Bin Stretching Revisited

Leah Epstein*

Abstract

We study three on-line models of bin stretching on two machines. For the case where the machines are identical and the jobs arrive sorted by non-increasing sizes, we show a tight bound of 10/9 on the competitive ratio. For two related machines, we show a preemptive algorithm with competitive ratio 1 for any speed ratio, and two new non-preemptive algorithms. We prove that the upper bound on the competitive ratio achieved by the non-preemptive algorithms is optimal for almost any speed ratio, and close to optimal for all other speed ratios.

1 Introduction

The name "Bin Stretching" was first introduced by Azar and Regev [2]. The problem is actually older and was studied by Coffman, Garey and Johnson [3], as an off-line problem, more than twenty years ago. In the on-line problem, a set of bins of fixed sizes is given. Items arrive on-line while it is known that there exists a valid packing of all items into the given bins. The goal of the algorithm is to pack all items into the bins assigning each item to one of the bins. It is allowed to overload the bin i.e. the sum of the items assigned to the bin need not be bounded by its size. The goal function is the stretch factor, which is the maximum ratio for any bin between the size to which any bin was stretched (the sum of all items assigned to it) and its original size. A possible application mentioned in [2] is moving files one by one from a system of m servers to a different and remote system. Note that [2] study the problem in the context of identical bins, i.e. all original bins are of size 1.

We can also see this problem as a scheduling problem on uniformly related machines. Jobs arrive on-line to be assigned one by one to a set of machines. Each job is assigned to a machine (or to non-overlapping time slots on the machines, for preemptive algorithms), before the next job is revealed. The smallest makespan achievable for the sequence is known in advance before any job arrives. The goal is to minimize the makespan of the on-line schedule. We used the scheduling terminology in this paper.

In this work we investigate the case where there are two related machines, i.e. $m = 2$. The first model we study is the case where the jobs arrive semi-online, sorted in non-increasing

*School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. lea@idc.ac.il. Research supported in part by the Israel Science Foundation, (grant No. 250/01-1)

order of sizes, and the machines are identical. In the other models we consider two machines of different speeds (or bins of different sizes). The speed ratio between the two machines is $q \geq 1$. Note that it is more convenient to use $q \in [1, \infty)$ rather than a speed ratio which is in $(0, 1]$. This approach is common also in other two machine problems, see e.g. [19, 9, 20]. Without loss of generality we assume that the speeds are 1 and $1/q$. We consider the preemptive and the non-preemptive cases.

We use the usual measure for on-line algorithms. This is the supremum ratio between the cost of the on-line (or semi-online) algorithm and the optimal off-line algorithm (this algorithm and its cost are both denoted by OPT). Without loss of generality we always assume $OPT = 1$. We denote the semi-online algorithm and its cost by $SONL$. For a sequence of jobs, the jobs are $J_1, J_2, \ldots$. The processing times of the jobs, also called "sizes" are denoted by $p_1, p_2 \ldots$. We sometimes use jobs J'_i and J''_i whose sizes are p'_i and p''_i respectively. For a job j we denote its size by p_j. For non-preemptive algorithms, the load of a machine is the sum of the sizes of jobs assigned to it. For identical machines we denote the loads by P_1 and P_2, whereas for related machines we denote the loads by P_1 (of the fast machine) and P_q (of the slow machine). The time to run a job of size p on a machine of speed $1/q$ is pq. Since $OPT = 1$, the optimal off-line algorithm has load of at most $1/q$ on that machine. The competitive ratio is the maximum between the costs of the two machines. The cost (also called "stretch factor") of a machine of speed 1 is simply its load. The cost of a machine of speed $1/q$ is qP_q since this is the time required to run the jobs on it.

Previous work: Both bin packing and scheduling problems were extensively studied in an on-line environment, for surveys see [4, 1, 16]. More specifically, there has been some work on semi-online scheduling [15, 8, 7, 14, 21, 12] where "semi-online" often means decreasing job sizes but it can also mean e.g. that the assignment of some jobs may be delayed [14, 21]. Generally "semi-online" can describe any situation which is not off-line, but some information is given in advance, or the jobs are revealed in some specific order. Considering the above problem as a bin packing model, the problem is similar to the *Extendable bin packing problem* [6, 18, 5] where an initial set of bins is also given, but the goal function is the sum of final sizes of bins rather than the maximum stretch factor. Each bin can be stretched by its own stretch factor, and the algorithm pays for both the original space and the extra space used.

As mentioned above, the first paper that studied bin stretching is [3]. The goal in that paper was to design an offline approximation for multiprocessor scheduling on identical machines. This was done adapting the "First-Fit-Decreasing" (FFD) technique which has a good performance (competitive ratio 11/9) for bin packing. The scheduling algorithm has an a-priori knowledge of the offline makespan, and a parameter C, which is the stretch factor, is given. Bins of size C are initialized and filled by the jobs (which are sorted in non-increasing order) using the First-Fit technique (assign a job into the bin of the lowest index where it fits). Ten years later, a PTAS for the basic multiprocessor scheduling problem was given by Hochbaum and Shmous [13]. However, the results of [3] are still interesting from the point of view of "bin stretching". In [3], it was shown that the stretch factor of FFD (i.e. the competitive ratio) is 8/7 for two machines, 15/13 for three machines, 20/17 for four, five, six and seven machines, and between 20/17 and 1.22 for eight machines or more.

Apart from [3], the most relevant results for our study are the results of [2] and results for semi-online scheduling on identical speed machines, where the total processing time is known in advance [14, 10]. In [2] an algorithm of competitive ratio $13/8 = 1.625$ is designed as a combination of two families of algorithms all with competitive ratio 5/3. That paper also gives a lower bound of 4/3 on the competitive ratio of any on-line algorithm any number of machines. For the case of two identical machines, [2] gives an algorithm of optimal competitive ratio 4/3. The second problem (known total processing time) is slightly different from bin stretching, as some jobs may have size larger than the average and hence OPT is not known. However the algorithm and lower bound of Kellerer et al. [14] (given independently) are quite similar to [2]. The m machine case was studied in [10] (also independently). That work gives an algorithm of competitive ratio 5/3, which resembles the algorithms in [2].

Previous results on preemptive algorithms in the off-line environment [11, 17] are also relevant to our study. Those papers give a polynomial solution to the problem of preemptive scheduling on related machines. The algorithms calculate OPT and use it to assign the jobs one by one. However the algorithms in both papers sort the jobs by non-increasing sizes before assigning them one by one, and hence this gives an optimal semi-online preemptive bin stretching algorithm.

Our results: We generalize the results of [2, 14] by considering a semi-online version of bin stretching (jobs arrive sorted) and the related machines (different speeds) version.

We give a tight analysis of bin stretching for two identical machines, where jobs arrive sorted by non-increasing running time (size). The tight competitive ratio of this problem is shown to be 10/9. We design a deterministic algorithm "SIZES" (of this competitive ratio) and show a matching lower bound. Note that the result of [3] for two machines gives an algorithm of competitive ratio 8/7 for this problem.

For two uniformly related machines, we give an optimal algorithm (i.e. of competitive ratio 1) for preemptive bin stretching. This algorithm, unlike [11, 17], is not semi-online and does not require the jobs to arrive sorted.

