On-line Bin-Stretching *

Yossi Azar[†] **Oded Regev**[‡]

Abstract

We are given a sequence of items that can be packed into m unit size bins. In the classical bin packing problem we fix the size of the bins and try to pack the items in the minimum number of such bins. In contrast, in the bin-stretching problem we fix the number of bins and try to pack the items while stretching the size of the bins as least as possible. We present two on-line algorithms for the bin-stretching problem that guarantee a stretching factor of $5/3$ for any number m of bins. We then combine the two algorithms and design an algorithm whose stretching factor is 1.625 for any m. The analysis for the performance of this algorithm is tight. The best lower bound for any algorithm is $4/3$ for any $m \geq 2$. We note that the bin-stretching problem is also equivalent to the classical scheduling (load balancing) problem in which the value of the makespan (maximum load) is known in advance.
Keywords. On-line algorithms, approximation algorithms, bin-stretching, load balancing, scheduling, bin-packing.

1 Introduction

The on-line bin-stretching problem is defined as follows. We are given a sequence of items that can be packed into m bins of unit size. We are asked to pack them in an on-line fashion minimizing the stretching factor of the bins. In other words, our goal is to stretch the sizes of the bins as least as possible to fit the sequence of items. Bin-stretching is somewhat related to the bin-packing problem [10, 13, 18]. In both cases all the items are to be packed in bins of a certain size. However, in bin-packing the goal is to minimize the number of bins while in bin-stretching the number of bins is fixed and the goal is to minimize the stretching factor of the bins. Hence, results for bin packing do not seems to imply results for the bin-stretching problem.

A bin-stretching algorithm is defined to have a stretching factor β if for every sequence

*A Preliminary version of this paper appears in the proceedings of RANDOM'98, 1998.
†Department of Computer Science, Tel Aviv University, Tel-Aviv, 69978, Israel. E-Mail: azar@math.tau.ac.il. Research supported in part by the Israel Science Foundation and by the US-Israel Binational Science Foundation (BSF).
‡Department of Computer Science, Tel Aviv University, Tel-Aviv, 69978, Israel. E-Mail: odedr@math.tau.ac.il

of items that can be assigned to m bins of a unit size, the the algorithm assigns the items to m bins of size of at most β.

The motivation for our problem comes from the following file allocation problem. Consider a case in which a set of files are stored on a system of m servers, each of some unit capacity. The files are sent one by one to a remote system of m servers in some order. The only information the remote system has on the files is that they were originally stored on m servers of unit capacity. Our goal is to design an algorithm that can assign the arriving sequence of files on the remote system with the minimum capacity required. An algorithm for our problem whose stretching factor is β can assign the sequence of jobs to servers of capacity β.

It is also natural to view the bin-stretching problem as scheduling (load balancing) problem. In the classical on-line scheduling (load balancing) problem there are m identical machines and n jobs arriving one by one. Each job has some weight and should be assigned to a machine upon its arrival. The makespan (load) of a machine is the sum of the weights of the jobs assigned to it. The objective of an assignment algorithm is to minimize the makespan (maximum load) over all machines. In the bin-stretching problem we have the additional information that the optimal load is some known value and the goal is to minimize the maximum load given this information.

It is clear that an upper bound for the classical scheduling (load balancing) problem is also an upper bound for the bin-stretching problem since we may ignore the knowledge of the optimal makespan (load). The classical scheduling problem was first introduced by Graham [14, 15] who showed that the greedy algorithm has a performance ratio of exactly $2 - \frac{1}{m}$ where m is the number of machines. Better algorithms and lower bounds are shown in [7, 8, 9, 11, 12, 19, 21]. Recently, Albers [1] designed an algorithm whose performance ratio is 1.923 and improved the lower bound to 1.852.

The only previous result on bin-stretching is for two machines (bins). Kellerer et al. [20] showed that the performance ratio is exactly 4/3 for two machines. For $m > 2$ there were no algorithms for bin-stretching that achieve a better performance than those for scheduling. In this paper we provide for the first time algorithms for bin-stretching on arbitrary number of machines (bins) that achieve better bounds than the scheduling/load-balancing results. Specifically, we show the following results:

- Two algorithms for the bin-stretching problem whose stretching factor is 5/3 for any number m of machines (bins).
- An improved algorithm which combines the above two algorithms whose stretching factor is 1.625 for any number m of machines (bins). Our analysis for the stretching factor of this algorithm is tight (for large m).
- For a fixed number $m \geq 3$ we get an upper bound $\frac{5m-1}{3m+1}$ which is better than 1.625 for $m \leq 20$.
- Also, we easily extend the lower bound of 4/3 on the stretching factor of any deterministic algorithm for $m = 2$ for any number $m \geq 2$.

Observe that the additional information that bin-stretching has over the scheduling problem really helps in improving the performance of the algorithms. Moreover, our upper bounds for the bin-stretching problem are lower than the lower bounds for the classical load balancing problem for all $m \geq 2$ and this fact separates the two problems.

Note that the notion of stretching factor has been already used for various problems and, in particular, for scheduling. A paradigm that is used for attacking many of the off-line and on-line problems is to design algorithms that know an upper bound on the value of the optimal algorithm. Binary search for the optimal value is used in the off-line setting. In fact, this is the way that scheduling is reduced to bin-stretching by the polynomial approximation scheme of [17]. This paradigm is also used for the related machines model [16] which corresponds to bins of different sizes. In the on-line case the paradigm of stretching factor is used with a doubling technique. Reducing the case of unknown optimal value to known optimal value results in loosing a factor of 4 [2]. The notion of stretching factor has also been used in the temporary jobs model where jobs arrive and depart at arbitrary times [3, 4, 5, 6].

2 Notation

Let M be a set of machines (bins) and J a sequence of jobs (items) that have to be assigned to the machines (bins). Each job j has an associated weight, $w_j \geq 0$. As job j arrives it must be permanently assigned to one of the machines. An assignment algorithm selects a machine i for each arriving job j. Whenever we speak about time j we mean the state of the system after the jth job is assigned. Let $l_i(j)$ denote the load on machine i at time j, i.e., the sum of the weights of all the jobs on machine i at time j. The cost of an assignment algorithm A on a sequence of n jobs J is defined as the maximum load over all machines, or, $C_A(J) = \max_{i \in M} l_i(n)$.

The objective of an on-line bin-stretching algorithm is to minimize the stretching factor β; i.e., the cost of a sequence of jobs given that the optimal off-line assignment algorithm (that knows the sequence of jobs in advance) assigns them at a unit cost. This is unlike the classical on-line scheduling (load balancing) problems where the optimal cost is not known in advance and the performance is measured by the regular competitive ratio which is defined as the supremum of the ratio between the cost of the on-line assignment and the cost of the optimal off-line assignment.

We say that a sequence of jobs can be assigned to m machines by an optimal off-line algorithm if it can be assigned with a unit cost. We note some simple properties of such sequences of jobs. First, the weight of all jobs must be at most 1 since a job that is larger than 1 cannot be assigned by any algorithm without creating a load larger than 1. Second, the sum of weights of all jobs in a sequence of jobs is at most m, the number of machines. That follows from the fact that the optimal off-line algorithm can assign jobs with total weight of at most 1 to each machine.

3 Two algorithms with 5/3 stretching factor

In this section we present two algorithms with a stretching factor of 5/3 for the on-line bin-stretching problem. These are actually two families of algorithms. For each family we prove the same 5/3 upper bound.

We start with a simple algorithm with a stretching factor of 2: put each arriving job on an arbitrary machine such that the resulting load on that machine will not exceed 2. Obviously, if the algorithm does not fail to find such machine it has a stretching factor of 2 by definition. In order to show that such a machine is always available we notice that there must be a machine whose load is at most 1. Otherwise, all the machines have loads larger than 1 which contradicts the fact that the optimal solution has maximal load 1. Since the weight of each job is at most 1, each arriving job can be assigned to some machine which implies that the algorithm never fails.

Our algorithms use a threshold α to classify machines according to their loads. An appropriate choice of α will lead as described later to an algorithm whose stretching factor is $1+\alpha$.

Definition 3.1 A machine is said to be short if its load is at most α. Otherwise, it is tall.

At the arrival time of job j, we define three disjoint sets of machines based on the current load and the job's weight.

