MB_{t'+1}, \ldots, MB_t$ have load at least 1.2 and the total weight of the items $w(I)$ would exceed m.

If $a = z_f$, due to $m_{t''} < m_{t'}$ for $t'' > t$ an item b must have been assigned to $MB_{t''}$ which did not fit into $MB_{t'}$, i.e. $b + w(MB_{t'}) > 1.6LB$. At that time $LB \geq 1.2$ holds. Therefore, $b > 1.6 \cdot 1.2 - m_{t'} \geq 1.92 - 0.8 = 1.12$. Thus, at time z_f there are $m + 1$ items not smaller than $m_{t'}$ and we conclude $\alpha^* \geq 2m_{t'} > w_{z_f}(MB_{t'}) + 0.6$. Lemma 3(b) with $w_{z_f}(MB_{t'}) \leq 0.8$ contradicts the assumption that z_f is assigned to a bin of the M-group.

b) z_f is assigned to bin $BB_{s'}$ ($s' \leq s$) of the B-group: Let $b_{s'}$ denote the big item from Phase 1 in $BB_{s'}$ and a the first item assigned to $BB_{s'}$ in Phase 1. If $a \neq z_f$, we can argue in a similar way to (a) for showing that the total weight of the items $w(I)$ would exceed m.

So assume $a = z_f$. Because of $w(BB_{s'}) > w(MB_i)$ for $i = 1, \ldots, t$ at the end of Phase 1, an item b must have been assigned to each MB_i which did not fit into $BB_{s'}$. From $LB \geq 1.2$, we get $b > 1.6 \cdot 1.2 - w(BB_{s'}) \geq 1.92 - 0.9 = 1.02$. Thus , item b is very big and z_f is at least the $(m+1)$-st large item. Thus at time z_f even $LB \geq 1.6$ holds and

we conclude $w_{z_f}(BB_{s'}) + z_f > 1.6 \cdot 1.6 = 2.56$. From Lemma 3(a) we get $z_f < \frac{5}{3} w_{z_f}(BB_{s'})$, which gives with the preceding inequality that $w_{z_f}(BB_{s'}) > 2.56/(1 + 5/3) = 0.96$, a contradiction to the fact that $w_{z_f}(BB_{s'}) \leq 0.9$.

c) z_f is assigned to bin $VB_{r'}$ ($r' \leq r$) of the V-group: Since at the end of Phase 1 the weight of each bin of the B-group and the M-group is smaller than $w(VB_{r'})$, at time z_f to each of these bins an item b has been assigned which did not fit into $VB_{r'}$. Similar to a) and b) we get $b > 1.6 \cdot 1.2 - w_{z_f}(VB_{r'}) \geq 1.92 - 1 = 0.92$, that means at time z_f each bin of the B-group, the M-group and the V-group contains a very big item.

If there are no full bins containing big items at the end of Phase 1, the B-group contains all big items which exist at the end of Phase 1. Therefore, z_f is at least the $(m + 1)$-st very big item. Hence, $\alpha^* > 0.9 + 0.9 = 1.8$, contradicting Lemma 3(a).

If there is a full bin $\tilde{B}$ with a big item b at the end of Phase 1, it contains also some small items. Let s denote the first small item assigned to bin $\tilde{B}$. Then at time s all bins with very big items but one are full (and one is at least nearly full). Otherwise s would have been assigned to one of these bins. (Note that very big bins with items greater than 1 would increase LB, so any small item fits into these bins.)

If b is the first item assigned to $\tilde{B}$, due to the definition of Phase 1, at time s all bins of $B(1, m/2)$ are non-empty. Consequently, all bins except one of $B(1, m/2)$ with very big items are full.

If s is the first item assigned to $\tilde{B}$, then no item greater than 0.6 is assigned to $\tilde{B}$ for $\emptyset B > m/2$. Hence, $\tilde{B}$ remains small while $\emptyset B > m/2$. Because of Lemma 1(a) also in this case all bins but one of $B(1, m/2)$ with very big items are full.

