Semi-Online Multiprocessor Scheduling with Given Total Processing Time

T.C. Edwin Cheng* Hans Kellerer†‡ Vladimir Kotov§

Abstract

We are given a set of identical machines and a sequence of jobs, the sum of whose weights is known in advance. The jobs are 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. These results improve on the recent results by Azar and Regev, who proposed an algorithm with performance ratio 1.625 for the less general problem that the optimal makespan is known in advance.

Keywords: On-line algorithms; Semi-online algorithms; Bin stretching; Multiprocessor scheduling; Approximation algorithms

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 is 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 list scheduling algorithm has a performance ratio of exactly $2 - 1/m$ [6, 7]

*Department of Logistics, The Hong Kong Polytechnic University, Kowloon, Hong Kong, lgtcheng@polyu.edu.hk

†Institut für Statistik und Operations Research, Universität Graz, Universitätsstraße 15, A–8010 Graz, Austria, hans.kellerer@uni-graz.at

‡corresponding author

§Byelorussian State University, Faculty of Applied Mathematics and Computer Science, Byelorussian State University, Skarina ave. 4, Minsk, 220050, Byelorussia, kotov@fpm.bsu.unibel.by

and is best possible for $m \leq 3$ [3]. A long list of improved algorithms has since been published. The best heuristic is due to Fleischer and Wahl [4]. They designed an algorithm with competitive ratio smaller than 1.9201 when the number of machines tends to infinity. The best lower bound of 1.8358 is due to Gormley et al. [5]. For a survey on on-line algorithms for scheduling problems, we refer the reader to Sgall [9].

We investigate a semi on-line version of this on-line multiprocessor scheduling problem, where we assume that the sum of processing times is given in advance. In a previous paper [8], an algorithm with performance ratio 4/3 for the problem with known processing times and two machines was given. Moreover, this bound is best possible. A less general semi on-line version has been introduced by Azar and Regev in [1], who labeled it 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 are 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 [1]. In analogy to Azar and Regev we call our problem the *generalized on-line bin stretching problem* (GOBSP). Obviously, any online algorithm for (GOBSP) with competitive ratio α turns into an algorithm for the online bin stretching problem with stretching factor α (after possibly adding some dummy items).

For the bin stretching problem, a sophisticated proof for an algorithm with stretching factor 1.625 was given by Azar and Regev in [1]. 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 a recent paper Epstein [2] studied several online models of bin stretching on two machines. Especially, she shows a tight bound of 10/9 for two identical machines assuming the jobs sorted by non-increasing order of processing times.

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

2 Exact Problem Definition and Notation

In the (GOBSP) we are given a set M of m identical machines (bins) of unit size and a sequence I of jobs (items), which are 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 sum of the weights of all items assigned to B, and is 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 speak of *time* j, we mean the state of the system just before item j is assigned. The sum of the item weights $w(I)$ is 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.

Denote the set of the first $\lceil \frac{m}{2} \rceil$ bins to which items are assigned by $B(1, m/2)$ and the other bins by $B(m/2, m)$.

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; ∞[. 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	nearly full	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 will decide, depending on the structure of the bins, how the algorithm will continue 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 at the end, it will decide with which stage we will 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.

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 is very big or if a is big and $\emptyset B \leq m/2$, assign a to the largest bin B with $w(B) \leq 0.6$.
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 small 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 an empty bin.

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

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

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

(e) All bins have weight $\leq 1.6LB$.

(f) After Phase 1 the following types of bins are possible: empty bins, little bins, medium bins, big bins, very big bins, full bins and at most one tiny bin or nearly full bin.

Proof: The proofs are straightforward. Assertions (a) and (b) are true by definition of the algorithm. Note that assigning a very big item a to a small bin results always in a bin with weight not exceeding $1.6LB$ since for $a > 1$ the lower bound LB is redefined. Therefore, assertion (e) is true. Assertions (c) and (d) follow by induction. In the following we will show (c), assertion (d) can be proven analogously. Assertion (f) will then follow directly from (a), (b) and (c).