Definition 3.2 When job j arrives, $1 \le j \le n$, define the following three disjoint sets:

- $S_1^\alpha(j) = \{i \in M \mid l_i(j-1) + w_j \le \alpha\}$
- $S_2^\alpha(j) = \{i \in M \mid l_i(j-1) \le \alpha,\ \alpha < l_i(j-1) + w_j \le 1+\alpha\}$
- $S_3^\alpha(j) = \{i \in M \mid l_i(j-1) > \alpha,\ l_i(j-1) + w_j \le 1+\alpha\}$

The set S_1 is of machines that are short and remain short if the current job is placed on them. The second set S_2 is of machines that are short but become tall if the job is placed on them. The last set S_3 is of machines that are tall but remain below $1+\alpha$ if the job is placed on them. Note that there may be machines which are not in any of the sets. We omit the indices j and α when they are clear from the context.

Using this definition we can now describe the two algorithms:

$ALG1_\alpha$**:** When job j arrives:

- Put the job on any machine from the set S_3 or S_1 but not on an empty machine from S_1 if there is a non-empty machine from S_1.
- If $S_1 = S_3 = \phi$ then put the job on the least loaded machine from the set S_2.
- If $S_1 = S_2 = S_3 = \phi$ then report failure.

$ALG2_\alpha$**:** When job j arrives:

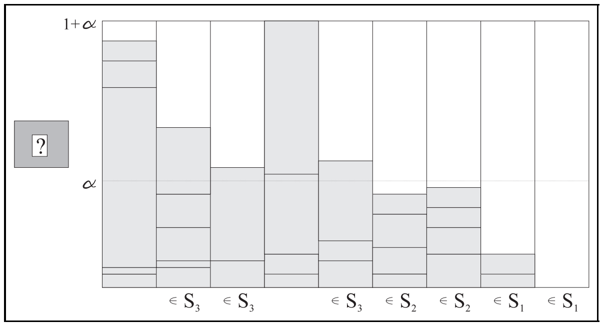

Figure 1: S_1, S_2 and S_3

- Put the job on any machine from the set S_1.
- If $S_1 = \phi$ then put the job on any machine from the set S_3.
- If $S_1 = S_3 = \phi$ then put the job on the least loaded machine from the set S_2.
- If $S_1 = S_2 = S_3 = \phi$ then report failure.

Notice that these two algorithms are actually families of algorithms. In the first algorithm we are free to choose how to select a machine from S_3 and whether we put a job on a machine from S_1 or from S_3. In the second algorithm we are free to choose how to select a machine from S_1 and from S_3.

Note that since the algorithms assign job j only to machines from the sets $S_1(j)$, $S_2(j)$ and $S_3(j)$, their stretching factor is at most $1+\alpha$ as long as they do not fail. For $1 \leq i \leq 3$ let J_i be the set of jobs j assigned to a machine in $S_i(j)$ at their arrival time by the algorithm.

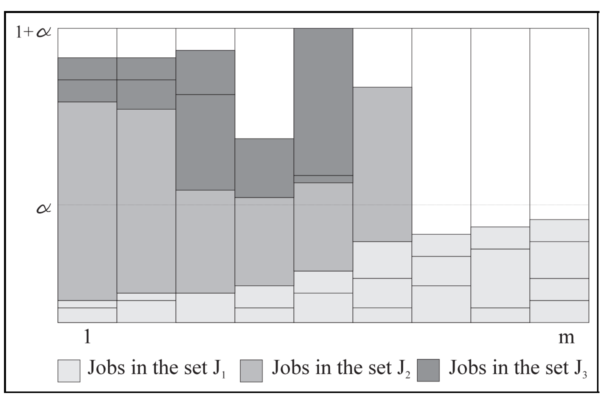

Figure 2: J_1, J_2 and J_3

Theorem 3.3 $ALG1_\alpha$ above never fails for $\alpha \geq 2/3$. Therefore, for $\alpha = 2/3$ it has a stretching factor of $5/3$.

Theorem 3.4 $ALG2_\alpha$ above never fails for $\alpha \geq 2/3$. Therefore, for $\alpha = 2/3$ it has a stretching factor of $5/3$.

In order to prove the above theorems we assume by contradiction that $ALG1_\alpha$ or $ALG2_\alpha$ fail on the last job of some sequence of $n+1$ jobs and that this sequence can be assigned by an optimal algorithm. We start with the following simple lemmas:

Lemma 3.5 At time n all the machines are tall and there are at least two machines whose load is less than 1.

Proof: At time n, when the last job arrives, the three sets, S_1, S_2 and S_3 are empty. Hence, $l_i(n) + w_{n+1} > 1 + \alpha$ for all $1 \leq i \leq m$. Since the weight of each job is at most 1, $l_i(n) > 1 + \alpha - w_{n+1} \geq \alpha$ for all $1 \leq i \leq m$. Thus, all the machines are tall. Assume by contradiction that except a machine i, all the machines have loads of 1 or more. When the last job comes, $l_i(n) + w_{n+1} > 1 + \alpha > 1$ and since all other machines also have loads of 1 or more it implies that the sum of all loads is above m which contradicts the fact that the sequence of jobs can be assigned by an optimal algorithm. ■

Corollary 3.6 The last job is larger than α.

Proof: At time n, when the last job arrives, there is a machine i whose load is less than 1 by lemma 3.5. Since the algorithm fails to assign the last job, $1 + w_{n+1} > 1 + \alpha$ or $w_{n+1} > \alpha$. ■

To utilize some of our lemmas for the improved algorithm we use a more general formulation. Consider a subset $M' \subseteq M$ of machines. We define the notion of composed algorithm $D(ALG, M')$ where ALG is $ALG1_\alpha$ or $ALG2_\alpha$ on a sequence of jobs I and a set of machines M as follows: The algorithm decides on an arbitrary set $I' \subseteq I$ and assigns it to a machine in M' and it assigns the rest of the jobs to a machine in $M - M'$. The assignment of jobs I' is done by running algorithm ALG on the set of machines M'. However, the jobs in $I - I'$ are assigned to a machine in $M - M'$ in any arbitrary way. Moreover, we make no assumption on the sequence I, for example, the optimal algorithm may not be able to assign them in M without exceeding a load of 1 (in particular, jobs of weight larger than 1 may exist).

Note that $D(ALG, M')$ is the same as ALG for $M' = M$. We already proved that if $ALG1_\alpha$ or $ALG2_\alpha$ fail on the $n+1$ job of sequence J of jobs then at time n all the machines are tall and there are two machines whose load is less than 1. Meanwhile, for the composed algorithms we assume that after a sequence of n jobs I was assigned by $D(ALG, M')$ all the machines from the set M' are tall and two of them have loads below 1. This assumption is used until (including) lemma 3.13. Also, we assume that $0 \leq \alpha \leq 1$ unless otherwise specified.

Define the *raising job* k_i of machine $i \in M'$ as the job that raises machine i from being short to being tall. More formally, $l_i(k_i) > \alpha$ and $l_i(k_i - 1) \leq \alpha$. The raising jobs are well defined since we assumed that all machines from M' are tall. Rename the indices of the machines in M' to $1, \ldots, m'$ such that $k_1 < k_2 < \ldots < k_{m'}$ i.e., the order of the machines in M' is according to the time the machines crossed α. From now on, all the indices are according the the new order. Note that the set of the raising job is J_2. Denote by s_1, s_2 the two machines in M' ($s_1 < s_2$) whose load is less than 1 at time n.

Lemma 3.7 If at time n, the load of some machine $i \in M'$ is at most l then $w_{k_{i'}} > 1 + \alpha - l$

for $i' > i, i' \in M'$.

Proof: Both $ALG1_\alpha$ and $ALG2_\alpha$ assign jobs to machines from S_2 only if the two other sets are empty. By definition of $k_{i'}$, at time $k_{i'} - 1$, job $k_{i'}$ arrived and was assigned to machine i'. By the definitions of S_2 and $k_{i'}$, machine i' was in the set $S_2(k_{i'})$ and therefore the sets $S_1(k_{i'})$ and $S_3(k_{i'})$ were empty. Machine i was already tall at that time since $i < i'$. This implies that at time $k_{i'} - 1$ machine i was not in $S_2(k_{i'})$. Hence $l_i(k_{i'} - 1) + w_i > 1 + \alpha$ or $w_i > 1 + \alpha - l_i(k_{i'} - 1) \geq 1 + \alpha - l_i(n) \geq 1 + \alpha - l$. ■

Since we assumed the load of machine s_1 is at most 1 at time n, the lemma above implies:

Corollary 3.8 Jobs k_i for $s_1 < i \leq m'$ are larger than α.

Let $f_i = l_i(k_i - 1)$ for $1 \leq i \leq m'$. This is the load of each machine just before it was raised by the raising job.