Finally we give a comprehensive study of the competitive ratio for the non-preemptive case as a function of the speed ratio between the two machines. We give upper bounds which are tight for almost any q (the total length of the interval where they are not tight is less than 0.5, and the largest gap between the upper bound and the lower bound is about 0.073). Specifically the intervals where an algorithm of tight competitive ratio is given are $[(1+\sqrt{65})/2 \approx 1.13278, (1+\sqrt{3})/2 \approx 1.36603] \cup [\sqrt{3} \approx 1.73205, \infty]$. The overall upper bound on the competitive ratio proved in this paper is $\sqrt{2}$, and the overall (highest) lower bound proved in this paper is $(1+\sqrt{3})/2$, achieved at $(1+\sqrt{3})/2$ and at $\sqrt{3}$ (note that for those points this is the tight competitive ratio, therefore the lower bound cannot be improved). Let

$r(q)$ be the competitive ratio as a function of q. The results are given in the following list:

$$r(q): \begin{cases} r(q) \in \left[1+\frac{1}{3q}, \frac{2q+2}{2q+1}\right] & \text{for } q \in [1, q_1 \approx 1.12433] \\ r(q) \in \left[q\left(\frac{3}{4}+\frac{\sqrt{65}}{20}\right), \frac{2q+2}{2q+1}\right] & \text{for } q \in \left[q_1, \frac{1+\sqrt{65}}{8} \approx 1.13278\right] \\ r(q) = \frac{2q+2}{2q+1} & \text{for } q \in \left[\frac{1+\sqrt{65}}{8}, \frac{1+\sqrt{17}}{4} \approx 1.28078\right] \\ r(q) = q & \text{for } q \in \left[\frac{1+\sqrt{17}}{4}, \frac{1+\sqrt{3}}{2} \approx 1.36603\right] \\ r(q) \in \left[1+\frac{1}{2q}, q\right] & \text{for } \frac{1+\sqrt{3}}{2} \le q \le \sqrt{2} \approx 1.41421 \\ r(q) \in \left[1+\frac{1}{2q}, \frac{q+2}{q+1}\right] & \text{for } \sqrt{2} \le q \le \phi = \frac{1+\sqrt{5}}{2} \approx 1.61803 \\ r(q) \in \left[\frac{q+1}{2}, \frac{q+2}{q+1}\right] & \text{for } \phi \le q \le \sqrt{3} \approx 1.73205 \\ r(q) = \frac{q+2}{q+1} & \text{for } q \ge \sqrt{3} \end{cases}$$

The parameter q_1 is a solution of $36x^4 - 135x^3 + 45x^2 + 60x + 10 = 0$.

The structure of the paper is as follows. In Section 2 we present the results for identical machines. In Section 3 we present the preemptive algorithm for related machines and in Section 4 we give the non-preemptive results for related machines.

2 Semi-online bin stretching on identical machines

In this section we prove the following theorem.

Theorem 2.1 *The tight competitive ratio for semi-online bin stretching on two identical machines is* $\frac{10}{9}$.

We start with showing a lower bound on the competitive ratio:

Lemma 2.1 *Any semi-online algorithm for bin stretching has competitive ratio of at least* $\frac{10}{9}$.

Proof. Consider the following sequence for which $OPT = 1$. We start with two jobs J_1 and J_2, both of size $4/9$. If they are assigned to the same machine, we continue with four jobs $(J_3, \ldots, J_6)$, each of size $5/18$. If all four new jobs run on the other machine then $SONL \ge 20/18 = 10/9$. Otherwise if at least one of the new jobs runs with the two first jobs we get $SONL \ge 8/9 + 5/18 = 21/18 > 10/9$. OPT runs J_1, J_3, J_4 on one machine and the other jobs on the second machine.

If the algorithm assigns J_1 and J_2 to different machines, we continue by three jobs J'_3, J'_4, J'_5, each of size $1/3$. At least two of them must run on one machine and $SONL \ge 4/9 + 2/3 = 10/9$. OPT runs J_1 and J_2 on one machine, and the three other jobs on the second machine. □

Next we describe the algorithm **SIZES**

The first step is to assign J_1 to M_1.
We define the algorithm for various cases. Before the application of the algorithm for any job J, two additional properties are checked. If one of them is valid, the assignment of J and future jobs is not done by the basic algorithm, but by using an appropriate rule.

Property 1. If the assignment of J to one of the machines, machine M_i, where $i = 1$ or $i = 2$, causes $8/9 \leq P_i \leq 10/9$, assign J to M_i and apply rule 1 with $\ell = 3 - i$ (the other machine) for all future jobs.

Property 2. If $p(J) \leq 2/9$, assign J and all future jobs by rule 2.

The rules:
Rule 1. Assign all jobs to machine M_ℓ.
Rule 2. Assign each arriving job to the least loaded machine.
If both properties hold at the same time, one of the two rules is chosen arbitrarily. (It would have been correct to always invoke rule 2, but for the convenience of proof we introduce rule 1 as well.)

We next discuss the assignment of the other jobs as a function of the sizes of J_1 and J_2. Note that either $p_1 + p_2 < 8/9$ or $p_1 + p_2 > 10/9$, otherwise property 1 holds for J_2 and rule 1 is invoked. Note also that $J_2 > 2/9$ otherwise property 2 holds and rule 2 is invoked.

We now define the basic algorithm. For each case we specify where jobs are assigned until some job has one of the properties.

Note that in all cases, we consider only jobs that do not have any of the properties, those are jobs of size more than $2/9$ and hence OPT can run at most four such jobs on one machine. We call those jobs "large jobs". If there exists a job of size more than $1/3$, OPT can accommodate only three large jobs on the machine running it. If a job of size more than $5/9$ exists, the maximum number of large jobs on its machine is two (for OPT), and if a job of size more than $7/9$ exists, no other large jobs are assigned to the same machine by OPT. In the case where it does not directly follow from the above, we later show directly why there cannot be more large jobs than the number of jobs considered in each one of the cases.

1. $p_1 > 7/9$. Assign the jobs J_2, J_3, J_4 and J_5 to M_2. If the first five jobs do not have any of the two properties, J_6 must have property 2 as shown later.

2. $p_2 \leq 1/3$ and $5/9 < p_1 \leq 7/9$. There are at most six large jobs. Assign J_2, J_3 and J_4 to M_2, J_5 to M_1 and J_6 to M_2.

3. $p_2 > 1/3$ and $5/9 < p_1 \leq 7/9$. Assign the first job J_k such that $p_1 + p_k \leq 10/9$ to M_1, and all other jobs $(J_{k+1}, \ldots, J_6)$ to M_2 (clearly $k > 2$ otherwise $8/9 < p_1 + p_2 \leq 10/9$).

4. $4/9 < p_1 \leq 5/9$ and $p_2 \leq 1/3$. In this case there are at most seven large jobs. Assign J_2 and J_3 to M_2, J_4 to M_1, J_5 to M_2, J_6 to M_1 and J_7 to M_2.

5. $p_2 > 1/3$, $7/9 < p_1 + p_2 \leq 8/9$ and $p_1 - p_2 \leq 1/9$. Note that $p_2 \leq 4/9$ and $p_1 \leq 5/9$.

There are at most six large jobs. Assign J_2 to M_2, J_3 to M_1, J_4 to M_2, J_5 to M_2 and J_6 to M_1.

6. $p_2 > 1/3$ and $p_1 > 1/9 + p_2 > 4/9$, and $p_1 + p_2 < 8/9$. It can be easily seen that $p_1 < 5/9$, and $p_2 < 7/18$. Also $p_1 + p_2 > 7/9$. There can be at most six large jobs. Assign J_2 and J_3 to M_2, J_4 to M_1, J_5 to M_2 and J_6 to M_1.

7. $p_1 + p_2 \leq 7/9$, and $p_2 > 1/3$. Clearly $p_1 \leq 4/9$. There can be at most seven large jobs. Assign J_2 to M_1, J_3 and J_4 to M_2, J_5 to M_1, J_6 and J_7 to M_2.

8. $p_1 \leq 4/9$ and $p_2 \leq 1/3$. There can be at most eight jobs that do not have a property. Assign J_2 to M_1, J_3 and J_4 and J_5 to M_2, J_6 to M_1, J_7 to M_2 and J_8 to M_1.

We now show that all possible cases are covered. If there is an overlap between cases, the jobs are assigned by the case with the smaller index, hence we need to show that all cases are covered at least once. Case 1 covers $p_1 > 7/9$ and cases 2 and 3 cover $5/9 < p_1 \leq 7/9$. It is left to consider $2/9 < p_2 \leq p_1 \leq 5/9$. The cases $p_2 \leq 1/3$ is covered by cases 4,8. We are left with $1/3 < p_2 \leq p_1 \leq 5/9$. Those cases are dealt with according to the sum of the first two jobs. Clearly $p_1 + p_2 \leq 10/9$ (otherwise $p_1 > 5/9$). If $p_1 + p_2 \geq 8/9$ holds then property 1 holds and we are done. The case $p_1 + p_2 \leq 7/9$ is covered by case 7 and $7/9 < p_1 + p_2 < 8/9$ is covered by cases 5 and 6.