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 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. Thus a large item assigned to a small bin gives a full bin. Consider now a small item a. 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 or nearly full bin and empty bins. Moreover we have

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

Proof: We distinguish four cases according to the different stages. The possible types of bins in Stages 1 to 3 follow directly from Lemma 1(c), (d) and (f).

(a): The claim follows directly from above since by assumption $\emptyset B = 0$ and $sB = 0$ holds.

(b): Only inequality (1) has to be shown. Denote by mB' and $\ell iB'$ the number of medium bins and little bins, before assigning an item to the last empty bin. Analogously, denote by mB and ℓiB the number of medium bins and little bins, after assigning an item to the last empty bin. Since we do not enter Stage 4 while there are empty bins, the inequality

$$2(mB' + 1) \geq \ell iB' + 1$$

must hold for $\emptyset B = 1$. With the preceding inequality we get

$$2mB \geq 2(mB' - 1) = 2(mB' + 1) - 2 \geq \ell iB' - 1 \geq (\ell iB - 1) - 1.$$

This proves (1).

(c): Due to $|B(m/2, m)| = \lfloor m/2 \rfloor$ it is sufficient 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 last bin in $B(1, m/2)$ becomes nonempty, big items can be combined with small items and vice versa.

The following statement can be easily seen by induction on the number of empty bins: 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.

Since $bB > 0$ and $sB > 0$ hold at the end of Phase 1, it follows that $bB > 0$ and $sB > 0$ must hold already after the last bin in $B(1, m/2)$ becomes nonempty and also afterwards. 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, only medium items are assigned to empty bins. Consequently, only medium bins can be elements of $B(m/2, m)$.

(d): The left-hand side of inequality (3) follows directly from the definition of Stage 4 and the right-hand side of (3) is shown with an argumentation analogously to (b):

Consider the bins just before the last item a is assigned. Denote by mB', $\ell iB'$ and $\emptyset B'$ the number of medium bins, little bins and empty bins, just before item a is assigned. Analogously, denote by mB, ℓiB and $\emptyset B$ the number of medium bins, little bins and empty bins, after item a is assigned. Then, $mB \geq mB'$, $\emptyset B \geq \emptyset B' - 1$ and $\ell iB \leq \ell iB' + 1$.

Then, $2(mB' + \emptyset B') > \ell iB'$ but after assigning item a we get

$$\ell iB \leq \ell iB' + 1 \leq 2(mB' + \emptyset B') \leq 2(mB + \emptyset B + 1) = 2(mB + \emptyset B) + 2.$$

It remains to show that there are no bins with a single large item in it. 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 $m/2$ because $2(mB + \emptyset B) \leq \ell iB$ cannot hold for $\emptyset B > m/2$. This means that there is some time when small items are allowed to be packed with big items and vice versa.

From $\emptyset B > 0$ and the left-hand side of (3) we conclude $\ell iB \geq 2$, hence $\ell iB' \geq 1$.

If a is medium, the number $2(mB' + \emptyset B')$ remains constant, i.e. $2(mB + \emptyset B) = 2(mB' + \emptyset B')$. A large item a would be assigned to a small bin which exists due to $\ell iB' \geq 1$. Consequently, a is small and is assigned to a big or very big bin if there is one. But a can only be the last item if the number of little bins is increased or the number of empty bins is decreased. Thus, there can be no big bin or very big bin. ∎

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 an item into the largest bin in which it fits, and if this is not possible, we assign it to the bin with the smallest weight.

Before we continue the description of the algorithm, we introduce some further notation. Consider the bins at the end of Phase 1. Then the set of the very big bins is called the *V-group*. Analogously, the set of the big bins, medium bins and small bins are called *B-group*, *M-group* and *S-group*, respectively. Note that the M-group consists of all medium items assigned to separate bins in Phase 1.

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

$$w(B_1) \geq w(B_2) \geq \ldots \geq w(B_m), \tag{4}$$

and we will keep this notation for the bins even after some new items are assigned.