Figure 3: The series f_i. Only machines from M' are shown for clarity.

Lemma 3.9 For $i' > i$, both in M', $f_i \leq l_{i'}(k_i - 1) \leq f_{i'}$.

Proof: At time $k_i - 1$ the load of machine i is f_i by definition. At this time, by definition of k_i, machine i is in the set S_2 which means that S_1 and S_3 are empty. Thus, at the same time, each machine $i' > i$ must be in S_2 or not in any of the sets. Note that if the load of machine i' is below f_i at time $k_i - 1$ then it is in $S_2(k_i)$ since machine i, whose load is higher, is in $S_2(k_i)$. Therefore, the load of machine i' is at least f_i since both algorithms choose the least loaded machine from S_2. Machine i' is still short so its load is at most $f_{i'}$. ■

Corollary 3.10 The series f_i , $1 \leq i \leq m'$, is non-decreasing.

Lemma 3.11 For $i \leq s_2$, $f_i < 1 - \alpha$.

Proof: According to corollary 3.8, $w_{k_{s_2}} > \alpha$. Since the load of machine s_2 is below 1 at time n,

$$1 > l_{s_2}(n) \geq l_{s_2}(k_{s_2}) = l_{s_2}(k_{s_2} - 1) + w_{k_{s_2}} = f_{s_2} + w_{k_{s_2}}.$$

Therefore,

$$f_{s_2} < 1 - w_{k_{s_2}} < 1 - \alpha.$$

By corollary 3.10, $f_i < 1 - \alpha$ for $i \leq s_2$. ■

Note that up to now our proof was not specific to one of the algorithms. Now we focus our attention on the first algorithm. Recall that we still assume that the set of jobs I is assigned by algorithm $D(ALG1_\alpha, M')$ or $D(ALG2_\alpha, M')$ to the set of machines M.

Lemma 3.12 At any time of the activity of $D(ALG1_\alpha, M')$, there is at most one non empty machine in M' whose load is at most $\frac{\alpha}{2}$.

Proof: Assume by contradiction that at a certain time there are two such machines. Let j be the first job that its assignment created two such machines. Thus, job j arrived and was placed on an empty machine i_2 while another non empty machine i_1 had a load of at most $\frac{\alpha}{2}$. Clearly $w_j \leq \frac{\alpha}{2}$ and $l_{i_1}(j-1) + w_j \leq \alpha$. Therefore $i_1 \in S_1(j)$ and job j should have been assigned to i_1. ■

Lemma 3.13 Assume $\alpha \geq 2/3$ and $D(ALG1_\alpha, M')$ assigns a set of n jobs I to a set of machines M. Then the weight of each job k_i, $1 \leq i \leq m'$, is more than α.

Proof: We have already seen in corollary 3.8 that jobs k_i for $s_1 < i \leq m'$ are larger than α. Now we show that jobs k_i for $i \leq s_1$ are also larger than α.

By lemma 3.11, f_{s_1} and f_{s_2} are both below $1 - \alpha$. According to lemma 3.9, $f_{s_1} \leq l_{s_2}(k_{s_1} - 1) \leq f_{s_2}$. Recall that the load of machine s_1 at time $k_{s_1} - 1$ is f_{s_1}. At that time, the loads of machines s_1 and s_2 are below $1 - \alpha \leq \frac{\alpha}{2}$. Thus, by lemma 3.12, the less loaded machine, s_1, is empty, or $l_{s_1}(k_{s_1} - 1) = f_{s_1} = 0$. By corollary 3.10, $f_i = 0$ for all machines $i \leq s_1$. A small f_i implies that machine i has a large raising job. More formally, for $i \leq s_1$:

$$w_{k_i} = l_i(k_i) - l_i(k_i - 1) = l_i(k_i) - 0 > \alpha.$$

■

Now we are ready to complete the proof of theorem 3.3. Assume that $ALG1_\alpha$ fails on the $n+1$ job of a sequence J of jobs. After the n jobs have been assigned, all the machines are tall and there are two machines whose load is less than 1 by lemma 3.5. We take $M' = M$ and therefore $I' = I$ where I is the set of jobs J without the last job. The previously defined series k_i is now defined over all machines since we took $M' = M$. By lemma 3.13, for $\alpha \geq 2/3$, this implies that there are m jobs larger than α. Corollary 3.6 shows that the last job is also larger than α. We showed there are $m+1$ jobs larger than α. This contradicts the fact that the number of jobs of weight larger than $1/2$ is at most m since the optimal algorithm can assign at most one such job to each machine. This completes the proof of theorem 3.3.

The proof of theorem 3.4, i.e. $ALG2_\alpha$ has the same stretching factor, is in subsection 7.1 of the Appendix.

4 Improved Algorithm

In this section we present an improved algorithm whose stretching factor is 1.625. The improved algorithm combines both of the previous algorithms into a single algorithm.

At the arrival time of job j we define five disjoint sets of machines based on the current load and the job's weight.

Definition 4.1 When job j arrives, $1 \le j \le n$, define the following five sets:

- $S_{11}^{\alpha}(j) = \{i \in M \mid l_i(j-1) + w_j \le \alpha,\ l_i(j-1) + w_j \le 2\alpha - 1\}$
- $S_{12}^{\alpha}(j) = \{i \in M \mid l_i(j-1) + w_j \le \alpha,\ l_i(j-1) \le 2\alpha - 1,\ l_i(j-1) + w_j > 2\alpha - 1\}$
- $S_{13}^{\alpha}(j) = \{i \in M \mid l_i(j-1) + w_j \le \alpha,\ l_i(j-1) > 2\alpha - 1\}$
- $S_{2}^{\alpha}(j) = \{i \in M \mid l_i(j-1) \le \alpha,\ \alpha < l_i(j-1) + w_j \le 1 + \alpha\}$
- $S_{3}^{\alpha}(j) = \{i \in M \mid l_i(j-1) > \alpha,\ l_i(j-1) + w_j \le 1 + \alpha\}$

Note that the previously defined S_1 is split into three sets according to a low threshold of $2\alpha - 1$. We still use the notation S_1 for the union of these three sets. We omit the indices j and α when they are clear from the context. The sets J_1, J_2 and J_3 are defined as in the previous section.

Improved Algorithm: When job j arrives:

- Put the job on a machine from the set S_1 according to:
  - Put the job on any machine from the set S_{13} or S_{11} but not on an empty machine from the set S_{11} if there is a non-empty machine from the set S_{11}.
  - If $S_{11} = S_{13} = \phi$ then put the job on the least loaded machine from the set S_{12}.
- If $S_1 = \phi$ then put the job on the *earliest* machine from the set S_3, that is, the machine that was the first to cross the threshold α from all machines in S_3.
- If $S_1 = S_3 = \phi$ then put the job on the least loaded machine from the set S_2.
- If $S_1 = S_2 = S_3 = \phi$ then report failure.

This improved algorithm is contained in the family of $ALG2_{\alpha}$ presented in the last section. Our algorithm, however, defines the methods used in placing jobs on machines from the sets S_1 and S_3. The way we choose a machine from S_1 is by the method presented in $ALG1_{\alpha}$. In choosing a machine from S_3 we prefer the earliest machine according to the order of crossing the threshold. The proof of the theorem below appears in subsection 7.2 of the Appendix.

Theorem 4.2 The improved algorithm above never fails for $5/8 \le \alpha \le 2/3$. Thus, for $\alpha = 5/8$ it has a stretching factor of $13/8$.

5 Fixed number of machines

In this section we present an improvement to $ALG1_\alpha$ when m is fixed. For $m \geq 5$ we show that α can be slightly reduced without causing the algorithm to fail. In order to improve the performance also for $m = 3, 4$ we use an algorithm called $ALG12_\alpha$ which is the intersection of $ALG1_\alpha$ and $ALG2_\alpha$. For $m = 2$ we use a simple algorithm that has a $4/3$ stretching factor.

The proof of the theorem below appears in subsection 7.3 of the Appendix.

Theorem 5.1 For $m \geq 5$, $ALG1_\alpha$ never fails for $\frac{2m-2}{3m+1} \leq \alpha \leq 2/3$. Therefore, for $m \geq 5$, its stretching factor is $\frac{5m-1}{3m+1}$.

We overcome the $m \geq 5$ limitation by introducing the following algorithm.

$ALG12_\alpha$**:** When job j arrives:

- Put the job on any machine from the set S_1 but not on an empty machine from S_1 if there is a non-empty machine from S_1.
- If $S_1 = \phi$ then put the job on any machine from the set S_3.
- If $S_1 = S_3 = \phi$ then put the job on the least loaded machine from the set S_2.
- If $S_1 = S_2 = S_3 = \phi$ then report failure.