Lemma 2.2 *The competitive ratio of algorithm SIZES is* $\frac{10}{9}$.

Proof. If property 1 is valid for machine i, and all other jobs are assigned to machine $3-i$, then machine $3-i$, no matter how many jobs are assigned to it, cannot have load that exceeds $2 - P_i \leq 10/9$. Note that once property 2 is valid, all future jobs have size of at most $2/9$. If property 2 is valid at a time when no machine exceeds $10/9$, we can show that all future jobs are assigned without exceeding $10/9$. Let j be a new job. Let P^* be the total size of other jobs. Clearly $P^* \leq 2 - p_j$. The less loaded machine ℓ has $P_\ell \leq P^*/2 \leq (2-p_j)/2$. Assigning j to it gives $P_\ell \leq 1 + p_j/2 \leq 1 + 1/9 = 10/9$. To show that the basic algorithm never lets any machine exceed $10/9$, we prove each case separately.

Case 1. There can be at most five large jobs. This is true since OPT assigns J_1 to one machine and all other large jobs to the other. Hence $p_2 + p_3 + p_4 + p_5 \leq OPT = 1$.

Case 2. As long as there are at most four jobs, $SONL \leq 1$ since $p_4 \leq p_3 \leq p_2 \leq 1/3$. After five jobs $SONL = p_1 + p_5 \leq p_1 + p_2 \leq 10/9$. After six jobs $SONL = p_2 + p_3 + p_4 + p_6$. OPT assigns some four jobs on one machine hence $p_6 + p_5 + p_4 + p_3 \leq OPT = 1$. We get $SONL \leq 1 + p_2 - p_5 \leq 1 + 1/3 - 2/9 = 10/9$ (since p_5 is large).

Case 3. Note that in this case $p_1 + p_2 > 8/9$, but property 1 does not hold and so $p_1 + p_2 > 10/9$. If OPT runs J_1 and J_2 on the same machine, then this machine has no other large jobs and there can be at most six jobs in this case. If OPT runs J_1 and J_2 on different machines, there can be at most three extra large jobs. Clearly $k \leq 5$, since $p_5 \leq 1/3$ (some machine has at least three jobs) and $p_1 \leq 7/9$. For all jobs J_i, $2 \leq i < k$, OPT does not run J_i together with J_1 and so $\sum_{i=2}^{k-1} p_i \leq 1$. We get that before the arrival of J_k, both machines do not exceed the load 1. When J_k is assigned, if OPT assigns J_1 without other large jobs then all jobs assigned to M_2 run on one machine of OPT and $P_2 \leq OPT = 1$. Otherwise let ℓ be the number of large jobs, and let J_i be the large job that OPT runs with J_1 on the same machine. Then $P_2 \leq p_2 + \ldots + p_\ell - p_k$. On the other hand $1 = OPT \geq p_2 + \ldots + p_\ell - p_i$. Note that OPT cannot run J_1 with any of the jobs $J_2 \ldots, J_{k-1}$ and so $i \geq k$ and $p_i \leq p_k$. We conclude that $P_2 \leq 1$.

Case 4. As long as there are at most four jobs, and since $p_1 \leq 5/9$, no machine exceeds $10/9$. Given $p_2 \leq 1/3$ we get that $p_2 + p_3 + p_5 \leq 3p_2 \leq 1$. After six jobs we consider two cases. If OPT has a machine with four jobs, $p_3 + p_4 + p_5 + p_6 \leq OPT = 1$. Using the fact $p_1 \leq 5/9$ and that J_3 and J_5 are large we get $p_1 + p_4 + p_6 \leq (p_1 - p_3 - p_5) + 1 \leq 10/9$. Otherwise OPT runs J_1 with at least two jobs and $p_1 + p_5 + p_6 \leq OPT = 1$. Both p_4 and p_5 are in the interval $[2/9, 1/3]$ and so $p_4 - p_5 \leq 1/9$ which gives $p_1 + p_4 + p_6 \leq 10/9$.

After seven jobs, there are two cases. Clearly OPT runs four jobs on one machine and three on the other. If OPT runs all jobs among J_1,J_2 and J_3 on one machine we get $p_1 + p_2 + p_3 \leq OPT = 1$ and $2(p_5 + p_7) \leq p_4 + p_5 + p_6 + p_7 \leq OPT = 1$. Therefore $p_2 + p_3 + p_5 + p_7 \leq 1 - p_1 + p_5 + p_7 \leq 3/2 - p_1 \leq 3/2 - 4/9 = 19/18 \leq 10/9$. Otherwise $1 = OPT \geq p_3 + p_7 + p_6 + p_5$. As $p_2 - p_6 \leq 1/3 - 2/9 = 1/9$ we get $p_2 + p_3 + p_5 + p_7 \leq 10/9$.

Case 5. In this case there can be at most six jobs, no matter whether J_1 and J_2 are run on the same machine by OPT or not. As long as there are at most four jobs, and since $p_1 \leq 5/9$, no machine exceeds $10/9$.

After five jobs, $OPT \geq p_3 + p_4 + p_5 \geq 3(p_4 + p_5)/2$. Using $p_2 \leq 4/9$ we get that $p_2 + p_4 + p_5 \leq 4/9 + 2/3 \leq 10/9$. After six jobs there are two cases. If OPT runs the first two jobs on one machine then it has four jobs on the other. Otherwise, there can be two additional jobs on each machine. In the first case $OPT \geq p_3 + p_4 + p_5 + p_6$. Since all jobs are large, $p_1 - p_4 - p_5 \leq 5/9 - 4/9 = 1/9$. We conclude that $p_1 + p_3 + p_6 \leq 10/9$. Otherwise OPT has a machine where it runs at least two of J_1,J_2 and J_3, hence $1 = OPT \geq p_2 + p_3 + p_6$. Using $p_1 \leq 1/9 + p_2$ we get $p_1 + p_3 + p_6 \leq 1 + (p_1 - p_2) \leq 10/9$.

Case 6. Using $p_1 \leq 5/9$, we get that assigning two jobs to a machine does not let its load exceed $10/9$ by more than $1/9$. When five jobs exist, $1 = OPT \geq p_5 + p_4 + p_3$. Hence $p_5 \leq 1/3$. We get $p_2 + p_3 + p_5 \leq 2p_2 + p_5 \leq 2 \cdot 7/18 + 1/3 = 10/9$. When six jobs exist, there are several cases. Note that (as in case 5) if J_1 and J_2 are on the same machine by OPT, then the second machine may have up to four jobs. If each of them runs on a different machine, then each machine may have two additional jobs. In the first case $1 = OPT \geq p_3 + p_4 + p_5 + p_6$. All jobs are large, hence $p_3 + p_5 > 4/9$. This gives $p_1 - p_3 - p_5 < 1/9$. We can conclude that $p_1 + p_4 + p_6 \leq 10/9$. If J_2 runs with both J_3 and J_4 then $1 = OPT = p_2 + p_3 + p_4$ and $p_4 \leq 1/3$. Then J_1 runs with J_5 and J_6 and $1 = OPT = p_1 + p_5 + p_6 \geq 1 + 2p_6$ which gives

$p_6 \leq (1-p_1)/2$. We get $p_1 + p_4 + p_6 \leq 1/2 + 1/3 + p_1/2 \leq 9/18 + 6/18 + 5/18 = 10/9$. In the last case J_1 runs with either J_3 or J_4, which gives $1 = OPT \geq p_1 + p_4 + p_6$. In this case $P_1 \leq 1$.

Case 7. If OPT runs the first two jobs on different machines, then there can be at most six large jobs. If they run on the same machine, that machine may have one more job, which gives a total of seven large jobs. As long as there are at most four jobs, no load exceeds 8/9. When five jobs exist, $1 = OPT \geq p_5 + p_4 + p_3$. Hence $p_5 \leq 1/3$. We get $SONL = p_1 + p_2 + p_5 \leq 7/9 + 1/3 = 10/9$. After six jobs clearly $p_3 + p_4 + p_6 \leq p_1 + p_2 + p_5 \leq 10/9$. After seven jobs let J_i be the job which runs with J_1 and J_2. Then $p_i < 1/3$ and $1 = OPT \geq p_7 + p_6 + p_5 + p_4 + p_3 - p_i$. In this case $P_2 \leq p_3 + p_4 + p_6 + p_7 \leq 1 + p_i - p_5 \leq 1 + 1/3 - 2/9 \leq 10/9$ (since p_5 is large).