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 let β denote the current $(m+1)$-st largest item. Set $LB := \max\{LB, a, 2\beta\}$. Assign item a to the largest bin B, for which $w(B) + a \leq 1.6LB$, else assign a to the bin with the smallest weight. In case of ties, bins with smaller index, as defined in (4), are considered to have "larger" 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 that 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 the current item a be assigned to bin B with $w_a(B) \leq 0.9$ and $\alpha^* \geq w_a(B) + 0.6$. Then, $w_a(B) + a \leq 1.6\,\alpha^*$.

(c) Assume the algorithm fails and the failure item z_f is assigned to a bin B. Let a denote the first item assigned to B in Phase 2. If $w_a(B) \geq 0.4$ and if all bins which have weight less than 0.4 at the end of Phase 1, have weight at least 1 after Phase 2, then $z_f = a$ holds.

Proof: (a): The inequality follows directly from $w_{z_f}(B) + \alpha^* \geq w_{z_f}(B) + z_f > 1.6\,\alpha^*$. Now let B be a bin with $w(B) \leq 0.6$. If item a is assigned to B we get with $a \leq LB$ that $w(B) + a \leq 0.6 + LB \leq 1.6LB$.

(b): Assume the assertion does not hold, i.e., $a = z_f$. Then we get from $w_a(B) + a > 1.6\,\alpha^* \geq 1.6\,(w_a(B) + 0.6)$ that $a > 0.96 + 0.6\,w_a(B)$. Inserting $a = z_f < \frac{5}{3} w_a(B)$ from Lemma 3(a) into the preceding inequality, we get $w_a(B) > 0.9$, a contradiction.

(c): Note that a failure item z_f is always assigned to a bin B with smallest weight. Let $B = B_f$ as defined in (4). Bin B_f has weight smaller than 1 since $w(I) = m$, i.e.

$$w_{z_f}(B_f) < 1. \tag{5}$$

Assume $a \neq z_f$. We get from (5) that $w_a(B_f)+a < 1$. From the assumption that $w_a(B_f) \geq 0.4$ follows that $a < 0.6$. All bins $B_1, \ldots, B_{f-1}$ have weight at least $w_a(B_f)$ in the beginning of Phase 2. They are considered for item a before B_f and were not used. This means that they are full by more than 1. Now consider bins $B_{f+1}, \ldots, B_m$. Those bins had weight smaller than or equal to $w_a(B_f)$ in the beginning of Phase 2. Yet just before the assignment of z_f, they have larger weight than $w_a(B_f)$. This means they all received at least one item in the meantime. However B_f has weight smaller than 1 until z_f arrives. An item will be put in a bin with smaller weight if it does not fit in B_f. Thus it has weight larger than 0.6. Thus, the bins $B_{f+1}, \ldots, B_m$ also have weight at least 1 a contradiction to $w(I) = m$. ∎

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

Proof: Assume the algorithm fails. Recall from Lemma 2(a) that at the beginning of Stage 1 we have full bins, very big bins, big bins, medium bins plus at most one nearly full bin. Especially there are no small bins and due to Lemma 1(b) each bin contains an item with weight greater than 0.6.

Because the minimum weight of a bin at the beginning of Stage 1 is at least 0.6, by Lemma 3(c) a failure item z_f is according to (4) assigned to a bin B_f, $f \in \{1, \ldots, m\}$, and z_f is the first item assigned to B_f in Phase 2.

Since z_f is always assigned to a bin with smallest weight, the bins B_j, $j = f+1, \ldots, m$, all received at least one item b_j in the meantime. Items b_j did not fit in bin B_f and have weight larger than 0.6.