This algorithm is actually a family of algorithms since we have some freedom in choosing a machine. Notice that this family is the intersection of the two families of algorithms, $ALG1_\alpha$ and $ALG2_\alpha$. Our proof which appears in subsection 7.4 combines both of the methods used in the proofs of these two algorithms.

Theorem 5.2 The algorithm for small m above never fails for $\min(\frac{2}{3}, \frac{m-1}{m+1}) \geq \alpha \geq \frac{2m-2}{3m+1}$ for $m \geq 3$. Therefore, for $m \geq 3$, its stretching factor is $\frac{5m-1}{3m+1}$.

Next, we prove that for two machines the following simple algorithm has a stretching factor of $4/3$: Put each job on machine 1 if the resulting load is at most $\frac{4}{3}$ and, otherwise, put the job on machine 2.

Theorem 5.3 The simple algorithm for $m = 2$ has a stretching factor of $4/3$.

Proof: Consider the first job j from a set of n jobs that cannot be assigned to the first machine. If at time $j - 1$ the load of the first machine is above $2/3$ then all jobs $j, \ldots, n$ can be assigned to the second machine since the sum of the weights of all jobs is at most 2. Otherwise, job j is larger than $2/3$. Thus, the weight of all jobs except j sum up to at most $4/3$ and can be assigned to the first machine. ∎

6 Lower Bounds

In this section we prove a general lower bound of $4/3$ on the stretching factor of deterministic algorithms for any number of machines. We show a lower bound of $5/3 - \epsilon$ for arbitrary small ϵ for the family of $ALG1_\alpha$ and a lower bound of $13/8 - \epsilon$ for arbitrary small ϵ on the stretching factor of our improved algorithm. Note that it is impossible to show a lower bound of $5/3 - \epsilon$ for $ALG2_\alpha$ since the improved algorithm is in that family. In these two cases we assume the number of machines is large enough. The details of all the lower bounds are in the Appendix.

References

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

[2] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proc. 25th ACM Symposium on the Theory of Computing*, pages 623–631, 1993. Also in *Journal of the ACM* 44:3 (1997) pp. 486–504.

[3] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Competitive routing of virtual circuits with unknown duration. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 321–327, 1994.

[4] Y. Azar, A. Broder, and A. Karlin. On-line load balancing. In *Proc. 33rd IEEE Symposium on Foundations of Computer Science*, pages 218–225, 1992. Also in *Theoretical Compute Science* 130 (1994) pp. 73-84.

[5] Y. Azar and L. Epstein. On-line load balancing of temporary tasks on identical machines. In *5th Israeli Symp. on Theory of Computing and Systems*, pages 119–125, 1997.

[6] Y. Azar, B. Kalyanasundaram, S. Plotkin, K. Pruhs, and O. Waarts. On-line load balancing of temporary tasks. In *Proc. Workshop on Algorithms and Data Structures*, pages 119–130, August 1993.

[7] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. In *Proc. 24th ACM Symposium on Theory of Algorithms*, pages 51–58, 1992. To appear in *Journal of Computer and System Sciences.*

[8] B. Chen, A. van Vliet, and G. Woeginger. A lower bound for randomized on-line scheduling algorithms. *Information Processing Letters*, 51:219–222, 1994.

[9] B. Chen, A. van Vliet, and G. J. Woeginger. New lower and upper bounds for on-line scheduling. *Operations Research Letters*, 16:221–230, 1994.

[10] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: a survey. In D. Hochbaum, editor, *Approximation algorithms*. 1996.

[11] U. Faigle, W. Kern, and G. Turan. On the performance of online algorithms for partition problems. *Acta Cybernetica*, 9:107–119, 1989.

[12] G. Galambos and G. J. Woeginger. An on-line scheduling heuristic with better worst case ratio than graham's list scheduling. *SIAM J. Computing*, 22:349–355, 1993.

[13] M.R. Garey and D.S. Johnson. *Computers and Intractability*. W.H. Freeman and Company, San Francisco, 1979.

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

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

[16] D. Hochbaum and D. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

[17] 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, January 1987.

[18] D. S. Johnson. *Near-optimal bin packing algorithms*. PhD thesis, MIT, Cambridge, MA, 1973.

[19] D. R. Karger, S. J. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. In *Proc. of the 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 132–140, 1994.

[20] H. Kellerer, V. Kotov, M. G. Speranza, and Zs. Tuza. Semi on-line algorithms for the partition problem. *Operations Research Letters*. To appear.

[21] J. Sgall. On-line scheduling on parallel machines. Technical Report Technical Report CMU-CS-94-144, Carnegie-Mellon University, Pittsburgh, PA, USA, 1994.

7 Appendix

7.1 Upper bound for $ALG2_\alpha$

We prove theorem 3.4. We show that there are many large jobs here as in the proof of theorem 3.3. We first assume that $\alpha \geq 1/2$.

Definition 7.1 Let s_0 be the smallest non-negative integer such that $w_{k_i} > \alpha$ for all $i > s_0$.

Figure 4: Machine s_0

Note that s_0 is always defined and $0 \leq s_0 \leq m$. As a matter of fact, $1 \leq s_0$. Otherwise, the m jobs k_i are larger than α and by corollary 3.6 the last job is also larger than α. This is a contradiction since we assumed $\alpha \geq 1/2$ and as we saw before, there are at most m jobs larger than $1/2$.

Lemma 7.2 $f_i < 1 - \alpha$ for all $1 \leq i \leq s_0$.

Proof: By corollary 3.8, all jobs k_i for $i > s_1$ are larger than α and therefore, from the definition of s_0, $s_0 \leq s_1$. By lemma 3.11, $f_i \leq 1 - \alpha$ for $i \leq s_0$. ■

Lemma 7.3 For all $1 \leq i \leq m$, the weight of every job from the sets J_2 and J_3 that arrived before k_i is larger than $\alpha - f_i$. Job k_i itself is also larger than $\alpha - f_i$.

Proof: Consider a job j that arrived before k_i and is in one of the sets J_2 and J_3. By the description of $ALG2_\alpha$, the fact that the job was assigned to a machine from the sets S_2 or S_3 means that the set S_1 was empty when job j arrived. At that time, the load of machine i is at most f_i since k_i has not arrived yet and therefore $w_j > \alpha - f_i$. Job k_i is the raising job and therefore must also be larger than the difference between f_i and the threshold α. ■

Lemma 7.4 Each machine $i < s_0$ either contains a job that is larger than $1 - f_{s_0}$ or two jobs each is larger than $\alpha - f_{s_0}$. Machine s_0 itself contains a job that is larger than $\alpha - f_{s_0}$.

Proof: Fix a certain $i < s_0$. At time $k_{s_0} - 1$ machine i is already tall since $i < s_0$. By definition of s_0, the weight of k_{s_0} is at most α. At time $k_{s_0} - 1$ the sets S_1 and S_3 are empty since the job k_{s_0} is assigned to a machine in S_2. In particular, machine i is not in S_3. Thus, at time $k_{s_0} - 1$, the load of machine i is larger than $1 + \alpha - w_{k_{s_0}} \geq 1$.

If the raising job of machine i also raised it above 1 then by corollary 3.10 the weight of the raising job must be larger than $1 - f_i \geq 1 - f_{s_0}$. Otherwise, there was another job that arrived after the raising job which raised the machine above 1. Those two jobs arrived before k_{s_0} and by lemma 7.3 they are both larger than $\alpha - f_{s_0}$. ■

Lemma 7.5 Assume that there are n_1 jobs larger than β and other n_2 jobs larger than $1-\beta$ such that $1/2 \leq \beta \leq 2/3$. Then $n_1 + \frac{n_2}{2} \leq m$.

Proof: The optimal algorithm can assign to one machine at most one job that is larger than β or at most two jobs each is larger than $1-\beta$. The n_1 jobs are assigned to n_1 machines and the remaining $m - n_1$ machines can hold at most 2 jobs, each is larger than $1-\beta$. This implies that $2(m-n_1) \geq n_2$, or $n_1 + \frac{n_2}{2} \leq m$. ■

Now we complete the proof of theorem 3.4. Assume that $ALG2_\alpha$ fails. By lemma 3.5 at time n all the machines are tall and there are two machines whose load is below 1. By definition of s_0, all jobs k_i for $i > s_0$ are larger than α. Lemma 7.2 implies that $f_{s_0} < 1-\alpha$. Therefore, by lemma 7.4, each machine $i < s_0$ either contains a job that is larger than α or two jobs each is larger than $\alpha - (1-\alpha) = 2\alpha - 1$ and machine s_0 contains a job larger than $2\alpha - 1$. The last job is larger than α by corollary 3.6.