Case 8. All jobs except J_1 have size of at most 1/3, hence the first six jobs do not violate the competitive ratio. When seven jobs exist, $1 = OPT \geq p_7 + p_6 + p_5 + p_4$. Using $p_3 \leq 1/3$ and $p_6 > 2/9$ we get $p_3 + p_4 + p_5 + p_7 \leq 1 + p_3 - p_6 \leq 10/9$. When eight jobs exist, OPT assigns J_1 with three other jobs, and $1 = OPT \geq p_1 + p_8 + p_7 + p_6$. Similarly, $p_2 - p_7 \leq 1/3 - 2/9 = 1/9$, and $p_1 + p_2 + p_6 + p_8 \leq 10/9$. $\square$

3 An optimal preemptive algorithm for two related machines

In this section we show that an optimal algorithm for the preemptive problem can be given already in an on-line environment. We prove the following theorem:

Theorem 3.1 *There exists a preemptive deterministic on-line algorithm for two related machines with competitive ratio 1.*

We need to construct an algorithm which assigns each job to a set of intervals, such that for each machine, the intervals assigned there are contained in the interval $[0, 1]$. Each point in the intervals $[0, 1]$ on each one of the two machines (except end-points of intervals assigned to jobs) should be dedicated to a single job. Moreover, two intervals assigned to a job j (on the same machine or on different machines) cannot be overlapping. Let L_q and L_1 be the combined length of all intervals assigned to job j on the machines M_q and M_1 respectively. Then in order to make it possible to complete the job we need $L_q/q + L_1 \geq p_j$.

The following algorithm "Perfect" assigns a job j in a way that it receives at most one interval on each machine and that $L_q/q + L_1 = p_j$. Each machine has a continuous non-occupied interval at every time. The intervals I_q and I_1 denote the free intervals of machines M_q and M_1 respectively. The algorithm keeps a state as a parameter which is the phase it is in. It starts at phase 1, and increases the index of the state it is in from time to time. All jobs are assigned as defined for the current state until the algorithm moves to the next state. Roughly speaking, the algorithm starts with assigning each job to the slow machine only. As soon as this is not possible, it splits the job in a way that either the slow machine becomes completely occupied, or the free intervals left on the two machines are disjoint. From that point any legal assignment of each job is fine, since no overlaps between intervals on different machines can happen. For completeness, we define the assignment after that point as well.

Algorithm PERFECT:

Initialization: $I_1 = [0, 1]$, and $I_q = [0, 1]$. Start at phase 1.

phase 1: During this phase I_1 does not change and I_q always keeps the shape $[0, a]$. In the beginning of the phase $a = 1$. Let j be an arriving job. If $qp_j \leq a$, assign it to the interval $[a - qp_j, a]$ on M_q and let $a = a - qp_j$. Otherwise go to phase 2 and assign j by the rules of that phase.

phase 2: In the beginning of this phase $I_1 = [0, 1]$, and $I_q = [0, a]$ and $qp_j > a$. This phase processes only one job and moves to phase 3 or 5. The phase results in the shapes $I_1 = [0, b']$ $I_q = [b, a]$ of the intervals. The values of a, b and b' are such that $b' \leq b$ (and possibly $a = b$ i.e. I_q may be empty). If $a/q + (1 - a) \geq p_j$, assign it to the intervals $[0, a]$ on M_q and $[1 - (p_j - a/q), 1]$ on M_1. In this case I_q becomes empty and $I_1 = [0, 1 - (p_j - a/q)]$. Move to phase 5. Otherwise let $\beta = q(1 - p_j)/(q - 1)$. Assign the job into the intervals $[0, \beta]$ on M_q and $[\beta, 1]$ on M_1. The new free intervals are $I_1 = [0, \beta]$ and $I_q = [\beta, a]$. Move to phase 3.

phase 3: As mentioned above, the phase starts with the shapes $I_1 = [0, b]$ and $I_q = [b, a]$. The phase also ends with the same shapes. If $qp_j \leq a - b$, assign it to the interval $[a - qp_j, a]$ on M_q, and $a = a - qp_j$. Otherwise go to phase 4.

phase 4: In the beginning of this phase $I_1 = [0, b]$ $I_q = [b, a]$ and $qp_j > a - b$. This phase processes only one job and moves to phase 5. The phase results in the shape $[0, d]$ for M_1 whereas the interval of M_q becomes empty. Let $p'_j = p_j - (a - b)/q$. Assign j to $[b - p'_j, b]$ on M_1 and to $[b, a]$ on M_q. $I_1 = [0, b - p'_j]$. Move to phase 5.

phase 5: This phase starts with $I_1 = [0, d]$ and empty I_q. Assign job j to $[d - p_j, d]$ on M_1 and $d = d - p_j$. This phase ends when no more jobs arrive.

Lemma 3.1 *The above algorithm has competitive ratio 1.*

Proof. : Note that $OPT = 1$, which means that the maximum size of a job is 1 and also the total size of all jobs is at most $1 + 1/q$. The intervals can accommodate exactly this size. We show that a job never gets too much room and hence there is always enough room for all jobs in the intervals.

However, it is not allowed to assign jobs into overlapping sub-intervals. It is required to show that each job received the correct amount of space, and that in each phase the intervals defined for the job do not overlap. It is not difficult to see that after each assignment the intervals indeed have the claimed properties. For most cases it is easy to see that the intervals assigned to each job are indeed free at that point, and that the job receives enough space. We clarify those points in the cases where it is less straightforward.

In phase 1 a job j gets an interval of length qp_j on M_q. In phase 2. In the first case $a/q + (1 - a) \geq p_j$ holds. The lengths of intervals assigned to it are $L_q = a$ and $L_1 = p_j - a/q$, we get $L_q/q + L_1 = p_j$. The intervals are disjoint since $1 - (p_j - a/q) \geq a$. In the other case the intervals are clearly disjoint and $L_q/q + L_1 = 1 - \beta + \beta/q = p_j$ by the definition of β. In phase 3, $a - qp_j \geq b$ holds and hence the interval is valid. In phase 4, we know that j can fit into I_1 and I_q. The value of P'_j is positive since $qp_j > a - b$. In phase 5, job j gets an interval of length p_j on M_1.

4 Non preemptive bin stretching on two related machines

In this section we give the results for non-preemptive on-line bin stretching. We start with the algorithms and later prove lower bounds.

4.1 Non-preemptive algorithms

In this section we introduce two algorithms SLOW and FAST. SLOW is valid for $q \geq \sqrt{2}$ and FAST is valid for $q \leq \sqrt{2}$. Some of the ideas of the two algorithms are similar. SLOW works for the cases where the slow machine is relatively slow, and starts with assigning jobs to the slow machine. FAST works for the cases where the slow machine is relatively fast, and starts with assigning jobs to the faster machine. In general, it gives better results to fill the fast machine first. However, recall that the largest job in the sequence has size at most 1 (since $OPT = 1$). For large values of q, an assignment of such a job to the slow machine violates the competitive ratio right away. Hence we need to keep the fast machine relatively free, in order to be able to receive such jobs. Later we state the intervals where we can show that those are algorithms of tight competitive ratio (by matching lower bounds) and give very close (but not tight) lower bounds for the other intervals.

We show that the competitive ratio of SLOW is $r_s(q) = (q+2)/(q+1)$ and the competitive ratio of FAST is $r_f(q) = \max\{q, (2q+2)/(2q+1)\}$.

The definition of the algorithm **SLOW** is as follows:
Initialization: $P_1 \leftarrow 0$ and $P_q \leftarrow 0$.
On arrival of a job j with size p_j the algorithm does the following.
1. If $P_q \geq 1/(q^2+q)$ assign j to M_1 and $P_1 \leftarrow P_1 + p_j$.
2. Else if $P_q + p_j \leq r_s(q)/q$ assign j to M_q and $P_q \leftarrow P_q + p_j$.
3. Else assign j to M_1 and $P_1 \leftarrow P_1 + p_j$.