We conclude that all bins, besides bins with very big items of $B(m/2, m)$ (and at most one further bin) are full at the end of

Phase 2. At time z_f the total item weight is then greater than $1.1\lceil m/2\rceil + 0.9\lfloor m/2 - 0.3\rfloor$, an obvious contradiction to $w(I) = m$. ∎

Let G be a group of bins. If at least one item has been assigned to each bin of G in Phase 2 (Stages 1 to 3), group G is called *filled*. The following lemma is simple but very useful:

Lemma 4 *Let G be a filled group of bins each having weight greater than w at the end of Phase 1. Then all bins but one have weight greater than $(0.8 + w/2)$. Moreover, the average weight of the bins is greater than $(0.8 + w/2)$ if G contains at least two bins.*

Proof: It is sufficient to show that there is always at most one bin B of group G to which items have been assigned in Phase 2, but which has weight not exceeding $(0.8 + w/2)$. Let a be the new arriving item. Assume a does not fit into B but into a bin B' with smaller weight but no items added. Then, $w(B') + a > w + 1.6 - (0.8 - w/2) = 0.8 + w/2$. The claim follows. ∎

We show now that the algorithm works well for Stages 2 and 3.

Proposition 2 *For Stage 2, the algorithm has stretching ratio 1.6.*

Proof: Recall from Lemma 2(b) that we have at the beginning of Phase 2, full bins, an M-group of bins, an S-group, and at most one extra bin (nearly full or tiny). Recall also that as long as there are bins with weight smaller or equal to 0.6, especially the S-group is not filled, by Lemma 3(a) any item a of arbitrary weight can be assigned to a small bin B without getting $w(B) + a > 1.6\alpha^*$. We distinguish two cases with respect to the item group which is filled first:

a) The M-group is filled before the S-group: Then also the S-group is filled before item z_f is assigned. By Lemma 4 all bins but one of the S-group have weight greater than 0.95 (the last one having weight at least 0.8) and all bins but one of the M-group have weight greater than 1.1 (the last one having also weight at least 0.8). According to inequality (1) we have $m \geq t + u \geq \frac{3}{2}u - 1$ and hence $u \leq \frac{2}{3}(m+1)$. Thus the total weight of the items can be estimated as follows:

$$\begin{aligned} w(I) &> 1.1(t-1) + 0.95(u-1) + 0.8 \cdot 2 + 1.1(m-t-u) + z_f = \\ &= 1.1m - 0.15u - 0.45 + z_f \geq m - 0.55 + z_f, \end{aligned}$$

a contradiction to $w(I) = m$.

b) The S-group is filled before the M-group: Then the first item in Phase 2 assigned to each bin of the S-group is a big item, since items smaller than 0.8 are assigned to a bin of the M-group or to a bin of the S-group to which already a big item has been assigned in Phase 2. Thus, the failure item z_f must be assigned to a bin $MB_{t'}$ of the M-group. Like in Proposition 1 it is easy to see that z_f is the first item assigned to $MB_{t'}$ in Phase 2, otherwise the total weight of the items would exceed m. Also the first items assigned in Phase 2 to bins $MB_{t'+1}, \ldots, MB_t$ are big, otherwise these items would have been put into $MB_{t'}$. Thus, at time z_f there are $m+1$ items not smaller than $m_{t'}$ and we conclude $\alpha^* \geq 2m_{t'} > w_{z_f}(MB_{t'}) + 0.6$. Lemma 3(b) contradicts the assumption that z_f is assigned to a bin of the M-group.

■

Proposition 3 *For Stage 3, the algorithm has stretching ratio 1.6.*

Proof: The proof for Stage 3 is similar to the proof for Stage 2. We have at the beginning of Phase 2 full bins, an M-group of bins, an S-group, at most one extra bin (nearly full or tiny) and additionally

a B-group. Again we do a case distinction with respect to the group which is filled first.

a) The B-group is filled before the other groups: We can continue as in the proof for Proposition 2, noting that inequality (1) is replaced by inequality (2).

b) The M-group is filled before the other groups: Then the first items assigned in Phase 2 to bins of the M-group are all greater than 0.7 since otherwise they could be assigned to a bin of the B-group. Thus, each bin of the M-group has weight greater than 1.3. By Lemma 3(a) the S-group must be filled before the failure item can arrive and by Lemma 4 all bins but one of the S-group have weight greater than 0.95. Even the smallest bin of the S-group must have weight greater than 0.65. Adding the weights of the items after z_f yields with inequality (1) that

$$w(I) > \lfloor m/2 \rfloor \, 1.3 + (\lceil m/2 \rceil - 1) \; 0.8 + 0.65 + z_f > m,$$

a contradiction to $w(I) = m$.