Recall that at the beginning of Phase 2 there are already m items with weight greater than 0.6. Thus, when the first of the items b_j arrives, there are at least $m+1$ items with weight greater than 0.6 and so $LB > 1.2$ holds at that time. Therefore, $b_j + w(B_f) > 1.6LB > 1.6 \cdot 1.2 = 1.92$. Because of $w_{z_f}(B_f) < 1$, we have $b_j > 0.92$ for $j = f+1, \ldots, m$. With the same argumentation also $z_f > 0.92$ holds. We distinguish now several cases with respect to the item group to which z_f is assigned.

a) B_f is a bin of the M-group: By definition all bins $B_1, \ldots, B_{f-1}$ contain an item with weight at least m_f where $m_f = w_{z_f}(B_f)$ denotes the medium item in B_f. Using that all full bins contain a large item, at time z_f there are $m+1$ items with weight at least m_f and we conclude $\alpha^* \geq 2m_f > w_{z_f}(B_f) + 0.6$. Lemma 3(b) with $w_{z_f}(B_f) = m_f \leq 0.8$ contradicts the assumption that z_f is assigned to a medium bin.

b) B_f is a bin of the B-group:

Now to each medium bin B_j a very big item $b_j > 0.92$ is assigned in Phase 2. Thus, including z_f, there are at least $(m+1)$ large items and at time z_f even $LB \geq 2 \cdot 0.8 = 1.6$ holds. We conclude $w_{z_f}(B_f) + z_f > 1.6 \cdot 1.6 = 2.56$. From Lemma 3(a), we get $z_f < \frac{5}{3} w_{z_f}(B_f)$, which gives with the preceding inequality $(1 + 5/3) w_{z_f}(B_f) > 2.56$, and so $w_{z_f}(B_f) > 0.96$, contradicting that B_f is a bin of the B-group.

c) B_f is a bin of the V-group: Since all items b_j have weight greater than 0.92, at time z_f all bins except those which are already full after the end of Phase 1 contain a very big item.

If all full bins contain very big items at the end of Phase 1, then there are including z_f at least $(m+1)$ very big items. Hence, $\alpha^* > 0.9 + 0.9 = 1.8$, contradicting Lemma 3(a).

Thus, assume there is a full bin $\tilde{B}$ with big item b and some small items at the end of Phase 1. Let s denote the first small item assigned to bin $\tilde{B}$. Then at time s all bins with very big items except one are full and at most one is at least nearly full with weight greater than 1 by Lemma 1(c). Otherwise s would have been assigned to one these bins. (Note that very big bins with items greater than 1 would increase LB, so any small item fits in these bins.)

If b is the first item assigned to $\tilde{B}$, due to the definition of Phase 1, s can only be assigned to bin $\tilde{B}$ if at time s all bins of $B(1, m/2)$ are nonempty. Thus, s can only be assigned to $\tilde{B}$ after all bins in $B(1, m/2)$ with very big items except one are full and one is at least nearly full with weight greater than 1. Since all bins of the B-group and the M-group are full at time z_f, in the end all bins of $B(1, m/2)$ except one are full and one is at least nearly full with weight greater than 1.

If s is the first item assigned to $\tilde{B}$, then no big item is assigned to $\tilde{B}$ for $\emptyset B > m/2$. Hence, $\tilde{B}$ remains small while $\emptyset B > m/2$ and due to Lemma 1(d) there is at most one bin with a very big item which is not full. Again in this case all bins of $B(1, m/2)$ except one are full and one is at least nearly full with weight greater than 1.

We summarize that all bins of $B(1, m/2)$ except one are full and one is at least nearly full with weight greater than 1 and all bins in $B(m/2, m)$ have weight at least 0.9 since they contain very big items or they are full. At time z_f the total item weight is then greater than $1.1(\lceil m/2 \rceil - 1) + 1 + 0.9 \lfloor m/2 \rfloor + z_f$, an obvious contradiction to $w(I) = m$. ∎

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

Lemma 4 *Let G be a filled set of bins each having weight greater than w at the end of Phase 1. Then all bins except 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: We show that the assertion holds during Phase 2 for all bins of G which are filled already. Let a be a new arriving item in Phase 2. Assume there is exactly one bin B of set G to which items have been assigned at time a, but which has weight not exceeding $(0.8+w/2)$. If a fits in B, there is still at most one filled bin of G with weight less than $(0.8 + w/2)$. Thus, assume a does not fit into B but is put into a bin $B' \in G$ with smaller weight but no items added. Hence, $a > 0.8-w/2$ and $w(B')+a > w+0.8-w/2 = 0.8+w/2$. The first claim follows. The second claim is straightforward. ∎