Note that if case 3 happens, then both $P_q < 1/(q^2+q)$ and $P_q + p_j > r_s(q)/q$ hold and therefore $p_j > r_s(q)/q - P_q > (q+2)/(q^2+q) - 1/(q^2+q) = 1/q$.

Lemma 4.1 *The competitive ratio of algorithm SLOW is* $r_s(q) = \frac{q+2}{q+1}$.

Proof. By the definition of the algorithm, M_q never exceeds $r_s(q)/q$ and so the slow machine can never violate the competitive ratio .

We need to show that the same holds for M_1. We do that by showing that each job, that is assigned to M_1, does not let P_1 exceed $r_s(q)$. If a job is assigned to M_1 by rule 1, then this job and all future jobs are legally assigned, since $M_1 \leq 1+1/q-M_q \leq 1+1/q-1/(q^2+q) = (q^2+q+q+1-1)/(q^2+q) = (q+2)/(q+1)$. (Using $P_q \geq 1/(q^2+q)$ and the fact that the total size of all jobs is at most $1+1/q$.)

The only case left to consider is the case where M_1 receives a job by rule 3, and never got any jobs by rule 1. All the jobs assigned to M_1 are of size more than $1/q$. Since $OPT = 1$, the optimal offline algorithm runs on the show machine only jobs of size at most $1/q$. Consequently, the sum of jobs larger than $1/q$ is at most 1, and hence in this case $M_1 \leq 1$. $\square$

The definition of the algorithm **FAST** is as follows:

Initialization: $P_1 \leftarrow 0$ and $P_q \leftarrow 0$.

On arrival of a job j with size p_j the algorithm does the following.

0. If a job was ever assigned to M_q by rule 3, assign j to M_1 and $P_1 \leftarrow P_1 + p_j$.

1. If $P_1 \geq 1+1/q-r_f(q)/q$, assign j to M_q and $P_q \leftarrow P_q + p_j$.

2. Else if $P_1 + p_j \leq r_f(q)$, assign j to M_1 and $P_1 \leftarrow P_1 + p_j$.

3. Else Assign j to M_q and $P_q \leftarrow P_q + p_j$.

Note that if case 3 happens then both $P_1 < 1+1/q-r_f(q)/q$ and $P_1 + p_j > r_f(q)$ hold and then $p_j > r_f(q) - P_1 > r_f(q) - 1 - 1/q + r_f(q)/q = (1+1/q)(r_f(q)-1)$.

Lemma 4.2 *The competitive ratio of algorithm FAST is $r_f(q) = \max\{q, (2q+2)/(2q+1)\}$, i.e. $\frac{2q+2}{2q+1}$ for $1 \leq q \leq \frac{1+\sqrt{17}}{4}$ and q for $\frac{1+\sqrt{17}}{4} \leq q \leq \sqrt{2}$.*

Proof. We discuss the case where rule 0 is never invoked first. By the definition of the algorithm, M_q never exceeds $r_f(q)$ and so the fast machine can never violate the competitive ratio.

To show that the same holds for M_q, we show that each job that is assigned to M_q does not let P_q exceed $r_f(q)/q$. If a job is assigned to M_q by rule 1, then this job and all future jobs are legally assigned since $M_q \leq 1+1/q-M_q \leq 1+1/q-1-1/q+r_f(q)/q = r_f(q)/q$.

If M_q receives a job by rule 3, and never got any jobs by rule 1, it has a single job (because after this happens, rule 0 is used for all jobs) and so $P_q \leq 1$, and the competitive ratio is at most q.

We are left with the case where rule 0 is invoked, that happens after rule 3 was invoked. We need to show that rule 0 does not violate the competitive ratio for the fast machine. After rule 3 was invoked, $P_q \geq (1+1/q)(r_f(q)-1)$. The load of M_1 is bounded by $P_1 \leq 1+1/q-P_q \leq (1+1/q)(2-r_f(q))$. Using $r_f(q) \geq (2q+2)/(2q+1)$ we get $P_1 \leq (1+1/q)\cdot(2q)/(2q+1) = (2q+2)/(2q+1) \leq r_f(q)$.

$\square$

4.2 Lower bounds for non-preemptive algorithms

In this section, we prove lower bounds on the competitive ratio.

We start with stating the intervals where SLOW and FAST have optimal competitive ratio.

SLOW has an optimal competitive ratio for $q \geq \sqrt{3}$. Specifically, we prove the following lemma:

Lemma 4.3 *Any deterministic online algorithm for bin stretching on related machines with speed ratio $q \geq \sqrt{3}$ has competitive ratio of at least $\frac{q+2}{q+1}$.*

Proof. Note that for $q \geq \sqrt{2}$, the value of the required competitive ratio satisfies $(q+2)/(q+1) \leq q$ or $(q+2)/(q(1+q)) \leq 1$. The sequence starts with at most $\lceil q \rceil$ jobs of size $1/(q(1+q))$. Such jobs arrive until either one of them is assigned to the slow machine, or until $\lceil q \rceil$ such jobs arrived. Let k be the number of such jobs, then $p_1 = \ldots = p_k = 1/(q^2+q)$.

If no jobs were assigned to M_q then $k = \lceil q \rceil$. In this case the sequence continues with one job of size $p_{k+1} = 1$. If J_k is assigned to M_q then $P_q = 1 \geq (q+2)/(q(1+q))$. Otherwise $P_1 \geq 1 + \lceil q \rceil/(q(1+q)) \geq 1 + 1/(1+q) = (q+2)/(1+q)$. OPT assigns all jobs but J_{k+1} to M_q, which is possible since $p_{k+1} = 1$ and $\lceil q \rceil/(q(1+q)) \leq (1+q)/(q(1+q)) = 1/q$.

Otherwise J_k is assigned to M_q, and the sequence continues with $J_{k+1} = 1/q$ and $J_{k+2} = 1-k/(q(1+q))$. OPT assigns J_{k+1} to M_q and since $\sum_{\ell=1}^{k} p_\ell + p_{k+2} = 1-k/(q(1+q))+k/(q(1+q)) = 1$, all other jobs can be assigned to M_1 together. If the algorithm assigns J_{k+1} to the slow machine we get $P_q \geq 1/(q(1+q))+1/q = (q+2)/(q(1+q))$, which gives competitive ratio of at least $(q+2)/(q+1)$. Otherwise, if the algorithm assigns J_{k+2} to the slow machine we get $P_q \geq 1-(k-1)/(q(1+q))$. For $q \geq 2$ we use $k/(q(1+q)) \leq 1/q$ and then $q^2 \geq q+2$ and get $P_q \geq q/(1+q) \geq (q+2)/(q(1+q))$. For $\sqrt{3} \leq q < 2$ we use $k = 2$ and $q^2+q-1 \geq q+2$ and get $P_q \geq (q^2+q-1)/(q(1+q)) \geq (q+2)/(q(1+q))$, which gives the competitive ratio . Otherwise, all jobs but J_k are assigned to M_1 and so $P_1 = (k-1)/(q(1+q)) + 1/q + 1 - k/(q(1+q)) = 1 + 1/q - 1/(q(1+q)) = (q^2+2q)/(q(1+q)) = (q+2)/(q+1)$, which completes the proof. $\square$

FAST has optimal competitive ratio for the interval $(1+\sqrt{65})/8 \approx 1.13278 \leq q \leq (1+\sqrt{3})/2 \approx 1.36602$. To show that we consider the three intervals: $[(1+\sqrt{65})/8, (1+\sqrt{13})/4 \approx 1.15139]$, $[(1+\sqrt{13})/4, (1+\sqrt{17})/4 \approx 1.28078]$ and $[(1+\sqrt{17})/4, (1+\sqrt{3})/2]$, and give a matching lower bound in each one of them.

Lemma 4.4 *Any deterministic online algorithm for bin stretching on related machines with speed ratio $(1+\sqrt{17})/4 \leq q \leq (1+\sqrt{3})/2$ has competitive ratio of at least q.*

Proof. The first job has size $p_1 = q-1$. If it is assigned to M_1 then the second job J'_2 has size $p'_2 = 1$. Otherwise there are two additional jobs of sizes $p_2 = 1/q$ and $p_3 = 2-q$ ($p_3 \geq 0$ as $q \leq 2$).