Thus we proved that there are certain numbers m_1 and m_2 such that $m_1 + m_2 = m$ and there are $m_1 + 1$ jobs larger than α and $2m_2 - 1$ jobs larger than $2\alpha - 1$. For $\alpha \geq 2/3$, this contradicts lemma 7.5 with a choice of $\beta = 2/3$ since

$$m_1 + 1 + \frac{2m_2 - 1}{2} = m_1 + m_2 + \frac{1}{2} > m.$$

7.2 Upper bound for the improved algorithm

We prove theorem 4.2. From now on, we assume $1/2 \leq \alpha \leq 2/3$. Since our algorithm is a special case of $ALG2_\alpha$ we can use the lemmas in the previous section. As before, we begin by assuming the algorithm fails on the last job of some sequence of $n+1$ jobs, J. According to lemma 3.5 at time n there are two machines whose load is less than 1, denoted s_1 and s_2, and all machines are tall which implies that we can define an order on the machines and rename them according to that order. The series k_i and the series f_i are defined as before. Note that by this order the earliest machine in S_3 is the one with the minimal index.

Lemma 7.6 As long as the load of a certain tall machine i is at most l, all arriving jobs in the sets J_2 and J_3 that are assigned to machines $i' > i$ are larger than $1 + \alpha - l$.

Proof: Take a certain job $j \in J_2$ assigned to a machine $i' > i$. It was placed on a machine in S_2 and therefore the set S_3 was empty when it arrived. In particular, $i \notin S_3$ and since machine i is tall, $w_j > 1 + \alpha - l_i(j-1) \geq 1 + \alpha - l$. In case job j is in J_3, we know the algorithm placed it on the earliest machine from the set S_3. Since $i < i'$, machine i was not in the set S_3 and as before, $w_j > 1 + \alpha - l$. ■

Recall that s_0 is the minimum index such that $w_{k_i} > \alpha$ for all $i > s_0$ and $1 \leq s_0 \leq m$.

Lemma 7.7 $f_{s_0} > 2\alpha - 1$.

Proof: Assume by contradiction that $f_{s_0} \leq 2\alpha - 1$. By definition of s_0, all jobs k_i for $i > s_0$ are larger than α. According to lemma 7.4 every machine $i < s_0$ either contains a job that is larger than $1-(2\alpha-1) = 2(1-\alpha) \geq \alpha$ or two jobs each is larger than $\alpha-(2\alpha-1) = 1-\alpha$ and machine s_0 contains a job that is larger than $1-\alpha$. The last job is larger than α by corollary 3.6. Since $1/2 \leq \alpha \leq 2/3$ this contradicts lemma 7.5. ■

Definition 7.8 Define the following three disjoint sets of machines that include all machines except s_0:

- $M_1 = \{i \in M \mid f_i \leq 2\alpha - 1\}$
- $M_2 = \{i \in M \mid i < s_0, f_i > 2\alpha - 1\}$
- $M_3 = \{i \in M \mid i > s_0\}$

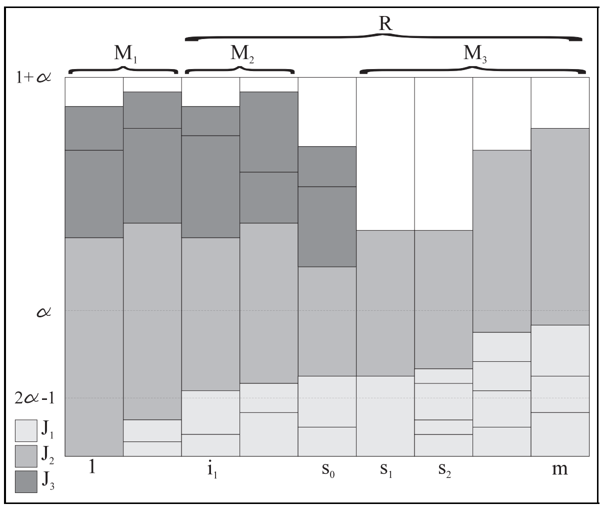

Figure 5: The sets M_1, M_2, M_3 and R.

Lemma 7.9 Assuming that $\alpha \geq 5/8$, every machine from the set $R = M_2 \cup \{s_0\} \cup M_3$ contains a job that is larger than $2\alpha - 1$ in J_1.

Proof: Let i_1 be the first machine in the set R. We look on the set of jobs $I = \{1, \ldots, k_{i_1} - 1\}$ and a subset of jobs $I' \subseteq I$ that are assigned to a machine in R. Notice that $I' \subseteq J_1$ since until time $k_{i_1} - 1$ all machines from R are short. Thus, it is enough to show that every machine from R contains a job in I' that is larger than $2\alpha - 1$.

We prove that for the set I our improved algorithm is a scale down by $1-\alpha$ of an algorithm in the family $D(ALG1_\beta, R)$ for $\beta = \frac{2\alpha-1}{1-\alpha}$. Recall that an algorithm in the family $D(ALG1_\beta, R)$ only specifies the method used in placing jobs to machines from R. The scale down by $1-\alpha$ of $ALG1_\beta$ defines the three scaled down sets of S_1^β, S_2^β and S_3^β. Notice that the three scaled down sets are exactly S_{11}^α, S_{12}^α and S_{13}^α used in the improved algorithm. Since the improved algorithm assigns all the jobs in I' to one of the sets S_{11}^α, S_{12}^α or S_{13}^α it is equivalent to a scaled down version of $D(ALG1_\beta, R)$.

By definition 7.8, $f_{i_1} > 2\alpha - 1$ and by lemma 3.9, at time $k_{i_1} - 1$ the load of each machine from the set R is also above $2\alpha - 1 = \beta(1-\alpha)$. According to corollary 3.8, $s_0 \le s_1 < s_2$ and therefore both s_1 and s_2 are in R. Lemma 3.11 implies that at time $k_{i_1} - 1$ the loads of both machines s_1 and s_2 are below $1 - \alpha = 1(1-\alpha)$. Thus, both of the scaled down assumption of lemma 3.13 hold here. Thus, by lemma 3.13 we conclude that there is a job of weight larger than $\beta(1-\alpha) = 2\alpha - 1$ from the set I' in every machine from the set R. ■

Definition 7.10 A job is said to be of type 1 if it is larger than α. Jobs of type 2 are larger than $1-\alpha$ and type 3 are larger than $2\alpha - 1$.

Using this definition, lemma 7.9 implies that every machine from the set R contains a job of type 3 in J_1. Next we prove that there are additional large jobs in the sets J_2 and J_3. We consider in lemma 7.11 and lemma 7.12 two possible cases according to the minimum load of machines of M_1 at time $k_{s_0} - 1$.

Lemma 7.11 Assume that at time $k_{s_0} - 1$ there is a machine from the set M_1 whose load is at most 2α. Then every machine must hold the following large jobs from the sets J_2 and J_3:

- All machines in M_1 contain one of the following:
  - **a.** A job that is larger than $2(1-\alpha)$,
  - **b.** Two jobs, the first of type 1 and the second of type 3,
  - **c.** Two jobs of type 2,
  - **d.** At most one machine contains two jobs: the first of type 2 and the second of type 3.
- Each machine from the set M_2 either contains a job of type 1 or two jobs of type 2.
- Machine s_0 contains a job of type 2.
- All machines in M_3 contain a job of type 1.

Proof: First, the raising jobs of machines from the set M_3 are of type 1 by the definition of the set. Denote by i_s a machine from the set M_1 whose load is at most 2α at time $k_{s_0} - 1$. By definition of s_0, $k_{s_0} < \alpha$ and thus at time $k_{s_0} - 1$ the loads of all machines $i < s_0$ are already larger than 1. Fix a certain $i \in M_2$. Since $i < s_0$, by time $k_{s_0} - 1$ the load of machine i is already above 1. If it is raised above 1 by its raising job, then the raising job is larger than $1 - (1-\alpha) = \alpha$ by lemma 7.2. Otherwise, it is raised above 1 by at least two jobs. These two jobs arrive before k_{s_0} and therefore the load of machine i_s is still below 2α at their arrival time. Since $i_s \in M_1$, $i_s < i$ and by lemma 7.6 both of the jobs are larger than $1 + \alpha - 2\alpha = 1 - \alpha$. When job k_{s_0} arrived, the load of machine i_s was still below 2α and by the same lemma, $w_{k_{s_0}} > 1 - \alpha$.