c) The S-group is filled before the other groups: Then the first items assigned in Phase 2 to bins of the S-group are all greater than 0.8 since otherwise they could be assigned to a bin of the M-group. Thus, each bin of the S-group has weight greater than 1.1 (with the possible exception of one bin having weight at least 0.8). Analogously, to part b) of the proof for Proposition 2 we can exclude that the failure item is assigned to a bin of the M-group. Consequently, z_f must be assigned to a bin of the B-group and the M-group must be filled before the B-group. Then, the first items assigned in Phase 2 to bins of the M-group are all greater than 0.7 since otherwise they could be assigned to a bin of the B-group. Again, adding all the item weights gives a contradiction to $w(I) = m$. ∎

It remains to present the algorithm of Phase 2 for Stage 4. In this case instead of a best fit approach the bins are collected in batches of three

bins each depending on their type after the end of Phase 1. A set of three bins B_1, B_2, B_3 forms a *3-batch*, if is generated from two little bins and one empty bin or one medium. If only small items are assigned to a 3-batch, it is called *small 3-batch*, if only items greater 0.6 are assigned to a 3-batch, it is called *large 3-batch*. At the time, when a 3-batch is *opened*, the "counted" number of little bins, medium bins and empty bins, is reduced appropriately. At the first time when a small item does not fit into a small 3-batch or an item with weight greater than 0.6 does not fit into a large 3-batch, we *close* the corresponding 3-batch, i.e. no more items are assigned to it. Phase 2 for Stage 4 is depicted in detail in Figure 3.

Lemma 5 *Any closed 3-batch has total weight greater than 3.*

Proof: a) The assertion is trivially true for small 3-batches, since small items are not greater than 0.6.

b) Now consider a large 3-batch which consists of two little bins B_1, B_2 and one medium bin B_3. When opened, this batch has weight at least $2 \cdot 0.3 + 0.6 = 1.2$. In this case the algorithm assigns the items to the largest bins in which they fit. If an item can be assigned to B_3, then at least three items with weight greater than 0.6 can be assigned to the 3-batch. The total weight is then greater than $1.2 + 3 \cdot 0.6 = 3$. If no item can be assigned to B_3, we have $a_i > \max\{0.8,\ 1.6 - w(B_3)\}$ for the items a_i $(i = 1, 2)$ assigned to B_1 or B_2, respectively. Thus, the total weight of the 3-batch exceeds $(1.6 - w(B_3)) + w(B_3) + +0.8 + 0.6 \geq 3$.

c) Finally, consider a large 3-batch which consists of two little bins B_1, B_2 and one empty bin B_3. If the first item a_1 to be assigned is medium, we can continue like in b). Assume a_1 is large. Then, a_1 is assigned to B_1 or B_2. The following item a_2 (at least medium) is assigned to the empty bin B_3. If the next item a_3 can be assigned to B_3, at least four items with weight greater than 0.6 are assigned to

Phase 2 for Stage 4 of the Algorithm

Let a denote the current item to be assigned.

1. If a does not fit into the corresponding small 3-batch (large 3-batch), close the batch and open a new small 3-batch (large 3-batch) if possible. If this is not possible, goto 2.

   1.1 a is small: Assign a to the largest bin of the small 3-batch in which it fits. Goto 1.

   1.2 a is medium: Assign a to an empty bin of the large 3-batch, or otherwise to the largest bin in which it fits. Goto 1.

   1.3 a is large: If the large 3-batch contains an empty bin and one large item which has been already assigned to this batch, assign a to the empty bin. Otherwise, assign it to the largest bin in which it fits. Goto 1.

2. Use best fit, to assign the remaining items into the current open 3-batch and the remaining bins not in batches.

Figure 3 Algorithmic description of Phase 2 for Stage 4

the 3-batch, yielding a total weight greater than 3. If a_3 cannot be assigned to B_3, it is put into the remaining small bin, yielding total weight greater than $(1.6 - w(B_3)) + 0.8 + w(B_3) + 0.6 \geq 3$. ∎

Now we are ready to show that the algorithm works well for Stage 4.