In the first case, OPT assigns J_1 to M_q and J'_2 to M_1. This is possible, as $q-1 \leq 1/q$ for $q \leq \phi$. If the algorithm assigns J'_2 to M_1, its load is q which gives the required competitive ratio. If the algorithm assigns J'_2 to M_q, its load is 1 and we get the competitive ratio as well.

In the second case, OPT assigns J_2 to M_q and J_1 and J_3 to M_1. Note that for each positive q, $q + 1/q \geq 2$ holds, and hence $p_3 \leq p_2$. If at least one additional job is assigned to M_q we get $P_q \geq (q-1)+(2-q) = 1$ which gives the competitive ratio as in the previous case. If both last jobs are assigned to M_1, $P_1 = 2 - q + 1/q$. For $q \leq (1+\sqrt{3})/2$ we have $2q - q^2 + 1 \geq q^2$ and hence $P_1 \geq q$. This completes the proof. $\Box$

Lemma 4.5 *Any deterministic online algorithm for bin stretching on related machines with speed ratio* $(1+\sqrt{13})/4 \leq q \leq (1+\sqrt{17})/4$ *has competitive ratio of at least* $\frac{2q+2}{2q+1}$.

Proof. In this sequence we use jobs of the following sizes: $a_1 = 1/(2q+1)$, $a_2 = (2q^2 - q - 1)/(2q^2+q)$, $a_3 = (q+1)/(2q^2+q)$, $a_4 = 1$, $a_5 = 2a_1$, $a_6 = 1 - a_1$, $a_7 = 1/q - a_2$ and $a_8 = 1/(2q^2+q)$. Note that $a_2 \geq 0$ for all $q \geq 1$.

The sequence starts with two jobs of sizes $p_1 = a_1$ and $p_2 = a_2$.

Case 1: If both jobs are assigned to M_q, continue with two jobs of sizes a_6 and a_7.
Case 2: If both jobs are assigned to M_1, continue with another job of size a_1.
Subcase 2.1: If the last job is assigned to M_1, continue with two jobs of size a_3 each.
Subcase 2.2: If the last job is assigned to M_q, continue with a job of size a_4.
Case 3: If J_1 was assigned to M_q and J_2 to M_1, continue with a job of size a_8.
Subcase 3.1: If the last job is assigned to M_q, continue with two jobs of sizes a_3 and a_5.
Subcase 3.2: If the last job is assigned to M_1, continue with a job of size a_4.
Case 4: If J_1 was assigned to M_1 and J_2 to M_q, continue with a job of size a_4.

We analyze OPT for all cases. If a job of size $a_4 = 1$ arrives, then OPT runs this job on M_1 and all other jobs on M_q. This is relevant for cases 2.2, 3.2 and 4. In those cases the load on M_q for OPT is $a_1 + a_2 + a_1$, $a_1 + a_2 + a_8$ and $a_1 + a_2$ respectively. Using $a_1 \geq a_8$, (which holds for all $q \geq 1$) we need to show that $a_1 + a_2 + a_1 \leq 1/q$, or that $2/(2q+1) + (2q^2 - q - 1)/(q(1+2q)) \leq 1/q$. Simplifying we get $2q^2 - q - 2 \leq 0$ which holds in the current interval. In case 1, as $a_1 + a_6 = 1$ and $a_2 + a_7 = 1/q$, OPT can assign the two jobs of sizes a_1 and a_6 to M_1, and the others to M_q. In case 2.1, OPT assigns one job of size a_3 with one job of size a_1 to M_q, since $a_1 + a_3 = 1/q$, and the other jobs are assigned to M_1. Their total size is $a_1 + a_2 + a_3 = ((q+1) + q + (2q^2 - q - 1))/(2q^2+q) = 1$. In case 3.1 OPT also assigns one job of size a_3 with one job of size a_1 to M_q. The rest of the jobs can be assigned to M_1 since $a_2 + a_5 + a_8 = ((2q^2 - q - 1) + 2q + 1)/(2q^2+q) = 1$.

To show that the competitive ratio is achieved we do the following. In cases 2.2, 3.2 and 4, we need to show that if M_q receives the last job its load becomes at least $(2q+2)/(2q^2+q)$, and if M_1 receives the last job its load becomes at least $(2q+2)/(2q+1)$.

In case 4 this means $a_1 + a_4 \geq (2q+2)/(2q+1)$ which holds with equality, and $a_2 + a_4 \geq (2q+2)/(2q^2+q)$ which gives $4q^2 - 2q - 3 \geq 0$. The last inequality holds for $q \geq (1+\sqrt{13})/4$.

In case 2.2 this means $a_1 + a_2 + a_4 \geq (2q+2)/(2q+1)$ and $a_1 + a_4 \geq (2q+2)/(2q^2+q)$. The first inequality holds since $a_2 \geq 0$. The second inequality gives $2q^2 \geq 2$ which holds for all $q \geq 1$.

In case 3.2 this means $a_2 + a_8 + a_4 \geq (2q+2)/(2q+1)$ and $a_1 + a_4 \geq (2q+2)/(2q^2+q)$. The first inequality gives $2q^2 \geq 2q$, which holds for all $q \geq 1$. The second inequality is the same as in case 2.2.

In case 2.1 we would like to show that in both options, M_q receives both last jobs, and M_1 receives at least one last job, the competitive ratio is achieved. This means that $2a_1+a_2+a_3 \geq (2q+2)/(2q+1)$ and $2a_3 \geq (2q+2)/(2q^2+q)$. Both inequalities hold with equality.

In cases 1 and 3.1, we would like to show that in both cases, M_q receives one of the last jobs, and M_1 receives both last jobs, the competitive ratio is achieved.

In case 1 this means that $a_1+a_2+a_6 \geq (2q+2)/(2q^2+q)$, $a_1+a_2+a_7 \geq (2q+2)/(2q^2+q)$, and $a_6 + a_7 \geq (2q+2)/(2q+1)$. The first inequality gives $4q^2 - 2q - 3 \geq 0$, which holds for $q \geq (1+\sqrt{13})/4$. The second inequality simply gives $q \geq 1$. The third inequality gives $2q^2 - q - 2 \leq 0$, which holds for $q \leq (1+\sqrt{17})/4$.

In case 3.1 this means that $a_1+a_8+a_3 \geq (2q+2)/(2q^2+q)$, $a_1+a_8+a_5 \geq (2q+2)/(2q^2+q)$, and $a_2 + a_3 + a_5 \geq (2q+2)/(2q+1)$. The first and third inequalities hold with equality and the second inequality simply gives $q \geq 1$. $\square$

Lemma 4.6 *Any deterministic online algorithm for bin stretching on related machines with speed ratio* $(1+\sqrt{65})/8 \leq q \leq (1+\sqrt{13})/4$ *has competitive ratio of at least* $\frac{2q+2}{2q+1}$.

Proof. In this sequence we use jobs of the following sizes: $b_1 = (2q^2-2)/(2q^2+q)$, $b_2 = (-3q^2+q+4)/(2q^2+q)$, $b_3 = 1/(2q^2+q)$, $b_4 = 1/(2q+1)$, $b_5 = 1$, $b_6 = (q+1)/(2q^2+q)$, and $b_7 = (q+1)/(2q+1)$. Note that $b_2 \geq 0$ for $q \leq 4/3$. The sequence starts with a job of size $p_1 = b_1$.

Case 1: If J_1 is assigned to M_q, continue with a job of size b_2.

Subcase 1.1: If the last job is assigned to M_1, continue with a job of size b_5.

Subcase 1.2: If the last job is assigned to M_q, continue with two jobs of sizes b_7 each.

Case 2: If J_1 is assigned to M_1, continue with two jobs of sizes b_3 and b_4.

Subcase 2.1: If the last two jobs are assigned to the same machine, continue with two jobs of size b_6 each.

Subcase 2.2: If the last two jobs are assigned to different machines, continue with a job of size b_5.