Next we look on machines from the set M_1. As before, at time $k_{s_0} - 1$ the loads of all machines $i < s_0$ are already larger 1. If a certain machine i was raised by its raising job above 1 then it satisfies case **a** since the weight of its raising job is $w_{k_i} > 1 - f_i \ge 2(1-\alpha)$ by definition 7.8. All other machines are raised above 1 by at least two jobs. The second job entered before k_{s_0} and by lemma 7.3 and lemma 3.11 its weight is above $2\alpha - 1$ i.e., it is of type 3. If a machine contains a raising job that is larger than α (of type 1) then it satisfies case **b**.

We are left with a set of machines that are raised to 1 by at least two jobs, with the raising job's weight being at most α. Let i_2 be the last machine from this set assuming it is not empty. Thus, as the raising job k_{i_2} arrives, the loads of all previous machines are already above 1 since $w_{k_{i_2}} \leq \alpha$. That means that both the raising and the second jobs of all previous machines have already arrived and they are larger than $\alpha - f_{i_1} \geq \alpha - (2\alpha - 1) = 1 - \alpha$ by lemma 7.3. Hence, they satisfy case **c**. The only machine that may not satisfy **a**, **b** or **c** is i_2. It contains two jobs, by lemma 7.3 the raising job, k_{i_2}, is of type 2 and the second is of type 3 since it entered before k_{s_0}. Thus, it satisfies case **d.**

Note that all jobs in the proof are from the sets J_2 and J_3, as required. ■

Lemma 7.12 Assume that at time $k_{s_0} - 1$ the loads of all machines from the set M_1 are more than 2α. Every machine must hold the following large jobs from the sets J_2 and J_3:

- All machines in M_1 contain one of the following:
  - **a.** Two jobs that the sum of their weights is above one,
  - **b.** Three jobs, one of type 1 and two of type 3,
  - **c.** Three jobs, two of type 2 and one of type 3,
  - **d.** At most one machine contains three jobs: one of type 2 and two of type 3.
- Each machine from the set M_2 either contains a job of type 1 or two jobs of type 3.
- Machine s_0 contains a job of type 3.
- All machines in M_3 contain a job of type 1.

Proof: All jobs from the sets J_2 and J_3 that arrived before k_{s_0} are of type 3 according to lemma 7.3. This fact will be used throughout the proof.

The raising job of all machines from the set M_3 is of type 1 by definition of the set. If a machine in M_2 is raised above 1 by its raising job then the raising job is of type 1. Otherwise, there are at least two jobs, both arriving before k_{s_0} and therefore both are of type 3. The raising job of machine s_0 is also of type 3.

We assumed that at time $k_{s_0} - 1$ the loads of all machines from the set M_1 are above 2α. Notice that $f_i \leq 2\alpha - 1$ for all $i \in M_1$ and therefore the raising job itself cannot raise a machine from M_1 above 2α. In case there are two jobs that raise a machine above 2α then the sum of their weights is above 1 which satisfies case **a**. All other machines in M_1 are raised above 2α by at least three jobs. The first is the raising job and at least two other jobs from J_3, all arriving before k_{s_0}. If the raising job of some machines is of type 1 the machine satisfies case **b** since the two other jobs are of type 3.

We are left with a set of machines that contain a job in the set J_2 whose weight is at most α and at least two jobs from the set J_3. Let i_2 be the last machine from the above set, assuming it is not empty. Since $w_{k_{i_2}} < \alpha$, the loads of all previous machines are above

1 when k_{i_2} arrives. Fix a certain machine $i < i_2$. Since the weight of its raising job is at most α and $(2\alpha - 1) + \alpha \leq 1$, it cannot raise the machine above 1. Therefore, machine i contains at least two jobs that arrive before k_{i_2} and by lemma 7.3 they are both larger than $\alpha - (2\alpha - 1) = 1 - \alpha$. As before, the third job is of type 3 and therefore machine i satisfies case **c**. Machine i_2 itself contains a raising job of type 2 since $f_{i_2} \leq 2\alpha - 1$ and two other jobs of type 3.

Note that as in the previous proof, all the indicated jobs are from the sets J_2 and J_3. ■

Next we prove that the combinations of jobs presented in the previous lemmas together with the last job cannot be assigned by an optimal algorithm. The number and types of jobs are taken from lemmas 7.9, 7.11 and 7.12 as indicated in parenthesis.

Lemma 7.13 Each of the following two sets of jobs cannot be assigned by an optimal algorithm assuming $5/8 \leq \alpha \leq 2/3$:

- m_1 times a job that is larger than $2(1-\alpha)$, (M_1, case **a**)
  m_2 times a job of type 1 and a job of type 3, (M_1, case **b**)
  m_3 times two jobs of type 2, (M_1, case **c**)
  $m_4 \leq 1$ times a job of type 2 and a job of type 3, (M_1, case **d**)
  m_5 times a job of type 1, (M_2, first case)
  m_6 times two jobs of type 2, (M_2, second case)
  a job of type 2, (s_0)
  m_7 times a job of type 1, (M_3)
  $m_5 + m_6 + 1 + m_7$ times a job of type 3, (R, jobs from J_1)
  a job of type 1, (the last job)
  such that $m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + 1 + m_7 = m$.

- m_1 times two jobs that the sum of their weights is above 1, (M_1, case **a**)
  m_2 times a job of type 1 and two jobs of type 3, (M_1, case **b**)
  m_3 times two jobs of type 2 and a job of type 3, (M_1, case **c**)
  $m_4 \leq 1$ times a job of type 2 and two jobs of type 3, (M_1, case **d**)
  m_5 times a job of type 1, (M_2, first case)
  m_6 times two jobs of type 3, (M_2, second case)
  a job of type 3, (s_0)
  m_7 times a job of type 1, (M_3)
  $m_5 + m_6 + 1 + m_7$ times a job of type 3, (R, jobs from J_1)
  a job of type 1, (the last job)
  such that $m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + 1 + m_7 = m$.

Proof: We begin by determining all the combinations of jobs of types 1,2,3 an optimal algorithm can assign to a single machine. The optimal algorithm can assign a job of type 1 with at most one job of type 3. It can assign two jobs of type 2 to a single machine, but no other job can be assigned with them. Another possibility is to assign only one job of type 2 and at most two jobs of type 3 with it. It can assign at most 3 jobs of type 3 to a single machine.

We first prove that the first set cannot be assigned by the optimal algorithm. Since jobs larger than $2(1-\alpha)$ cannot be assigned together with any of the other jobs, $m - m_1$ machines are left for the other jobs. The other jobs are $m_2 + m_5 + m_7 + 1$ jobs of type 1, $2m_3+m_4+2m_6+1$ jobs of type 2 and $m_2+m_4+m_5+m_6+m_7+1$ jobs of type 3. Each machine can hold at most one job of type 1. A machine that holds a job of type 1 can hold at most one job of type 3 and no jobs of type 2. That leaves $m-m_1-(m_2+m_5+m_7+1) = m_3+m_4+m_6$ machines that are supposed to hold at least $2m_3 + m_4 + 2m_6 + 1$ jobs of type 2 and $m_2 + m_4 + m_5 + m_6 + m_7 + 1 - (m_2 + m_5 + m_7 + 1) = m_4 + m_6$ jobs of type 3. In case $m_4 = 0$ we reach a contradiction because the number of jobs of type 2 in more than twice the number of machines and each machine can hold at most two jobs of type 2. In case $m_4 = 1$, the number of jobs of type 2 is exactly twice the number of machines. That means each machine holds 2 jobs of type 2. However, the number of jobs of type 3 is at least 1 but none of the machines can hold a job of type 3 since it already holds two jobs of type 2. This completes the proof of the first case.

Next we prove that the second set cannot be assigned by the optimal algorithm. We begin by showing that the m_1 pairs of jobs that the sum of their weight is above 1 can be ignored. More formally, assume that a set of jobs J contains two jobs, j_1 and j_2 that the sum of their weights is above 1. We show that if J can be assigned by an optimal algorithm to m machines then the set of jobs $J-(\{j_1\}\cup\{j_2\})$ can be assigned by an optimal algorithm to $m-1$ machines. Consider the assignment of the set J. Denote by i_1 and i_2 the two machines to which j_1 and j_2 are assigned. $i_1 \neq i_2$ since the sum of their weights is above 1. Thus, the sum of the weights of the other jobs in i_1 and i_2 is less than 1 and can be assigned to one machine.