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

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

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

a) The M-group is filled: Then the S-group and a tiny extra bin are also filled before item z_f is assigned. By Lemma 4 all bins of the S-group except one have weight greater than 0.95 and all bins of the M-group except one have weight greater than 1.1. Consider the set of the two bins with smallest weight in the S-group and the M-group. By Lemma 4 with $w = 0.3$ the average weight of these two bins is at least 0.95. Since z_f did not fit in the extra bin B, we get $z_f+w_{z_f}(B) > 1.6$. Denote by t and u the number of bins in the M-group and in the S-group, respectively. 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:

$$\begin{aligned} w(I) &> 1.1(t-1) + 0.95(u-1) + 2 \cdot 0.95 + 1.1(m-t-u-1) + 1.6 \\ &= 1.1m - 0.15u + 0.35 \geq m + 0.25, \end{aligned}$$

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

b) The M-group is not filled: Then the first item in Phase 2 assigned to each bin of the S-group or the extra tiny bin is a large item, since items not greater than 0.8 are assigned preferably to a non-filled bin of the M-group. Thus, after the S-group is filled, all bins of this group are full. The remaining bins consist of bins with a medium item and at most one nearly full bin.

Denote the bin to which the failure item z_f is assigned by B_f according to the enumeration in (4). B_f is a bin of the M-group or a nearly full bin. Since z_f is always assigned to a bin with smallest weight and the M-group is not filled, z_f is assigned to a bin of the M-group. Hence, all conditions of Lemma 3(c) are fulfilled and z_f is the first item assigned to B_f in Phase 2.

The argumentation for the remainder of the proof is similar to the proof of part (a) of Proposition 1: The bins B_j, $j = f+1, \ldots, m$ all received at least one item b_j before z_f arrives. These items b_j did not fit in bin B_f and because of $w_{z_f}(B_f) \leq 0.8$, they have weight larger than 0.8.

Thus, at time z_f there are $m+1$ items not smaller than m_f, and we conclude $\alpha^* \geq 2m_f > w_{z_f}(B_f) + 0.6$. 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 assume there is a failure item z_f. At the beginning of Phase 2 there are full bins, an M-group of bins, an S-group, at most one extra bin B, nearly full or tiny, and additionally a B-group. Again by Lemma 3(a) the S-group and a tiny extra bin will be filled in Phase 2 of the algorithm.

a) The M-group and the B-group are filled: Then the S-group and a tiny extra bin are also filled before item z_f is assigned. We continue as in the proof for part (a) of Proposition 2, noting that inequality (1) is replaced by (2). By Lemma 4 all bins of the S-group except one have weight greater than 0.95, all bins of the M-group except one have weight greater than 1.1 and

all bins of the B-group except one have weight greater than 1.2. Consider the set of the three bins with smallest weight in the S-group, the M-group and the B-group. By Lemma 4 with $w = 0.3$ the total weight of these three bins is at least $3 \cdot 0.95 = 2.85$. Since z_f did not fit in the extra bin B, we get $z_f + w_{z_f}(B) > 1.6$. Denote by t, u, v the number of bins in the M-group, the S-group and the B-group, respectively. According to inequality (2), we have $t \geq \lfloor m/2 \rfloor$. Thus, the total weight of the items can be estimated as follows using $u \leq m - t \leq m/2 + 1$:

$$\begin{aligned} w(I) &> 1.2(v-1) + 1.1(t-1) + 0.95(u-1) + 1.1(m-t-u-v-1) \\ &\quad +2.85 + 1.6 = 1.1m + 0.1v - 0.15u + 0.1 \geq 1.025m - 0.05. \end{aligned}$$

We conclude $w(I) > m$ for $m \geq 2$, a contradiction to $w(I) = m$.