We analyze OPT for all cases. If a job of size 1 arrives, then OPT runs this job on M_1 and all other jobs on M_q. This is relevant for cases 1.1 and 2.2. In those cases the load on M_q for OPT is $b_1 + b_2$ and $b_1 + b_3 + b_4$ respectively. The inequality $b_1 + b_2 \leq 1/q$ gives $q^2 + q - 1 \geq 0$ which holds for $q \geq 1$. The inequality $b_1 + b_3 + b_4 \leq 1/q$ gives $2q^2 - q - 2 \leq 0$, which holds for $q \leq (1+\sqrt{17})/4$ and throughout the interval.

In case 1.2, OPT assigns a job of size b_2 with a job of size b_7 to M_1, and the other jobs (of sizes b_1 and b_7) to M_q. We need to show $b_2 + b_7 \leq 1$ and $b_1 + b_7 \leq 1/q$. The first gives $4q^2 - q - 4 \geq 0$, and the second gives $3q^2 - q - 3 \leq 0$. To make both true, q needs to satisfy $(1+\sqrt{65})/8 \leq q \leq (1+\sqrt{37})/6 \approx 1.18046$, which is true in the current interval. In case 2.1 OPT assigns one job of size b_4 with one job of size b_6 to M_q, their sum is $1/q$. The rest of the jobs (of sizes b_1, b_3 and b_6) can be assigned to M_1 since $b_1 + b_3 + b_6 = ((2q^2) + 1 + (q+1))/(2q^2+q) = 1$.

To show that the competitive ratio is achieved we do the following. In cases 1.1 and 2.2, we need to show that if M_q receives the last job its load becomes at least $(2q+2)/(2q^2+q)$, and if M_1 receives the last job its load becomes at least $(2q+2)/(2q+1)$.

In case 1.1 this means $b_2 + b_5 \geq (2q+2)/(2q+1)$ and $b_1 + b_5 \geq (2q+2)/(2q^2+q)$. The first inequality gives $3q^2 \leq 4$, which holds for all $q \leq \sqrt{(4/3)} \approx 1.15470$. Using $(1+\sqrt{13})/4 \leq \sqrt{(4/3)}$ we get that it holds in the current interval. The second inequality gives $4q^2-q-4 \geq 0$, which holds for $q \geq (1+\sqrt{65})/8$.

In case 2.2, since $b_3 \leq b_4$, we need to show $b_1 + b_3 + b_5 \geq (2q+2)/(2q+1)$ and $b_3 + b_5 \geq (2q+2)/(2q^2+q)$. Both inequalities give $2q^2 - q - 1 \geq 0$ which holds for all $q \geq 1$.

In case 1.2, we would like to show that in both cases, M_q receives one of the last jobs, and M_1 receives both last jobs, the competitive ratio is achieved. This means that $b_1 + b_2 + b_7 \geq (2q+2)/(2q^2+q)$, $2b_7 \geq (2q+2)/(2q+1)$. Both hold with equality.

In case 2.1 we can see the two jobs of sizes b_3 and b_4 as one job of size b_6 since the algorithm assigned them to one machine. For the algorithm (but not for OPT) the sequence consists of one job of size b_1 and three jobs of size b_6. We would like to show that in both cases, M_q receives at least two jobs of size b_6, and M_1 receives at least two such jobs, the competitive ratio is achieved. This means that $a_1 + 2a_6 \geq (2q+2)/(2q+1)$ and $2a_6 \geq (2q+2)/(2q^2+q)$. Both inequalities holds with equality.

□

We are left with the intervals $(1, (1+\sqrt{65})/8)$ and $((1+\sqrt{3})/2, \sqrt{3})$. The next four lemmas deal with those cases, and give relatively high lower bounds. We discuss the gap between the upper bound and the lower bound in each case.

Lemma 4.7 *Any deterministic online algorithm for bin stretching on related machines with speed ratio* $1 \leq q \leq q_1$ *(where* $q_1 \approx 1.12433$ *is a solution of* $36x^4 - 135x^3 + 45x^2 + 60x + 10 = 0$*) has competitive ratio of at least* $1 + \frac{1}{3q}$.

The difference function between the upper bound and the lower bound is $(q-1)/(6q^2+3q)$ which is monotonically increasing in this interval. Hence the largest gap occurs at q_1 and is ≈ 0.011346.

Proof. The sequence starts with a job of size $p_1 = 1/(3q)$. If J_1 is assigned to M_q, then the second job has the same size $p_2 = 1/(3q)$ and otherwise $p'_2 = 1 - 1/(3q) - 1/(3q^2)$. Consider the following four cases.

If both jobs are assigned to the slow machine, the next two jobs have sizes $p_3 = 2/(3q)$ and $p_4 = 1 - 1/(3q)$. Note that $p_3 \leq p_4$. OPT assigns p_1 and p_3 to M_q and the two other jobs to M_1.

If at least one extra job is assigned to M_q by the algorithm, we get $P_q \geq p_1+p_2+p_3 = 4/(3q)$, which gives the competitive ratio $4/3 \geq (3q+1)/(3q)$ (holds for all $q \geq 1$). Otherwise J_3 and J_4 are assigned to M_1 and $P_1 = 1 + 1/(3q)$.

If J_1 is assigned to M_q and J_2 to M_1, there is only one extra job J_3' of size $p_3' = 1$. In this case, no matter which machine receives it, its load becomes $1 + 1/(3q)$, which gives the required competitive ratio. OPT assigns J_1 and J_2 to M_q and J_3' to M_1.

If J_1 to M_1 and J_2' is assigned to M_q, then there is one extra job as in the previous case with $p_3' = 1$. In this case, since $3q^2 - 2q - 1 \geq 0$ for any $q \geq 1$, then $p_2 \leq p_1$. After the first two jobs, M_1 has the same load as in the previous case and M_q is more loaded, so the assignment of J_3' also gives load of at least $1 + 1/(3q)$ on each machine. However $p_2' + p_1 = (3q^2 - 1)/(3q^2) \leq 1/q$ holds for $q \leq (3 + \sqrt{21})/6 \approx 1.26376$, and hence the same assignment for OPT is still valid. This again gives the needed competitive ratio .

In the last case both J_1 and J_2' are assigned to M_1 by the algorithm. The jobs J_3'' and J_4'' have sizes $p_3'' = 2/(3q)$ and $p_4'' = (q + 1)/(3q^2)$. Note that $p_3'' \geq p_4''$ for any $q \geq 1$. OPT assigns J_1 and J_3'' to M_q and the other two jobs to M_1. If the algorithm assigns both last jobs to M_q, its load is $P_j = (3q + 1)/(3q^2)$, which gives the required competitive ratio. Otherwise, at least one extra job is assigned to M_1. The load of M_1 becomes at least $p_1 + p_2' + p_4'' = 1/(3q) + (3q^2 - q - 1)/(3q^2) + (q + 1)/(3q^2) = (3q^2 + q)/(3q^2) = 1 + 1/(3q)$, that also gives the required competitive ratio . $\square$

Lemma 4.8 *Any deterministic online algorithm for bin stretching on related machines with speed ratio* $q_1 \approx 1.12433 \leq q \leq (1 + \sqrt{65})/8$ *has competitive ratio of at least* $(\frac{15+\sqrt{65}}{20})q$.

Proof. Applying the lower bound instance for $q_2 = (1 + \sqrt{65})/8$ we get a valid lower bound sequence for the current interval. The reason is that OPT can assign the jobs in the same way (it now has more room on the slow machine). The slow machine is faster by a factor of q/q_2 and so the value that the lower bound gives decreases by a factor of q/q_2. $\square$

The difference function between the upper bound and the lower bound is clearly monotonically decreasing in this interval. Hence the largest gap occurs at q_1 and is ≈ 0.011346.

Lemma 4.9 *Any deterministic online algorithm for bin stretching on related machines with speed ratio* $(1 + \sqrt{3})/2 \leq q \leq \phi = (1 + \sqrt{5})/2 \approx 1.61803$ *has competitive ratio of at least* $1 + \frac{1}{2q}$.