The last paragraph implies that if the entire set of jobs can be assigned by the optimal algorithm to m machines then the set of jobs without the m_1 pairs of jobs can be assigned to $m - m_1$ machines. There are $m_2 + m_5 + m_7 + 1$ jobs of type 1, $2m_3 + m_4$ jobs of type 2 and $2m_2 + m_3 + 2m_4 + m_5 + 3m_6 + m_7 + 2$ jobs of type 3 that are supposed to be assigned to $m - m_1$ machines. Each machine can hold at most one job of type 1. Together with that job, only one job of type 3 can be assigned. That leaves $2m_3 + m_4$ jobs of type 2 and at least $m_2 + m_3 + 2m_4 + 3m_6 + 1$ jobs of type 3 that are supposed to be assigned to $m - m_1 - (m_2 + m_5 + m_7 + 1) = m_3 + m_4 + m_6$ machines. Out of these machines assume m' machines contain two jobs of type 2 and all other machines contain at most one job of type 2. Obviously, $0 \leq m' \leq \lfloor \frac{2m_3+m_4}{2} \rfloor = m_3$. The m' machines cannot hold any other job so we are left with $2m_3 + m_4 - 2m'$ jobs of type 2 and $m_2 + m_3 + 2m_4 + 3m_6 + 1$ jobs of type 3 that are supposed to be assigned to $m_3 + m_4 + m_6 - m'$ machines. These machines hold at most one job of type 2, and thus $2m_3 + m_4 - 2m'$ machines hold a job of type 2. The optimal algorithm can assign at most two jobs of type 3 with the job of type 2 so we are left with $m_2 + m_3 + 2m_4 + 3m_6 + 1 - 2(2m_3 + m_4 - 2m')$ jobs of type 3 and $m_3 + m_4 + m_6 - m' - (2m_3 + m_4 - 2m')$ machines. Since each machine can hold up to three jobs of type 3,

$$3(m_3+m_4+m_6-m'-(2m_3+m_4-2m')) \geq m_2+m_3+2m_4+3m_6+1-2(2m_3+m_4-2m')$$

$$3m_6 - 3m_3 + 3m' \geq m_2 - 3m_3 + 3m_6 + 4m' + 1$$

$$0 \geq m_2 + 1 + m'$$

which is impossible since all the variables are non-negative. ■

Now we complete the proof of theorem 4.2. Assume that the improved algorithm fails. Note that the jobs of lemma 7.9 are from the set J_1 while jobs of lemmas 7.11 and 7.12 are from J_2 and J_3 and thus, they are disjoint. Therefore, one of the two cases of lemma 7.13 occurs. That contradicts the assumption that the algorithm fails.

7.3 Upper bound for $ALG1_\alpha$ for fixed $m \geq 5$

We prove theorem 5.1. Our proof is very similar to the original proof of theorem 3.3 with some minor changes. Again, we begin by assuming the algorithm fails on the last job of some sequence of $n+1$ jobs. By lemma 3.5, all the machines are tall at time n and we can define the jobs k_i for $1 \leq i \leq m$ and the series f_i. The following lemma somewhat improves lemma 3.5:

Lemma 7.14 At time n there are two machines whose load is below $1 - \frac{\alpha}{m-1}$.

Proof: Assume by contradiction that the loads of all machines except machine i are at least $1 - \frac{\alpha}{m-1}$. Since the algorithm failed when job $n+1$ arrived, $l_i(n) + w_{n+1} > 1 + \alpha$ and the total load of all machines is above $1 + \alpha + (m-1)(1 - \frac{\alpha}{m-1}) = m$ which is a contradiction. ■

Let s_1 and s_2 be two machines whose load is below $1 - \frac{\alpha}{m-1}$. According to lemma 3.7 all jobs k_i for $i > s_1$ are larger than $\alpha + \frac{\alpha}{m-1}$. In particular $w_{k_{s_2}} > \alpha + \frac{\alpha}{m-1}$. Since the load of machine s_2 is below $1 - \frac{\alpha}{m-1}$ at time n,

$$1 - \frac{\alpha}{m-1} > l_{s_2}(n) \geq l_{s_2}(k_{s_2}) = l_{s_2}(k_{s_2} - 1) + w_{k_{s_2}} = f_{s_2} + w_{k_{s_2}}.$$

Therefore,

$$f_{s_2} < 1 - \frac{\alpha}{m-1} - w_{k_{s_2}} < 1 - \alpha - \frac{2\alpha}{m-1}$$

and by corollary 3.10 $f_i < 1 - \alpha - \frac{2\alpha}{m-1}$ for all $i \leq s_2$.

According to lemma 3.9, at time $k_{s_1} - 1$, the load of machine s_2 was at most f_{s_2} but more than the load of machine s_1. Therefore, at that time, the loads of both machines are below $1 - \alpha - \frac{2\alpha}{m-1}$ and by our choice of α, this is at most $\frac{\alpha}{2}$. By lemma 3.12 this implies that $f_i = 0$ for $i \leq s_1$. Therefore, the raising jobs of machines i, for $1 \leq i \leq s_1$, is larger than α.

We proved that all the m raising jobs are larger than α. The last job is also larger than α by corollary 3.6. Note that $\alpha \geq \frac{2m-2}{3m+1} \geq 1/2$ for $m \geq 5$. Thus, we reached a contradiction.

7.4 Upper bound for $ALG12_\alpha$ for fixed $m \geq 3$

We prove theorem 5.2. Assume the algorithm fails on the last job of some sequence of $n+1$ jobs. By lemma 7.14, at time n there are two machines whose load is below $1 - \frac{\alpha}{m-1}$. Let s_1

and s_2 be two machines ($s_1 < s_2$) whose load is below $1 - \frac{\alpha}{m-1}$. By the discussion following lemma 7.14 all jobs k_i for $i > s_1$ are larger than $\alpha + \frac{\alpha}{m-1}$ and $f_i = 0$ for $i \leq s_1$.

The next lemma improves corollary 3.6:

Lemma 7.15 The last job is larger than $\alpha + \frac{\alpha}{m-1}$.

Proof: By lemma 7.14, at time n there was a machine i whose load is less than $1 - \frac{\alpha}{m-1}$. Since the algorithm failed to assign the last job $1 - \frac{\alpha}{m-1} + w_{n+1} > 1 + \alpha$ or $w_{n+1} > \alpha + \frac{\alpha}{m-1}$. ■

Definition 7.16 Let s_0 be the smallest non-negative integer such that $w_{k_i} > \alpha + \frac{\alpha}{m-1}$ for all $i > s_0$.

Note that the definition of s_0 is slightly different from the original definition. As before, s_0 is always defined and $0 \leq s_0 \leq m$. If $s_0 = 0$ then there are m jobs of weight larger than $\alpha + \frac{\alpha}{m-1}$. The last job is also larger than $\alpha + \frac{\alpha}{m-1}$ by lemma 7.15. This is a contradiction since we found $m+1$ jobs larger than $\alpha + \frac{\alpha}{m-1} > 1/2$ for our choice of m and α.

By definition of s_0, $s_0 \leq s_1$ and therefore $f_{s_0} = 0$. Now we present an improvement to lemma 7.4.

Lemma 7.17 Each machine $i < s_0$ either contains a job that is larger than $1 - \frac{\alpha}{m-1}$ or two jobs each is larger than α. Machine s_0 itself contains a job that is larger than α.

Proof: Fix a machine $i < s_0$. Since $w_{k_{s_0}} < \alpha + \frac{\alpha}{m-1}$, at time $k_{s_0} - 1$, the load of machine i was already larger than $1 + \alpha - (\alpha + \frac{\alpha}{m-1}) = 1 - \frac{\alpha}{m-1} \geq \alpha$ by our choice of α. This implies that machine i is tall before job k_{s_0} arrives. If the raising job of machine i raised it above $1 - \frac{\alpha}{m-1}$ then it is larger than $1 - \frac{\alpha}{m-1}$ since $f_i = 0$. Otherwise, there are at least two jobs, one from the set J_2 and the other from J_3 that both arrived before k_{s_0}. By lemma 7.3, this implies that both of these jobs are larger than α. ■

By our choice of α, $1 - \frac{\alpha}{m-1} \geq \alpha + \frac{\alpha}{m-1}$. Therefore, by definition 7.16 and lemmas 7.17, 7.15 there are $2m_1 - 1$ jobs larger than α and $m_2 + 1$ jobs larger than $\alpha + \frac{\alpha}{m-1}$ for certain numbers m_1, m_2 such that $m_1 + m_2 = m$. By our choice of α and m, $\alpha + (\alpha + \frac{\alpha}{m-1}) \geq 1$. This contradicts lemma 7.5.