b) The M-group is filled, but not 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 should be assigned to a non-filled bin of the B-group which has weight at most 0.9. Thus, each bin of the M-group has weight greater than $0.6 + 0.7 = 1.3$. By Lemma 3(a) the S-group and the tiny bin must be filled before the failure item can arrive and by Lemma 4 all bins of the S-group except one have weight greater than 0.95. Failure item z_f did not fit in the smallest bin among the bins of the S-group and the extra bin B. Thus, the total weight of this bin and z_f is at least 1.6. With inequality (2) the total weight of the items can be estimated by

$$w(I) > \lfloor m/2 \rfloor \, 1.3 + (\lceil m/2 \rceil - 1) \, 0.8 + 1.6 > m,$$

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

c) The M-group is not filled: Then the first items assigned in Phase 2 to bins of the S-group are all greater than 0.8 since otherwise they should be assigned to an non-filled bin of the M-group which has weight at most 0.8. Thus, each bin of the S-group has weight greater than $0.3 + 0.8 = 1.1$. Since z_f is always assigned to a bin with smallest weight and the M-group is not filled, z_f is assigned to a bin of the M-group. The rest of the proof is completely identical to the end of part b) of the proof for Proposition 2.

■

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 it 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 current 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.

Phase 2 for Stage 4 of the Algorithm

Let a denote the current item to be assigned.

1. If $a \leq 0.6$ ($a > 0.6$) does not fit in the current open small 3-batch (large 3-batch) or there is none, 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 that 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 to bins of a possible open 3-batch and to the remaining bins not in batches.

Figure 3: Algorithmic description of Phase 2 for Stage 4

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 that 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. In any case, two items a_1, a_2 with weight greater than 0.6 are assigned to B_1 and B_2. If an item is assigned to B_3, the total weight is greater than $1.2 + 3 \cdot 0.6 = 3$. If no item is assigned to B_3, none of the items a_1 and a_2, assigned to B_1 and B_2, fits in B_3. Consequently, $a_i > \max\{0.8,\, 1.6 - w(B_3)\}$ for $i = 1, 2$. 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 that 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, a_1 is put in the empty bin and 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, there is also a fourth item with weight greater than 0.6 which can be assigned to 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 little 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 for Stage 4.

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

Proof: Assume there is a failure item z_f. Consider the bins to which items are assigned in Step 2 of Stage 4. By inequality (3) of Lemma 2(d) there can be two little bins and (at most) one tiny bin or nearly full bin that could not be assigned to 3-batches. After going to Step 2 at most one open 3-batch remains. It consists of three bins, two of which are at least little at the end of Phase 1. Altogether there are at most four bins which are little at the end of Phase 1 but not part of a 3-batch. These little bins are filled by Lemma 3(a). Hence, by Lemma 4 at time z_f the total weight of these bins is greater than $4 \cdot 0.95 = 3.8$, and the total weight of the possible 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 obtain the claimed stretching ratio by a 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 in 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 an elementary algorithm with stretching ratio 1.6 for the (GOBSP). Note that the proof of the algorithm can be shortened substantially if we apply our algorithm to the on-line bin stretching problem by Azar and Regev. There remain some further interesting open problems to address: 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.

Acknowledgment This research is partially supported by The Hong Kong Polytechnic University under grant number G-T397. The authors would like to thank an anonymous referee for his comments which improved the presentation of this paper a lot.

References

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

[2] L. Epstein, Bin stretching revisited, *Acta Informatica* **39** (2003) 97–117.

[3] U. Faigle, W. Kern, G. Turan, On the performance of on-line algorithms for partition problems, *Acta Cybernetica* **9** (1989) 107–119.

[4] R. Fleischer, M. Wahl, On-line scheduling revisited, *Journal of Scheduling* **3** (2000) 343–353.

[5] T. Gormley, N. Reingold, E. Torng, J. Westbrook, Generating adversaries for request-answer games, in: *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms* (SODA'00), 2000, pp. 564–565.

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

[7] R.L. Graham, Bounds on multiprocessing timing anomalies, *SIAM Journal of Applied Mathematics* **17** (1969) 263–269.

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

[9] J. Sgall, On-line scheduling – a survey, in: A. Fiat, G.J. Woeginger (Eds.), Online Algorithms: the State of the Art, Lecture Notes in Computer Science, Vol. 1442, Springer, Berlin, 1998, pp. 196–238.