Proposition 4 *For Stage 4, the algorithm has stretching ratio 1.6.*

Proof: Consider the bins to which items are assigned in Step 2 of Stage 4. By Lemma 2(d) there can be two little bins and (at most) one tiny bin which could not be assigned to 3-batches. Moreover, we have three bins (two of them at least little) from the current open 3-batch. Then it can be easily seen that at time z_f the total weight of the four former little bins is greater than $4 \cdot 0.95 = 3.8$ and the total weight of the two other bins exceeds 1.6. By Lemma 5 the bins in batches have average weight 1. Therefore, $z_f < 6 - (3.8 + 1.6) = 0.4$, contradicting that z_f is a failure item. ∎

We summarize our results in the following theorem:

Theorem 5 *The presented algorithm has stretching ratio 1.6. Moreover, the stretching ratio of any deterministic on-line algorithm for (GOBSP) is at least 1.5 for any number $m \geq 6$ of machines.*

Proof: We get the claimed stretching ratio as combination of Propositions 1, 2, 3 and 4. The lower bound can be obtained from an easy example:

Send m items of weight 0.75. If the algorithm puts two of them to the same bin, then send m items of weight 0.25. We would get $\alpha = 1.5$ and $\alpha^* = 1$. Thus, the algorithm must distribute the m items of weight 0.75 on different bins. The final item will have now weight 1.5. We get $\alpha = 2.25$ and $\alpha^* = 1.5$. ∎

5 Conclusions

In this paper we have presented a relatively simple algorithm with stretching ratio 1.6 for the (GOBSP). Note that the proof of the algorithm could be shortened substantially if we apply our algorithm to the on-line bin stretching problem by Azar and Regev. There are still some further interesting open problems: We believe that our algorithm for (GOBSP) can be further improved. Is it possible to adapt our algorithm so that we get a stretching factor of at most 1.5 for the on-line bin stretching problem? The two lower bounds for (GOBSP) and for the on-line bin stretching problem are very simple. An improvement of these bounds is not obvious. Specific algorithms for a small number of machines could be developed. For $m \leq 5$, there is still only the lower bound 4.3 for (GOBSP) known.

References

[1] S.Albers, *Better bounds for on-line scheduling problems.* In: Proc. 29th ACM Symp. on Theory of Computing, pp. 130–139, 1997.

[2] Y.Azar, O.Regev, *On-line bin-stretching,* Theoretical Computer Science, **268**, pp. 17–41, 2001.

[3] E.G.Coffman, J.Csirik, G.Woeginger, *Approximate solutions to bin packing problems,* In: P.M. Pardalos, M.G.C. Resende (ed.), *Handbook of Applied Optimization*, Oxford University Press, 2002.

[4] R.L.Graham, *Bounds for certain multiprocessor anomalies,* Bell System Technical Journal, **45**, pp. 1563–1581, 1966.

[5] R.L.Graham, *Bounds on multiprocessing timing anomalies,* SIAM J. Appl. Math., **17**, pp. 263–269, 1969.

[6] H. Kellerer, V. Kotov, M.G. Speranza and Z. Tuza, *Semi on–line algorithms for the partition problem,* Operations Research Letters, **21**, pp. 235–242, 1997.

T.C. Edwin Cheng, Hans Kellerer, Vladimir Kotov, Received Mai 6, 2003

T.C. Edwin Cheng,
Department of Management,
The Hong Kong Polytecnic University,
Kowloon, Hong Kong,
E–mail:*mscheng@inet.polyu.edu.hk*

Hans Kellerer,
Institut für Statistik und Operations Research,
Universität Graz,
Universitätsstraße 15, A–8010 Graz, Austria,
E–mail:*hans.kellerer@uni − graz.at*

Vladimir Kotov,
Belarusian State University,
Faculty of Applied Mathematics and Computer Science,
Belarusian State University,
Skarina ave. 4, Minsk, 220050, Belarus
E–mail:*kotov@fpm.bsu.unibel.by*