The difference function between the upper bound and the lower bound is $(2q^2 - 2q - 1)/(2q)$ for $(1 + \sqrt{3})/2 \leq q \leq \sqrt{2}$, and $(q - 1)/(2q(1 + q))$ for $\sqrt{2} \leq q \leq \phi$. In the first interval it is monotonically increasing and in the second interval it is monotonically decreasing. Hence the largest gap occurs at $\sqrt{2}$ and is $3\sqrt{2}/4 - 1 \approx 0.06066$.

Proof. The first job has size $p_1 = 1/(2q)$. If it is assigned to M_1 then the second job has size $p_2' = 1$. Otherwise there are two additional jobs of sizes $p_2 = 1/q$ and $p_3 = 1 - 1/(2q)$.

In the first case, OPT assigns J_1 to M_q and J_2' to M_1. If the algorithm assigns J_2' to M_1, its load is $1 + 1/(2q)$ which gives the required competitive ratio . If the algorithm assigns J_2' to M_q its load is 1, using $1 \geq (2q+1)/(2q^2)$ for $q \geq (1+\sqrt{3})/2$, we get the competitive ratio.

In the second case, OPT assigns J_2 to M_q and J_1 and J_3 to M_1. If at least one additional job is assigned to M_q we get $P_q \geq \min\{1/q, (2q-1)/(2q)\}$. For $q \leq 3/2$, $1 - 1/(2q) \leq 1/q$, and $P_q \geq p_1 + p_3 \geq 1$ which gives the competitive ratio as in the previous case. For $q \geq 3/2$, $P_q \geq 1/(2q) + 1/q = 3/(2q)$, which gives the competitive ratio (since $3/2 \geq 1 + 1/(2q)$ for all $q \geq 1$). The only case left is when both last jobs are assigned to M_1. In this case $P_1 = 1 + 1/(2q)$ and we are done. □

Lemma 4.10 *Any deterministic online algorithm for bin stretching on related machines with speed ratio $\phi \leq q \leq \sqrt{3}$ has competitive ratio of at least $\frac{q+1}{2}$.*

The difference function between the upper bound and the lower bound is $(3 - q^2)/(2q + 2)$, which is monotonically decreasing in this interval. Hence the largest gap occurs at $q = \phi$ and is $7/4 - 3\sqrt{5}/4 \approx 0.072949$.

Proof. The first job has size $p_1 = (q-1)/(2q)$. If it is assigned to M_1, another job of the same size arrives. Let k be the number of jobs so far ($k = 1$ or $k = 2$).

If there is at least one job assigned to M_q, the next jobs are $p_{k+1} = 1 - kp_1$ (if $k = 1$ $p_{k+1} = (q+1)/(2q)$ and if $k = 2$ then $p_{k+1} = 1 - (q-1)/q = 1/q$) and $p_{k+2} = 1/q$. Clearly OPT assigns all jobs but J_{k+2} to M_1 (their sum is exactly 1) and J_{k+2} to M_q. In both cases $p_{k+1} \geq 1/q$, hence the smallest job that can be now assigned to M_q is of size $1/q$. If at least one of the two new jobs is assigned to M_q we get $P_q \geq (q-1)/(2q) + 1/q = (q+1)/(2q)$, which gives the competitive ratio. Otherwise $P_1 \geq (1 + 1/q) - p_1 = (q+3)/(2q)$. For $q \leq \sqrt{3}$ we have $(q+3)/(2q) = 1/2 + 3/(2q) \geq 1/2 + q/2$, which also gives the competitive ratio.

Otherwise $k = 2$, and the next job has size $p_3 = 1$. For $q \leq 2$, $p_1 + p_2 = (q-1)/2 \leq 1/q$ hence OPT can run J_1 on M_q and J_2 on M_1. If J_2 is assigned to M_q then $P_q = 1 \geq (q+1)/(2q)$ (holds for any $q \geq 1$). Otherwise $P_1 = 1 + 2(q-1)/(2q) = (2q-1)/q$. Since for $1 \leq q \leq 2$ we have $2(2q-1) \leq q(1+q)$, we get the competitive ratio . □

5 Conclusions

We studied bin stretching on two related machines, and semi-online bin stretching on two identical machines. Both are natural generalizations of bin stretching. Our study of non-preemptive bin stretching on two related machines leaves some small gaps which can be interesting to close.

Another interesting problem is the combination of the two variants studied in this paper. Specifically the problem of identifying the best competitive ratio as a function of the speed ratio for non-preemptive semi-online bin stretching (non-increasing sizes of jobs) is an interesting natural problem that was not studied in this paper.

Acknowledgments

The author would like to thank Hadas Shachnai and Tami Tamir from the Technion for interesting discussions on [5, 18] and [17] that led to this work.

References

[1] Y. Azar. On-line load balancing. In A. Fiat and G. Woeginger, editors, *Online Algorithms - The State of the Art*, chapter 8, pages 178–195. Springer, 1998.

[2] Y. Azar and O. Regev. On-line bin-stretching. *Theoretical Computer Science*, 268(1):17–41, 2001. Also in the proc. of RANDOM'98, pages 71-81.

[3] E.G. Coffman, M.R. Garey, and D.S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, 7(1):1–17, 1978.

[4] J. Csirik and G. J. Woeginger. On-line packing and covering problems. In A. Fiat and G. Woeginger, editors, *Online Algorithms - The State of the Art*, chapter 7, pages 147–177. Springer, 1998.

[5] P. Dell'Olmo, H. Kellerer, M. G. Speranza, and Zs. Tuza. A 13/12 approximation algorithm for bin packing with extendable bins. *Information Processing Letters*, 65(5):229–233, 1998.

[6] P. Dell'Olmo and M. G. Speranza. Approximation algorithms for partitioning small items in unequal bins to minimize the total size. *Discrete Applied Mathematics*, 94:181–191, 1999.

[7] L. Epstein and L. M. Favrholdt. Optimal non-preemptive semi-online scheduling on two related machines. In *Proc. of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS'2002)*, pages 245–256, 2002.

[8] L. Epstein and L. M. Favrholdt. Optimal preemptive semi-online scheduling to minimize makespan on two related machines. *Operations Research Letters*, 30(4):269–275, 2002.

[9] L. Epstein, J. Noga, S. S. Seiden, J. Sgall, and G. J. Woeginger. Randomized Online Scheduling on Two Uniform Machines. *Journal of Scheduling*, 4(2):71–92, 2001.

[10] E. Girlich, V. Kotov, and M. Kovalev. Semi on-line algorithm for multiprocessor scheduling problem. Technical Report 98-05, The Mathematical Department, University Magdeburg, Germany, 1998.

[11] T. F. Gonzales and S. Sahni. Preemptive scheduling of uniform processor systems. *Journal of the ACM*, 25:92–101, 1978.

[12] Y. He and G. Zhang. Semi on-line scheduling on two identical machines. *Computing*, 62(3):179–187, 1999.

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

[14] H. Kellerer, V. Kotov, M. G. Speranza, and Zs. Tuza. Semi online algorithms for the partition problem. *Operations Research Letters*, 21:235–242, 1997.

[15] S. Seiden, J. Sgall, and G. Woeginger. Semi-online scheduling with decreasing job sizes. *Operations Research Letters*, 27(5):215–221, 2000.

[16] J. Sgall. On-line scheduling. In A. Fiat and G. Woeginger, editors, *Online Algorithms - The State of the Art*, chapter 9, pages 196–231. Springer, 1998.

[17] H. Shachnai, T. Tamir, and G. J. Woeginger. Minimizing makespan and preemption costs on a system of uniform machines. In *Proc of the 10th Annual European Symposium on Algorithms (ESA'2002)*, pages 859–871, 2002.

[18] M. G. Speranza and Zs. Tuza. On-line approximation algorithms for scheduling tasks on identical machines with extendable working time. *Annals of Operations Research*, 86:491–506, 1999.

[19] Z. Tan and Y. He. Ordinal on-line scheduling on two uniform machines. In *Proc. of the 6th Annual International Conference on Computing and Combinatorics, (COCOON'2000)*, pages 232–241, 2000.

[20] J. Wen and D. Du. Preemptive On-Line Scheduling for Two Uniform Processors. *Operations Research Letters*, 23:113–116, 1998.

[21] G. Zhang. A simple semi on-line algorithm for $P2//C_{\max}$ with a buffer. *Information Processing Letters*, 61:145–148, 1997.