7.5 General Lower Bound

Theorem 7.18 The stretching factor of any deterministic on-line algorithm for the bin-stretching problem is at least $4/3$ for any number $m \geq 2$ of machines.

Proof: Look at the following two sets jobs:

- m jobs of weight $1/3$ and another m jobs of weight $2/3$.
- m jobs of weight $1/3$ and a job of weight 1.

Obviously, these two sets can be assigned to $m \geq 2$ machines by an optimal off-line algorithm.

Assume a certain deterministic on-line algorithm receives m jobs of weight $1/3$. If the algorithm assigns the m jobs on m different machines then the algorithm receives a job of weight 1 as in the second set. Since the loads of all the machines are $1/3$, the load of the machine to which the algorithm assigns the last job is $4/3$.

Otherwise, there is a machine to which the algorithm assigned two jobs of weight $1/3$. Then we continue with m jobs of weight $2/3$ as in the first set. The algorithm can either put all the m last jobs on m different machines or put at least two of the last m jobs on a single machine. In both cases, there is a machine whose load is $4/3$. ∎

7.6 Lower bound for $ALG1_\alpha$

In this section we show that $ALG1_\alpha$ does not have a stretching factor of $1+\alpha$ for a fixed $\alpha < 2/3$. For any fixed $\alpha < 2/3$ we show an example in which the algorithm fails. The number of machines increases as α is closer to $2/3$ and therefore the lower bound is valid for a large number of machines. From now on we fix $\alpha < 2/3$.

In phase 1 a sequence of infinitesimal jobs of total weight αm_1 arrives where m_1 will be chosen later. By the description of $ALG1_\alpha$, the algorithm fills each machine up to a load of α and then continues to the next machine. Therefore, m_1 machines have loads of α and all other machines are empty.

In phase 2 a sequence of jobs whose weight is $\frac{\alpha}{2}+\epsilon$ arrives. We choose a very small constant $\epsilon > 0$. Recall that the algorithm assigns a job to a machine in S_2 only if $S_1 = S_3 = \phi$. Therefore, the algorithm assigns the first $m - m_1$ jobs to the empty $m - m_1$ machines. As the next job arrives, all the machines are in S_2 and the algorithm assigns it to the least loaded machine which is a machine with one job of weight $\frac{\alpha}{2}+\epsilon$. Denote this machine i_1. The next few jobs are assigned to i_1 since it is in S_3 and all other machines are in S_2. The number of jobs is so that the load of i_1 is at least $1+\alpha-(\frac{\alpha}{2}+\epsilon) = 1+\frac{\alpha}{2}-\epsilon$.

In phase 3 a sequence of m jobs of weight $\frac{1}{2}+\frac{\alpha}{4}$ arrives. Notice that the minimal load before placing these jobs is $\frac{\alpha}{2}+\epsilon$ and therefore the algorithm cannot place two of these jobs on the same machine. The algorithm cannot place any of these jobs on machine i_1 since

$$1+\frac{\alpha}{2}-\epsilon+\frac{1}{2}+\frac{\alpha}{4} = \frac{3}{2}+\frac{3\alpha}{4}-\epsilon > 1+\alpha$$

for small ϵ. Thus, the algorithm cannot assign these m jobs.

Now we show that the optimal algorithm can assign the same set of jobs for an appropriate choice of m_1. We choose m_1 to be the number of excessive jobs of machine i_1. It is important to see that this number is constant and does not depend on m since it is bounded by $\frac{1+\alpha}{\frac{\alpha}{2}+\epsilon}$. Note that the number of jobs of size $\frac{\alpha}{2}+\epsilon$ is exactly m by our choice of m_1. The optimal algorithm assigns each machine with a job of weight $\frac{1}{2}+\frac{\alpha}{4}$ and a job of weight $\frac{\alpha}{2}+\epsilon$. This results in a load of

$$\frac{1}{2}+\frac{\alpha}{4}+\frac{\alpha}{2}+\epsilon = \frac{1}{2}+\frac{3\alpha}{4}+\epsilon < 1$$

on each machine by our choice of ϵ and α.

Figure 6: After phase 3

This leaves only the infinitesimal jobs. Their total weight is $m_1\alpha$ and therefore constant. Each machine can hold a load of $1 - (\frac{1}{2} + \frac{3\alpha}{4} + \epsilon) > 0$ infinitesimal jobs. Since this is a positive constant, we can choose m to be large enough for all the infinitesimal jobs to fit.

7.7 Lower bound for the improved algorithm

In this section we show that the improved algorithm does not have a stretching factor of $1+\alpha$ for a fixed $\alpha < 5/8$. For any fixed $\alpha < 5/8$ we show an example in which the algorithm fails. The number of machines increases as α is closer to $5/8$ and therefore the lower bound is valid for a large number of machines.

In phase 1 a sequence of infinitesimal jobs of total weight $m_1(2\alpha - 1)$ arrives where m_1 will be chosen later. The algorithm fills each machine to a load of $2\alpha - 1$ and then continues to the next machine. Therefore, m_1 machines have loads of $2\alpha - 1$ and all other machines are empty.

In phase 2 a sequence of jobs of weight $\alpha - \frac{1}{2} + \epsilon$ arrives. We choose $\epsilon > 0$ to be a very small constant. The algorithm assigns the first $m - m_1$ jobs to the empty machines since it prefers a machine from S_{11} over a machine from S_{12}. The next job must cross the lower threshold, and therefore it is assigned to the least loaded machine. The following jobs fill the machine that crossed the lower threshold until no other job can be assigned to it without crossing the upper threshold. We continue with these jobs until m_2 machines are filled.

In phase 3 jobs of weight $\frac{1}{4}$ arrive. The first $m - m_2$ are assigned to $m - m_2$ different machines. The next job must cross the upper threshold and therefore it is assigned to the least loaded machine. We continue until the load of the first machine i_1 that crossed the upper threshold is above $1 + \alpha - \frac{1}{4} = \frac{3}{4} + \alpha$.

The minimal load is above $\alpha - \frac{1}{4} + \epsilon$ and the maximal load is above $\frac{3}{4} + \alpha$. In phase 4 m jobs of weight $\frac{5}{8}$ arrive. Assigning two of these jobs to the same machine results in a load above $\alpha - \frac{1}{4} + \epsilon + 2 \cdot \frac{5}{8} = 1 + \alpha + \epsilon$ and therefore all jobs must be assigned to different

Figure 7: After phase 4

machines. However, that results in a load of at least $\frac{3}{4}+\alpha+\frac{5}{8}>1+\alpha$. Thus, the algorithm for $\alpha<\frac{5}{8}$ fails.

Next we show that by choosing m_1 and m_2 appropriately, the jobs can be assigned by an optimal algorithm. Each machine in the optimal algorithm assignment contains a job of weight $\frac{5}{8}$, a job of weight $\frac{1}{4}$, a job of weight $\alpha-\frac{1}{2}+\epsilon$ and a certain part of the infinitesimal jobs.

There are $m-m_2$ machines which contain one job of size $1/4$ except one machine which contains several excessive jobs of size $1/4$. Clearly, the number of excessive jobs is at most 6 since $(6+1)\cdot\frac{1}{4}>1+\alpha$. We choose m_2 to be this number of excessive jobs.

There are $m-m_1-m_2$ machines which contain one job of size $\alpha-\frac{1}{2}+\epsilon$ and m_2 machines that contain an extra number of these jobs. The number of these excessive jobs is bounded by $m_2\cdot\frac{\alpha}{\alpha-\frac{1}{2}+\epsilon}$ which is constant. We choose m_1 to be this number of excessive jobs.

Choosing m_1 and m_2 in that way allows the optimal algorithm to assign one job of each weight to each machine: one of weight $5/8$, one of weight $1/4$ and another one of weight $\alpha-\frac{1}{2}+\epsilon$. Therefore, the load of each machine is $\alpha+\frac{3}{8}+\epsilon<1$, by our choice of α. The total weight of the infinitesimal jobs is $m_1(2\alpha-1)$ which is constant. We choose m to be large enough so that this weight of infinitesimal jobs can be spread over all m machines. The exact number of machines should be at least $\frac{m_1(2\alpha-1)}{1-(\alpha+\frac{3}{8}+\epsilon)}$.